CNTA

HANDBUCH

Version 10.0

25. April 2018

Eulenkrugstraße 83 - 22359 Hamburg - Tel: 0 40 - 6 03 05 62

CNTA-Handbuch
Stand 25. April 2018

Eulenkrugstraße 83
22359 Hamburg

Tel: (0 40) 6 03 05 62
E-Mail: info@wesselhoeft.de

Inhaltsverzeichnis

Einführung

CNTA

Stand 25. April 2018

Übersicht

CNTA ist das zentrale Programm zur Aufbereitung und Auswertung statistischer Daten aus dem CNT-Programmpaket. Es umfasst die Funktionen: Dateneingabe aus den verschiedensten Formaten, Datenprüfung und -korrektur, Datenaufbereitung, Vergabe von Fall- und Repräsentanzgewichten, unterschiedliche Grobauswertungen, Auswertungen in Tabellenform und Datenausgabe in den verschiedensten Formaten.

Die Arbeit mit CNTA ist leicht zu erlernen, insbesondere sind keine Programmierkenntnisse erforderlich. Andererseits besitzt CNTA ein sehr umfangreiches Funktionsspektrum, das es erlaubt, auch extreme Aufgabenstellungen zu lösen.

Die Auswertungsaufträge für CNTA sind mit Hilfe sogenannter Statements zu formulieren. Diese Befehle werden mit Hilfe eines beliebigen Texteditors oder Textverarbeitungsprogramms in einer Datei gespeichert und danach an CNTA zur Bearbeitung übergeben. CNTA liest diese Datei, prüft sie auf Fehler und führt danach sofort die Auswertung durch. Dieses Prinzip *Compile and Go* verlangt keinen zusätzlichen Übersetzungsschritt mit einem anderen Compilerprogramm.

Dieses Handbuch gibt im ersten Teil eine Einführung in die Arbeitsweise des Programms. Es werden die Regeln für die Erstellung der Statements beschrieben ohne sofort alle Möglichkeiten aufzuführen. Grundlegend für alle nachfolgenden Abschnitte wird gezeigt, wie Eingabedaten über externe Datenfelder mit Wertezuweisungen in Variable gestellt und damit für CNTA zur weiteren Verarbeitung verfügbar gemacht werden. Dazu gibt es viele Beispiele.

Der zweite Teil beschreibt die einzelnen Statements vollständig, zum leichteren Nachschlagen alphabetisch geordnet.

Der dritte Teil zeigt alle Fehlermeldungen, die während eines Verarbeitungslaufs ausgegeben werden können.

Es folgt eine Übersicht über die wichtigsten Funktionen und Statements des Programms:

Externe Dateien

Die auszuwertenden Datenbestände liegen in der Regel als sogenannte externe Dateien vor.
Hier kann CNTA neben den klassischen Column-Binary-Formaten der Marktforschung auch Dateien im EBCDIC- oder ASCII-Format direkt auswerten. Die zu verarbeitende Datei und ihr Aufbau werden durch das Statement INPUT-EXT beschrieben.

Externe Datenfelder

Innerhalb der verschiedenen Dateiformate kennt CNTA eine Vielzahl unterschiedlicher Typen von Datenfeldern. Dazu gehören alle praktisch relevanten Formen von statistischen Merkmalen (Kategorien-Werte, auch mit Mehrfachnennungen), numerische Felder (aus Zeichen, als gepackte Dezimalzahlen oder Integer-Werte) sowie Texte.

INPUT-EXT, OUTPUT-EXT

Die eingelesenen und gegebenenfalls aufbereiteten, bereinigten und selektierten Daten können durch das Statement OUTPUT-EXT wieder als externe Datei ausgegeben werden.
Hierfür stehen wie bei INPUT-EXT alle möglichen Dateiformate und Feldtypen zur Verfügung. Praktisch alle denkbaren Umformungen oder Recodierungen sind möglich. Zum Beispiel können die Daten von einer externen Datei im ASCII-Format eingelesen und mit einem völlig veränderten Spaltenplan auf eine Datei im Column-Binary-Format wieder ausgegeben werden.

INPUT-INT, OUTPUT-INT, INPUT-SEC

CNTA besitzt ein eigenes internes Datenformat, das die auszuwertenden Daten komprimiert in Form von Variablen enthält. Obendrein werden alle zu den Variablen eingegebenen Texte in einer solchen Datei für die spätere Auswertung gespeichert. Diese internen Dateien lassen sich besonders leicht, schnell und fehlerarm verarbeiten. Sie empfehlen sich insbesondere für Datenbestände, die nach ihrer Erstellung mehrfach auszuwerten sind. Solche internen Dateien werden durch das Statement OUTPUT-INT erstellt und können danach mit den Statements INPUT-INT oder INPUT-SEC wieder eingelesen werden.

Weiter können in einem Verarbeitungslauf interne Dateien und eine externe Datei gleichzeitig verarbeitet und dabei die verschiedenen statistischen Fälle zusammengeführt werden. Dies ermöglicht, eine bestehende interne Datei mit neuen oder geänderten Daten zu versehen und gegebenenfalls als fortgeschriebene Datei wieder auszugeben. Hierzu sind zusätzlich die Statements ADD-PROC und UPD-PROC erforderlich.

FETCH-FILE

Mit diesen Statements ist es möglich, zu den Datenwerten der statistischen Fälle Zusatzinformationen aus anderen Dateien einzulesen. Zum Beispiel kann mit einer Gemeindekennziffer auf eine Gemeindatei zugegriffen werden um die dort zu der Kennziffer gespeicherten Daten zum statistischen Fall hinzuzufügen.

DEFS

Um die Daten eines jeden statistischen Falles verarbeiten zu können, sind diese in die Form von Variablen zu bringen. Variable werden mit Hilfe des Statements DEFS und seinen Folgestatements definiert und vertextet.

ADD-PROC, UPD-PROC, MOD-PROC

Nach ihrer Definition werden den Variablen Werte aus den statistischen Fällen zugewiesen.
Dies geschieht durch ADD-PROC, UPD-PROC und MOD-PROC und deren Folgestatements. Neben diesen Wertezuweisungen erlauben die Folgestatments umfangreiche Prüfungen und Bereinigungen der Eingabedaten. Weiter können Werte neu berechnet und umgeschlüsselt werden, und es gibt verschiedene Möglichkeiten, die Daten einzelner statistischer Fälle auszudrucken. Weitere Einzelheiten finden sich auch in den Abschnitten *Logische Ausdrücke*, *Numerische Ausdrücke* und *Wertezuweisungen* sowie bei den Statements IF, DO, PRT und TST.

SELECT

CNTA verfügt über verschiedene Möglichkeiten, mit Hilfe des Statements SELECT aus den Eingabedaten Teilstichproben zu erstellen.

WEIGHT

Das Statement WEIGHT weist den statistischen Fällen aufgrund vorgegebener Sollwerte Fallgewichte zu. Dabei sind sowohl Zellen- als auch Randgewichtungen in mehreren Dimensionen möglich. CNTA ermittelt die Randgewichte mit einem Optimierungsverfahren, der Methode der kleinsten Quadrate mit Lagrange-Multiplikatoren. Dies arbeitet schnell und liefert Wichtungsergebnisse, die den üblichen heuristischen Iterationsverfahren in der Qualität meist deutlich überlegen sind.

FUSION

Dieses Statement ordnet den statistischen Fällen der laufenden Verarbeitung (Empfänger; Rezipient) einen Datensatz aus einer weiteren internen Datei (Spender, Donor) zu.
Dies geschieht durch Vergleich von Variablenwerten der Empfänger mit entsprechenden Variablenwerten der Spender, wobei möglichst ähnliche Fälle zusammengeführt werden.
Die Spenderdaten lassen sich in bestehende oder neu definierte Variable übernehmen.

PAGE, PAGEP

Mit Hilfe der Statements PAGE und PAGEP lassen sich die Druckausgaben gestalten.
Dazu werden zunächst die Größe, Format und Orientierung der Ausgabeseiten festgelegt.
Möglich ist aber auch die Einteilung der Seiten in mehrere Spalten. Auf diese Weise lassen sich mehrere kleine Tabellen sehr einfach auf einem Blatt platzieren. Weiter können Kopf- und Fußtexte festgelegt und die Seitennummerierung gesteuert werden. Schließlich werden Linien und Schriften festgelegt und Druckzeichen für Sonderzwecke angegeben: Zum Beispiel Zeichen für punktierte Linien, Zeichen für die Zählergebnisse Null und gerundete Null.
Bei Druckausgabe auf Seitendrucker kann auch mit Proportionalschriften, verschiedenen Schriftgrößen, Fettung, Kursivschrift, Unterstreichungen und Tonflächen gearbeitet werden.
Weiter lassen sich in die Auswertungen gescannte Bilder einfügen.
Derartige Druckausgaben können direkt für Setzmaschinen im PostScript-Format aufbereitet werden. Damit besteht die Möglichkeit, Berichtsbände ohne weitere Zwischenschritte für den Satz fertig zu stellen, Zeitaufwand und Zusatzkosten entfallen.

XTAB

Die eigentliche Auswertung in Form von Tabellen erfolgt durch das Statement XTAB.
Dies ist einer der Schwerpunkte von CNTA mit einer Fülle von Möglichkeiten zur Gestaltung der Auswertungen.

XADD, PAGEP

Die Statements XADD und XADDB erzeugen ebenfalls Kreuztabellen, die jedoch nicht ausgedruckt werden. Stattdessen addiert CNTA deren Ergebnisse auf die Werte von XTAB-Tabellen, bevor es diese zum Drucken aufbereitet. Damit ist es unter anderem möglich, *unechte Kreuztabellen* zu erzeugen.

INDEX

Zu den mit XTAB erzeugten Tabellen liefert das Statement INDEX ein Inhaltsverzeichnis, das neben verschiedenen Textelementen aus den Statements die entsprechenden Seitennummern ausweist.

HOLECOUNT, CODEBOOK

Grobauswertungen in Form von Globalzählungen liefert das Statement HOLECOUNT.
Eine vertextete und stärker aufbereitete Form lässt sich mit CODEBOOK erzeugen.

REPORT

Mit diesen Statements lassen sich die Daten der einzelnen statistischen Fälle beliebig aufbereiten und anzeigen.

TEXT

Dieses Statement stellt beliebige freie Texte in die Auswertungsergebnisse, zum Beispiel als Legenden auf den Tabellenseiten oder als eigene Textseiten vor und zwischen den verschiedenen Auswertungsergebnissen.

Makros

CNTA verfügt über eine Makrotechnik, mit der umfangreiche Auswertungen leichter, schneller und übersichtlicher zu erstellen sind. Sie erlaubt es aber auch, Auswertungen zu standardisieren und nur noch durch geringe Eingriffe an die jeweilige Aufgabe anzupassen.

INCLUDE

Häufig wiederkehrende Statements oder Gruppen von Statements können in getrennten Teildateien gespeichert und mit Hilfe des Statements INCLUDE in spätere Auswertungen eingefügt werden.

DO

Mit dem Statement DO verfügt CNTA über eine Schleifentechnik, die die Schreibarbeit bei umfangreicheren Auswertungen stark reduzieren kann.

OPTIONS

Das Statement OPTIONS dient dazu, grundsätzliche Verarbeitungsregeln anzugeben.
Hier kann zum Beispiel festgelegt werden, ob bestimmte Fehlermeldungen auszugeben sind oder nach wie vielen Fehlermeldungen eine Verarbeitung abzubrechen ist.

PRINT OFF, PRINT ON

Die Statements PRINT OFF und PRINT ON sorgen dafür, dass die dazwischen liegenden Statements zwar verarbeitet aber nicht in dem von CNTA erstellten Zählprotokoll ausgedruckt werden.

PROCESS ON / OFF

Ähnlich arbeiten die Statements PROCESS OFF und PROCESS ON. Die dazwischen liegenden Statements werden weder verarbeitet noch im Zählprotokoll gedruckt. Sie können zum Beispiel dazu verwendet werden, Teile einer umfangreichen Auswertung zu wiederholen, ohne gleich eine neue Statementdatei zu erstellen.

Alle hier aufgeführten Funktionen können in einem Verarbeitungslauf miteinander kombiniert werden. Für die verschiedenen Teilfunktionen sind keine unterschiedlichen Programmläufe oder gar Programme erforderlich. So könnten zum Beispiel in einem Durchlauf eine externe Datei eingelesen, die Eingabedaten geprüft und aufbereitet, Kreuztabellen und eine Globalzählung erstellt sowie eine interne Datei wieder ausgegeben werden.

CNTA erlaubt auch sehr umfangreiche Auswertungen auf kleinen Rechnern: Reicht der Hauptspeicher nicht aus, um alle Statements in einem Durchlauf zu verarbeiten, so wird der Auftrag zerlegt und in mehreren Durchläufen ausgewertet.

Variable und Textelemente

Die Daten eines statistischen Falles müssen in Form von Variablen vorliegen, bevor CNTA sie auswerten kann. Eine Variable ist ein Speicherelement, das zu jedem statistischen Fall Werte aufnimmt.

Jede Variable ist zunächst hinter dem Statement DEFS zu definieren. Dort ist ein Variablenname anzugeben und festzulegen, welche Werte die Variable aufnehmen soll. Zusätzlich kann sie beschreibende Texte erhalten, die in den Auswertungen zur Vertextung der Zählergebnisse dienen.

Variablennamen dürfen aus den Buchstaben A....Z, den Ziffern 0 ... 9 sowie dem Zeichen _ bestehen. Dabei wird nicht zwischen Groß- und Kleinbuchstaben unterschieden. Umlaute Ä, Ö, Ü sowie ß sind nicht möglich. Variablennamen müssen mit dem Buchstaben C, N oder T beginnen und können zwei bis zehn Zeichen lang sein. Endet ein Variablenname mit einer Zahl, so ergeben führende Nullen keine neue Variable.
So bezeichnen N001, N01 und N1 dieselbe Variable. Dagegen sind NX und NX0 voneinander verschieden.

Zu Beginn der Verarbeitung eines jeden statistischen Falles werden die Werte der Variablen automatisch gelöscht, es sei denn, sie wurden mit der Angabe DUP definiert. Während der Verarbeitung des Falles können ihnen Werte zur späteren Auswertung zugewiesen werden.

Das Programm kennt drei Typen von Variablen:

C-Variable

C-Variable (Bedingungsvariable, Kategorienvariable) beginnen mit dem Buchstaben C. Sie speichern *Merkmale* von statistischen Fällen wie Antwortvorgaben, Kategorien, Bewertungen und andere diskrete Werte.

C-Variable erlauben bis zu 10 000 *Merkmale*, die durch die Merkmalsnummern 1 bis 10 000 angesprochen werden. Jedes Merkmal kann den Wert 1 für *erfüllt* oder 0 für *nicht erfüllt* annehmen. Beliebig viele Merkmale einer C-Variablen dürfen gleichzeitig erfüllt sein.

Die Statements:

```
DEFS
-C105:2 'Geschlecht' 1='Männer' 2='Frauen'
```

definieren die Variable C105 mit zwei Merkmalen. Sie erhält den Variablentext *Geschlecht* und ihre beiden Merkmale die Merkmalstexte *Männer* und *Frauen*. Mit den Statements

```
ADD-PROC
-C105=M1(10,1)
```

könnte diese Variable Werte aus einer Eingabedatei erhalten. Einzelheiten dazu finden sich in der Beschreibung der Statements *DEFS* und *ADD-PROC* sowie im Abschnitt *Wertezuweisungen*.
In den Auswertungen kann in der Form *C105.1* auf die Männer und mit *C105.2* auf die Frauen zugegriffen werden.

N-Variable

Numerische Variable beginnen mit dem Buchstaben N. Sie speichern Zahlen mit bis zu 18 Dezimalziffern, auch mit Nachkommastellen und negativen Vorzeichen, jedoch keine Mehrfachnennungen. Neben Zahlen ist auch der künstliche Wert Z0 für *keine Angabe* oder *fehlender Wert* möglich.

Die Statements:

```
DEFS
-N10:4 'Nettoeinkommen'
```

definieren die Variable N10, die Zahlenwerte aus maximal vier Dezimalziffern ohne Nachkommastellen aufnehmen kann, also Werte zwischen –9 999 und +9 999. Sie erhält den Variablentext *Nettoeinkommen*. Einzelheiten zur Definition der N-Variablen und ihrer Wertezuweisungen finden sich in der Beschreibung der Statements *DEFS* und *ADD-PROC* sowie

im Abschnitt *Wertezuweisungen*.

T-Variable

Textvariable beginnen mit dem Buchstaben N. Sie speichern Texte zu den statistischen Fällen, wie den Wohnort eines Befragten, einen unverschlüsselten Markennamen oder den Antworttext auf eine offene Frage.

Jede Textvariable kann Texte von bis zu 4 000 Zeichen Länge aufnehmen.

Die Statements

```
DEFS
-T10:30  'Wohnort'
```

definieren die Textvariable T10, die Texte von bis zu 30 Zeichen Länge aufnehmen kann. Sie erhält den Variablentext *Wohnort*. Einzelheiten zur Definition der T-Variablen und ihrer Wertezuweisungen finden sich in der Beschreibung der Statements *DEFS* und *ADD-PROC* sowie im Abschnitt *Wertezuweisungen*.

Variable mit Index

Diese Einrichtung erleichtert die Arbeit mit ähnlich aufgebauten Variablen. Jede Variable kann unter DEFS mit einem Index [...] versehen werden, zum Beispiel:

```
DEFS
-C200[5]:4   1='0 bis 5 Jahre'
             2='6 bis 10 Jahre'
             3='10 bis 15 Jahre'
             4='16 und mehr Jahre'
-C200[1]  'Alter des 1. Kindes'
-C200[2]  'Alter des 2. Kindes'
-C200[3]  'Alter des 3. Kindes'
-C200[4]  'Alter des 4. Kindes'
-C200[5]  'Alter des 5. Kindes'
```

Das Statement -C200[5]:4 definiert dabei nicht eine, sondern fünf Variable, alle mit den gleichen vier Merkmalen und Merkmalstexten. Die dahinter stehenden Statements geben jeder dieser fünf Variablen einen individuellen Variablentext *Alter des 1. Kindes* bis *Alter des 5. Kindes*.

Als Index sind Zahlen zwischen 1 und 99 999 zulässig. Auf diese Weise vervielfacht sich die Anzahl der verfügbaren Variablen.

Wird eine mit Index definierte Variable später verwendet, so ist stets ein Index anzugeben, zum Beispiel in den Wertezuweisungen

```
ADD-PROC
-C200[1]=M1(13)
```

Hier erhält nur die Variable C200[1] einen Wert aus den Eingabedaten, die Variablen mit dem Index [2] ... [5] bleiben unverändert.

Als Index sind auch N-Variable und Rechenausdrücke möglich. Zum Beispiel:

```
ADD-PROC
-N1=D1(10)
-C200[N1-7]=M1(11)
```

Das Statement -N1=D1(10) versieht hier die Variable N1 für jeden statistischen Fall mit einem Wert aus den Eingabedaten. Von diesem Wert wird 7 subtrahiert und das Ergebnis als Index für die Variable C200 verwendet. Nur die Variable mit diesem Index erhält danach einen Wert aus dem statistischen Fall zugewiesen. In solchen Rechenausdrücken sind die Operatoren + - * und / sowie Klammern möglich.

Textelemente

Textelemente speichern Texte, die während eines Verarbeitungslaufs unverändert bleiben, im Gegensatz zu Textvariablen, deren Inhalt sich für jeden statistischen Fall ändern kann.
In jedem Verarbeitungslauf lassen sich maximal 100 000 Textelemente definieren, mit den vorgegebenen Bezeichnungen (0) ... (99999)

Die Statements:

```
DEFS
-(21) 'Ich lese Ihnen jetzt einmal einige Aussagen vor'/
      'und Sie sagen mir bitte inwieweit diese zutreffen'
```

definieren das zweizeilige Textelement (21).

Ein solches Textelement kann Texte für die Auswertungen liefern, aber auch als Baustein für weitere Texte dienen, zum Beispiel bei sich wiederholenden Teilen von Variablentexten.
Zum Beispiel erzeugt das Statement

```
XTAB  TITLE=(21) ...
```

eine Kreuztabelle mit dem hier definierten Text als Überschrift. Einzelheiten dazu finden sich in der Beschreibung der Statements *DEFS* und im Abschnitt *Textzeilen*.

Abkürzungen

Hinter DEFS können mehrere Variable mit Index in einer Zeile definiert werden:

```
Statt:      –C1[5]:12
            –C2[5]:12
            –C3[5]:12
genügt:     –C1 .. 3[5]:12

Statt       –C1[3]  'Einkommen'
            –C2[3]  'Einkommen'
            –C3[3]  'Einkommen'
genügt:     –C1 .. 3[3]  'Einkommen'
```

In den Folgestatements von ADD-PROC, MOD-PROC, UPD-PROC und OUTPUT-EXT können mehreren Variablen mit Index in einer Zeile Werte zugewiesen werden:

```
Statt:      –N10[1] = 0
            –N10[2] = 0
            –N10[3] = 0
genügt:     –N10[1] .. 3 = 0
oder auch:  –N10 = 0      wenn die Variable N10 mit drei Indexwerten definiert wurde.

Statt:      –N1[2] = D1(1,2)
            –N2[2] = D1(3,2)
            –N3[2] = D1(5,2)
genügt:     –N1 .. 3[2] = D1(1,2)     INC=2
```

Wenn in einem Statement mehrmals hintereinander auf die gleiche Variable mit Index zugegriffen wird, kann die Variablenbezeichnung weggelassen werden:

```
Statt:      XTAB   ROW = N10[1] ; N10[2] ; N10[3] ; N10[4] ; N10[5]
genügt:     XTAB   ROW = N10[1] ; .. [5]
oder auch:  XTAB   ROW = N10[1] ; .. 5

Statt:      –N0 = N10[1] + N10[2] + N10[3] + N10[4] + N10[5]
genügt:     –N0 = N10[1] + .. [5]
oder auch:  –N0 = N10[1] + .. 5

Statt:      XTAB   ROW = N10[1] ; N10[3] ; N10[5] ; N10[7]
genügt:     XTAB   ROW = N10[1] ; [3] ; [5] ; [7]

Statt:      –N0 = N10[1] + N10[3] + N10[5] + N10[7]
genügt:     –N0 = N10[1] + [3] + [5] + [7]
```

Ein Verarbeitungslauf

Hier wird als Beispiel ein vollständiger Verarbeitungslauf mit CNTA beschrieben, der die Datei DATEN.EXT auswertet. Sie soll folgende Datenzeilen enthalten:

```
001 1 2 31 BRAUN
002 1 2 35 POHL
003 2 2 36 SCHARF
004 1 3 45 MÜLLER
005 1 3 53 LEHMANN
006 1 1 25 HUBER
007 2 2    BOSSLER
008 1 3 34 HEITMANN
009 2 2 38 MAYER
010 1 2 21 KERN
011 2 2 40 LANGE
012 2 3 48 WITTHÖFT
```

Jede Zeile enthält die Werte eines statistischen Falles, der aus einem Fragebogen hervorgegangen ist. Dieser Datenbestand kann aus einer beliebigen Quelle stammen, zum Beispiel einem speziellen Erfassungsprogramm, einem Tabellenkalkulations-, Text- oder Editorprogramm.

In den Positionen 1 bis 3 der Datensätze sind die Fragebogennummern 001 bis 012 erfasst.
Solche Fragebogennummern, Versuchspersonennummern, Fallkennzeichen usw. werden in CNTA als *Fall-Identifikation* bezeichnet. CNTA verwendet sie unter anderem zur Prüfung der Reihenfolge, zum Erkennen doppelter Fälle und zur Kennzeichnung von Fehlermeldungen.

In der Position 5 ist das Geschlecht der Befragten verschlüsselt: 1 für *Männer* und 2 für *Frauen.*

Die Position 7 enthält das ebenfalls verschlüsselte Haushaltseinkommen:
1 für *unter 2000 EUR,* 2 für *2000 bis unter 3000 EUR* und 3 für *3000 EUR und mehr*.

Die Positionen 9 und 10 enthalten das Alter des Befragten als zweistellige Zahlen. Der Fall 007 enthält hier keine Angabe.

Ab der Position 12 ist der Nachname der Befragten als Text erfasst.

Die Positionen 4, 6, 8 und 11 wurden leer gelassen, um die Lesbarkeit der Daten zu verbessern.

Zur Auswertung dieser Daten dienen CNTA-Statements. Diese Anweisungen an das Programm können mit einem Textverarbeitungs- oder Editor-Programm erstellt werden. In unserem Beispiel soll die Datei den Namen 0085.STM haben und folgendermaßen aussehen:

```
INPUT-EXT  FNAME=DATEN.EXT  ID=D(1..3)  ASCII  SIZE=80,EOL

DEFS
-C6:2   'Geschlecht'  1='Männer'
                      2='Frauen'
-C10:3  'Haushaltseinkommen'   1='bis  2000 EUR'
                               2='2000-3000 EUR'
                               3='über 3000 EUR'
-N10:2 'Alter'
-C20:4 'Altersklassen'  1='...20 Jahre'
                        2='21...35 Jahre'
                        3='36...59 Jahre'
                        4='60... Jahre'
-T100:30  'Name des Befragten'

ADD-PROC
-C6=M1(5)
-C10=M1(7)
-N10=D1(9,2)   >Z0
-C20.1= N10<=20
-C20.2= N10>=21 & N10<=35
-C20.3= N10>=36 & N10<=59
-C20.4= N10>=60
-T100=A1(12,30)

XTAB  FILTER=TOTAL
      COL=TOTAL;C6.1;C6.2;MEAN(N10)
      ROW=TOTAL:'Basis';;C10;C10.Z0:'keine Angabe'

OUTPUT-INT  FNAME=0085.INT
```

Das Programm liest diese Statements aus der Datei 0085.STM ein und verarbeitet danach die Daten der Datei DATEN.EXT. Dabei wird die Protokolldatei CNTLST erstellt, mit einer Auflistung der eingelesenen Statements und eventuellen Fehlermeldungen zu den Statements und Daten:

Es folgt eine kurze Beschreibung der Statements aus obigem Beispiel. Ausführliche Angaben findet sich im Kapitel *Statements* dieses Handbuchs.

INPUT-EXT FNAME=DATEN.EXT ASCII SIZE=80,EOL ID=D(1..3)

Dieses Statement beschreibt die Datei mit den auszuwertenden Daten. INPUT-EXT selbst besagt, dass ein externes Datenformat vorliegt. FNAME liefert den Dateinamen und ASCII bestimmt das Format der Eingabesätze.

SIZE=80 legt die Länge der einzelnen Datensätze pro Fall auf maximal 80 Positionen fest. Die Angabe EOL dahinter besagt, dass die einzelnen Datensätze mit Zeilenende-Zeichen abschließen.

ID=D(1..3) besagt, dass in den Positionen 1 ... 3 der Datensätze Fall-Identifikationen als numerische Werte gespeichert sind.

DEFS

Die zu verarbeitenden Datenwerte der statistischen Fälle müssen vor ihrer Verarbeitung in Variablen gespeichert werden. Das Statement DEFS leitet die Definition dieser Variablen ein.
Jede Variablendefinition beginnt mit einem Strich am Zeilenanfang:

- C6:2 'Geschlecht' 1='Männer' 2='Frauen'

Dieses Folgestatement von DEFS definiert die Variable C6. Die Angabe :2 hinter C6 legt fest, dass die Variable zwei Merkmale besitzen soll. Dahinter werden die Texte vorgegeben, die in den Auswertungen für die Variable C6 zu verwenden sind. *Geschlecht* ist dabei der Variablentext, *Männer* der Text des ersten und *Frauen* der Text des zweiten Merkmals.

- C10:3 'Haushaltseinkommen' 1=' bis 2000 EUR'
2='2000-3000 EUR'
3='über 3000 EUR'

In diesem Statement wird die Variable C10 mit drei Merkmalen für das Haushaltseinkommen definiert. Hier wurde die Definition auf drei Zeilen verteilt. Jede Zeile, die mit einer Leerstelle beginnt, wird als Fortsetzung der vorigen Zeile verstanden.

- N10:2 'Alter'

In diesem Folgestatement wird die numerische Variable N10 definiert. Die Angabe :2 legt fest, dass die Variablenwerte zwei Dezimalstellen lang sein dürfen. Es sind also Werte zwischen -99 und +99 möglich. Der Variablen N10 erhält den Text *Alter*.

- C20:4 'Altersklassen' 1='...20 Jahre'
2='21..35 Jahre'
3='36...59 Jahre'
4='60... Jahre'

Die Variable N10 enthält die Altersangabe als Zahlenwert. Dagegen soll die in diesem Folgestatement definierte Bedingungsvariable C20 ebenfalls das Alter aufnehmen, jedoch in Form von vier Altersklassen.

- T100:30 'Name des Befragten'

Dieses Folgestatement definiert die Textvariable T100. Wegen der Angabe :30 kann sie für jeden statistischen Fall maximal 30 Zeichen aufnehmen.

ADD-PROC

Dieses Statement leitet die Wertezuweisungen ein, durch die die Variablen Werte aus den Datensätzen erhalten. Alle Statements, die direkt hinter ADD-PROC liegen und mit einem Strich in der ersten Position beginnen, sind solche Wertezuweisungen:

- C6=M1(5)

Dieses Statement entnimmt die die Werte für die Variable C6 aus der Position 5 der Datensätze im M-Format: Befindet sich dort die Ziffer 1, so wird das Merkmal 1 für *Männer* auf *erfüllt* gesetzt, die Ziffer 2 setzt das Merkmal 2 für *Frauen* auf *erfüllt*. Alle anderen Werte setzen beide Merkmale auf *nicht erfüllt*. Details zum M-Format und allen anderen Datenformaten enthält der Abschnitt *Externe Datenfelder*.

- C10=M1(7)

Dieses Statement setzt analog zum vorigen Statement die Merkmale 1 ... 3 der Variablen C10 aus den Werten der Position 7 der Datensätze.

- N10=D1(9,2) >Z0

Dieses Folgestatement versieht die numerische Variable N10 mit Werten. Die Angabe D1 besagt, dass die Werte in der Eingabedatei im ersten oder einzigen Datensatz pro Fall in Form von Dezimalzahlen vorliegen, in Position 9 beginnen und zwei Stellen lang sind.

Die Angabe >Z0 sorgt für eine Datenprüfung. Unter Z0 versteht CNTA den Wert *keine Angabe*. So legt >Z0 fest, dass die Eingabe für die Variable N10 einen Wert *größer als Z0*, also einen echten Zahlenwert, enthalten muss.
Für jeden statistischen Fall, der diese Prüfbedingung nicht erfüllt, stellt CNTA eine Fehlermeldung in die Protokolldatei. In unserem Beispiel wird für den Fall 007 eine solche Meldung ausgegeben:

```
FALL-ID: 7
  FEHLER 508: FALSCHER WERT: D1(9,2)='Z0'           Stmt: 11
```

Über jeder Fehlermeldung steht die Identifikation des Falles, hier die Fragebogennummer 007.
Die Fehlernummer 508 verweist auf einen erläuternden Text im Anhang dieses Handbuchs. Dahinter wird das betroffene Eingabefeld D1(9,2) aufgeführt und der gefundene Wert, hier Z0 für *keine Angabe*. Zusätzlich wird die Nummer 11 des Statements ausgegeben, das die Prüfbedingung enthält.

- C20.1= N10<=20
- C20.2= N10>=21 & N10<=35
- C20.3= N10>=36 & N10<=59
- C20.4= N10>=60

Diese Statements setzen die Merkmale der Variablen C20 anhand der Werte der Variablen N10. Das erste Statement setzt das Merkmal 1 der Variablen C20 auf *erfüllt*, wenn der Wert der Variablen N10 kleiner oder gleich 20 ist. Andernfalls ist Merkmal 1 *nicht erfüllt.*
Das nächste Statement setzt Merkmal 2 der Variablen C20 auf *erfüllt*, wenn der Wert der Variablen N10 größer gleich 21 und kleinergleich 35 ist. Analog werden die Merkmale 3 und 4 gesetzt.

- T100=A1(12,30)

Mit diesem Folgestatement wird der Textvariablen T100 der Name des Befragten als Wert zugewiesen. Die Angabe A1(12,30) besagt dabei, dass die Eingabewerte als alphanumerische Daten vorliegen und ab Position 12 in der Länge von 30 Zeichen vorliegen.

XTAB FILTER=TOTAL
COL=TOTAL ; C6.1 ; C6.2 ; MEAN(N10)
ROW=TOTAL:'Basis' ; ; C10 ; C10.Z0 : 'keine Angabe'

Dieses Statement erzeugt folgende Kreuztabelle:

TOTAL 12

	TOTAL	Geschlecht		Alter
		Männer	Frauen	
Basis	12	7	5	37
Haushaltseinkommen				
bis 2000 EUR	1	1	-	25
2000-3000 EUR	7	3	4	34
über 3000 EUR	4	3	1	45
keine Angabe	-	-	-	-

Die Angabe FILTER=TOTAL bewirkt, dass alle statistischen Fälle der Eingabedatei in dieser Tabelle ausgewertet werden. Sie erzeugt auch die erste Zeile oberhalb der Tabelle.

COL = TOTAL ; C6.1 ; C6.2 ; MEAN(N10) liefert die die Spaltenbedingungen und die Beschriftung des Tabellenkopfes mit den Variablentexten

Dabei ist TOTAL eine spezielle Zählbedingung, die immer erfüllt ist und die TOTAL-Spalte erzeugt.

C6.1 bewirkt die Auswertung des Merkmals *Männer* der Variablen C6 und C6.2 die Auswertung des Merkmals *Frauen.* MEAN(N10) erzeugt die Mittelwerte aus dem Alter der Befragten. Im Tabellenkopf steht dafür der Variablentext von N10.

ROW = TOTAL : 'Basis' ; ; C10 ; C10.Z0 : 'keine Angabe' liefert analog die Zeilenbedingungen.
Da hier alle Merkmale der Variablen C10 auszuwerten sind, wird anstelle der komplizierteren Angabe C10.1 ; C10.2 ; C10.3 die Abkürzung C10 verwendet.
Der Text *Basis* ersetzt in der ersten Zeile den Standardtext des Operanden TOTAL.
Das zweite Semikolon zwischen TOTAL und C10 sorgt für eine zusätzliche Leerzeile.

Der Operand C10.Z0 steht für *keine Angabe in der Variablen C10*. Diese Bedingung wird selbsttätig von CNTA ermittelt und muss nicht gesondert unter DEFS definiert oder in einem Merkmal gespeichert werden. Der Text *keine Angabe* hinter *Z0 :* liefert den Zeilentext.

OUTPUT-INT FNAME=0085.INT

Dieses Statement erstellt eine Datei mit dem Dateinamen *0085.INT* im CNT-internen Format. Diese Datei enthält sämtliche Variablendefinitionen mit ihren Texten sowie zu jedem statistischen Fall einen Datensatz, der alle dazugehörigen Variablenwerte enthält.

Eine interne Datei ist ein selbstdokumentierender Datenbestand, der spätere Auswertungen wesentlich erleichtert und beschleunigt. Die Einrichtung einer solchen internen Datei ist vor allem zu empfehlen, wenn aus einem einmal erstellten Datenbestand wiederholt Auswertungen erfolgen sollen.

Logische Ausdrücke

Ein logischer Ausdruck liefert das Ergebnis 1 oder wahr, wenn er erfüllt ist, und das Ergebnis 0 oder falsch , wenn er nicht erfüllt ist. Er besteht im einfachsten Fall aus nur einem der folgenden Operanden:

Logische Ausdrücke dienen zur Auswertung komplexer Bedingungen.
Sie sind Bestandteil der Statements SELECT, IF, IFN, TST und der rechten Seite von Wertezuweisungen.
Logische Ausdrücke können nur auf der Ebene der einzelnen statistischen Fälle ausgewertet werden.
Sie sind daher nicht in Auswertungsstatements wie XTAB, CODEBOOK usw. einsetzbar.

0 Konstanter Merkmalswert *falsch* : Er ist nie erfüllt.

1 Konstanter Merkmalswert *wahr* : Er ist immer erfüllt.

Cn **.** a Dieser Operand ist erfüllt, wenn das Merkmal a = 1 ... 9999 der Bedingungsvariablen Cn erfüllt ist.
Zum Beispiel besitzt der Operand C10.7 den Wert *wahr* oder 1, wenn das Merkmal 7 der Variablen C10 erfüllt ist. Sonst hat der Operand C10.7 den Wert *falsch* oder 0.
Auch negative Merkmalsnummern können für *a* angegeben werden: C10.-1 ist das letzte Merkmal der Variablen, C10.-2 das vorletzte usw.

Cn . **Z**i Dieser Operand wird wahr oder falsch , je nachdem wie viele Merkmale in der Variablen Cn erfüllt sind.
Cn.Zi ist *wahr* , wenn die Variable Cn die Anzahlbedingung Zi erfüllt, andernfalls *falsch*. Dabei sind für Zi folgende Angaben möglich:

Z0	kein Merkmal ist erfüllt, alle haben den Wert *falsch*,
Z1	genau eines der Merkmale ist erfüllt, hat also den Wert *wahr*,
Z2	genau zwei der Merkmale sind erfüllt, besitzen also den Wert *wahr*,
Z3	drei oder mehr Merkmale sind erfüllt, besitzen also den Wert *wahr*.

Es sind auch kombinierte Anzahlbedingungen möglich:

Z01	keines oder genau ein Merkmal ist erfüllt,
Z123	mindestens ein Merkmal ist erfüllt, usw.

So verlangt C10.Z12, dass genau eines oder zwei der Merkmale der Variablen C10 erfüllt sind.

Anzahlbedingungen lassen sich auch auf ausgewählte Merkmale einer Variablen beschränken.
So ist die Bedingung

```
C10.Z1(8,13..17,21)
```

nur dann erfüllt, wenn genau eines der Merkmale 8, 13, 14, 15, 16, 17 oder 21 der Variablen C10 den Wert *wahr* besitzt und die anderen den Wert *falsch* . Auch negative Merkmalsnummern können hier angegeben werden: -1 ist das letzte Merkmal der Variablen, -2 das vorletzte usw.

ADD Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle mit Daten aus INPUT-EXT und ohne Daten aus INPUT-INT oder INPUT-SEC. In allen anderen Fällen besitzt ADD den Wert 0 = *falsch*.

OLD Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle mit Daten aus INPUT-INT oder INPUT-SEC und ohne Daten aus INPUT-EXT. In allen anderen Fällen besitzt OLD den Wert 0 = *falsch*.

UPD Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle, zu denen gleichzeitig Daten aus INPUT-EXT und Daten aus INPUT-INT oder INPUT-SEC vorliegen.
In allen anderen Fällen besitzt UPD den Wert 0 = *falsch*.

UPDSEC Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle aus INPUT-EXT oder INPUT-INT, zu denen ein Fall mit passender Identifikation aus einer INPUT-SEC-Datei eingelesen wurde. In allen anderen Fällen besitzt UPDSEC den Wert 0 = *falsch.*

ADDSEC Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle aus INPUT-SEC, zu denen kein Fall aus INPUT-EXT oder INPUT-INT mit passender Identifikation vorhanden ist. Die Bedingung ADDSEC ist auch für alle statistischen Fälle aus INPUT-SEC mit der Angabe ADD oder ADDP erfüllt. In allen anderen Fällen besitzt ADDSEC den Wert 0 = *falsch*.

CIDi . B
CIDi . E Diese Operanden sind wahr, wenn ein Gruppenwechsel der Fall-Identifikation IDi eintritt. Findet kein Gruppenwechsel statt, sind sie falsch .
Mehrere direkt hintereinander liegende statistische Fälle werden als Gruppe aufgefasst, wenn sie in ihren Fall-Identifikationen übereinstimmen. Um eine solche Gruppenverarbeitung zu ermöglichen, müssen die Eingabedaten eine entsprechende Fall-Identifikation besitzen. Dazu muss bei externen Dateien in INPUT-EXT eine solche Angabe ID ... ID4 gemacht werden. Bei internen Dateien muss diese Identifikation bei der Erstellung der Dateien vorgelegen haben.

Durch die Angabe CID, CID1 ...CID4 wird festgelegt, mit welchen Fall-Identifikationen die Gruppenbildung durchzuführen ist. Mit CID werden diejenigen Fälle zu einer Gruppe zusammengefasst, die direkt hintereinander liegen und die gleiche Fall-Identifikation ID besitzen. Bei CID1 bilden die Fälle eine Gruppe, die sowohl in ID als auch in ID1 übereinstimmen.
CID4 schließlich verlangt, dass die statistischen Fälle in allen Fall-Identifikationen ID ... ID4 gleiche Werte besitzen.
Durch die zusätzliche Angabe .B und .E wird festgelegt, ob die entsprechende Bedingung zu Beginn oder am Ende einer Gruppe erfüllt sein soll:
CID.B ist nur für den ersten statistischen Fall jeder Gruppe erfüllt und für alle anderen Fälle nicht,
CID.E gilt entsprechend nur für den letzten Fall jeder Gruppe.

FIRST Dieser Operand besitzt für den ersten eingelesenen statistischen Fall den Wert 1 = *wahr*, für alle anderen Fälle den Wert *falsch* . Liegt ein SELECT-Statement vor, so muss der erste eingelesene Fall nicht der erste verarbeitete Fall sein.

LAST Dieser Operand besitzt für den letzten eingelesenen statistischen Fall den Wert 1 = *wahr*, für alle anderen Fälle den Wert 0 = *falsch* . Liegt ein SELECT-Statement vor, so muss der letzte eingelesene Fall nicht der letzte verarbeitete Fall sein.

Diese logischen Operanden lassen sich mit Hilfe der folgenden Operatoren zu komplexen Ausdrücken zusammenfügen:

&	logisches und	
|	logisches oder	(stattdessen kann auch das Zeichen **!** verwendet werden)
¬	logisches nicht	(stattdessen kann auch das Zeichen **^** verwendet werden)

Zum Beispiel besitzt der logische Ausdruck

C10.1 | C10.2

den Wert wahr , wenn das Merkmal C10.1 oder das Merkmal C10.2 wahr ist oder wenn beide gleichzeitig erfüllt sind.

Dagegen ist der logische Ausdruck

C10.1 & C10.2

nur dann wahr , wenn die beiden Merkmale C10.1 und C10.2 gleichzeitig erfüllt sind.

Der Ausdruck

¬ C10.Z0

ist genau dann wahr , wenn C10.Z0 falsch ist, also mindestens eines der Merkmale der Variablen C10 erfüllt ist. Dieser logische Ausdruck ist gleichwertig mit

C10.Z1 | C10.Z2 | C10.Z3

oder auch mit

C10.Z123.

Die folgenden Vergleichsoperatoren können in logischen Ausdrücken verwendet werden:

< kleiner
> größer
= gleich
<= kleiner oder gleich
>= größer oder gleich
¬= ungleich (stattdessen kann auch **^=** geschrieben werden)
¬< nicht kleiner (stattdessen kann auch **^<** geschrieben werden)
¬> nicht größer (stattdessen kann auch **^>** geschrieben werden)

Mit den Vergleichsoperatoren > < = ... lassen sich je zwei der folgenden numerischen Operanden miteinander vergleichen. Ist eine solche Vergleichsrelation erfüllt, liefert sie das Ergebnis wahr oder 1, ist sie nicht erfüllt, den Wert falsch oder 0:

Nn Wert der numerischen Variablen Nn.
So ist zum Beispiel die Vergleichsrelation N7>N8 für die statistischen Fälle erfüllt, bei denen der Wert der Variablen N7 größer als der von N8 ist.

k Numerische Konstante.
So ist die Vergleichsrelation N100<=18.5 für die statistischen Fälle erfüllt, bei denen der Wert der Variablen N100 kleiner oder gleich der numerischen Konstanten 18.5 ist. Eine Konstante kann maximal 19 Ziffern besitzen. Als Dezimalzeichen ist der Punkt zu verwenden, das Komma wird dafür nicht akzeptiert. Die Schreibweisen .9 und 9. sind nicht zulässig.

Z0 Numerische Konstante: Keine Angabe.
Dieser künstliche numerische Wert ist kleiner als jede Zahl, also auch kleiner als die kleinste negative Zahl. So ist N11>Z0 immer dann erfüllt, wenn die Variable N11 überhaupt eine Zahl enthält. Jede numerische Variable besitzt vor der ersten Wertezuweisung den Wert Z0.

IDi Numerische Fall-Identifikationen ID...ID4.
Diese Operanden sind hier nur zulässig, wenn die entsprechende Fall-Identifikation auch numerische Werte besitzt, nicht jedoch bei Textwerten. So ist ID1>=112 für alle statistischen Fälle erfüllt, deren Fall-Identifikation ID1 größer oder gleich 112 ist.

Cn **. S** Numerischer Wert: Summe der erfüllten Merkmale.
Dieser Operand liefert die Anzahl aller erfüllten Merkmale der Bedingungsvariablen Cn.
So steht C10.S für die Summe aller Merkmale der Variablen C10, die den Wert *wahr* besitzen.

Diese Summe lässt sich auch auf ausgewählte Merkmale einer Variablen beschränken. So zählt

```
C10.S(8,13..15,21)
```

von den Merkmalen 8, 13, 14, 15, 21 der Variablen C10 diejenigen, die den Wert 1 besitzen. Die übrigen Merkmale von C10 werden nicht berücksichtigt. Auch negative Merkmalsnummern können für a angegeben werden: -1 ist das letzte Merkmal der Variablen, -2 das vorletzte usw.

ERR Fehlernummer 500 ... der zuletzt ausgeführten Wertezuweisung.
Der Operand ERR wird zu Beginn jeder Wertezuweisung auf 0 gesetzt. Läuft die Wertezuweisung fehlerfrei ab, so behält ERR den Wert 0. Wird hingegen ein Fehler gefunden, so enthält ERR anschließend die Nummer 500 ... 599 der entsprechenden Fehlermeldung. Mit den Statements

```
-N1=D1(10,3)
-IF ERR > 0
-N1=0
-EIF
```

erhält die Variable N1 den Wert 0, wenn das Feld D1(10,3) einen fehlerhaften Wert besitzt. Ohne das IF-Statement hätte N1 im Fehlerfall den Wert Z0.

numerischer Ausdruck
Als Operanden für die Vergleichsoperatoren lassen sich auch ganze Ausdrücke verwenden, so wie sie im Abschnitt *Numerische Ausdrücke* beschrieben sind. Jeder numerische Ausdruck liefert wieder *einen* numerischen Wert. So ist zum Beispiel die Vergleichsrelation N7+N8>500 für die statistischen Fälle erfüllt, bei denen die Summe von N7 und N8 größer als 500 ist.

logischer Ausdruck
In diesen Vergleichen können auch alle oben aufgeführten logischen Operanden erscheinen. Sie werden dann als numerische Werte verstanden:
Der logische Wert *wahr* hat dabei den numerischen Wert 1, und der logische Wert *falsch* besitzt den numerischen Wert 0. So ist zum Beispiel N20 > C10.3 erfüllt, wenn C10.3 *wahr* und der Wert von N20 größer als 1 ist. Er ist ebenfalls erfüllt, wenn C10.3 *falsch* und N20 größer als 0 ist.

Mit den Vergleichsoperatoren > < = ... lassen sich auch je zwei der folgenden Textoperanden miteinander vergleichen. Ist eine solche Vergleichsrelation erfüllt, liefert sie das Ergebnis wahr oder 1, ist sie nicht erfüllt, den Wert falsch oder 0:

Tn Durch diese Angabe wird der Text aus einer Textvariablen mit anderen Textwerten verglichen. So ist zum Beispiel die Vergleichsrelation T10='HANNOVER' erfüllt, wenn die Variable T10 den Text *HANNOVER* enthält. Sind die zu vergleichende Texte unterschiedlich lang, wird nur in der Länge des kürzeren Textes verglichen. So ist die Vergleichsrelation T10='H' immer dann erfüllt, wenn der Text in der Variablen T10 mit dem Buchstaben *H* beginnt.
Es ist auch möglich, nur Teile des Textes aus der Variablen Tn zu untersuchen. So ist die Vergleichsrelation T10(3..5)='NNO' immer dann erfüllt, wenn sie einen Text enthält, dessen Buchstaben drei bis fünf gleich NNO sind.

' text **'** Textkonstante: Eine in Hochkommas eingeschlossene Textzeile.
Zum Beispiel ist die Bedingung T17 > 'Müller' für die statistischen Fälle erfüllt, in denen die Variable T17 einen Text enthält, der in der alphabetischen Sortierfolge hinter dem Namen *Müller* liegt.

Mit der im Abschnitt *Textzeilen* beschriebenen Funktion UNDO lassen sich auch Variablentexte oder Texte aus Textelementen vergleichen. Besitzt das Textelement (10) zum Beispiel den Text *Müller*, so ist die Bedingung T10 > 'Müller' gleichwertig mit T10 > UNDO(10).

IDi Eine Fall-Identifikation vom Text-Typ kann mit anderen Texten verglichen werden.
Wenn zum Beispiel die Fall-Identifikation ID1 jeweils den Namen eines Befragten oder einer Testperson enthält, so ist die Vergleichsrelation ID1 = 'Müller' für alle Befragten mit dem Namen *Müller* erfüllt.

Dabei ist die Sortierfolge der einzelnen Zeichen abhängig vom Zeichensatz der verwendeten Maschine.
Die Tabelle Zeichenzuordnung ANSI, ASCII, EBCDIC zeigt die Sortierfolge der Zeichen in den verschiedenen Codes.

Logische Ausdrücke können bis zu 255 Operatoren und Operanden besitzen. Sie können zur Steuerung der Reihenfolge der Verarbeitung praktisch unbegrenzt viele Klammern (und) enthalten.
Die Operatoren eines Ausdrucks werden nach dieser Prioritätenfolge abgearbeitet:

hohe Priorität	( )	Klammern um Teilausdrücke und Funktionsargumente
	+ –	Vorzeichen Plus und Minus
	**	Potenzierung
	* /	Multiplikation und Division
	+ –	Addition und Subtraktion
	< > ...	Vergleichsoperatoren in logischen Teilausdrücken
	¬ ^	Negation in logischen Teilausdrücken
	&	*und* in logischen Teilausdrücken
	| !	*oder* in logischen Teilausdrücken
niedrige Priorität	=	Wertezuweisung in Variable oder externes Datenfeld

Stehen mehrere Operatoren gleicher Priorität ohne Klammern direkt hintereinander, wird von links nach rechts ausgewertet.
So wird zum Beispiel der logische Ausdruck C10.1 & N1 + N2 * 100 < 50 auf folgende Weise berechnet: Zunächst wird der Wert von N2 mit 100 multipliziert. Zu dem Ergebnis wird der Wert aus N1 addiert. Danach wird geprüft, ob diese Summe kleiner als 50 ist. Das Ergebnis 0 oder 1 dieses Vergleichs wird schließlich durch das logische und & mit dem Wert des Merkmals C10.1 verknüpft.

Beispiel

```
MOD-PROC
-IF N17>10 & N18>100
-    C10.7 = (C1.1|C1.2|C1.3) & C1.Z23
-    C10.8 =  C1.1|C1.2|C1.3 & C1.Z23
-EIF
-C10.1=OLD
-C10.2=ADD|UPD
SELECT C10.S>5 & ID<1000
```

Die Folgestatements des Statements MOD-PROC werden für alle statistischen Fälle der Eingabedateien ausgeführt.

Im IF-Statement wird zunächst geprüft, ob der Wert der Variablen N17 für den laufenden Fall größer als 10 ist. Wenn gleichzeitig der Wert der Variablen N18 größer 100 ist, so ist die IF-Bedingung erfüllt. Nur in diesem Fall werden die Statements vor dem EIF ausgeführt.

Durch das erste Folgestatement nach dem IF erhält das Merkmal 7 der Variablen C10 den Wert 1, wenn wenigstens eines der Merkmale 1...3 der Variablen C1 erfüllt ist und zusätzlich zwei oder mehr beliebige Merkmale von C1 erfüllt sind.

Das zweite Folgestatement nach dem IF zeigt im Vergleich zum ersten die Wirkung von Klammern:
Das Merkmal 8 der Variablen C10 wird bereits gesetzt, wenn in C1 eines der Merkmale 1 oder 2 erfüllt ist. Nur wenn diese beiden Merkmale den Wert 0 besitzen, so wird weiter geprüft: Ist Merkmal 3 erfüllt und sind gleichzeitig zwei oder mehr beliebige Merkmale in C1 gesetzt, so wird C10.8 ebenfalls auf 1 gesetzt. Der Unterschied entsteht dadurch, dass ohne Angabe von Klammern der *und*-Operator & stärker bindet als der *oder* -Operator |.

Das Folgestatement nach dem EIF setzt das Merkmal 1 der Variablen C10 auf 1, wenn die Bedingung OLD erfüllt ist. Diese Angabe ist nur sinnvoll, wenn sowohl ein Statement INPUT-INT als auch ein Statement INPUT-EXT zu verarbeiten ist. Das Merkmal OLD besitzt dann den Wert 1, wenn zu einem statistischen Fall nur Daten aus INPUT-INT oder INPUT-SEC vorliegen, nicht jedoch aus INPUT-EXT. In allen anderen Fällen hat OLD den Wert 0.

Die Merkmale ADD und UPD besitzen eine ähnliche Funktion: UPD ist erfüllt, wenn zu einem statistischen Fall gleichzeitig Daten aus INPUT-EXT und aus INPUT-INT zu verarbeiten sind. ADD ist erfüllt, wenn nur Daten aus INPUT-EXT vorliegen und keine aus INPUT-INT. So wird das Merkmal 2 der Variablen C10 immer dann auf *wahr* gesetzt, wenn zu einem statistischen Fall irgendwelche Daten aus INPUT-EXT zu verarbeiten sind, unabhängig von der Eingabe aus INPUT-INT.

Statt ADD | UPD könnte auch ¬OLD geschrieben werden.

Das Statement SELECT schließlich wählt diejenigen statistischen Fälle zur weiteren Verarbeitung aus, in denen mehr als 5 beliebige Merkmale von C10 gleichzeitig erfüllt sind und deren Fall-Identifikation ID zusätzlich kleiner als 1000 ist.

Abkürzungen

Stehen in einem logischen Ausdruck mehrere Merkmale der gleichen Bedingungsvariablen direkt hintereinander, so muß die Variable nicht wiederholt werden. Das Gleiche gilt für den Operanden Cn.Zi:
Statt: C80.1 & (C80.2 | C80.3) | C80.Z0
genügt: C80.1 & (.2 | .3) | .Z0

Stehen in einem logischen Ausdruck mehrere Merkmale der gleichen Bedingungsvariablen direkt hintereinander, lückenlos aufsteigend und mit dem gleichen Operator verknüpft, so genügt es, das erste und letzte Merkmal aufzuführen:
Statt: C80.1 & .2 & .3 & .4 & .5
genügt: C80.1 & .. 5

Ebenso mit dem Negationszeichen:
Statt: C80.1 &¬ .2 &¬ .3 &¬ .4 &¬ .5
genügt: C80.1 &¬ ..5

Auch negative Merkmalsnummern sind möglich:
Statt: C80.1 & .2 & .3 & .4 & .5
genügt: C80.1 & .. -2
sofern die Variable C80 gerade 6 Merkmale besitzt.

Wird ein numerischer Wert nacheinander mit verschiedenen anderen Werten verglichen, so muß der erste Wert nur einmal geschrieben werden:
Statt: N212 > 2 & N212 < 8 & N212 = N99
genügt: N212 > 2 & < 8 & = N99

Dies gilt auch, wenn der erste Wert keine numerische Variable ist.
Statt: C1.S > 2 & C1.S < 8 & C1.S = N99
genügt: C1.S > 2 & < 8 & = N99

Entsprechende Regeln gelten für Textvariable und konstante Texte:
Statt: T1 = 'GB' | T1 = 'IR' | T1='US'
genügt: T1='GB' | ='IR' | ='US'

Einige Vergleiche lassen sich auch durch einfache Zähloperanden ersetzen:

C7.Z0	ist gleichwertig mit	C7.S < 1 oder C7.S = 0
C7.Z1	ist gleichwertig mit	C7.S = 1
C7.Z2	ist gleichwertig mit	C7.S = 2
C7.Z3	ist gleichwertig mit	C7.S > 2
C7.Z01	ist gleichwertig mit	C7.S < 2
C7.Z123	ist gleichwertig mit	C7.S > 0 oder C7.S ¬= 0

usw.

Numerische Ausdrücke

Numerische Ausdrücke dienen der Ermittlung numerischer Werte für jeden einzelnen statistischen Fall.
Sie werden in den Folgestatements von ADD-PROC, MOD-PROC, UPD-PROC und OUTPUT-EXT verwendet sowie im SELECT-Statement. Sie können auf der rechten Seite von Wertezuweisungen erscheinen oder Teil von logischen Ausdrücken sein, zum Beispiel in IF-Statements oder in SELECT.

Bei der Auswertung von Ausdrücken wird grundsätzlich mit doppelt genauen Gleitkommazahlen gearbeitet. Das erlaubt eine Genauigkeit von etwa 16 Dezimalstellen. Erst zum Abschluss der Auswertung wird das Ergebnis eines Ausdrucks gegebenenfalls abgeschnitten und dabei gerundet.

Besitzt einer der Operanden eines Ausdrucks auf Fallebene den Wert Z0 (keine Angabe), so ist das Ergebnis des Ausdrucks stets auch Z0, egal welche Werte die übrigen Operanden besitzen. Bei Wertezuweisungen lässt sich diese Regel durch die Angabe von BZ verändern.

Rechenausdrücke sind auch in XTAB- oder XADD-Statements zulässig, unterscheiden sich aber wesentlich von den in diesem Abschnitt beschriebenen numerischen Ausdrücken. Während hier mit den Daten der einzelnen statistischen Fälle gerechnet wird, werden in XTAB und XADD zunächst die Werte der Operanden für alle statistischen Fälle aufaddiert und erst danach der Ausdruck mit den so entstandenen Summen ausgewertet.

So wird hinter ADD-PROC der numerische Ausdruck

-N100=N10*C20.1

für jeden statistischen Fall ausgewertet, indem der Wert der Variablen N10 und des Merkmals C20.1 miteinander multipliziert und das Ergebnis der Variablen N100 zugewiesen wird.

In XTAB dagegen wird für den Ausdruck

ROW=N10*C20.1

zunächst die Summe der Werte der Variablen N10 für alle statistischen Fälle gebildet. Getrennt davon wird die Summe der erfüllten Merkmale C20.1 für alle statistischen Fälle gebildet. Erst am Ende der Auswertung, beim Ausdruck der Tabelle, werden diese beiden Summen miteinander multipliziert und das Ergebnis in die Tabelle gestellt. Einzelheiten dazu finden sich im Abschnitt *Numerische Ausdrücke* bei XTAB unter COL/ROW/FILTER.

Ein numerischer Ausdruck besteht im einfachsten Fall aus nur einem der folgenden Operanden:

k Numerische Konstante.
Sie kann aus maximal 9 Ziffern bestehen.
Als Dezimalzeichen ist der Punkt zu verwenden, das Komma wird nicht akzeptiert.
Die Schreibweisen .9 und 9. sind unzulässig, 0.9 und 9.0 sind korrekt.

Z0 Numerische Konstante: Keine Angabe.
Dieser künstliche numerische Wert ist kleiner als jeder andere numerische Wert, also auch kleiner als die kleinste negative Zahl.
Jede numerische Variable besitzt vor der ersten Wertezuweisung den Wert Z0.

IDi Numerische Fall-Identifikationen ID...ID4.
Diese Operanden sind hier nur zulässig, wenn die entsprechende Fall-Identifikation auch numerische Werte besitzt, nicht jedoch bei Textwerten.

Nn Numerische Variable N0 ... N99999.

Nn **(** a ... b **)** Klasseneinteilung zu einem numerischen Wert.
Dieser Operand erzeugt aus dem numerischen Wert der Variablen Nn Klassenmerkmale 1 ....

Enthält zum Beispiel die Variable N10 Altersangaben als Zahlen 0 ... 99, so setzt das Statement

```
-C10 = N10(0 30 60 100)
```

folgende drei Altersklassen in der Variablen C10:

C10.1: 0 bis 29 Jahre
C10.2: 30 bis 59 Jahre
C10.3: 60 bis 99 Jahre

Enthält die Variable N10 den Wert Z0 für keine Angabe, so liefert N10(0 30 60 100) das Ergebnis Z0: Keines der Merkmale in C10 wird gesetzt.
Das Gleiche geschieht, wenn N10 einen Wert kleiner als 0 oder größer gleich 100 besitzt.

Mit diesem Operanden lässt sich auch rechnen. Zum Beispiel setzt

```
-C10 = 2 + N10(0 30 60 100)
```

die Merkmale 3 bis 5 der Variablen C10.

Schließlich lassen sich mit damit auch numerische Variable füllen. Zum Beispiel stellt

```
-N21 = 2 + N10(0 30 60 100)
```

die Werte 3 bis 5 oder Z0 in die Variable N21.

Cn **. S** Anzahl der erfüllten Merkmale der Variablen Cn.
Dieser Operand liefert die Anzahl aller erfüllten Merkmale der Bedingungsvariablen Cn.
So steht C10.S für die Summe aller Merkmale der Variablen C10, die den Wert 1 besitzen.

Diese Summe lässt sich auch auf ausgewählte Merkmale einer Variablen beschränken. So zählt

```
C10.S(8,13..15,21)
```

von den Merkmalen 8, 13, 14, 15, 21 der Variablen C10 diejenigen, die den Wert 1 besitzen. Die übrigen Merkmale von C10 werden nicht berücksichtigt. Auch negative Merkmalsnummern können für a angegeben werden: –1 ist das letzte Merkmal der Variablen, –2 das vorletzte usw.

ERR Fehlernummer 500 ... der zuletzt ausgeführten Wertezuweisung.
Der Operand ERR wird zu Beginn jeder Wertezuweisung auf 0 gesetzt. Läuft die Wertezuweisung fehlerfrei ab, so behält ERR den Wert 0. Wird hingegen ein Fehler gefunden, so enthält ERR anschließend die Nummer 500 ... 599 der entsprechenden Fehlermeldung. Mit den Statements

```
-N1=D1(10,3)
-N2=ERR
```

erhält die Variable N2 den Wert 0, wenn das Feld D1(10,3) einen korrekten Wert besitzt, andernfalls die Nummer 500 ... 599 der entsprechenden Fehlermeldung.

Die oben aufgeführten Operanden lassen sich mit Hilfe der folgenden Operatoren zu komplexen numerischen Ausdrücken zusammenfügen:

		Beispiel
+ –	positives und negatives Vorzeichen	-25 oder -N132
**	Potenzierung	N10 ** 5.8
*	Multiplikation	N10 * N20
/	Division	N10 / 10
+	Addition	N10 + N20
–	Subtraktion	N10 - 15.3

Eine Division durch 0 führt zum Ergebnis 0.

In den numerischen Ausdrücken können auch logische Operanden auftreten.
Sie werden dann als numerische Werte verstanden:

Der logische Wert *wahr* hat dabei den numerischen Wert 1, und
der logische Wert *falsch* besitzt den numerischen Wert 0.

So ist zum Beispiel C10.3 + C10.10 + C10.23 gleichwertig mit C10.S(3,10,23).
Der Ausdruck 50*C10.1 + 70*C10.2 liefert das Ergebnis:
- 50, wenn C10.1 *wahr* und C10.2 *falsch* ist,
- 70, wenn C10.1 *falsch* und C10.2 *wahr* ist,
- 120, wenn C10.1 und C10.2 beide *wahr* sind, und
- 0, wenn C10.1 und C10.2 beide *falsch* sind.

Folgende logische Operanden können auf diese Weise als numerische 0-1-Ausdrücke verwendet werden:

Cn . a Dieser Operand besitzt den Wert 1, wenn das Merkmal a = 1 ... 9999 der Bedingungsvariablen Cn erfüllt ist. Andernfalls hat Cn.a den Wert 0. Zum Beispiel liefert C10.7 den Wert 1, wenn das Merkmal 7 der Variablen C10 erfüllt ist.
Auch negative Merkmalsnummern können für a angegeben werden: C10.-1 ist das letzte Merkmal der Variablen, C10.-2 das vorletzte usw.

Cn **. Z**i Dieser Operand erhält den Wert 1, wenn die Variable Cn die Anzahlbedingung Zi erfüllt.
Sonst besitzt er den Wert 0. Es bedeutet:
- Z0 kein Merkmal ist erfüllt, alle haben den Wert 0,
- Z1 genau eines der Merkmale besitzt den Wert 1,
- Z2 genau zwei der Merkmale besitzen den Wert 1,
- Z3 drei *oder mehr* Merkmale besitzen den Wert 1.

Es sind auch kombinierte Anzahlbedingungen möglich:
- Z01 keines oder genau ein Merkmal besitzt den Wert 1,
- Z123 mindestens ein Merkmal besitzt den Wert 1, usw.

So verlangt C10.Z12 zum Beispiel, dass genau eines oder zwei der Merkmale der Variablen C10 erfüllt sind, damit dieser Operand den Wert 1 erhält - in allen anderen Fällen hat er den Wert 0.
Anzahlbedingungen lassen sich auch auf ausgewählte Merkmale einer Variablen beschränken.

So ist die Bedingung
C10.Z1(8,13..17,21)
nur dann erfüllt, wenn genau eines der Merkmale 8, 13, 15, 16, 17 oder 21 der Variablen C10 den Wert 1 besitzt und die anderen Merkmale den Wert 0. Auch negative Merkmalsnummern können für a angegeben werden: -1 ist das letzte Merkmal der Variablen, -2 das vorletzte usw.

CIDi **. B**
CIDi **. E**
Diese Operanden besitzen den Wert 1, wenn ein Gruppenwechsel der Fall-Identifikation i eintritt. Findet kein Gruppenwechsel statt, haben sie den Wert 0.
Mehrere direkt hintereinander liegende statistische Fälle werden als Gruppe aufgefasst, wenn sie in ihren Fall-Identifikationen übereinstimmen.
Um eine solche Gruppenverarbeitung zu ermöglichen, müssen die Eingabedaten eine entsprechende Fall-Identifikation besitzen. Dazu muss bei externen Dateien in INPUT-EXT eine solche Angabe ID...ID4 gemacht werden. Bei internen Dateien muss diese Identifikation bei der Erstellung der Dateien vorgelegen haben.
Durch die Angabe CID, CID1 ...CID4 wird festgelegt, mit welchen Fall-Identifikationen die Gruppenbildung durchzuführen ist. Mit CID werden diejenigen Fälle zu einer Gruppe zusammengefasst, die direkt hintereinander liegen und die gleiche Fall-Identifikation ID besitzen. Bei CID1 bilden die Fälle eine Gruppe, die sowohl in ID als auch in ID1 übereinstimmen. CID4 schließlich verlangt, dass die statistischen Fälle in allen Fall-Identifikationen ID...ID4 gleiche Werte besitzen.
Durch die zusätzliche Angabe .B und .E wird festgelegt, ob die entsprechende Bedingung zu Beginn oder am Ende einer Gruppe erfüllt sein soll:
CID.B ist nur für den ersten statistischen Fall jeder Gruppe erfüllt und für alle anderen Fälle nicht, CID.E entsprechend nur für den letzten Fall jeder Gruppe.

ADD Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle mit Daten aus INPUT-EXT und ohne Daten aus INPUT-INT oder INPUT-SEC. In allen anderen Fällen besitzt ADD den Wert 0 = *falsch*.

OLD Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle mit Daten aus INPUT-INT oder INPUT-SEC und ohne Daten aus INPUT-EXT. In allen anderen Fällen besitzt OLD den Wert 0 = *falsch*.

UPD Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle, zu denen gleichzeitig Daten aus INPUT-EXT und Daten aus INPUT-INT oder INPUT-SEC vorliegen.
In allen anderen Fällen besitzt UPD den Wert 0 = *falsch*.

UPDSEC Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle aus INPUT-EXT oder INPUT-INT, zu denen ein Fall mit passender Identifikation aus einer INPUT-SEC-Datei eingelesen wurde.
In allen anderen Fällen besitzt UPDSEC den Wert 0 = *falsch*.

ADDSEC Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle aus INPUT-SEC, zu denen kein Fall aus INPUT-EXT oder INPUT-INT mit passender Identifikation vorhanden ist. Die Bedingung ADDSEC ist auch für alle statistischen Fälle aus INPUT-SEC mit der Angabe ADD oder ADDP erfüllt. In allen anderen Fällen besitzt ADDSEC den Wert 0 = *falsch*.

FIRST Dieser Operand besitzt für den ersten eingelesenen statistischen Fall den Wert 1 = *wahr*, für alle anderen Fälle den Wert *falsch* . Liegt ein SELECT-Statement vor, so muss der erste eingelesene Fall nicht der erste verarbeitete Fall sein.

LAST Dieser Operand besitzt für den letzten eingelesenen statistischen Fall den Wert 1 = *wahr*, für alle anderen Fälle den Wert 0 = *falsch* . Liegt ein SELECT-Statement vor, so muss der letzte eingelesene Fall nicht der letzte verarbeitete Fall sein.

logischer Ausdruck

Als Teile von numerischen Ausdrücken lassen sich auch komplexe logische Ausdrücke verwenden, die aus den obigen Operanden zusammengesetzt sind. Siehe Abschnitt *Logische Ausdrücke*.
Sie besitzen den numerischen Wert 1 = *wahr*, wenn sie erfüllt sind, und sonst den numerischen Wert 0.

In den numerischen Ausdrücken lassen sich auch die folgenden Funktionen verwenden Als Argumente sind neben Variablen und Konstanten auch Ausdrücke möglich. Zum Beispiel: ABS((N10+N20)*3.1)

ABS Absolutbetrag.
Zum Beispiel liefert ABS(N10) den absoluten Betrag des Wertes der Variablen N10.

CL Ceiling = kleinste ganze Zahl, die größer oder gleich dem Argument ist.
Zum Beispiel liefern CL(1.1) und CL(2) beide den Wert 2.

COS Cosinus. Die Argumente müssen im Bogenmaß angegeben werden.
Zum Beispiel liefert COS(3.1416) den Wert -1.
Zur Umrechnung: Bogenmaß = Grad * 2 π / 360.

EXP Exponentialfunktion.
Zum Beispiel liefert EXP(N10) die gleichen Werte wie (2.7182...)**N10.

FL Floor = größte ganze Zahl, die kleiner oder gleich dem Argument ist.
Zum Beispiel liefern FL(1.8) und FL(1) beide den Wert 1.

LN Logarithmus zur Basis e.
Zum Beispiel liefert LN(1) den Wert 0. Zu nicht positiven Argumenten ergibt LN den Wert 0.

SIN Sinus. Die Argumente müssen im Bogenmaß angegeben werden.
Zum Beispiel liefert SIN(3.1416) den Wert 0. Zur Umrechnung: Bogenmaß = Grad * 2 π / 360.

SQRT Quadratwurzel.
Zum Beispiel liefert SQRT(N10) die Quadratwurzel der Werte der Variablen N10.
Zu negativen Argumenten ergibt SQRT den Wert 0.

WEEK Wochennummer eines Datums nach der ISO-Norm 8601.
Enthält die Variable N2 das Datum 30.6.2016 in der Form 20160630, so stellt das Statement

```
-N1=WEEK(N2)
```

die Wochennummer 201626 in die Variable N1. Die Jahresangabe in dem Ergebnis ist erforderlich, da die ersten Tage eines Jahres in die letzte Woche des Vorjahres fallen können und die letzten Tage eines Jahres in die erste Woche des Folgejahres.
Das folgende Statement löst die Wochennummer 26 des obigen Beispiel aus der Variablen N1 heraus und stellt sie in die Variable N3:

```
-N3=N1-FL(N1/100)*100
```

Ein ungültiges Datum beantwortet die Funktion mit dem Ergebnis Z0 (keine Angabe) und einer Fehlermeldung im Zählprotokoll. Der Eingabewert Z0 liefert ebenfalls das Ergebnis Z0, jedoch ohne Fehlermeldung.

DAYW Wochentag 1 ... 7 eines Datums für Montag ... Sonntag.
Enthält die Variable N2 das Datum 30.6.2016 in der Form 20160630, so stellt das Statement
Das Statement

```
-C1=DAYW(N2)
```

setzt die Merkmale 1 ... 7 der Variablen C1 abhängig vom Datum in der Variablen N2. Enthält N2 das Datum 30.6.2016 in der Form 20160630, so wird das Merkmal 4 der Variablen C1 für *Donnerstag* gesetzt.
Ein ungültiges Datum beantwortet die Funktion mit dem Ergebnis Z0 (keine Angabe) und einer Fehlermeldung im Zählprotokoll. Der Eingabewert Z0 liefert ebenfalls das Ergebnis Z0, jedoch ohne Fehlermeldung.

DAYY Tag 1 ... 366 eines Datums im Jahr.
Enthält die Variable N2 das Datum 30.6.2016 in der Form 20160630, so stellt das Statement

```
-N1=DAYY(N2)
```

den Tag 182 des Jahres in die Variable N1.
Ein ungültiges Datum beantwortet die Funktion mit dem Ergebnis Z0 (keine Angabe) und einer Fehlermeldung im Zählprotokoll. Der Eingabewert Z0 liefert ebenfalls das Ergebnis Z0, jedoch ohne Fehlermeldung.

RCH (Nn_1 : m_1 Nn_2 : m_2 ... < ; k_1 : w_1 k_2 : w_2 ... >)

Diese Funktion ermittelt die Nettoreichweite eines Medienstreuplanes. Hierbei müssen die Variablen *Nn* Kontaktwahrscheinlichkeiten zwischen 0.0 und 1.0 für die Medien des Streuplans enthalten. Der Wert *m* hinter jeder Variablen und dem Doppelpunkt muss die Belegungshäufigkeit oder Frequenz 1, 2 ... des entsprechenden Mediums sein. Es ist eine praktisch unbegrenzte Anzahl von Argumentpaaren erlaubt, Ausdrücke sind als Argumente aber nicht möglich.
Die Funktion RCH ist sowohl in Wertezuweisungen als auch in XTAB zulässig.
Die Berechnung erfolgt durch:

$$r = \sum (1 - (1 - p_1)^{m_1} (1 - p_2)^{m_2} \ldots)$$

p_i = Kontaktwahrscheinlichkeit aus den Variablen Nn_i.
m_i = Belegungshäufigkeit des Mediums der Variablen Nn_i.

Besitzt eine der N-Variablen in RCH den Wert Z0 (keine Angabe), so wird dieser von der Funktion RCH wie 0 verarbeitet. Wahrscheinlichkeitswerte über 1 werden wie 1 behandelt.

Die Funktion RCH akzeptiert auch Wirkungskurven zur Ermittlung von wirksamen Reichweiten. dazu sind die gewünschten Kontaktklassen anzugeben, mit den Gewichten zur Bewertung des Reichweitenanteils der Klassen:

k_i = obere Grenze 1 ... 65 635 der jeweiligen Kontaktklasse. Diese muss größer sein, als die Grenze der vorausgehenden Klasse.
w_i = Kontaktgewicht 0.0 ... 1.0 der entsprechenden Kontaktklasse. Jedes Gewicht darf bis zu 8 Nachkommastellen besitzen.

Zur exakten Auswertung einer solchen Wirkungskurve ist ein hoher Rechenaufwand erforderlich (Faltung von binomialverteilten Wahrscheinlichkeiten). CNTA vermeidet diesen Aufwand mit Hilfe eines Approximationsverfahrens, das bei guten Laufzeiten sehr gute Näherungswerte liefert.
CNTA ist auch in der Lage, Nettoreichweiten pro Kontaktklasse zu ermitteln. Siehe hierzu in XTAB, COL/ROW/FILTER den Abschnitt *Kontaktklassen zur Mediaplanung*.
COL/ROW/FILTER den Abschnitt *Nettoreichweiten zur Mediaplanung*.

RCHM (Nn_1 Nn_2 ...)

Diese Funktion ermittelt die Mehrfachleser der angegebenen Titel, also diejenigen, die alle Titel gleichzeitig lesen. Sie ist sowohl in Wertezuweisungen als auch in XTAB zulässig.
Die Berechnung erfolgt durch:

$$r = p_1 \cdot p_2 \cdots$$

p_i = Kontaktwahrscheinlichkeit aus den Variablen Nn_i.

RCHX (Nn_1 **N**n_2 ... **)**

Diese Funktion ermittelt die Exklusivleser des ersten Titels, also die Leser des ersten Titels, die die übrigen Titel nicht lesen. Sie ist sowohl in Wertezuweisungen als auch in XTAB zulässig.
Die Berechnung erfolgt zum Beispiel bei drei Titeln durch:

$$r(p_1\ p_2\ p_3) = p_1(1 - p_2 - p_3 + p_2 \cdot p_3)$$

p_i = Kontaktwahrscheinlichkeit aus den Variablen Nn_i.

COMP | **COMPI** | **COMPX** **(** ' sss ' | **T**n | **ID**i , ' ttt ' | **T**n | **ID**i **)**

Diese Funktionen vergleichen zwei Texte und liefern als Ergebnis einen Ähnlichkeitswert zwischen 0 und 100. Dabei steht 100 für sehr große und 0 für sehr geringe Ähnlichkeit. Die Ähnlichkeit der Texte wird anhand der kleinstmöglichen Anzahl von Editierschritten (Zeichen einfügen, löschen oder vertauschen) ermittelt, mit denen sich die beiden Texte ineinander überführen lassen. Der Vergleich der Texte findet stets im ANSI-Code statt.

Zum Beispiel vergleicht das Statement
−N1=COMP('Hannover' T12)
in den Wertezuweisungen den Text *Hannover* mit dem Wert der Variablen T12 und stellt das Ergebnis 0 ... 100 in die Variable N1.

Die Funktionen lassen sich auch in logischen Ausdrücken verwenden:
−C1.1=COMP('Hannover' T12)>80 | COMP('Hildesheim' T12)>80

Vor dem Vergleich werden die Leerzeichen vor und hinter den Texten entfernt sowie mehrfache Leerzeichen zwischen den Worten auf ein Leerzeichen verkürzt. Tabulatorzeichen gelten dabei als Leerzeichen. Die Funktion liefern folgende Werte:

Vor dem Vergleich werden die Leerzeichen vor und hinter den Texten entfernt sowie mehrfache Leerzeichen zwischen den Worten auf ein Leerzeichen verkürzt. Tabulatorzeichen gelten dabei als Leerzeichen. Die Funktionen liefern folgende Ähnlichkeitswerte:

100	Die Texte sind bis auf die Leerzeichen genau gleich.
99	Nach Umsetzung in Kleinbuchstaben sind die Texte gleich.
98	Nach zusätzlicher Umsetzung der Umlaute *ä* ... in *a* ... und von *ß* in *s* sind die Texte gleich.
97	Nach zusätzlicher Umsetzung aller Sonderzeichen in Leerzeichen sind die Texte gleich.
96	Nach zusätzlicher Umsetzung aller doppelten Zeichen in ein Zeichen sind die Texte gleich.
0 ... 95	In den verbleibenden Fällen vergleicht das Programm die Texte nach diesen Umsetzungen. Es wird ein Ähnlichkeitswert 0 ... 95 ermittelt, abhängig von der kleinstmöglichen Anzahl von Editierschritten (Zeichen einfügen, löschen oder vertauschen, Verfahren von Damerau-Levenshtein) mit denen sich zwei Worte ineinander überführen lassen. 95 steht für große Ähnlichkeit und 0 für völlig unähnliche Texte. Bestehen die Texte aus mehreren durch Leerzeichen getrennten Worten, so werden zunächst Ähnlichkeitswerte zu den beteiligten Wortpaaren gebildet. Daraus wird die am besten passende Wortkombination ermittelt und der durchschnittliche Ähnlichkeitswert der beteiligten Worte ausgewiesen.

COMP
Diese Funktion prüft, wie stark sich die beiden Texte ähneln. Die Reihenfolge der Worte ist dabei wesentlich.

COMPI
Diese Funktion prüft, ob die Worte des ersten Textes im zweiten Text enthalten sind. Die Reihenfolge der Worte spielt keine Rolle.

COMPX
Diese Funktion prüft, wie stark sich die beiden Texte ähneln. Die Reihenfolge der Worte spielt keine Rolle.

Die folgende Tabelle zeigt die Werte, die diese Funktionen für verschiedene Texte liefern:

T1	COMP		COMPI	COMPX
	(T1 'Anna Müller')	(T1 'Anna Maria Müller')	(T1 'Anna Maria Müller')	(T1 'Anna Müller')
Anna Müller	100	67	100	100
anna müller	99	67	99	99
anna muller	98	67	98	98
anna-müller	97	67	97	97
ana muler	96	67	96	96
Müller Anna	51	34	100	100
Anna Maler	77	51	77	77

Jeder numerische Ausdruck kann bis zu 255 Operatoren und Operanden besitzen.
Er kann praktisch unbegrenzt viele Klammern (und) enthalten.
Die Operatoren eines Ausdrucks werden nach dieser Prioritätenfolge abgearbeitet:

hohe Priorität	( )	Klammern um Teilausdrücke und Funktionsargumente
	+ –	Vorzeichen Plus und Minus
	**	Potenzierung
	* /	Multiplikation und Division
	+ –	Addition und Subtraktion
	< > ...	Vergleichsoperatoren in logischen Teilausdrücken
	¬ ^	Negation in logischen Teilausdrücken
	&	*und* in logischen Teilausdrücken
	| !	*oder* in logischen Teilausdrücken
niedrige Priorität	=	Wertezuweisung in Variable oder externes Datenfeld

Stehen mehrere Operatoren gleicher Priorität ohne Klammern hintereinander, wird von links nach rechts ausgewertet.

So wird der Ausdruck

```
N10+N20/10+ABS(N30-10)*ID
```

auf folgende Weise aufgelöst:

Zunächst wird der Quotient N20/10 ermittelt und zwischengespeichert. Danach wird, der Klammern wegen, die Differenz N30-10 gebildet und der absolute Betrag daraus ermittelt. Dieser wird mit dem Wert aus ID multipliziert und das Ergebnis zwischengespeichert. Schließlich werden die gespeicherten Summanden zum Wert von N10 addiert.

Beispiel

```
ADD-PROC
-N20=(N11+N12)/1.5
-N21=N11+N12/1.5
-N22=ID+C10.S + CID.B
-N23=FL((N101-N100)/60)
```

Die Folgestatements von ADD-PROC werden für alle statistischen Fälle ausgeführt, zu deren Fall-Identifikation Daten aus INPUT-EXT vorliegen, nicht jedoch gleichzeitig aus INPUT-INT.

Durch das erste Folgestatement werden für jeden statistischen Fall die Werte der Variablen N11 und N12 addiert und diese Summe anschließend durch 1.5 dividiert. Dieses Ergebnis wird dann in die Variable N20 gestellt. Besitzt die Variable N20 weniger Kommastellen als das Ergebnis, so werden die überschüssigen Kommastellen mit Rundung abgeschnitten. Besitzt eine der Variablen N11 oder N12 den Wert Z0 (keine Angabe), so ist das Ergebnis des numerischen Ausdrucks ebenfalls Z0.

Das zweite Folgestatement zeigt im Vergleich zum ersten die Wirkung von Klammern:
Die Division / besitzt höhere Priorität als die Addition +. Daher wird zunächst der Wert der Variablen N12 durch 1.5 dividiert. Das Ergebnis wird danach zum Wert aus N11 addiert und diese Summe in die Variable N21 gestellt.

Im dritten Folgestatement wird der numerische Wert der laufenden Fall-Identifikation ID ermittelt. Danach werden die erfüllten Merkmale der Bedingungsvariablen C10 gezählt. Schließlich wird der Gruppenwechsel der Fall-Identifikation ID geprüft. Hat sich der Wert von ID im laufenden Fall gegenüber dem vorigen Fall geändert, so erhält CID.B den Wert 1, sonst den Wert 0. Diese drei Werte werden addiert und die Summe in N22 gestellt.

Im letzten Folgestatement wird für jeden statistischen Fall der Wert der Variablen N100 von dem der Variablen N101 abgezogen und die Differenz durch 60 geteilt. Die Funktion FL sorgt dann dafür, dass das Ergebnis abgerundet wird, auf die größte ganze Zahl, die kleiner oder gleich dem Ergebnis ist.

Abkürzungen

Stehen in einem numerischen Ausdruck mehrere Merkmale der gleichen Bedingungsvariablen direkt hintereinander, so muß die Variable nicht wiederholt werden. Das Gleiche gilt für die Operanden Cn.Zi und Cn.S:

Statt: C80.1 + C80.2 * C80.Z3 + C80.S
Genügt: C80.1 + .2 * .Z3 + .S

Stehen in einem numerischen Ausdruck mehrere Merkmale der gleichen Bedingungsvariablen direkt hintereinander, lückenlos aufsteigend und mit dem gleichen Operator verknüpft, so genügt es, das erste und letzte Merkmal aufzuführen:

Statt: C80.1 + C80.2 + C80.3 + C80.4 + C80.5
Genügt: C80.1 + ..5

Sind in einem numerischen Ausdruck mehrere direkt hintereinander liegende numerische Variable mit dem gleichen Operator verknüpft, so muß nur die erste und die letzte Variable aufgeführt werden:

Statt: N12 + N13 + N16 + N17
Genügt: N12 + ..17

In diesem Beispiel wird unterstellt, dass die Variablen N14 und N15 weder unter DEFS noch in INPUT-INT definiert sind.

Wird in einem Ausdruck mehrmals hintereinander auf die gleiche Variable mit Index zugegriffen, kann die Variablenbezeichnung weggelassen werden:

Statt: N10[1] + N10[2] + N10[3] + N10[4] + N10[5]
genügt: N10[1] + ...[5]
oder: N10[1] + ...5

Statt: N10[1] + N10[3] + N10[5] + N10[7]
genügt: N10[1] + [3] + [5] + [7]

Externe Dateien

Externe Dateien bestehen aus Datensätzen, die alle gleich lang sind oder bei unterschiedlicher Länge durch Zeilenende-Zeichen abgeschlossen werden. Dabei ist eine Vielzahl verschiedener Formate (ASCII, ANSI, Column Binary, QUANTUM, TRIPLE-S usw.) möglich.

Die Datenwerte können innerhalb der Datensätze feste Positionen und Längen besitzen oder im CSV-Format vorliegen, bei dem die einzelnen Werte durch Separatorzeichen wie Semikolon oder Komma voneinander getrennt sind.

Externe Dateien werden über die Statements INPUT-EXT und FETCH-FILE eingelesen. Danach können sie bereinigt, ausgewertet, umgeformt oder als neue externe Datei durch OUTPUT-EXT wieder ausgegeben werden.

Das Programm kann externe Dateien der verschiedenen Formate untereinander umwandeln, zum Beispiel von COLBIN nach ASCII, aber auch die Größe und der Inhalt der einzelnen Datensätze.

Statistische Fälle

Ein statistischer Fall besteht aus einem oder mehreren Datensätzen. Die Sätze eines Falles müssen direkt hintereinander liegen, nicht unbedingt vollzählig oder in fester Reihenfolge.

Bestehen die statistischen Fälle aus mehreren Datensätzen, muss erkennbar sein, welche Datensätze einen statistischen Fall bilden. Am einfachsten ist es, wenn alle Fälle aus der gleichen Anzahl von Datensätzen bestehen. Dann genügt es, die Anzahl der Sätze pro Fall anzugeben.

Enthalten statistische Fälle einer Datei unterschiedlich viele Datensätze, so müssen sie Satz-Kennzeichen enthalten, um sie innerhalb des Falles einordnen zu können. Solche Satz-Kennzeichen sind auch erforderlich, wenn die Datensätze innerhalb der Fälle keine feste Reihenfolge besitzen.

Datenfelder

Die Datenelemente eines statistischen Falles werden als externe Datenfelder oder kurz Felder bezeichnet.
Die vollständige Beschreibung eines Datenfeldes verlangt die folgenden Angaben:
Format des Datenfeldes, wie im Abschnitt *Externe Datenfelder* beschrieben.
Nummer des Datensatzes im statistischen Fall.
Position und Länge des Feldes im Datensatz.

Der Inhalt eines Datenbytes kann auf verschiedene Weisen interpretiert und werden. Dem Programm wird durch den Feldtyp der externen Datenfelder mitgeteilt, wie der Inhalt eines Bytes zu verarbeiten ist.

Wenn zum Beispiel ein Byte einer externen Datei den Wert 9 als Textzeichen enthält, so hängt die weitere Verarbeitung vom verwendeten Feldtyp ab: Als M-Feld stellt der Wert das Merkmal 9 dar, das in eine Bedingungsvariable übernommen werden kann. Im B-Format werden daraus die Merkmalsnummern 3, 4, 5 und 8. Als D-Feld liefert das Zeichen den Zahlenwert 9 und in einem A-Feld ist es ein Textzeichen usw.

Dateiformate

Dateien der folgenden Formate lassen sich einlesen und ausgeben:

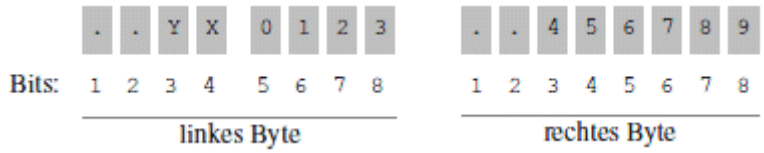
ASCII
Dies ist das bevorzugte Format aus der Zeit der DOS-PCs. Alle Feldtypen sind möglich.
Die Textverschlüsselung ist im Anhang in der Tabelle Zeichenzuordnung ANSI, ASCII, EBCDIC beschrieben.

ANSI
Dies ist das von WINDOWS bevorzugte Format. Es unterscheidet sich von ASCII lediglich durch die Verschlüsselung der Umlaute und einiger Sonderzeichen. Auch hier sind alle Feldtypen möglich, die Tabelle Zeichenzuordnung ANSI, ASCII, EBCDIC im Anhang beschreibt die Textverschlüsselung.

EBCDIC
Dieses Format stammt von Großrechnern. Alle Feldtypen sind möglich. Sämtliche Textzeichen sind anders verschlüsselt wie bei ASCII und ANSI. Die Tabelle *Zeichenzuordnung ANSI, ASCII, EBCDIC* im Anhang beschreibt die Textverschlüsselung, die sich wesentlich von ASCII und ANSI unterscheidet.

COLBIN
Dieses Format (Dualkarten-Code, Column binary) stammt aus der Zeit der Lochkarten mit 80 Spalten.
Jede Spalte belegt dabei zwei Bytes in den Datenbeständen. In diesem Format verlangt das Programm alle Längen- und Positionsangaben in Spalten. Jede Spalte kann die Werte 1 ...9, 0, X und Y enthalten oder leer sein. Insbesondere sind in jeder Spalte mehrere dieser Angaben gleichzeitig möglich.

Die 12 COLBIN-Werte einer Spalte werden auf folgende Weise den Bits zweier benachbarter Bytes gespeichert.



Ist in der Spalte ein Wert vorhanden, so erhält das entsprechende Bit den Wert 1, ohne Lochung den Wert 0. Die beiden linken Bits werden in beiden Bytes nicht benötigt. Sie sind hier durch . . gekennzeichnet und sollten stets den Wert 0 enthalten. Findet CNTA den Wert 1 in einem solchen Bit einer COLBIN-Spalte, so wird die Fehlermeldung 500 ausgegeben.

Die Tabelle Lochkombinationen der COLBIN-Spalten enthält weitere Informationen dazu.

UTF8
Dieser Zeichencode umfasst alle Zeichen des 16-Bit-UNICODE, also praktisch alle international vorkommenden Schriftzeichen. Die einzelnen Zeichen haben dabei eine Länge von 1 bis 3 Bytes.

QUANTUM
Hier handelt es sich um ein spezielles Format aus dem Markforschungsbereich. Es ist eine Mischform aus ASCII- und COLBIN-Dateien. Position und Länge der Datenfelder sind als Bytes anzugeben. Jedes Byte kann wie bei COLBIN die Werte 1 ...9, 0, X und Y enthalten oder leer sein. Mehrfachnennungen sind gleichfalls möglich: Diese werden am Ende eines jeden Datensatzes gespeichert, ohne dass der Anwender davon Kenntnis nehmen muss.

BINARY1
Dieses Format arbeitet mit ein Byte langen Dualzahlen. Es sind nur Datenfelder für die Merkmalsnummern 1 bis 8 sowie die Zahlenwerte 0 bis 255 und -127 bis +127 möglich.

BINARY2
Dieses Format arbeitet mit zwei Byte langen Dualzahlen. Es sind nur Datenfelder für die Merkmalsnummern 1 bis 8,
die Zahlenwerte 0 bis 65 535 und -32 767 bis +32 767 möglich. Position und Länge der Datenfelder sind als Feldnummern 1 ... anzugeben, wobei jedes Feld zwei Bytes lang ist.

TRIPLES
Diese Dateien enthalten sowohl die Daten der statistischen Fälle als auch die dazugehörigen Variablendefinitionen samt Texten. Dieses Format ist nur für die Ausgabe mit OUTPUT-EXT verfügbar.

CSV-Dateien
CSV steht für Comma Separated Values. Diese Dateien eignen sich unter anderem zur Ein- und Ausgabe von Daten für Tabellenkalkulationsprogramme wie EXCEL oder Datenbankprogramme wie ACCESS.
Sie sind in den Dateiformaten ASCII, ANSI, UTF8 und EBCDIC möglich, nicht jedoch im COLBIN, QUANTUM- und BINARY-Format.

In CSV-Dateien besitzen die einzelnen Datenfelder keine feste Position und Länge, sondern sind durch ein Separatorzeichen (meistens Komma, Semikolon oder Tabulatorzeichen) voneinander getrennt. Der Zugriff auf ein Feld erfolgt über die Reihenfolgenummer 1 ... innerhalb des Datensatzes. Die Länge der Felder ist variabel.

Mit der Angabe SEP= ... in den Statements INPUT-EXT, FETCH-FILE oder OUTPUT-EXT wird festgelegt, dass CSV-Dateien zu verarbeiten sind.

In CSV-Dateien besitzen die einzelnen Datenfelder keine feste Position und Länge, sondern sind durch ein bestimmtes Separatorzeichen (meistens Komma, Semikolon oder Tabulatorzeichen) voneinander getrennt. Die Länge der einzelnen Felder ist variabel. Jedes Feld ist durch die Reihenfolgenummer 1 ... innerhalb der Datensätze bestimmt. Zum Beispiel können zwei Datensätze mit dem Semikolon als Separatorzeichen so beginnen:

```
0001;35;2; ...
0002;25;1; ...
```

Folgende Datenfelder sind dabei möglich:

AS-Felder: Textwerte,
DS-Felder: Zahlenwert aus Dezimalziffern,
MS-Felder: Ein- bis vierstellige Merkmalsnummern aus den Zeichen 0 ... 9,
US-Felder: Merkmalswerte aus den Zeichen 0 und 1.

Darüber hinaus können vor den Datenwerten der CSV-Dateien Schlüsselworte mit einem Gleichheitszeichen stehen. Zum Beispiel können zwei Datensätze folgendermaßen beginnen:

```
ID=0001;  frage3=5; alter=34; ...
ID=0002;  alter=25; frage7=2; ...
```

Die Datenwerte dürfen dabei in beliebigen Spalten stehen und in einzelnen Datensätzen auch ganz fehlen.
Folgende Datenfelder sind dafür vorhanden:

AK-Felder: Textwerte mit Schlüsselwort
DK-Felder: Dezimalzahlen mit Schlüsselwort
MK-Felder: Merkmalsnummern mit Schlüsselwort
UK-Felder: Merkmalswerte aus 0 und 1 mit Schlüsselwort

Schließlich ist es möglich, die Schlüsselworte in die ersten Zeilen der CSV-Datei zu stellen, ebenfalls durch Separatorzeichen voneinander getrennt und in der gleichen Spalte wie die dazugehörigen Datenwerte. Eine solche Datei könnte folgendermaßen beginnen:

```
ID  ; frage3; alter; frage7;  ...
0001;      5;    34;      1;  ...
0002;       ;    25;      2;  ...
```

Hierfür sind ebenfalls die AK-, DK-, MK- und UK-Felder zu verwenden. Zusätzlich ist jedoch die Angabe KEYLINE in dem entsprechenden Statement INPUT-EXT, FETCH-FILE oder OUTPUT-EXT erforderlich.

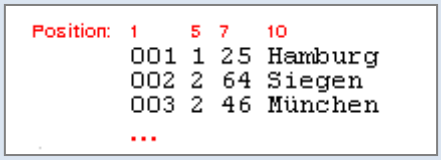
Umwandlung der Formate

Es ist sehr leicht, Dateien von einem Format in ein anderes umzuwandeln. Durch die Statements:

```
INPUT-EXT   FNAME=S140.EXT  COLBIN  SIZE=4500
OUTPUT-EXT  FNAME=S141.EXT  ASCII   COPY
```

wird zum Beispiel die externe Datei S140.EXT im COLBIN-Format eingelesen, mit einer Länge von 4 500 Spalten pro Eingabesatz. Es wird die externe Datei S141.EXT im ASCII-Format ausgegeben. Die Angabe COPY im Statement OUTPUT-EXT sorgt dafür, dass die Eingabesätze aus dem COLBIN-Format in das ASCII-Format umgewandelt und als 4 500 Byte lange Datensätze in die Ausgabedatei geschrieben werden.

Hier soll eine ASCII-Datei verarbeitet werden, die mit den folgenden drei Datensätzen beginnt:



Am Anfang jedes Datensatzes steht eine dreistellige Fall-Identifikation, zum Beispiel eine Fragebogennummer. Die Position 5 enthält die Angaben 1 und 2 für *männlich* und *weiblich*. In Position 7 ist eine Altersangabe gespeichert und in Position 10 der Wohnort.

Diese Daten können mit den folgenden Statements zu Auswertung aufbereitet werden:

```
INPUT-EXT     ASCII
              SIZE = 80,EOL
              ID = D(1,3)
DEFS
–C1:2   'Geschlecht'   1 = 'männlich'   2 = 'weiblich'
–N1:2   'Alter'
–T1:30  'Wohnort'
ADD-PROC
–C1 = M1(5)
–N1 = D1(7,2)
–T1 = A1(10,30)
```

Das Statement INPUT-EXT verlangt eine ASCII-Datei mit maximal 80 Byte langen Datensätzen. EOL besagt, dass diese Datensätze variabel lang sind und mit einem Zeilen-Ende-Zeichen abschließen. Mit ID=D(1,3) wird eine numerische Fall-Identifikation in den ersten drei Positionen der Datensätze angefordert.

Das Statement DEFS leitet die Definition der Variablen ein:

C1 ist eine Bedingungsvariable mit dem Variablentext *Geschlecht* und den beiden Merkmalen *männlich* und *weiblich.*
N1 ist eine numerische Variable mit dem Text *Alter,* die zweistellige Zahlenwerte aufnehmen kann.
T1 ist eine Textvariable mit dem Variablentext *Wohnort,* die bis maximale 30 Zeichen lange Texte aufnehmen kann.

Das Statement ADD-PROC leitet die Wertezuweisungen ein, die die Datenwerte aus den Sätzen der externen Datei in die Variablen übertragen:

-C1=M1(5) entnimmt dem ersten - und einzigen - Datensatz jedes statistischen Falles aus der Position 5 einen Merkmalswert und überträgt diesen in die Variable C1.
-N1=D1(7,2) überträgt ab Position 7 einen zweistelligen Zahlenwert in die Variable N1.
-T1=A1(10,30) stellt einen 30 Zeichen langen Text ab Position 10 der Datensätze in die Textvariable T1.

Danach lassen sich die eingelesenen Daten auf vielfältige Weise, zum Beispiel mit den Statements CODEBOOK, XTAB, OUTPUT-INT, auswerten und verarbeiten.

Externe Datenfelder

Externe Datenfelder beschreiben einzelne Informationselemente in externen Dateien.
Mit Hilfe der externen Datenfelder werden Werte aus den Dateien INPUT-EXT und FETCH-FILE entnommen und in Variable gestellt. Umgekehrt lassen sich Werte aus Variablen in externe Datenfelder der Dateien OUTPUT-EXT ausgeben.

Diese Übertragung von Informationen geschieht mit den Folgestatements von ADD-PROC, UPD-PROC, MOD-PROC und OUTPUT-EXT. Zusätzliche Angaben dazu liefern die Abschnitte Wertezuweisungen und Einlesen der statistischen Fälle.

Die Beschreibung eines externen Datenfeldes beginnt mit einem oder zwei Buchstaben. Dahinter stehen Positions- und Längenangaben. Ein externes Datenfeld muss immer ganz in einem Datensatz enthalten sein.
Es darf nicht über das Ende eines Datensatzes hinaus in den Anfang des nächsten hineinragen.

Folgende Datenfelder sind möglich:

A-Feld:	Textwert
AK-Feld:	Textwert mit Schlüsselwort in CSV- und XLSX-Dateien
AS-Feld:	Textwert in festen Spalten von CSV- und XLSX-Dateien
B-Feld:	Merkmalsnummern als Bits
D-Feld:	Numerischer Wert aus Dezimalziffern
DK-Feld:	Dezimalzahl mit Schlüssel in CSV- und XLSX-Dateien
DS-Feld:	Dezimalzahl in festen Spalten von CSV- und XLSX-Dateien
F-Feld:	Dualzahl mit Vorzeichen
G-Feld:	Dualzahl ohne Vorzeichen
I-Feld:	Merkmalsnummern als Dualzahlen
K-Feld:	Lochkartenspalten im COLBIN-Format
L-Feld:	Merkmalsnummern als Zeichen 1...9, 0, X und Y
M-Feld:	Merkmalsnummern als Dezimalzahlen
MK-Feld:	Merkmalsnummern mit Schlüssel in CSV- und XLSX-Dateien
MS-Feld:	Merkmalsnummern in festen Spalten von CSV- und XLSX-Dateien
P-Feld:	gepackte Dezimalzahl
U-Feld:	Merkmalswerte aus 0 und 1
UK-Feld:	Merkmalswerte aus 0 und 1 mit Schlüssel in CSV- und XLSX-Dateien
US-Feld:	Merkmalswerte aus 0 und 1 in festen Spalten von CSV- und XLSX-Dateien

Abhängig vom Format der Dateien INPUT-EXT, OUTPUT-EXT und FETCH-F'ILE sind folgende externe Datenfelder zulässig:

	zulässige Datenfelder																	
Dateiformat	***A***	***B***	***I***	***K***	***L***	***M***	***U***	***D***	***F***	***G***	***P***	***AK***	***DK***	***MK***	***UK***	***AS***	***DS***	***MS***	***US***
ASCII ohne SEP	x	x	x	x	x	x	x	x	x	x	x								
ASCII mit SEP												x	x	x	x	x	x	x	x
ANSI ohne SEP	x	x	x	x	x	x	x	x	x	x	x								
ANSI mit SEP												x	x	x	x	x	x	x	x
UTF8 ohne SEP	x	x	x	x	x	x	x	x	x	x	x								
UTF8 mit SEP												x	x	x	x	x	x	x	x
EBCDIC ohne SEP	x	x	x	x	x	x	x	x	x	x	x								
EBCDIC mit SEP												x	x	x	x	x	x	x	x
COLBIN	x		x	x	x	x	x	x	x	x									
QUANTUM	x			x	x	x	x	x											
BINARY1		x	x						x	x									
BINARY2		x	x						x	x									
XLSX												x	x	x	x	x	x	x	x

A-Feld: Alphanumerisch

A-Felder enthalten alphanumerische Werte an fester Position und in fester Länge in den Datensätzen.
Diese Werte bestehen aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen. Sie können in ASCII-, ANSI-, UTF8-, EBCDIC-, COLBIN- und QUANTUM-Dateien vorkommen.

A-Felder erlauben folgende Angaben:

Ad **(** a **)**
Ad **(** a **..** e **)**
Ad **(** a **,** c **)**

- a Erstes Byte 1 ... des Feldes im Datensatz.
  In COLBIN-Dateien ist dies die erste Spalte des Feldes. Anstelle einer Zahl kann auch ein Rechenausdruck mit + - * / Klammern und N-Variablen verwendet werden. Zum Beispiel
  ```
  A1((N2+8)*4-1,4).
  ```
- c Länge 1 ... 32 767 des Feldes im Datensatz in Bytes.
  Bei COLBIN-Dateien können hier 1 ... 16 383 Spalten angegeben werden.
  Fehlen die Angaben *c* und *e*, so wird als Feldlänge 1 angenommen.
- d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.
- e Letztes Byte 1 ... des Feldes im Datensatz.
  In COLBIN-Dateien ist dies die letzte Spalte des Feldes. Wie beim ersten Byte *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable.
  Fehlen die Angaben *c* und *e*, so wird als Feldlänge 1 angenommen.

Beginnt zum Beispiel eine ASCII-Datei mit folgenden Datensätzen:

```
0001 34 Schmidt      ...
0002 25 Peters       ...
```

so würde die Wertezuweisung

```
-T1=A1(9,20)
```

die darin enthaltenen Namen in die Textvariable T1 übertragen.

AK-Feld: Alphanumerisch mit Schlüssel in CSV-Dateien

AK-Felder enthalten alphanumerische Werte, die in der Datei durch Schlüssel gekennzeichnet sind.
Die Schlüssel bestehen aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen. AK-Felder sind nur in CSV-Dateien möglich. Siehe SEP in den Statements INPUT-EXT, FETCH-FILE oder OUTPUT-EXT.

Die Schlüssel können - durch ein Gleichheitszeichen getrennt - vor den Datenwerten stehen; sie dürfen dabei in beliebigen Spalten der Datensätze erscheinen oder ganz fehlen. Die Schlüssel können aber auch in den ersten Datensätzen enthalten sein, in der gleichen Spalte wie die dazugehörigen Werte; in diesem Fall muss das Statement INPUT-EXT, FETCH-FILE oder OUTPUT-EXT die Angabe KEYLINE enthalten.

AK-Felder erlauben folgende Angaben:

AKd **( )**
AKd **(** s **)**

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

s Schlüssel in den Datensätzen.
Dieser darf maximal 64 Zeichen lang sein und aus beliebigen Zeichen bestehen.
Dabei wird nicht zwischen Groß- und Kleinschrift unterschieden.
Enthält der Schlüssel Klammern (oder) so ist er in Hochkommas einzuschließen.
Zum Beispiel: `AK1('AB()')` für den Schlüssel `AB()`.

Der Schlüssel *s* kann entfallen, wenn auf der anderen Seite des Gleichheitszeichens eine Variable steht: Bei `-T1=AK1()` und `-AK1()=T1` wird *T1* zum Schlüsselwort.

Entsprechend wird in Ein- und Ausgabestatements für `ID=AK()` der Schlüssel *ID* verwendet, für `KEY=AK()` der Schlüssel *KEY* und für `RC=AK()` der Schlüssel *RC*.

Eine Datei mit KEYLINE könnte folgendermaßen beginnen:

```
  ID; c2; n1;    name; ...
0001;   ; 34; Schmidt; ...
0002;  5; 25;  Peters; ...
```

Die Wertezuweisung

```
-T1=AK1(name)
```

stellt die Namen aus diesen Datensätzen in die Textvariable T1.

Die gleiche Datei ohne KEYLINE könnte folgendermaßen beginnen:

```
ID=0001;n1=34;name=Schmidt;              ...
 ID=0002; name = Peters ;c2=5 ;n1=25; ...
```

Die Reihenfolge der Felder in den Datensätzen ist hier beliebig. Mit der Wertezuweisung

```
-T1=AK1(name)
```

werden die Namen aus diesen Datensätzen in die Textvariable T1 gestellt.

AS-Feld: Alphanumerisch in festen Spalten von CSV-Dateien

AS-Felder enthalten alphanumerische Werte. Sie bestehen aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen.
AS-Felder sind nur in CSV-Dateien möglich: Siehe SEP in den Statements INPUT-EXT, FETCH-FILE oder OUTPUT-EXT.
Die Position eines AS-Feldes im Datensatz wird durch die Anzahl davor stehender Separatorzeichen festgelegt.

Ein AS-Feld erlaubt folgende Angaben:

ASd **(** r **)**

- d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

- r Die Angabe r = 1 ... ist die Reihenfolgenummer des Wertes im Datensatz d.
  Datenwerte bis zu 4 000 Zeichen Länge werden akzeptiert. Führende Leerzeichen werden als Teil des Datenwertes verstanden. Leerzeichen hinter dem Wert werden ignoriert.
  Anstelle einer Zahl kann für die Reihenfolgenummer auch ein Rechenausdruck mit `+ - * /` Klammern und N-Variablen verwendet werden. Zum Beispiel `AS1((N2+8)*4-1)`.

Beginnt zum Beispiel eine CSV-Datei mit den folgenden Datensätzen:

```
0001;   ; 34;Schmidt; ...
0002;  5; 25;Peters ; ...
```

so überträgt die Wertezuweisung

```
-T1=AS1(4)
```

die darin enthaltenen Namen in die Textvariable T1.

B-Feld: Bits

B-Felder enthalten Merkmalswerte in Form einzelner Bits an fester Position und in fester Länge in den Datensätzen. Jedes Bit steht für ein statistisches Merkmal und kann den Wert 1 für erfüllt und 0 für nicht erfüllt annehmen. Diese Felder können in ASCII-, ANSI- , UTF8-, EBCDIC-, BINARY1- und BINARY2-Dateien vorkommen.

In BINARY2-Dateien sind Positionen in Feldern anzugeben, wobei jedes Feld zwei Bytes lang ist und 16 Merkmale aufnehmen kann. Alle anderen Dateien verlangen Positionsangaben in Bytes mit 8 Merkmalen pro Position.

Ein B-Feld erlaubt folgende Angaben:

Bd **(** a **)**	**B**d **(** a.p **)**
Bd **(** a **,** b **)**	**B**d **(** a.p **,** b **)**
Bd **(** a **...** e **)**	**B**d **(** a.p **...** e **)**
Bd **(** a **...** e.q **)**	**B**d **(** a.p **...** e.q **)**

a Erste Position 1 ... im Datensatz.
Fehlen die Angaben *b c* und *e*, so werden nur die Merkmale aus Feld *a* verwendet.
Anstelle einer Zahl ist auch ein Rechenausdruck mit + - * / Klammern und N-Variablen möglich.
Zum Beispiel B1 ((N2 + 8) * 4 - 1,4).

b Anzahl 1 ... 10 000 Bits oder Merkmale, die das Feld enthalten soll.

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

e Letzte Position 1 ... des Feldes im Datensatz.
Wie bei der ersten Position *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable.

p Erstes Bit 1 ... 8 in Position *a* bei dem das Feld beginnen soll.
Fehlt diese Angabe, so wird bei Bit 1 in Position *a* begonnen.

q Letztes Bit 1 ... 8 in Position *e* bei dem das Feld enden soll.

Jedes Byte kann 8 Bits oder Merkmale aufnehmen. Ein B-Feld darf 1 ... 10 000 Bits lang sein und bis zu 10 000 Mehrfachnennungen aufnehmen.

Zum Beispiel könnte ein Datensatz in Byte 24 folgende Werte enthalten:

Byte:	24							
Inhalt:	0	0	0	1	0	0	0	1
Bit:	1	2	3	4	5	6	7	8

Mit der Wertezuweisung

```
-C1=B1(24.3,2)
```

werden die Daten aus den Bits 3 und 4 des Datenbytes entnommen und in die Merkmale 1 und 2 der Variablen C1 gestellt. C1.1 erhält den Wert 0 und C1.2 den Wert 1.

D-Feld: Dezimalzahl

D-Felder enthalten numerische Werte in Form von Textzeichen an fester Position und in fester Länge in den Datensätzen. Sie können in ASCII-, ANSI-, UTF8-, EBCDIC-, COLBIN- und QUANTUM-Dateien vorkommen.

Bei der Eingabe aus D-Feldern sind die Zeichen 0 ... 9, Vorzeichen + und - sowie Dezimalpunkt oder Dezimalkomma zulässig. Vor und hinter einer Zahl dürfen auch Leerzeichen im Datenfeld stehen.

Bei der Ausgabe in D-Felder werden alle Werte rechtsbündig in die D-Felder gestellt. Bei negativen Zahlen wird ein Minuszeichen davor gesetzt. Ist ein numerischer Wert zu lang für ein D-Feld, so wird er nicht übertragen. Dazu erscheint eine Fehlermeldung. Im Statement OUTPUT-EXT entscheidet die Angabe DSTD, ob die D-Felder mit führenden Nullen oder führenden Leerzeichen ausgegeben und wie Dezimalzeichen aufbereitet werden.

Ein D-Feld erlaubt folgende Angaben:

Dd **(**a **..** e**)** < n >
Dd **(**a **,** c**)** < n >

a Erstes Byte 1 ... des Feldes im Datensatz.
In COLBIN-Dateien ist dies die erste Spalte des Feldes. Anstelle einer Zahl kann auch ein Rechenausdruck mit + - * / Klammern und N-Variablen verwendet werden. Zum Beispiel `D1((N2+8)*4-1,4).`

c Länge 1 ... 30 des Feldes im Datensatz in Bytes.
In COLBIN-Dateien ist die Länge in Spalten anzugeben.

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

e Letztes Byte 1 ... des Feldes im Datensatz.
In COLBIN-Dateien ist dies die letzte Spalte des Feldes. Wie beim ersten Byte *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable. Ein D-Feld darf 1 ... 30 Bytes oder Spalten lang sein.

n Anzahl 1 ... 18 Nachkommastellen im Feld.
Bei der Eingabe legt *n* fest, wie viele Ziffern des Eingabewertes als Nachkommastellen verarbeitet werden sollen, wenn der Eingabewert kein Dezimalkomma oder Dezimalpunkt enthält.
Besitzt der Eingabewert ein Dezimalzeichen, so hat dieses Vorrang vor der Angabe *n.*
Fehlt die Angabe *n,* so müssen Nachkommastellen im Eingabewert ein Dezimalzeichen besitzen.

Bei der Ausgabe ohne die Angabe *n* werden die Nachkommastellen mit Dezimalpunkt oder Dezimalkomma ausgegeben oder ganz unterdrückt, je nach Einstellung DSTD im Statement OUTPUT-EXT.

Mit der Angabe *n* werden die Ausgabewerte immer mit *n* Nachkommastellen versehen. Dabei wird der Ausgabewert gegebenenfalls gerundet oder mit Nullen aufgefüllt.

Zum Beispiel könnte eine ASCII-Datei die folgenden Daten als Textzeichen enthalten:

Inhalt:		4	2	5	0	
Byte:	10	11	12	13	14	15

Mit der Wertezuweisung

```
-N1=D1(10..15)
```

wird der Variablen N1 der Wert 4 250 zugewiesen.

Die Wertezuweisung

```
-N1=D1(10..15)2
```

dagegen versteht die beiden letzten Ziffern als Nachkommastellen und weist den Wert 42,50 zu.

Enthält die ASCII-Datei dagegen den Wert

Inhalt:	+	4	2	5	.	0
Byte:	10	11	12	13	14	15

so liest die vorige Wertezuweisung das Ergebnis 425,0 ein: Das Dezimalzeichen im Eingabewert hat Vorrang vor der Angabe 2 der Wertezuweisung.

DK-Feld: Dezimalzahl mit Schlüssel in CSV-Dateien

DK-Felder enthalten numerische Werte in Form von Textzeichen. Die Werte sind in der Datei durch Schlüssel gekennzeichnet. Die Schlüssel bestehen aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen. DK-Felder sind nur in CSV-Dateien möglich: Siehe SEP in den Statements INPUT-EXT, FETCH-FILE oder OUTPUT-EXT.

Die Schlüssel können - durch ein Gleichheitszeichen getrennt - vor den Datenwerten stehen; sie dürfen dabei in beliebigen Spalten der Datensätze erscheinen oder ganz fehlen. Die Schlüssel können aber auch in den ersten Datensätzen enthalten sein, in der gleichen Spalte wie die dazugehörigen Werte; in diesem Fall muss das Statement INPUT-EXT, FETCH-FILE oder OUTPUT-EXT die Angabe KEYLINE enthalten.

Bei der Eingabe aus DK-Feldern sind in den Werten die Ziffern 0 ... 9, Vorzeichen + und - sowie Dezimalpunkt oder Dezimalkomma zulässig. Vor und hinter einer Zahl dürfen Leerzeichen stehen.

Bei der Ausgabe in DK-Felder wird bei negativen Zahlen ein Minuszeichen vor den Wert gestellt. Im Statement OUTPUT-EXT entscheidet die Angabe DSTD darüber, wie die Dezimalzeichen aufbereitet werden.

Ein DK-Feld erlaubt folgende Angaben:

DKd **( )** < n >
DKd **(** s **)** < n >

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

s Schlüssel in den Datensätzen.
Dieser darf maximal 64 Zeichen lang sein und aus beliebigen Zeichen bestehen.
Dabei wird nicht zwischen Groß- und Kleinschrift unterschieden.

Enthält der Schlüssel Klammern (oder) so ist er in Hochkommas einzuschließen.
Zum Beispiel: DK1('AB( )') für den Schlüssel *AB( )*.

Der Schlüssel *s* kann entfallen, wenn auf der anderen Seite des Gleichheitszeichens eine Variable steht: Bei -N1=DK1( ) und -DK1( )=N1 wird *N1* zum Schlüsselwort.
Entsprechend wird in Ein- und Ausgabestatements für ID=DK( ) der Schlüssel *ID* verwendet, für KEY=DK( ) der Schlüssel *KEY* und für RC=DK( ) der Schlüssel *RC*.

n Anzahl 1 ... 18 Nachkommastellen im Feld.
Bei der Eingabe legt *n* fest, wie viele Ziffern des Eingabewertes als Nachkommastellen verarbeitet werden sollen, wenn der Eingabewert kein Dezimalkomma oder Dezimalpunkt enthält.
Besitzt der Eingabewert ein Dezimalzeichen, so hat dieses Vorrang vor der Angabe *n*.
Fehlt die Angabe *n*, so müssen Nachkommastellen im Eingabewert ein Dezimalzeichen besitzen.

Bei der Ausgabe ohne die Angabe *n* werden die Nachkommastellen mit Dezimalpunkt oder Dezimalkomma ausgegeben oder ganz unterdrückt, je nach Einstellung DSTD im Statement OUTPUT-EXT. Mit der Angabe *n* werden die Ausgabewerte immer mit *n* Nachkommastellen versehen. Dabei wird der Ausgabewert gegebenenfalls gerundet oder mit Nullen aufgefüllt.

Eine Datei mit KEYLINE könnte folgendermaßen beginnen:

```
  ID; c2; alter; ...
0001;   ;    34; ...
0002;  5;    25; ...
```

Mit der Wertezuweisung

```
-N1=DK1(alter)
```

werden die Altersangaben aus diesen Datensätzen in die numerische Variable N1 gestellt.

Die gleiche Datei ohne KEYLINE könnte folgendermaßen beginnen:

```
ID=0001; alter = 34 ; …
ID=0002;c2=5 ;alter=25; …
```

Die Reihenfolge der Felder innerhalb der Datensätze ist hier beliebig. Mit der Wertezuweisung

```
-N1=DK1(alter)
```

werden die Altersangaben aus diesen Sätzen in die numerische Variable N1 gestellt.

DS-Feld: Dezimalzahl in festen Spalten von CSV-Dateien

DS-Felder enthalten numerische Werte in Form von Textzeichen. Diese Felder sind nur in CSV-Dateien möglich:
Siehe SEP in den Statements INPUT-EXT, FETCH-FILE oder OUTPUT-EXT.
Die Position eines DS-Feldes im Datensatz wird durch die Anzahl davor stehender Separatorzeichen festgelegt.

Bei der Eingabe aus DS-Feldern sind die Ziffern 0 ... 9, Vorzeichen + und - sowie Dezimalpunkt oder Dezimalkomma zulässig. Vor und hinter einer Zahl dürfen auch Leerzeichen stehen.

Bei der Ausgabe in DS-Felder wird bei negativen Zahlen ein Minuszeichen vor den Wert gestellt. Im Statement OUTPUT-EXT entscheidet die Angabe DSTD darüber, wie die Dezimalzeichen aufbereitet werden.

Ein DS-Feld erlaubt folgende Angaben:

DSd **(** r **)** < n >

- d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

- r Die Angabe r = 1 ... ist die Reihenfolgenummer des Wertes im Datensatz d.
  Datenwerte bis zu 30 Zeichen Länge werden akzeptiert. Leerzeichen vor und hinter den Werten sind bei der Eingabe zulässig, werden aber ignoriert.
  Anstelle einer Zahl kann für die Reihenfolgenummer *r* auch ein Rechenausdruck mit + - * / Klammern und N-Variablen verwendet werden. Zum Beispiel `DS1((N2+8)*4-1)`.

- n Anzahl 1 ... 18 Nachkommastellen im Feld.

  Bei der Eingabe legt *n* fest, wie viele Ziffern des Eingabewertes als Nachkommastellen verarbeitet werden sollen, wenn der Eingabewert kein Dezimalkomma oder Dezimalpunkt enthält.
  Besitzt der Eingabewert ein Dezimalzeichen, so hat dieses Vorrang vor der Angabe *n.*
  Fehlt die Angabe *n,* so müssen Nachkommastellen im Eingabewert ein Dezimalzeichen besitzen.

  Bei der Ausgabe ohne die Angabe *n* werden die Nachkommastellen mit Dezimalpunkt oder Dezimalkomma ausgegeben oder ganz unterdrückt, je nach Einstellung DSTD im Statement OUTPUT-EXT.

  Mit der Angabe *n* werden die Ausgabewerte immer mit *n* Nachkommastellen versehen. Dabei wird der Ausgabewert gegebenenfalls gerundet oder mit Nullen aufgefüllt.

Eine CSV-Datei mit DS-Feldern könnte folgendermaßen beginnen:

```
0001;  ; 34; ...
0002; 5; 25; ...
```

Mit der Wertezuweisung

```
-N1=DS1(3)
```

werden die Zahlenwerte 34, 25, ... aus diesen Datensätzen in die numerische Variable N1 gestellt.

F-Feld: Dualzahl mit Vorzeichen (fixed point binary, integer)

F-Felder enthalten numerische Werte als Dualzahlen mit Vorzeichen in fester Position und Länge in den Datensätzen. Diese Felder können in ASCII-, ANSI-, UTF8-, EBCDIC-, COLBIN-, BINARY1- und BINARY2-Dateien vorkommen. Sie dürfen 1 bis 8 Bytes lang sein und besitzen positive oder negative Vorzeichen.

In BINARY2-Dateien sind Positionen und Längen in Feldern anzugeben, wobei jedes Feld zwei Bytes lang ist.
In COLBIN-Dateien sind dafür Spalten erforderlich, jede Spalte ebenfalls zu zwei Bytes.
BINARY2- und COLBIN-Dateien erlauben daher nur F-Felder von der Länge 1 bis 4, entsprechend 2 bis 8 Bytes.
Alle anderen Dateien verlangen diese Angaben als 1 bis 8 Bytes.

Normalerweise enthalten die linken Bytes der F-Felder die höherwertigen Bits. Mit der Angabe
INTEGER=REVERSE in INPUT-EXT, FETCH-FILE oder OUTPUT-EXT wird diese Reihenfolge vertauscht.

F-Felder erlauben folgende Angaben:

Fd **(** a **)** < n >
Fd **(** a **..** e **)** < n >
Fd **(** a, c **)** < n >

a Erste Position 1 ... des Feldes im Datensatz.
Anstelle einer Zahl ist auch ein Rechenausdruck mit + - * / Klammern und N-Variablen möglich.
Zum Beispiel `F1((N2+8)*4-1,4).`

c Länge 1 ... 8 des Feldes im Datensatz in Bytes. Bei COLBIN- und BINARY2-Dateien ist hier die Anzahl 1 ... 4 der Spalten oder Felder anzugeben. Die Angabe *c* kann fehlen; dann wird dafür 1 angenommen. Abhängig von dieser Länge können die F-Felder folgende Werte aufnehmen:

Länge in Bytes	Werte
1	–127 ... +127
2	–32 767 ... +32 767
3	–8 388 607 ... +8 388 607
4	–2 147 483 647 ... +2 147 483 647
5	–549 755 813 887 ... +549 755 813 887
6	–140 737 488 355 327 ... +140 737 488 355 327
7	–36 028 797 018 963 967 ... +36 028 797 018 963 967
8	–9 223 372 036 854 775 807 ... +9 223 372 036 854 775 807

Fehlt die Längenangabe *c* so wird dafür 1 angenommen.

d Nummer 1 ... 1 000 des externen Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

e Letzte Position des Feldes im Datensatz.
Wie bei der ersten Position *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable.
Das Feld darf 1 bis 8 Positionen lang sein.

n Anzahl 1 ... 18 Nachkommastellen im Feld.
Bei Eingabefeldern legt die Angabe *n* legt fest, dass eine entsprechende Anzahl von Dezimalziffern als Nachkommastellen zu verstehen sind. Fehlt *n* bei Eingabefeldern, so wird dafür 0 angenommen, also ohne Nachkommastellen gearbeitet.
In Ausgabefeldern legt *n* fest, wie viele Nachkommastellen auszugeben sind.
Fehlt *n* bei Ausgabefeldern, so werden so viele Nachkommastellen ausgegeben, wie der Wert rechts vom Gleichheitszeichen besitzt.

Enthält ein Datensatz zum Beispiel die folgenden Angaben

Inhalt:	099	123	002	005	000
Byte:	10	11	12	13	14

so stellt die Wertezuweisung

-N1=F1(11)

die Zahl 123 in die numerische Variable N1.

G-Feld: Dualzahl ohne Vorzeichen (fixed point binary, unsigned integer)

G-Felder enthalten numerische Werte als nicht negative Dualzahlen in fester Position und Länge in den Datensätzen. Diese Felder können in ASCII-, ANSI-, UTF8-, EBCDIC-, COLBIN-, BINARY1- und BINARY2-Dateien vorkommen. Sie dürfen 1 bis 8 Bytes lang sein und können keine negativen Werte annahmen.

In BINARY2-Dateien sind Positionen und Längen in Feldern anzugeben, wobei jedes Feld zwei Bytes lang ist.
In COLBIN-Dateien sind dafür Spalten erforderlich, jede Spalte ebenfalls zu zwei Bytes.
BINARY2- und COLBIN-Dateien erlauben daher nur G-Felder von der Länge 1 bis 4, entsprechend 2 bis 8 Bytes. Alle anderen Dateien verlangen diese Angaben als 1 bis 8 Bytes.

Normalerweise enthalten die linken Bytes der G-Felder die höherwertigen Bits. Mit der Angabe INTEGER=REVERSE in INPUT-EXT, FETCH-FILE oder OUTPUT-EXT wird diese Reihenfolge vertauscht.

Ein G-Feld erlaubt folgende Angaben:

Gd **(** a **)** < n >
Gd **(** a **..** e**)** < n >
Gd **(** a, c **)** < n >

a Erste Position 1 ... des Feldes im Datensatz.
Anstelle einer Zahl kann auch ein Rechenausdruck mit + - * / Klammern und N-Variable verwendet werden. Zum Beispiel `F1((N2+8)*4-1,4).`

c Anzahl Positionen 1 ... des Feldes im Datensatz.
Abhängig von dieser Länge können die G-Felder folgende Werte aufnehmen:

Länge in Bytes	Werte
1	0 ... 255
2	0 ... 65 535
3	0 ... 16 777 215
4	0 ... 4 294 967 295
5	0 ... 1 099 511 627 775
6	0 ... 281 474 976 710 655
7	0 ... 72 057 594 037 927 935
8	0 ... 9 223 372 036 854 775 807

Fehlt die Längenangabe *,c* so wird dafür 1 angenommen.

d Nummer 1 ... 1 000 des externen Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

e Letzte Position 1 ... des Feldes im Datensatz.
Wie bei der ersten Position *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable.
Ein G-Feld darf 1 ... 8 Bytes lang sein; siehe Wert *c* oben.

n Anzahl 1 ... 18 Nachkommastellen im Feld.
Bei Eingabefeldern legt die Angabe *n* legt fest, dass eine entsprechende Anzahl von Dezimalziffern als Nachkommastellen zu verstehen sind. Fehlt *n* bei Eingabefeldern, so wird dafür 0 angenommen, also ohne Nachkommastellen gearbeitet.

In Ausgabefeldern legt *n* fest, wie viele Nachkommastellen auszugeben sind.
Fehlt *n* bei Ausgabefeldern, so werden so viele Nachkommastellen ausgegeben, wie der Wert rechts vom Gleichheitszeichen besitzt.

Enthält ein Datensatz zum Beispiel die folgenden Angaben:

Inhalt:	099	123	252	005	000
Byte:	10	11	12	13	14

so stellt die Wertezuweisung

-N1=F1(12)

die Zahl 252 in die numerische Variable N1.

I-Feld: Dualzahlen als Merkmalswerte

I-Felder enthalten Merkmalsnummern als Dualzahlen ohne Vorzeichen (unsigned integer) in fester Position und Länge in den Datensätzen.

I-Felder können in ASCII-, ANSI-, UTF8-, EBCDIC-, COLBIN-, BINARY1- und BINARY2-Dateien vorkommen. Mehrfachnennungen sind möglich, indem mehrere solcher Dualzahlen nebeneinander gestellt werden. Negative Werte sind nicht zulässig. Positive Werte werden als Merkmalsnummer, die Zahl 0 als keine Angabe verstanden. Ein I-Feld kann in BINARY1-Dateien Werte zwischen 0 und 255 speichern und in BINARY2-Dateien Werte zwischen 0 und 65 535.

In BINARY2-Dateien sind Positionen und Längen in Feldern anzugeben, wobei jedes Feld zwei Bytes lang ist. In COLBIN-Dateien sind dafür Spalten erforderlich, jede Spalte ebenfalls zu zwei Bytes.

Normalerweise enthalten die linken Bytes der I-Felder die höherwertigen Bits. Mit der Angabe INTEGER=REVERSE in INPUT-EXT, FETCH-FILE oder OUTPUT-EXT wird diese Reihenfolge vertauscht.

Ein I-Feld erlaubt folgende Angaben:

Id(a <,c>)
Id(a , c , m)
Id(a.p <,c>)
Id(a.p , c , m)
Id(a .. e <,m>)
Id(a.p .. e <,m>)

a Erste Position 1 ... des Feldes im Datensatz.
Anstelle einer Zahl kann auch ein Rechenausdruck mit + - * / Klammern und N-Variablen verwendet werden. Zum Beispiel `I1((N2+8)*4-1,9).`

c Anzahl 1 ... Positionen pro Merkmal.
Ein Byte lange I-Felder fassen die Werte 1 ... 255, längere Felder die Werte 1 ... 65 535.
Fehlt die Angabe *c*, so wird dafür 1 angenommen.
BINARY2- und COLBIN-Dateien erlauben nur I-Felder von der Länge 1 oder 2 Positionen, entsprechend 2 oder 4 Bytes. Alle anderen Dateien lassen Längen von 1, 2, 3 oder 4 Bytes zu.

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

e Letzte Position 1 ... des Feldes im Datensatz. Bei Feldern mit Mehrfachnennungen ist dies die letzte Position der letzten Nennung. Die Länge des Feldes muss durch die Anzahl *c* der Positionen pro Merkmal teilbar sein. Ein I-Feld darf nicht länger als 200 Bytes werden.
Wie beim ersten Byte *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable.

m Gesamtlänge des Feldes im Datensatz in Positionen für nebeneinander stehende Mehrfachnennungen. Diese Länge muss durch die Anzahl *c* der Positionen pro Merkmal teilbar sein.
Fehlt die Angabe *m*, so sind keine Mehrfachnennungen möglich.
Die Reihenfolge der Angaben *m* und *c* darf vertauscht werden:
I1(1, 4, 2) ist gleichwertig mit I1(1, 2, 4).

---

p Nummer des ersten Merkmals im Feld
Bei der Ausgabe von Merkmalen in ein I-Feld sorgt diese Angabe dafür, dass für das Merkmal 1 der Wert *p* ausgegeben wird, für das Merkmal 2 der Wert *p+1* usw. Werte, die über der höchstmöglichen Merkmalsnummer der Zielfeldes liegen, werden ignoriert.

Umgekehrt sorgt der Wert *p* bei der Eingabe aus einem I-Feld dafür, dass der Eingabewert *p* als Merkmal 1 verarbeitet wird, der Eingabewert *p+1* als Merkmal 2 usw. Eingabewerte, die kleiner als *p* sind, werden als *keine Angabe* verstanden.
Fehlt die Angabe *p*, so wird dafür 1 angenommen: Die Merkmalsnummern werden unverändert übertragen.

Enthält zum Beispiel ein Datensatz die folgenden Angaben:

Inhalt:	099	123	002	005	000
Byte:	10	11	12	13	14

so setzt die Wertezuweisung

```
-C1=I1(10..14,1)
```

in der Variablen C1 die Merkmale 2, 5, 99 und 123 auf erfüllt und die übrigen Merkmale auf nicht erfüllt.

K-Feld: Kartenspalten

K-Felder enthalten die Merkmalsangaben 1 ... 9, 0, X, Y an fester Position und in fester Länge in den Datensätzen. Dieses Format entstammt den früheren Lochkarten und erlaubt die Speicherung von bis zu 12 Mehrfachnennungen in einer Datenposition.

Die Lochung 0 wird als Merkmal 10 verstanden, die Lochung X oder - als 11 und die Lochung Y oder & als 12. Die K-Felder können in ASCII-, ANSI-, UTF8-, EBCDIC-, COLBIN- und QUANTUM-Dateien vorkommen.
K-Felder dürfen bis zu 10 000 Merkmale enthalten und können bei beliebigen Lochungen einer Spalte beginnen oder enden.

K-Felder erlauben folgende Angaben:

Kd **(** a **)**	**K**d **(** a.p **)**
Kd **(** a **,** b **)**	**K**d **(** a.p **,** b **)**
Kd **(** a **...** e **)**	**K**d **(** a.p **...** e **)**
Kd **(** a **...** e.q **)**	**K**d **(** a.p **...** e.q **)**

a Erste Spalte 1 ... des Feldes im Datensatz einer COLBIN-Datei.
In ASCII-, ANSI-, EBCDIC- oder QUANTUM-Dateien ist dies das erste Byte des Feldes.
Fehlen die Angaben *b c* und *e*, so werden nur die Merkmale aus Feld *a* verwendet.
Anstelle einer Zahl IST auch ein Rechenausdruck mit + - * / Klammern und N-Variablen möglich.
Zum Beispiel K1 ((N2 + 8) * 4 - 1,12).

b Anzahl 1 ... 10 000 Merkmale des Datenfeldes.

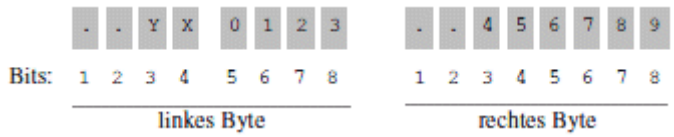
d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

e Letzte Spalte 1 ... des Feldes im Datensatz einer COLBIN-Datei.
In ASCII-, ANSI-, EBCDIC- oder UTF8-Dateien das letzte Byte der letzten Kartenspalte.
Wie bei der ersten Spalte *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable.

p Erste Lochung 1 ... 12 in Spalte a, bei der das Feld beginnen soll. Statt 10, 11 und 12 kann auch 0, X und Y angegeben werden. Fehlt diese Angabe, so wird mit der Lochung 1 begonnen.
Vorsicht bei der Ausgabe in ASCII-, ANSI- und UTF8 Dateien: Hier sind leere Datensätze mit Leerzeichen (hexadezimal 20) vorbelegt; das entspricht den Angaben 4 und 12, die vorher gegebenenfalls zu entfernen sind.

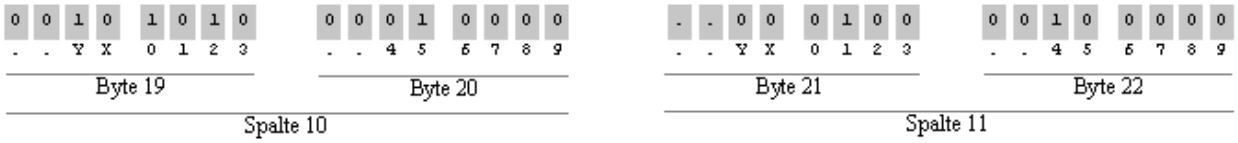
q Letzte Lochung 1 ... 12 in Spalte e, bei der das Feld enden soll. Statt 10, 11 und 12 kann auch 0, X und Y angegeben werden. Fehlt diese Angabe, so wird Spalte *e* bis zu ihrem Ende verwendet. Vorsicht bei der Ausgabe in ASCII-, ANSI- und UTF8 Dateien: Hier sind leere Datensätze mit Leerzeichen (hexadezimal 20) vorbelegt; das entspricht den Angaben 4 und 12, die vorher gegebenenfalls zu entfernen sind.

In K-Feldern werden die 12 Merkmale einer Kartenspalte auf folgende Weise in den Bits zweier benachbarter Bytes gespeichert:



Die hier mit . versehenen Bits nehmen im K-Format keine Daten auf. Sie sollten immer den Wert 0 besitzen.

Alle übrigen Bits können die Werte 0 oder 1 annehmen. 0 steht für keine Lochung oder Merkmal nicht erfüllt und 1 für Lochung oder Merkmal erfüllt. Beispielsweise könnten die Spalten 10 und 11 in einem COLBIN-Datensatz folgende Werte enthalten:



Dann würde die Wertezuweisung

```
-C1=K1(11..12)
```

in der Variablen C1 die Merkmale 2, 5, 10, 12, 13 und 16 auf erfüllt setzen und die übrigen Merkmale auf nicht erfüllt.

Im Anhang dieses Handbuchs zeigt die Tabelle Lochpositionen der COLBIN-Bytes alle möglichen Lochkombinationen mit dem dazugehörigen Zeichen und hexadezimalen Codes.

L-Feld: Einstellige Zeichen als Merkmalswerte

L-Felder enthalten Merkmalswerte als Textzeichen 1 ... 9, 0, - oder &. Sie stehen in fester Position und Länge in den Datensätzen. Sie können in ASCII-, ANSI-, UTF8-, EBCDIC-, COLBIN- und QUANTUM-Dateien vorkommen.

Das Zeichen 0 wird als Merkmal 10 verstanden, das Minuszeichen - als 11 und das Zeichen & als 12. Das Leerzeichen gilt als keine Angabe. Bei der Eingabe aus einem L-Feld wird auch das Zeichen X für 11 akzeptiert und Y für 12.

Ein Byte oder eine Spalte im L-Feld kann nur eines dieser Zeichen aufnehmen. Mehrfachnennungen lassen sich in einem längeren L-Feld direkt nebeneinander stellen.

L-Felder erlauben folgende Angaben:

Ld **(**a.. e **)**
Ld **(**a **,** c **)**
Ld **(**a.p **..** e **)**
Ld **(**a.p **,** c **)**

- a Erstes Byte 1 ... des Feldes im Datensatz. In COLBIN-Dateien ist dies die erste Spalte des Feldes. Anstelle einer Zahl kann auch ein Rechenausdruck mit + - * / Klammern und N-Variablen verwendet werden. Zum Beispiel `L1((N2+8)*4-1,9).`

- c Länge 1 ... 20 des Feldes in Bytes; in COLBIN-Dateien in Spalten.
  L-Felder mit mehr als einem Byte dienen der Speicherung von Mehrfachnennungen.
  Fehlt diese Angabe, so wird als Länge 1 angenommen.

- d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

- e Letztes Byte 1 ... des Feldes im Datensatz. In COLBIN-Dateien ist dies die letzte Spalte.
  Wie beim ersten Byte *a* sind hier Rechenausdrücke möglich, allerdings ohne N-Variable.

- p Nummer 1 ... 12 des ersten zu verarbeitenden Merkmals im Feld. Fehlt diese Angabe, so wird dafür 1 angenommen. Statt 10, 11 und 12 kann auch 0, X und Y angegeben werden.

Eine ASCII-Datei könnte zum Beispiel folgende Daten enthalten:

Inhalt:		4	2	&		0
Byte:	10	11	12	13	14	15

Dann setzt die Wertezuweisung

```
-C1=L1(10..15)
```

in der Variablen C1 die Merkmale 2, 4, 10 und 12 auf erfüllt und die übrigen Merkmale auf nicht erfüllt.

Die Wertezuweisung

```
-C1=L1(10.4,6)
```

Setzt mit dem Eingabewert 4 das Merkmal C1.1, mit dem Eingabewert 0 für 10 das Merkmal C1.7 und mit dem Eingabewert & für 12 das Merkmal C1.9 auf erfüllt. Eingabewerte kleiner als 4 werden ignoriert.

Bei der Ausgabe gibt die Wertezuweisung

```
-L1(10.4,6)=C1
```

für das Merkmal C1.1 den Wert 4 aus, für das Merkmal C1.2 den Wert 5 usw.
Werte über 12 werden bei der Ausgabe ignoriert.

M-Feld: Dezimalzahlen als Merkmalswerte

M-Felder enthalten ein bis vier Zeichen lange Merkmalsnummern in fester Position und Länge in den Datensätzen. Mehrfachnennungen lassen sich in einem längeren M-Feld direkt nebeneinander stellen.
M-Felder können in ASCII-, ANSI-, UTF8-, EBCDIC-, COLBIN- und QUANTUM-Dateien vorkommen.

Ein Byte lange Merkmale bestehen aus den Ziffern 0 ... 9. Die 0 wird dabei als Merkmal 10 verstanden.
Zwei Byte lange Merkmale bestehen aus den Werten 00 ... 99. Die Angabe 00 wird als Merkmal 100 verstanden.
Drei Byte lange Merkmale bestehen aus den Werten 000 ... 999. Die Angabe 000 wird als Merkmal 1000 verstanden. Vier Byte lange Merkmale bestehen schließlich aus den Werten 0000 ... 9999. Die Angabe 0000 wird als Merkmal 10 000 verstanden.

Die Merkmalswerte dürfen rechte und linke Leerzeichen besitzen. Führende Nullen sind ebenfalls zulässig.
Ein Wert nur aus Leerzeichen gilt als *keine Angabe*.

Ein M-Feld erlaubt folgende Angaben:

Md **(** a <,c> **)**
Md **(** a **,** c **,** m **)**
Md **(** a **.**p <,c> **)**
Md **(** a **.**p **,**c **,** m **)**
Md **(** a **..** e <,c> **)**
Md **(** a **.**p **..** e <,c> **)**

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

a Erstes Byte 1 ... des Feldes im Datensatz.
In COLBIN-Dateien ist dies die erste Spalte des Feldes. Anstelle einer Zahl kann auch ein Rechenausdruck mit + - * / Klammern und N-Variablen verwendet werden. Zum Beispiel
`M1((N2+8)*4-1,10)`.

c Anzahl 1 ... 4 Bytes oder Spalten pro Merkmal.
Fehlt diese Angabe, so wird dafür 1 angenommen.

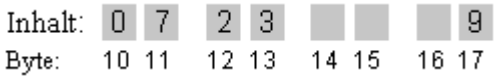
Ein Byte lange Merkmale bestehen aus den Ziffern 1 ... 9 oder 0 = 10,
zwei Byte lange Merkmale aus den Werten 1 ... 99 oder 0 = 100,
drei Byte lange Merkmale aus den Werten 1 ... 999 oder 0 = 1 000 und
vier Byte lange Merkmale aus den Werten 1 ... 9 999 oder 0 = 10 000.

m Gesamtlänge 1 ... 16 000 des Feldes in Bytes oder Spalten für nebeneinander stehende Mehrfachnennungen. Diese Länge muss durch die Anzahl *c* von Zeichen pro Merkmal teilbar sein. Fehlt die Angabe *m*, so sind keine Mehrfachnennungen möglich.
Die Reihenfolge der Angaben *m* und *c* darf vertauscht werden:
M1(1, 4, 2) ist gleichwertig mit M1(1, 2, 4).

e Letztes Byte 1 ... des Feldes im Datensatz.
In COLBIN-Dateien ist dies die letzte Spalte des Feldes.
Die Länge des Feldes muss durch die Anzahl *c* der Zeichen pro Merkmal teilbar sein. Wie beim ersten Byte *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable.

p Nummer 1 ... des ersten Merkmals im Feld. Fehlt diese Angabe, so wird dafür 1 angenommen.

Eine ASCII-Datei könnte zum Beispiel folgende Daten enthalten:



Dann setzt die Wertezuweisung

```
-C1=M1(10,2,8)
```

in der Variablen C1 die Merkmale 7, 9 und 23 auf erfüllt und die übrigen Merkmale auf nicht erfüllt.
Das gleiche Ergebnis liefern die Wertezuweisungen

```
-C1=M1(10..17,2)
```

und

```
-C1=M1(10,8,2)
```

MK-Feld: Merkmalswerte als Dezimalzahlen mit Schlüssel in CSV-Dateien

MK-Felder enthalten ein bis vier Zeichen lange Merkmalsnummern, die in der Datei mit einem Schlüssel gekennzeichnet sind. Die Schlüssel bestehen aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen. MK-Felder sind nur in CSV-Dateien möglich.

Die Schlüssel stehen vor den Datenwerten, durch ein Gleichheitszeichen von ihnen getrennt.
Sie dürfen in beliebigen Spalten der Datensätze erscheinen oder ganz fehlen.

Die Schlüssel können auch in den ersten Datensätzen der Datei enthalten sein, als Überschrift in der gleichen Spalte wie ihre Werte. Dazu muss das Statement INPUT-EXT, FETCH-FILE oder OUTPUT-EXT die Angabe KEYLINE enthalten.

Besitzt ein MK-Feld nur einen Schlüssel, so stehen Mehrfachnennungen zu diesem Schlüssel direkt hintereinander, jeweils durch Leerzeichen getrennt. Dabei sind auch Abkürzungen zulässig: 5-8 für 5 6 7 8.

Sind dagegen in einem MK-Feld mehrere Schlüssel angegeben, so werden die Mehrfachnennungen auf diese Schlüssel verteilt, pro Schüssel eine Nennung.

Ein MK-Feld erlaubt folgende Angaben:

MK d **(** < **.** p > **)**
MK d **(** s < **.** p > **)**
MK d **((**a**)** < **.** p > **)**
MK d **(** s **(**a**)** < **.** p > **)**

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

s Schlüssel des MK-Feldes in den Datensätzen.
Schlüssel dürfen maximal 64 Zeichen lang sein und aus beliebigen Zeichen bestehen.
Es wird nicht zwischen Groß- und Kleinschrift unterschieden.

Enthält der Schlüssel Klammern ( ) oder Punkte . so ist er in Hochkommas einzuschließen.
Zum Beispiel: `MK('AB()')` für den Schlüssel `AB().`

Der Schlüssel *s* kann entfallen, wenn auf der anderen Seite des Gleichheitszeichens eine Variable steht: Bei `-C1=MK1()` und `-MK1()=C1` wird *C1* zum Schlüssel.

Zur Verarbeitung von Mehrfachnennungen sind in einem MK-Feld auch mehrere Schlüssel möglich.
Zum Beispiel füllt

```
-C1=MK1(f11 f12 f13)
```

die Variable C1 aus den Angaben der drei Schlüssel aus dem Eingabesatz:

```
... f11=1; f12=3; f13=5; ...
```

Dabei kann auch abgekürzt werden: `MK1(f11...13)` oder `MK1('f11'...13)` liefern das gleiche Ergebnis.

p Nummer 1 ... 99 999 des ersten Merkmals im Feld. Fehlt diese Angabe, so wird dafür 1 angenommen.

Enthält eine Eingabedatei zum Beispiel folgende Daten

```
... c1=1 3 5; ...
```

so setzt die Wertezuweisung

```
-C1=MK1(.3))
```

mit dem Eingabewert 3 das Merkmal C1.1 und mit dem Eingabewert 5 das Merkmal C1.3 auf *erfüllt*. Eingabewerte kleiner als 3 werden ignoriert.

Bei der Ausgabe gibt die Wertezuweisung

```
-MK1(.3)=C1
```

für das Merkmal C1.1 den Wert 3 aus, für das Merkmal C1.2 den Wert 4 usw.

a Position 1 ... 9 999 eines einzulesenden Merkmals.
Mit dieser Angabe wird nur der Merkmalswert an der Position *a* eingelesen. Alle davor und dahinter stehenden Merkmale werden ignoriert. Enthält der Datensatz zum Beispiel die Angaben

```
... frage2=1 3 5-9 ...
```

so liest die Wertezuweisung

```
-C1=MK1(frage2(4))
```

die Merkmalsnummer 6 in die Variable C1 ein.

Eine Datei mit KEYLINE könnte folgendermaßen beginnen:

```
  ID; c2; n1; frage7;  ...
0001;   ; 34;  1 4 7;  ...
0002;  5; 25;  3 5-9;  ...
```

Die Wertezuweisung

```
-C1=MK1(FRAGE7)
```

setzt für den ersten Datensatz die Merkmale 1, 4 und 7 der Variablen C1 auf erfüllt und die übrigen Merkmale auf nicht erfüllt.

Die gleiche Datei ohne KEYLINE könnte folgendermaßen beginnen:

```
ID=0001;n1=34;frage7=1 4 7;  ...
ID=0002;frage7=3 5-9;;c2=5 ;n1=25...
```

Die Wertezuweisung

```
-C1=MK1(FRAGE7)
```

setzt für den ersten Datensatz die Merkmale 1, 4 und 7 der Variablen C1 auf erfüllt.

Mehrfachnennungen stehen in MK-Feldern direkt hintereinander, jeweils durch Leerzeichen voneinander getrennt. Zum Beispiel:

```
... ;frage4=3 5-9 12; ...
```

Dabei ist auch die Abkürzung 5 - 9 für die Merkmale 5, 6, 7, 8 und 9 möglich.

In MK-Feldern mit mehreren Schlüsseln ist zu jedem Schlüssel nur eine Nennung möglich:

```
... ;fr11=2;fr12=4;fr13=6; ...
```

Die Wertezuweisung

```
-C1=MK1(fr11 fr12 fr13 fr14)
```

setzt damit die Merkmale 2, 4 und 6 der Variablen C1 auf 1.

MS-Feld: Merkmalswerte als Dezimalzahlen in festen Spalten von CSV-Dateien

MS-Felder enthalten ein bis vier Zeichen lange Merkmalsnummern. Sie sind nur in CSV-Dateien möglich. Die Position eines MS-Feldes im Datensatz ist durch die Anzahl davor stehender Separatorzeichen bestimmt.

Ein MS-Feld erlaubt folgende Angaben:

MSd **(** r <.p>**)**
MSd **(** r <.p> **,** n **)**
MSd **(** r <.p> **..** s **)**
MSd **(** r <**(**a**)**> <.p> **)**
MSd **(** r <**(**a**)**> <.p> **,** n **)**
MSd **(** r <**(**a**)**> <.p> **..** s **)**

a Position 1 ... 10 000 eines einzulesenden Merkmals.
Mit dieser Angabe wird nur der Merkmalswert an der Position *a* eingelesen. Alle davor und dahinter stehenden Merkmale werden ignoriert.
Enthält der Datensatz zum Beispiel die Werte

```
;;1 3 5-9
```

so liest die Wertezuweisung

```
-C1=MS1(3(4))
```

die Merkmalsnummer 6 in die Variable C1 ein.
Die Angabe *(a*) ist nur bei der Eingabe sinnvoll, wenn das MS-Feld in der Wertezuweisung rechts vom Gleichheitszeichen steht.

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

r Reihenfolgenummer r = 1 ... des Feldes im Datensatz d vor dem Wert befinden sich r - 1 Separatorzeichen SEP.
Anstelle einer Zahl kann für die Reihenfolgenummer auch ein Rechenausdruck mit `+ - * /` Klammern und N-Variablen verwendet werden. Zum Beispiel `MS1((N2+8)*4-1)`.

n Anzahl 1 ... 10 000 direkt hintereinander liegender MS-Felder für Mehrfachnennungen.
Mit einer solchen Angabe *n* ist zwischen zwei Separatorzeichen nur ein Merkmalswert zulässig.
Ohne die Angaben *n* und *s* sind alle Mehrfachnennungen in einem Feld enthalten.

s Reihenfolgenummer 1 ... des letzten MS-Feldes für Mehrfachnennungen.
Mit einer solchen Angabe *s* ist zwischen zwei Separatorzeichen nur ein Merkmalswert zulässig.
Ohne die Angaben *n* und *s* so sind alle Mehrfachnennungen in einem Feld enthalten.

p Nummer 1 ... 99 999 des ersten Merkmals im Feld. Fehlt diese Angabe, so wird dafür 1 angenommen.

Enthält ein Eingabesatz zum Beispiel folgende Daten

```
1 3 5; ...
```

so setzt die Wertezuweisung

```
-C1=MS1(1.3))
```

aus dem Eingabewert 3 das Merkmal C1.1 und aus dem Eingabewert 5 das Merkmal C1.3 auf *erfüllt*. Eingabewerte kleiner als 3 werden ignoriert.

Bei der Ausgabe gibt die Wertezuweisung

```
-MS1(1.3)=C1
```

für das Merkmal C1.1 den Wert 3 aus, für das Merkmal C1.2 den Wert 4 usw.

Mehrfachnennungen können in ein MS-Feld hintereinander gestellt werden, jeweils durch Leerzeichen voneinander getrennt. Zum Beispiel:

```
... ;3 5 6 7 8 12; ...
```

Dabei sind auch Abkürzungen möglich. Die folgenden Angaben sind mit dem vorigen MS-Feld gleichwertig:

```
... ;3 5-8 12; ...
```

Mehrfachnennungen lassen sich aber auch in mehrere durch Separatorzeichen voneinander getrennte MS-Felder speichern, in jedem Feld eine Merkmalsnummer:

```
3;5;;7; ...
```

Die Wertezuweisung

```
-C1=MS1(1,4)
```

setzt damit die Merkmale 3, 5, und 7 der Variablen C1 auf erfüllt und die übrigen Merkmale auf nicht erfüllt.

P-Feld: Gepackte Dezimalzahlen

P-Felder enthalten numerische Werte in Form gepackter Dezimalzahlen in fester Position und Länge in den Datensätzen. Sie können in ASCII-, ANSI- , UTF8-, EBCDIC- und EBCDIC-Dateien vorkommen und dürfen 1 ... 10 Bytes lang sein.

Bei gepackten Dezimalzahlen werden in jedem Byte zwei Ziffern 0 ... 9 in jeweils 4 Bits als binäre Werte gespeichert. Die hexadezimalen Werte A ... F sind als Ziffern ungültig. Das letzte (rechte) Byte enthält nur im linken Halbbyte eine Ziffer und im rechten Halbbyte das Vorzeichen. Die hexadezimalen Werte A, C, E und F gelten dabei als positives Vorzeichen, B und D als negatives. Die Werte 0 ... 9 sind keine gültigen Vorzeichen. Gepackte Dezimalzahlen besitzen kein Dezimalzeichen.

P-Felder erlauben folgende Angaben:

Pd **(** a **..** e **)** < n >
Pd **(** a **,** c **)** < n >

- a Erstes Byte 1 ... des Feldes im Datensatz.
  Anstelle einer Zahl kann auch ein Rechenausdruck mit + - * / Klammern und N-Variablen verwendet werden. Zum Beispiel `P1((N2+8)*4-1,9).`

- c Länge 1 ... 10 des Zielfeldes im Datensatz in Bytes.

- d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

- e Letztes Byte 1 ... des Feldes im Datensatz. Ein P-Feld darf 1 ... 9 Bytes lang sein.
  Wie bei der Anfangsposition *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable.

- n Anzahl 1 ... 18 Nachkommastellen im Feld.

  Bei Eingabefeldern legt die Angabe *n* legt fest, dass eine entsprechende Anzahl von Dezimalziffern als Nachkommastellen zu verstehen sind. Fehlt *n* bei Eingabefeldern, so wird ohne Nachkommastellen gearbeitet.

  In Ausgabefeldern legt *n* fest, wie viele Nachkommastellen auszugeben sind.
  Fehlt *n* bei Ausgabefeldern, so werden so viele Nachkommastellen ausgegeben, wie der Wert rechts vom Gleichheitszeichen besitzt.

Enthält eine Eingabedatei zum Beispiel folgende Daten

Inhalt hexadezimal:	0	4	5	0	0	C
Byte:		10		11		12

so stellt die Wertezuweisung

```
-N1=P1(10..12)
```

den Wert 4 500 in die Variable N1.

U-Feld: Merkmalswerte aus den Zeichen 0 und 1

U-Felder enthalten Merkmalswerte als Folge der Zeichen 0 und 1 in fester Position und Länge in den Datensätzen.

Das Zeichen 1 steht für ein erfülltes Merkmal und das Zeichen 0 für nicht erfüllt. Leerzeichen vor und hinter einer Zeichenkette aus 0 und 1 sind zulässig, werden aber ignoriert. Leerzeichen zwischen den Zeichen 0 und 1 werden als 0 gewertet.

U-Felder dürfen maximal 9 999 Zeichen lang sein. Sie können in ASCII-, ANSI-, UTF8-, EBCDIC-, COLBIN- und QUANTUM-Dateien vorkommen.

U-Felder erlauben folgende Angaben:

Ud **(** a **)**
Ud **(** a **..** e **)**
Ud **(** a **,** b **)**

a Erstes Byte 1 ... des Feldes im Datensatz.
In COLBIN-Dateien ist dies die erste Spalte des Feldes. Anstelle einer Zahl ist auch ein Rechenausdruck mit + - * / Klammern und N-Variablen möglich. Zum Beispiel

```
U1((N1+8)*4-1,9).
```

b Länge 1 ... 255 des Feldes in Bytes.
Für COLBIN-Dateien ist diese Länge in Spalten anzugeben.
Fehlen die Angaben *b* und *e*, so wird als Länge 1 angenommen.

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

e Letztes Byte 1 ... des Feldes im Datensatz.
In COLBIN--Dateien ist dies die letzte Spalte des Feldes.
Dieser Wert ist so zu wählen, dass das U-Feld nicht mehr als 9 999 Merkmale enthält.
Wie bei der Anfangsposition *a* sind hier auch Rechenausdrücke möglich, allerdings ohne N-Variable.
Fehlen die Angaben *b* und *e*, so wird als Feldlänge 1 angenommen.

Zum Beispiel könnte eine ASCII-Datei folgende Daten enthalten:

Inhalt:	0	1	1	0	0			1
Byte:	10	11	12	13	14	15	16	17

Die Wertezuweisung

```
-C1=U1(10..17)
```

setzt damit die Merkmale 2, 3 und 8 der Variablen C1 auf 1 und die übrigen Merkmale auf 0.

UK-Feld: Merkmalswerte aus 0 und 1 mit Schlüssel in CSV-Dateien

UK-Felder enthalten Merkmalswerte als Folge der Zeichen 0 und 1, die innerhalb der Datei durch Schlüssel gekennzeichnet sind. Die Schlüssel bestehen aus Buchstaben, Ziffern, Sonderzeichen und Leerzeichen.
UK-Felder sind nur in CSV-Dateien möglich. Siehe SEP in den Statements INPUT-EXT, FETCH-FILE oder OUTPUT-EXT.

Die Schlüssel stehen vor den Datenwerten, durch ein Gleichheitszeichen von ihnen getrennt.
Sie dürfen in beliebigen Spalten der Datensätze erscheinen oder ganz fehlen.

Die Schlüssel können auch in den ersten Datensätzen der Datei enthalten sein, als Überschrift in der gleichen Spalte wie ihre Werte. Dazu muss das Statement INPUT-EXT, FETCH-FILE oder OUTPUT-EXT die Angabe KEYLINE enthalten.

Besitzt ein UK-Feld nur einen Schlüssel, so dürfen seine Werte maximal 9 999 Zeichen lang sein. Das Zeichen 1 steht für ein erfülltes Merkmal und das Zeichen 0 für ein nicht erfülltes.
Leerzeichen vor und hinter einer Zeichenkette aus 0 und 1 sind zulässig, werden aber ignoriert.
Leerzeichen zwischen den Zeichen 0 und 1 werden als 0 gewertet.

Sind dagegen in einem UK-Feld mehrere Schlüssel angegeben, so werden die Mehrfachnennungen auf diese Schlüssel verteilt, pro Schlüssel eine Angabe 0 oder 1.

Ein UK-Feld erlaubt folgende Angaben:

UKd **( )**
UKd **(** < s > **)**

d Nummer 1 ... 1 000 des Datensatzes, in den das Feld gestellt oder aus dem es entnommen werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

s Schlüssel in den Datensätzen.
Diese dürfen maximal 64 Zeichen lang sein und aus beliebigen Zeichen bestehen.
Es wird nicht zwischen Groß- und Kleinschrift unterschieden.

Enthält der Schlüssel Klammern (oder) so ist er in Hochkommas einzuschließen.
Zum Beispiel: `UK('AB()')` für den Schlüssel `AB()`.

Der Schlüssel *s* kann entfallen, wenn auf der anderen Seite des Gleichheitszeichens eine Variable steht: Bei `-C1=UK1()` und `-UK1()=C1` wird *C1* zum Schlüssel.

Zur Verarbeitung von Mehrfachnennungen sind in einem UK-Feld auch mehrere Schlüssel möglich.
Zum Beispiel füllt

```
-C1=UK1(f11 f12 f13)
```

die Variable C1 aus den Angaben der drei Schlüssel.
Dabei kann auch abgekürzt werden: UK1(f11...13) oder UK1('f11'...13) liefern das gleiche Ergebnis.

Eine Datei mit KEYLINE könnte folgendermaßen beginnen:

```
  ID; c2; n1;   frage7;  ...
0001;   ; 34;  1001001;  ...
0002;  5; 25;  0010111;  ...
```

Die Wertezuweisung

```
-C1=UK1(FRAGE7)
```

setzt zum ersten Datensatz die Merkmale 1, 4 und 7 der Variablen C1 auf 1 und die übrigen Merkmale auf 0.

Die gleiche Datei ohne KEYLINE könnte folgendermaßen beginnen:

```
ID=0001;n1=34;frage7=1001001;  ...
ID=0002;frage7=0010111;c2=5 ;n1=25; ...
```

Die Wertezuweisung

```
-C1=UK1(FRAGE7)
```

setzt für den ersten Datensatz die Merkmale 1, 4 und 7 der Variablen C1 auf 1, die übrigen Merkmale auf 0.

In UK-Feldern mit mehreren Schlüsseln ist zu jedem Schlüssel nur eine Nennung möglich:

```
... fr11=1;fr12=0;fr13=1; ...
```

Die Wertezuweisung

```
-C1=UK1(fr11 fr12 fr13)
```

setzt damit die Merkmale 1 und 3 der Variablen C1 auf 1 und die restlichen auf 0.

US-Feld: Merkmalswerte aus 0 und 1 in festen Spalten von CSV-Dateien

US-Felder enthalten Merkmalswerte als Folge der Zeichen 0 und 1. Sie sind nur in CSV-Dateien möglich.
Die Position eines US-Feldes im Datensatz ist durch die Anzahl davor stehender Separatorzeichen bestimmt.

US-Werte dürfen maximal 9 999 Zeichen lang sein. Das Zeichen 1 steht in einem U-Feld für ein erfülltes Merkmal und das Zeichen 0 für ein nicht erfülltes. Leerzeichen vor und hinter einer Zeichenkette aus 0 und 1 sind zulässig, werden aber ignoriert. Leerzeichen zwischen den Zeichen 0 und 1 werden als 0 gewertet.

Ein US-Feld erlaubt folgende Angaben:

USd **(** r **)**
USd **(** r **,** n **)**
USd **(** r **..** s **)**

d Nummer 1 ... 1 000 des Datensatzes, aus dem das Feld entnommen oder in den es gestellt werden soll. Die Nummerierung der Datensätze richtet sich nach der Angabe RC in den Statements INPUT-EXT, OUTPUT-EXT und FETCH-FILE.

r Die Angabe r = 1 ... ist die Reihenfolgenummer des Wertes im Datensatz d.
Vor dem Wert befinden sich *r - 1* Separatorzeichen SEP.
Anstelle einer Zahl ist für *r* auch ein Rechenausdruck mit + - * / Klammern und N-Variablen möglich. Zum Beispiel `US1((N2+8)*4-1)`.

n Anzahl 1 ... 9 999 direkt hintereinander liegender US-Felder für Mehrfachnennungen.
Mit einer solchen Angabe *n* ist zwischen zwei Separatorzeichen nur ein Wert 0 oder 1 zulässig.
Ohne die Angaben *n* und *s* sind alle Mehrfachnennungen in der Form *1011* ... in einem Feld enthalten.

s Reihenfolgenummer 1 ... des letzten US-Feldes für Mehrfachnennungen.
Mit einer solchen Angabe *s* ist zwischen zwei Separatorzeichen nur ein Wert 0 oder 1 zulässig.
Ohne die Angaben *n* und *s* sind alle Mehrfachnennungen in der Form *1011* ... in einem Feld enthalten.

Wenn ein Datensatz zum Beispiel mit den folgenden Daten beginnt

```
01100 1; ...
```

so setzt die Wertezuweisung

```
-C1=US1(1)
```

die Merkmale 2, 3 und 7 der Variablen C1 auf 1 und die restlichen Merkmale auf 0.

Mehrfachnennungen lassen sich aber auch in mehrere direkt hintereinander liegende US-Felder speichern, in jedem Feld höchstens eine 0 oder 1. Die folgenden Angaben sind gleichwertig mit dem obigen Beispiel:

```
0;1;1;0;0; ;1; ...
```

Hier setzt die Wertezuweisung

```
-C1=US1(1,7)
```

ebenfalls die Merkmale 2, 3 und 7 der Variablen C1 auf 1 und die restlichen Merkmale auf 0.

Wertezuweisungen

Diese Wertezuweisungen sind Folgestatements zu ADD-PROC, MOD-PROC, UPD-PROC und OUTPUT-EXT. Mit ihnen lassen sich für jeden statistischen Fall Variable mit Werten versehen oder Werte in die Dateien INPUT-EXT oder OUTPUT-EXT stellen.

Jede Wertezuweisung ist ein eigenes Statement. Es muss in der ersten Position der Zeile mit einem Strich - beginnen. Diese Statements haben stets die Form:

– x = y

Die Angabe x links vom Gleichheitszeichen steht für eine Variable oder ein externes Datenfeld als Ziel der Wertezuweisung.

Die Angabe y rechts vom Gleichheitszeichen steht für einen Eingabewert, der aus folgenden Quellen stammen kann:

- einem Datenfeld einer externen Datei INPUT-EXT oder FETCH-FILE,
- einer Variablen,
- einem logischen oder numerischen Ausdruck,
- einem im Statement festgelegten konstanten Wert.

Bei einer solchen Wertezuweisung wird der bestehende Wert der Zielvariablen oder des Zielfeldes x durch den Eingabewert y ersetzt. Zum Beispiel:

```
-C100=C201
-N9=12.85
-N25=(N10+N20+N30)/3
-C25.1=C99.7&(C10.1|C20.1|C30.1)
```

Ein externes Datenfeld auf der rechten Seite des Gleichheitszeichens bezieht sich stets auf die von INPUT-INT eingelesenen statistischen Fälle. Das Statement:

```
-N17=D1(5..7)
```

entnimmt aus einem D-Feld von INPUT-INT einen numerischen Wert und stellt ihn in die Variable N17.

In einer Wertezuweisung hinter OUTPUT-EXT stellt ein externes Datenfeld auf der linken Seite des Gleichheitszeichens Daten in einen Datensatz der Ausgabedatei. So überträgt das Statement

```
-D2(5..7)=G3(10,1)
```

hinter OUTPUT-EXT Daten aus einem G-Feld von INPUT-EXT in ein D-Feld von OUTPUT-EXT. Eventuelle Format-Umwandlungen erfolgen automatisch, wenn zum Beispiel die Eingabedaten im ASCII-Format vorliegen und die Ausgabedatei im COLBIN-Format erstellt wird.

Hinter ADD-PROC und UPD-PROC überträgt das gleiche Statement das G-Feld aus dem dritten eingelesenen Datensatz in ein D-Feld des zweiten ebenfalls aus INPUT-EXT eingelesenen Datensatzes. Dies geschieht nur temporär im Arbeitsspeicher; die Eingabedatei selbst wird dadurch nicht verändert.

Es gibt folgende Typen von Wertezuweisungen:

C-Typ = Merkmale, logische Werte
Bei dieser Wertezuweisung werden Merkmalswerte übertragen.
Auf beiden Seiten des Gleichheitszeichens darf eine Bedingungsvariable C... stehen oder ein externes Datenfeld vom Typ B, I, K, L, M, MK, MS, U, UK oder US.
Auf der rechten Seite des Gleichheitszeichens dürfen auch logische Ausdrücke und konstante Werte aus den Ziffern 0 und 1 verwendet werden.

N-Typ = Zahl, numerischer Wert
Bei dieser Wertezuweisung wird eine Zahl übertragen.
Auf beiden Seiten des Gleichheitszeichens darf eine numerischen Variable N... erscheinen oder ein externes Datenfeld vom Typ D, DK, DS, F, G oder P.
Auf der rechten Seite des Gleichheitszeichens dürfen auch numerische Ausdrücke und konstante Zahlenwerte verwendet werden.

T-Typ = Text, alphanumerischer Wert
Bei dieser Wertezuweisung wird ein Text übertragen.
Auf beiden Seiten des Gleichheitszeichens darf eine Textvariable T... erscheinen oder ein externes Datenfeld vom Typ A, AK oder AS.
Auf der rechten Seite des Gleichheitszeichens sind auch konstante Texte möglich.

Wertezuweisungen mit der FETCH-Funktion
Hier stammen die Eingabewerte nicht aus INPUT-EXT oder Variablen sondern werden aus zusätzlichen externen Dateien über Schlüsselwerte eingelesen. Siehe Abschnitt Wertezuweisungen mit der FETCH-Funktion.

Merkmale

Wertezuweisungen mit Merkmalen sind Folgestatements zu ADD-PROC, UPD-PROC, MOD-PROC und OUTPUT-EXT. Sie übertragen zu jedem statistischen Fall Merkmalswerte zwischen Variablen und Datenfeldern. Ein Merkmal nimmt für jeden auszuwertenden Fall entweder den Wert 1 für wahr beziehungsweise erfüllt oder 0 für falsch beziehungsweise nicht erfüllt an.

Der Eingabewert einer solchen Zuweisung kann aus einem Datensatz stammen, der von INPUT-EXT eingelesen wurde, er kann aus einer C-Variablen kommen, die von INPUT-INT eingelesen oder während der laufenden Verarbeitung definiert wurde. Er kann ein Merkmalswert sein, der in komplexen logischen Ausdrücken ermittelt wurde, oder auch aus konstanten Merkmalswerten 0 oder 1 bestehen, die in den Statements festgelegt sind.

Eine solche Übertragung kann Merkmalswerte in C-Variable stellen, sie kann Merkmale in eine externe Datei OUTPUT-EXT ausgeben, und sie kann schließlich Merkmale in Datensätze aus INPUT-EXT stellen und dabei die gerade eingelesenen Daten überschreiben. Dieses Überschreiben erfolgt aber nur für die laufende Verarbeitung. Die Datei selbst wird dadurch nicht verändert.

Weitere Angaben zu Merkmalswerten finden sich in den Abschnitten Variable, Logische Ausdrücke sowie Externe Datenfelder.

Mit einer Wertezuweisung können einzelne Merkmale oder mehrere Merkmale gleichzeitig übertragen werden. Dabei wird das erste Merkmal rechts vom Gleichheitszeichen dem ersten Merkmal der linken Seite zugewiesen, das zweite rechte dem zweiten linken usw.
Besitzt das Zielfeld auf der linken Seite des Gleichheitszeichens weniger Merkmale als das Eingabefeld auf der rechten Seite, so werden die Eingabemerkmale rechts abgeschnitten. Dazu erscheint keine Fehlermeldung.
Enthält das Zielfeld dagegen mehr Merkmale als die Eingabe, so werden die rechts überstehenden Merkmale des Zielfeldes auf 0 gesetzt. Auch dazu erscheint keine Fehlermeldung.

Folgende Wertezuweisungen sind für Merkmale möglich:

–**C**m...	**=**	Konstante						<	**SH**	>
–**B**z...	**!=**	°Konstante							**SL**	
–**I**z...	**&=**	numerischer Ausdruck		<**SKB**>	<**KKB**>	<**BZ**>			**SR**	
–**K**z...		**N**n...		<**SKB**>	<**KKB**>					
–**L**z...										
–**M**z...		logischer Ausdruck								
–**MK**z...										
–**MS**z...		**C**n...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>				
–**Zu**...										
–**UK**z...		**B**r...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>	<**INC=**w>			
–**US**z...		**I**r...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>	<**INC=**w>			
		Kr...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>	<**INC=**w>			
		Lr...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>	<**INC=**w>			
		Mr...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>	<**INC=**w>			
		MKr...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>				
		MSr...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>	<**INC=**w>			
		Ur...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>	<**INC=**w>			
		UKr...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>				
		USr...	<**Z**i>	<**SKE**>	<**SKB**>	<**KKB**>	<**INC=**w>			

Hierbei bedeutet:

= Das Zielfeld links vom Gleichheitszeichen wird vor der Wertezuweisung gelöscht.
Seine bisherigen Merkmalswerte gehen dabei verloren und werden durch neue ersetzt.

!= Links von dieser Angabe darf nur eine C-Variable stehen, rechts davon gibt es keine Einschränkung. Die Merkmale der Zielvariablen werden vor der Wertezuweisung nicht gelöscht. Die neuen Merkmalswerte rechts vom Gleichheitszeichen werden den alten Werten durch logisches 'oder' (bitweise) hinzugefügt. Statt != kann auch |= angegeben werden.
So verknüpft das Statement

```
-C1(1..4) |= C2(5..8)
```

die Merkmale 1 bis 4 von C1 und die Merkmale 5 bis 8 von C2 durch logisches 'oder' miteinander. Zum Beispiel könnten die Variablen C1 und C2 folgende Werte enthalten:

```
C1(1...4): 1100    vor der Wertezuweisung
C2(5...8): 1010    vor der Wertezuweisung
C1(1...4): 1110    nach der Wertezuweisung.
```

&= Links von dieser Angabe darf nur eine C-Variable stehen, rechts davon gibt es keine Einschränkung. Die Merkmale der Zielvariablen werden vor der Wertezuweisung nicht gelöscht. Die neuen Merkmalswerte rechts vom Gleichheitszeichen werden durch logisches 'und' (bitweise) mit den alten Werten verknüpft. So verrechnet das Statement

```
-C1(1..4) &= C2(5..8)
```

die Merkmale 1 bis 4 von C1 und die Merkmale 5 bis 8 von C2 durch logisches 'und' miteinander. Zum Beispiel könnten die Variablen C1 und C2 folgende Werte enthalten:

```
C1(1...4): 1100    vor der Wertezuweisung
C2(5...8): 1010    vor der Wertezuweisung
C1(1...4): 1000    nach der Wertezuweisung.
```

Bz...
Iz...
Kz...
Lz...
Mz...
MKz...
MSz...
Uz...
UKz...
USz...

Einem solchen externen Datenfeld wird, da es links vom Gleichheitszeichen steht, für jeden statistischen Fall ein neuer Wert zugewiesen. Dies ist nur in den Folgestatements von ADD-PROC, UPD-PROC und OUTPUT-EXT möglich.
Gehört die Wertezuweisung zu OUTPUT-EXT, so ist das Zielfeld ein Datenfeld der externen Ausgabedatei. Liegt sie dagegen hinter ADD-PROC oder UPD-PROC, so ist es ein Feld in dem zuletzt aus INPUT-EXT gelesenen statistischen Fall. Die gerade eingelesenen Daten werden dann durch die Wertezuweisung überschrieben, die Eingabedatei selbst aber nicht verändert.
Zum Beispiel sorgt das Statement

```
-B2(3.1,8) = C15
```

hinter OUTPUT-EXT dafür, dass die Merkmale der Variablen C15 in den zweiten Datensatz eines jeden ausgegebenen statistischen Falles in die Bits 1 bis 8 des dritten Bytes gestellt werden. Die Merkmale aus C15 werden linksbündig in das B-Feld gestellt: C15.1 in das erste (linke) Bit, C15.2 in das zweite bis hin zu C15.8 in das Bit 8. Wurde die Variable C15 mit weniger als 8 Merkmalen definiert, so erhalten die letzten Bits des B-Feldes den Wert 0. Besitzt C15 dagegen mehr als 8 Merkmale, so werden ab dem neunten keine weiteren Merkmale von C15 übertragen.

Br...
Ir...
Kr...
Lr...
Mr...
MKr...
MSr...
Ur...
UKr...
USr...

Einem solchen Datenfeld wird, da es auf der rechten Seite des Gleichheitszeichens steht, für jeden statistischen Fall ein Wert entnommen und in das Zielfeld oder die Zielvariable gestellt gestellt. Dies ist nur in den Folgestatements von ADD-PROC, UPD-PROC und OUTPUT-EXT möglich.
Diese Datenfelder stammen stets aus dem zuletzt aus INPUT-EXT gelesenen statistischen Fall.
Zum Beispiel sorgt das Statement

```
-C15=B2(3.1,8)
```

hinter ADD-PROC dafür, dass aus dem zweiten Datensatz jedes aus INPUT-EXT gelesenen Falles die Bits 1 bis 8 aus Byte 3 in die Merkmale der Bedingungsvariablen C15 gestellt werden.
Der Wert 0 oder 1 von Bit 1 wird in das Merkmal C15.1 übertragen, Bit 2 in Merkmal C15.2 bis hin zu Bit 8 in C15.8.
Besitzt die Variable C15 weniger als 8 Merkmale, so werden die letzten Bits aus Byte 3 nicht mehr übertragen. Besitzt C15 dagegen mehr als 8 Merkmale, so werden sie ab dem neunten Merkmal alle auf den Wert 0 gesetzt.

Ni Enthält die Variable N3 im Statement

```
-C1=N3
```

den Wert 7, so wird das Merkmal C1.7 auf 1 und die übrigen Merkmale von C1 auf 0.
Eventuelle Nachkommastellen der Variablen N7 werden dabei durch Rundung entfernt.

Enthält N3 den Wert Z0 für *keine Angabe*, so werden alle Merkmale von C1 gelöscht. Das Gleiche geschieht, wenn N3 einen Wert kleiner 1 oder einen Wert über dem höchsten Merkmal von C20 besitzt.

Eine solche Wertezuweisung lässt sich auch auf ausgewählte Merkmale der Variablen C1 beschränken. Mit dem Statement

```
-C1(5..7)=N3
```

wird das Merkmal C1.5 auf 1 gesetzt, wenn N3 den Wert 1 enthält.
Die Merkmale 6 und 7 werden gelöscht und die übrigen Merkmale nicht verändert.

Entsprechend wird mit den anderen Bedingungsfeldern auf der linken Seite des Gleichheitszeichens verfahren. Das Statement

```
-U1(5..8)=N3
```

stellt zum Beispiel den Wert 0100 in das Ausgabefeld, wenn N3 den Wert 2 besitzt.

Die Werte der numerischen Variablen lassen sich auch zu Gruppen zusammenfassen. Enthält zum Beispiel die Variable N3 Altersangaben als Zahlen 0 ... 99, so setzt das Statement

```
-C1=N3(0 30 60 100)
```

folgende drei Altersklassen in der Variablen C1:

C1.1: 0 bis 29 Jahre
C1.2: 30 bis 59 Jahre
C1.3: 60 bis 99 Jahre

Cm ... Bedingungsvariable, die neue Werte erhalten soll.
Zum Beispiel versieht das Statement

```
-C20=K1(12.1,10)
```

alle Merkmale der Bedingungsvariablen C20 für jeden statistischen Fall mit neuen Werten.
Die Werte dazu stammen aus einem K-Feld im ersten Datensatz jedes aus INPUT-EXT eingelesenen statistischen Falles. Es beginnt in Spalte 12 und besteht aus 10 Merkmalen oder 'Lochpositionen'. Der Wert aus Position 1 wird in das Merkmal C20.1 gestellt, der aus Position 2 in C20.2 bis hin zu dem Merkmal aus Position 0 in C20.10. Wurde C20 mit weniger als 10 Merkmalen definiert, so werden die Werte aus den letzten Positionen des K-Feldes nicht übertragen. Besitzt C20 dagegen mehr Merkmale, so erhalten die letzten den Wert 0.

Die Statements

```
-C20.1=1
-C20.2=0
```

setzen das Merkmal 1 der Variablen C20 auf 1 und löschen das Merkmal 2 auf 0. Allde übrigen Merkmale bleiben unverändert.

In dem Statement

```
-C20.3=(C112.1 | .2) & N17 > 25 & < 75
```

wird zunächst der logische Ausdruck auf der rechten Seite des Gleichheitszeichens ausgewertet.
Ist dieser erfüllt, so erhält das Merkmal C20.3 den Wert 1, andernfalls den Wert 0.

Das Statement

```
-C20=C10
```

überträgt alle Merkmale der Variablen C10 in die Variable C20. Besitzt C20 mehr Merkmale, so werden die rechten nicht übertragen. Besitzt C10 mehr Merkmale, so werden die rechten davon auf 0 gelöscht.

Ausgewählte, direkt hintereinander liegende Merkmale lassen sich in einem Statement verändern:

```
-C20(3...7)=°11111
```

Hiermit erhalten die Merkmale 3 bis 7 der Variablen C20 den Wert 1. Alle anderen bleiben unverändert.

Wird hinter einer Bedingungsvariablen in den Klammern nur ein Merkmal angegeben, so erhalten dieses und alle darauf folgenden Merkmale neue Werte. Zum Beispiel versorgt

```
-C20(3)=°11111
```

alle Merkmale der Variablen C20 ab Merkmal 3 mit neuen Werten. Besitzt C20 mehr als 7 Merkmale, so werden ab Merkmal 8 alle Merkmale auf 0 gesetzt: Der Eingabewert 11111 wird eine entsprechende Anzahl von Nullen ergänzt.

Mit

```
-C20=3
```

wird das Merkmal 3 der Variablen C20 auf 1 gesetzt. alle übrigen Merkmale erhalten den Wert 0.

Es können mehrere Variable mit einem Statement neue Werte erhalten. So löscht das Statement

```
-C100...150=0
```

alle Merkmale der Variablen C100 bis C150 auf 0. Dabei müssen nicht alle Variablen zwischen C100 und C150 definiert sind.

Cn ... Bedingungsvariable C0 ... C99999, aus der Merkmalswerte zu entnehmen sind.
Zum Beispiel stellt das Statement

```
-M1(12.1,4,2)=C30
```

die Merkmale der Variablen C30 für jeden statistischen Fall in das M-Feld eines externen Datensatzes. Dieses Feld kann zwei zweistellige Merkmalsnummern aufnehmen. Dazu werden die Nummern der beiden kleinsten Merkmale der Variablen C30 verwendet, die den Wert 1 besitzen. Enthält C30 mehr als zwei erfüllte Merkmale, so werden die höheren Merkmalsnummern nicht übertragen. Das M-Feld wird von links gefüllt. Nicht benötigte Bytes werden mit Leerzeichen gefüllt.

Es besteht auch die Möglichkeit, ausgewählte Merkmale der Variablen C30 zu übertragen.
Durch das Statement

```
-M1(12.1,4,2)=C30(10...15)
```

werden nur die Merkmale C30.10 bis C30.15 in das M-Feld gestellt. Ist C30.10 erfüllt, so wird dafür die Merkmalsnummer 01 ausgegeben, für C30.11 die Nummer 02 usw.
Dagegen überträgt

```
-M1(12.10,4,2)=C30(10...15)
```

die Merkmalsnummern unverändert.

Sollen ab einem Merkmal einer Variablen alle dahinterstehenden Merkmale übertragen werden, so genügt es, in den Klammern hinter der Variablen nur dieses erste Merkmal anzugeben.
Zum Beispiel werden durch

```
-C1=C30(10)
```

alle Merkmale der Variablen C30 ab dem Merkmal 10 in die Variable C1 zu übertragen.

Weiter kann auch ein einzelnes Merkmal der Bedingungsvariablen übertragen werden.
Das Statement

```
-K1(24.2,1)=C40.4
```

stellt eine 1 in die Position 2 des K-Feldes, wenn das Merkmal C40.4 erfüllt ist. Andernfalls erhält die Position 2 den Wert 0.

Konstante

Folgende Arten von Konstanten sind möglich:
Das Statement

```
-C20=12
```

setzt das Merkmal C20.12 auf 1 und alle übrigen Merkmale auf 0.

Mit

```
-C20=Z0
```

werden alle Merkmale der Variablen C20 auf 0 gelöscht.

Das Statement

```
-C20(5..10)=Z0
```

löscht die Merkmale 5 bis 10 der Variablen C20 auf 0 und lässt die übrigen unverändert.

Im Statement

```
-C20=°1011
```

werden die Ziffern hinter dem Zeichen ° als Merkmalswerte 0 für falsch oder 1 für wahr verstanden. Eine solche Kette darf bis zu 10 000 Zeichen lang sein.
Hier werden die Merkmale C20.1, C20.3 und C20.4 auf 1 gesetzt und die restlichen Merkmale auf 0 gelöscht.

Das Statement

```
-C20(5..7)=°111
```

setzt die Merkmale C20.5, C20.6 und C20.7 auf 1. Die restlichen Merkmale von C20 bleiben unverändert.

logischer Ausdruck

Logische Ausdrücke werden im Abschnitt *Logische Ausdrücke* ausführlich beschrieben.
Sie liefern den Wert 1, wenn sie erfüllt sind, und den Wert 0, wenn sie nicht erfüllt sind.
Zum Beispiel besitzt das Statement

```
-C20.3=(C112.1 | .2) & N17 > 25 & < 75
```

auf der rechten Seite des Gleichheitszeichens einen logischer Ausdruck. Ist dieser erfüllt, so erhält das Merkmal C20.3 den Wert 1, andernfalls den Wert 0.

numerischer Ausdruck

Numerische Ausdrücke werden im Abschnitt *Numerische Ausdrücke* ausführlich beschrieben.
Sie liefern als Ergebnis einen Zahlenwert. Mit einer solchen Wertezuweisung wird das Merkmal zu diesem Zahlenwert auf 1 oder *erfüllt* gesetzt und alle übrigen Merkmale auf 0 oder *nicht erfüllt.*

Ergibt zum Beispiel im Statement

```
-C1=N1-N2+3
```

der numerische Ausdruck N1+N2+3 den Wert 7, so wird das Merkmal C1.7 auf 1 gesetzt und die übrigen Merkmale von C1 gelöscht. Liefert der Ausdruck den Wert Z0 für *keine Angabe* , eine Zahl kleiner 1 oder größer als das höchste Merkmal von C1, so werden alle Merkmale der Variablen gelöscht.

Mit dem Statement

```
-C1(5..7)=N1-N2+3
```

wird das Merkmal C1.5 auf 1 gesetzt, wenn der numerische Ausdruck das Ergebnis 1 liefert.
Die Merkmale 6 und 7 werden gelöscht und die übrigen Merkmale nicht verändert.

Entsprechend wird mit den übrigen Bedingungsfeldern auf der linken Seite des Gleichheitszeichens verfahren. Das Statement

```
-U1(5..8)=N1-N2+3
```

stellt zum Beispiel den Wert 0100 in das Ausgabefeld, wenn der numerische Ausdruck den Wert 2 erzeugt.

Zi Bedingung zur Prüfung der zu übertragenden Merkmale. Es bedeutet:

Z0 = alle Eingabemerkmale müssen gleich 0 sein,
Z1 = genau eines der Eingabemerkmale muss gleich 1 sein,
Z2 = genau zwei der Eingabemerkmale müssen gleich 1 sein,
Z3 = drei *oder mehr* Eingabemerkmale müssen gleich 1 sein.

Es sind auch kombinierte Prüfbedingungen möglich:

Z01 = entweder ist keines oder genau ein Merkmal erfüllt,
Z123 = mindestens ein Merkmal muss erfüllt sein usw.

Die Prüfungen finden statt, nachdem die Eingabewerte so ergänzt oder abgeschnitten wurden, dass sie in das Zielfeld passen.
Ist eine solche Prüfbedingung nicht erfüllt, so wird eine Fehlermeldung gedruckt. Die falschen Eingabewerte werden trotzdem in das Zielfeld gestellt. Durch die zusätzliche Angabe von SKE kann aber erreicht werden, dass die Zuweisung von falschen Eingabewerten unterbleibt.

SKB 'skip on blank'
Diese Angabe lässt das Zielfeld oder die Zielvariable unverändert, wenn der Eingabewert fehlt oder nur aus Leerzeichen besteht.

SKE 'skip on error'
Diese Angabe ergänzt die Prüfbedingungen Zi. Ist eine solche Bedingung nicht erfüllt, so sorgt SKE dafür, dass die fehlerhaften Daten nicht übertragen werden, das Zielfeld also seine alten Werte behält.

KKB 'keep key on blank'
Diese Angabe wirkt nur auf die Zielfelder MK und UK. Sie gibt auch für leere Eingabewerte Schlüssel aus. Für leere Werte werden ohne KKB keine Daten ausgegeben.
Besitzt zum Beispiel die Variable C7 in der Wertezuweisung

```
-MK1()=C7  KKB
```

keine Angabe, so wird dafür C7= ausgegeben.

BZ 'blank to zero'
Hinter einem numerischen Ausdruck sorgt die Angabe BZ dafür, dass für alle numerischen Variablen im Ausdruck der Wert Z0 durch Null ersetzt wird. Ohne BZ liefert ein solcher Ausdruck den Wert Z0, wenn nur eine der beteiligten Variablen Z0 enthält.
In dem Statement

```
-C1=(N10+N11)/2
```

erhält die Variable N1 bereits dann den Wert Z0, wenn N10 den Wert Z0 besitzt und N11 etwa den Wert 10. Durch Angabe von BZ hinter dem Eingabefeld wird der Wert Z0 in N10 als 0 verarbeitet, so dass als Ergebnis das Merkmal 5 der Variablen C1 gesetzt wird.

SH 'select high'
SL 'select low'
SR 'select at random'

Die zu übertragenden Merkmale werden auf Mehrfachnennungen hin untersucht.
Sind Mehrfachnennungen vorhanden, so werden diese bis auf eine entfernt.
Bei SH bleibt die höchste Merkmalsnummer erhalten,
bei SL die niedrigste und
bei SR wird eine der Nennungen zufällig ausgewählt.

Die Beseitigung von Mehrfachnennungen geschieht, nachdem die Werte aus dem Eingabefeld entnommen sind. Das Eingabefeld selbst wird dadurch nicht verändert.
Zum Beispiel werden durch das Statement

```
-C45=K1(10,12) SL
```

die Lochungen der Spalte 10 nicht unbesehen in die Variable C45 übertragen.
Zunächst erhalten alle Merkmale von C45 den Wert 0. Enthalten mehrere Positionen in Spalte 10 den Wert 1, so erhält nur das Merkmal von C45 den Wert 1, das der ersten dieser Positionen entspricht. Enthält die Spalte 10 keine Mehrfachnennungen, so bleibt die Angabe SL ohne Wirkung.

In dem Statement

```
-C115=111111  SR
```

sorgt die Angabe SR dafür, dass eine der sechs Einsen auf der rechten Seite zufällig ausgewählt wird. Nur dieses Merkmal der Variablen C115 erhält den Wert 1, alle übrigen den Wert 0.

Durch das Statement:

```
-PRT C115
```

wird die ausgewählte Merkmalsnummer 1 ... 6 im Verarbeitungsprotokoll ausgedruckt:
In diesem Beispiel haben wir damit einen Würfel simuliert.
An der Prüfung auf Mehrfachnennungen nehmen nur die Merkmale teil, die auch im Zielfeld Platz finden. Besitzt die Variable C115 im obigen Beispiel etwa nur 4 Merkmale, so wählt SR die verbleibende 1 nur unter den ersten vier Einsen aus.

Die zufällige Auswahl von Merkmalen erfolgt durch einen Zufallsgenerator, der in Verarbeitungsläufen mit gleichen Statements und Daten auch die gleichen Ergebnisse liefert.
Im Statement OPTIONS kann mit RSTART erreicht werden, dass in verschiedenen Verarbeitungsläufen unterschiedliche Zufallswerte verwendet werden.

INC= w Schrittweite w der Eingabefelder.
Diese Angabe dient dazu, in einem Statement mehreren C-Variablen Werte aus verschiedenen externen Datenfeldern vom Typ B, I, K, L, M, MS, U oder US zuzuweisen.
So überträgt das Statement

```
-C1...5=M1(1,4)    INC=10
```

Werte aus M-Feldern in die Variablen C1 ... C5.

Dabei erhält C1 den Wert aus dem Feld M1(1,4). Die Angabe INC=10 verschiebt die Startposition des Eingabefeldes um jeweils 10 Stellen. Die Variable C2 erhält ihren Wert aus dem Feld M1(11,4) usw. Sind nur die Variablen C1 und C5 definiert, so erhält C1 den Wert aus M1(1,4) und N5 den Wert aus M1(11,4).

Die Regeln zum Weiterstellen der Eingabefelder hängen vom Typ des Feldes ab:

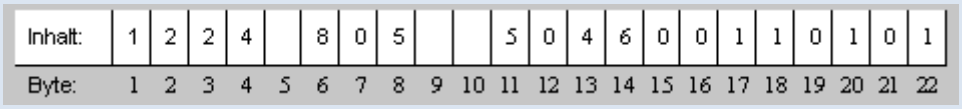
- für B-Felder wird das erste Eingabebit um die hier angegebene Anzahl w von Bits erhöht. Dabei werden gegebenenfalls auch mehrere Bytes übersprungen.
  Zum Beispiel füllt das Statement
  ```
  -C3..5=B1(6.1,8) INC=16
  ```
  zunächst die Variable C3 aus dem Eingabefeld B1(6.1,8). Danach wird C4 aus dem Feld B1(8.1,8) versorgt und C5 schließlich aus B1(10.1,8).
- für K-Felder wird die erste Eingabeposition um die angegebene Anzahl w von Lochpositionen erhöht. Dabei werden gegebenenfalls auch mehrere Spalten übersprungen. Zum Beispiel füllt das Statement
  ```
  -C3..5=K1(6.1,8) INC=24
  ```
  zunächst die Variable C3 aus dem Eingabefeld K1(6.1,8). Danach wird C4 bei einer COLBIN-Datei aus dem Feld K1(8.1,8) versorgt und C5 schließlich aus K1(10.1,8).
- für I-, L-, M- und U-Felder wird das Anfangsbyte oder die Anfangsspalte um die hier angegebene Anzahl w von Bytes oder Spalten erhöht.
  Zum Beispiel füllt das Statement
  ```
  -C3..5=M1(6.1,8,2) INC=8
  ```
  zunächst die Variable C3 aus dem Eingabefeld M1(6.1,8,2). Danach wird C4 aus dem Feld M1(14.1,8,2) versorgt und C5 schließlich aus M1(22.1,8,2).
- bei den MS- und US-Feldern der CSV-Dateien ist für w die Anzahl der zu überspringenden Separatorzeichen SEP anzugeben.

Die Schrittweite w und die Anzahl der Zielvariablen sind so zu wählen, dass das letzte zu übertragende Feld noch innerhalb des laufenden Datensatzes liegt.

Beispiel: ASCII-Daten

```
INPUT-EXT  FNAME=eingabe.ext  ASCII  ID=D(1,4)  SIZE=80,EOL
...
ADD-PROC
-C1=M1(6)
-C2=M1(7,2)
-C3=M1(5,4,2)
-C5=M1(5..16,2)
-C6=M1(5.45..16,2)
-C7=U1(17..22)
```

INPUT-EXT definiert eine externe Eingabedatei im ASCII-Format .
Ein Datensatz dieser Datei könnte folgendermaßen beginnen:



`SIZE=80, EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 80 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz.

`ID=D(1,4)`
Hiermit wird in den Bytes 1 bis 4 eines jeden Datensatzes eine numerische Fall-Identifikation angefordert. Im obigen Beispiel ist das die Nummer 1224.

`-C1=M1(6)`
Diese Wertezuweisung übernimmt den Wert 8 aus dem Byte 6 und setzt damit das Merkmal C1.8 auf 1. Die übrigen Merkmale der Variablen C1 werden auf 0 gelöscht.

`-C2=M1(7,2)`
Dieses Statement entnimmt den Wert 05 aus den zwei Spalten 7 und 8. Damit wird das Merkmal C2.5 auf 1 gesetzt und die übrigen Merkmale von C2 auf 0 gelöscht.

`-C3=M1(5,4,2)`
Hiermit werden Daten aus den vier Bytes 5 bis 8 entnommen und als zweistellige Werte verstanden, in diesem Beispiel also 8 und 05. Damit werden die beiden Merkmale C3.8 und C3.5 auf 1 gesetzt. Die übrigen Merkmale von C3 werden auf 0 gelöscht.

`-C5=M1(5..16,2)`
Mit dieser Wertezuweisung werden Daten aus den Bytes 5 bis 16 übernommen und als zweistellige Werte interpretiert. Daher werden die Merkmale C5.8, C5.5, C5.50, C5.46 und C5.100 auf 1 gesetzt: Der Wert 00 wird dabei als 100 verarbeitet.
Die übrigen Merkmale der Variablen C5 werden auf 0 gelöscht.

`-C6=M1(5.45..16,2)`
Dieses Statement arbeitet zunächst genauso wie das vorige. Die Angabe .45 sorgt dafür, dass die Eingabewerte ab 45 den Merkmalen C6.1, C6.2 ... zugewiesen werden. Alle Eingabewerte kleiner als 45 werden ignoriert. Daher werden die Merkmale C6.2, C6.6 und C6.56 auf 1 gesetzt und die übrigen auf 0 gelöscht..

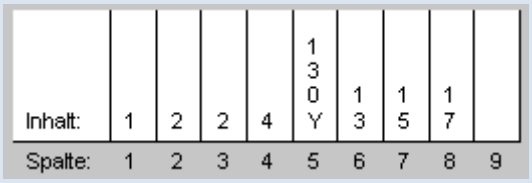
`-C7=U1(17..22)`
Diese Wertezuweisung entnimmt die Daten aus den Bytes 17 bis 22. Diese werden als Merkmalswerte 0 und 1 verstanden: Damit werden die Merkmale C7.1, C7.2, C7.4 sowie C7.6 auf 1 gesetzt und die übrigen auf 0 gelöscht.

Beispiel: COLBIN-Daten

```
INPUT-EXT  FNAME=eingabe.ext  COLBIN  SIZE=80  ID=D(1,4)
...
ADD-PROC
-C1=K1(5)
-C2=K1(5.8)
-C3=K1(7,24)
```

INPUT-EXT definiert eine externe Eingabedatei im COLBIN-Format (Column Binary).
In diesem Format kann jede Spalte die Werte 1 ...9, 0, X und Y enthalten oder leer sein.
Dabei wird 0 auch als Merkmal 10, X als Merkmal 11 und Y als Merkmal 12 verstanden. Das Besondere am COLBIN-Format ist, dass jede Spalte mehrere oder sogar alle dieser Angaben gleichzeitig enthalten kann.

Ein Datensatz dieser Datei könnte folgendermaßen beginnen:



`COLBIN  SIZE=80`
Diese Angaben im Statement INPUT-EXT legen das Datenformat Column Binary und die Satzlänge 80 fest. Das sind 80 Spalten, die Spalte zu je zwei Bytes.

`ID=D(1,4)`
Hiermit wird in den Spalten 1 bis 4 eines jeden Datensatzes eine numerische Fall-Identifikation angefordert. Im obigen Beispiel ist das die Nummer 1224.

`-C1=K1(5)`
Diese Wertezuweisung übernimmt die Angaben 1, 3, 0 und Y der Spalte 5 und stellt sie in die Variable C1: Der Wert 1 setzt das Merkmal C1.1, der Wert 3 das Merkmal C1.3 der Wert 0 das Merkmal C1.10 und der Wert Y das Merkmal C1.12. Die restlichen Merkmale der Variablen C1 werden gelöscht.

`-C2=K1(5 .8)`
Dieses Statement übernimmt ebenfalls Daten aus der Spalte 5 und stellt sie in die Variable C2. Die Angabe .8 sorgt dafür, dass der Wert 8 dem Merkmal C2.1 zugewiesen wird und der Wert 9 dem Merkmal C2.9 usw.

`-C3=K1(7, 24)`
Diese Wertezuweisung stellt die 24 möglichen Angaben aus den Spalten 7 und 8 in die Variable C3. Die Werte 1 und 5 aus Spalte 7 setzen die Merkmale C3.1 und C3.7.
Die Werte 1 und 7 aus Spalte 8 setzen die Merkmale C3.13 und C3.19.

Beispiel: Konstante, Variable, numerische Ausdrücke

```
...
ADD-PROC
-C1=C20
-C2(3..4)=C21(5,2)
-C3=N0
-C4=N1+N2+N3
-C5(5,3)=N3
-C6=101
-C7=°101
-M1(10,4)=N3+5
```

Diese Folgestatements von ADD-PROC zeigen, wie sich Merkmalswerte aus Variablen, numerischen Ausdrücken und Konstanten übertragen lassen.

`-C1=C20`
Dieses Statement überträgt sämtliche Merkmale der Variablen C20 in die Variable C1. Besitzt C1 mehr Merkmale als C20, so werden die höheren Merkmale von C1 auf 0 gelöscht.

`-C2(3..4)=C21(5,2)`
Hier werden zwei Merkmale der Variablen C21 in die Merkmale 3 und 4 der Variablen C2 übertragen: C21.5 in C2.3 und C21.6 in C2.4. Die übrigen Merkmale von C2 bleiben unverändert.

`-C3=N0`
Enthält die Variable N0 den Wert 7, so wird das Merkmal C3.7 auf 1 gesetzt und die übrigen Merkmale von C3 auf 0. Enthält N0 den Wert Z0 oder einen Wert außerhalb der für C3 definierten Merkmale, so werden alle Merkmale von C3 gelöscht.

`-C4=N1+N2+N3`
Dieses Statement bildet durch Addition der drei numerischen Variablen einen Merkmalswert, setzt das entsprechende Merkmal der Variablen C4 auf 1 und löscht alle übrigen Merkmale auf 0. Liefert der Ausdruck den Wert Z0 oder einen Wert außerhalb der für C4 definierten Merkmale, so werden alle Merkmale von C4 gelöscht.

`-C5(5,3)=N3`
Wenn die Variable N3 in diesem Statement den Wert 1 besitzt, erhält das Merkmal C5.5 den Wert 1. Die Merkmale 6 und 7 werden gelöscht und die übrigen Merkmale von N3 nicht verändert.

`-C6=101`
Dies Statement setzt stets das Merkmal C6.101 auf 1 und löscht die sonstigen Merkmale von C6.

`-C7=°101`
Das Zeichen ° sorgt dafür, dass die Konstante 101 als Kette von Werten 0 und 1 verstanden wird. Abweichend vom vorigen Statement werden dadurch die Merkmale C7.1 und C7.3 auf 1 gesetzt und die übrigen Merkmale von C7 auf 0.

`-M1(10,4)=N3+5`
Mit diesem Statement wird das Ergebnis des numerischen Ausdrucks N3 + 5 in die Datensätze von INPUT-EXT gestellt. Dies geschieht im M-Format, jeweils in den ersten Datensatz jedes Falles in die Positionen 10 bis 13. In den Wertezuweisungen hinter diesem Statement kann auf diese Daten zugegriffen werden, so als wären sie aus der Eingabedatei eingelesen worden.

Beispiel: CSV-Daten einlesen

```
INPUT-EXT  FNAME=eingabe.ext  ID=DS(1)  SIZE=1000,EOL
SEP=';'
...
ADD-PROC
-C1=MS1(3..6)
-C2=MS1(12)
-C3=US1(8,4)
-C4=US1(13)
```

INPUT-EXT definiert eine externe Eingabedatei als CSV-Datei mit dem Semikolon als Separatorzeichen. Ein Datensatz dieser Eingabedatei könnte folgendermaßen aussehen:

```
0001;1;5;7;8;;1-3 9;1;1;0;1;;100101; ...
```

`SIZE=1000, EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 1000 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz. Kürzere Datensätze werden mit Leerzeichen auf 1000 Bytes aufgefüllt.

`ID=DS(1)`
Wegen dieser Angabe in INPUT-EXT wird eine Fall-Identifikation als numerischer Wert im ersten Datenfeld der Eingabesätze erwartet, also vor dem ersten Semikolon.
In diesem Beispiel ist das der Wert 0001.

`-C1=MS1(3..6)`
Diese Wertezuweisung entnimmt die Merkmalsnummern aus den Feldern 3 bis 6 und stellt sie in die Variable C1. Da mehrere Eingabefelder angegeben sind, werden Mehrfachnennungen in verschiedenen, durch Semikolon getrennten Feldern erwartet. Hier werden die Merkmale C1.5, C1.7 und C1.8 auf 1 gesetzt und die übrigen auf 0.

`-C2=MS1(12)`
Da in diesem Statement nur das eine Eingabefeld 12 angegeben ist, sind Mehrfachnennungen hier in der Form 1-3 und 9 möglich: Die Merkmale C2.1, C2.2; C2.3 sowie C2.9 werden auf 1 gesetzt und die übrigen auf 0.

`-C3=US1(8, 4)`
Dieses Statement sucht in den Feldern 7 bis 12 die Werte 1 für erfüllte und 0 für nicht erfüllte Merkmale und überträgt sie in die Variable C3.
Da mehrere Eingabefelder angegeben sind, werden Mehrfachnennungen in verschiedenen, durch Semikolon getrennten Feldern erwartet. Hier werden die Merkmale C3.1, C3.2 und C3.4 auf 1 gesetzt und die anderen auf 0.

`-C4=US1(13)`
Da in diesem Statement nur das eine Eingabefeld 13 angegeben ist, sind Mehrfachnennungen hier in der Form 100101 in einem Datenfeld möglich: Die Merkmale C4.1, C4.4 sowie C4.6 werden auf 1 gesetzt und die übrigen auf 0.

Beispiel: CSV-Daten einlesen

```
INPUT-EXT  FNAME=eingabe.ext  ID=DS(1)  SIZE=1000,EOL  SEP=';'
...
ADD-PROC
-C1=MS1(3..6)
-C2=MS1(12)
-C3=US1(8,4)
-C4=US1(13)
```

INPUT-EXT definiert eine externe Eingabedatei als CSV-Datei mit dem Semikolon als Separatorzeichen. Ein Datensatz dieser Eingabedatei könnte folgendermaßen aussehen:

```
0001;1;5;7;8;;1-3 9;1;1;0;1;;100101; ...
```

`SIZE=1000, EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 1000 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz. Kürzere Datensätze werden mit Leerzeichen auf 1000 Bytes aufgefüllt.

`ID=DS(1)`
Wegen dieser Angabe in INPUT-EXT wird eine Fall-Identifikation als numerischer Wert im ersten Datenfeld der Eingabesätze erwartet, also vor dem ersten Semikolon.
In diesem Beispiel ist das der Wert 0001.

`-C1=MS1(3..6)`
Diese Wertezuweisung entnimmt die Merkmalsnummern aus den Feldern 3 bis 6 und stellt sie in die Variable C1. Da mehrere Eingabefelder angegeben sind, werden Mehrfachnennungen in verschiedenen, durch Semikolon getrennten Feldern erwartet. Hier werden die Merkmale C1.5, C1.7 und C1.8 auf 1 gesetzt und die übrigen auf 0.

`-C2=MS1(12)`
Da in diesem Statement nur das eine Eingabefeld 12 angegeben ist, sind Mehrfachnennungen hier in der Form 1-3 und 9 möglich: Die Merkmale C2.1, C2.2; C2.3 sowie C2.9 werden auf 1 gesetzt und die übrigen auf 0.

`-C3=US1(8,4)`
Dieses Statement sucht in den Feldern 7 bis 12 die Werte 1 für erfüllte und 0 für nicht erfüllte Merkmale und überträgt sie in die Variable C3.
Da mehrere Eingabefelder angegeben sind, werden Mehrfachnennungen in verschiedenen, durch Semikolon getrennten Feldern erwartet. Hier werden die Merkmale C3.1, C3.2 und C3.4 auf 1 gesetzt und die anderen auf 0.

`-C4=US1(13)`
Da in diesem Statement nur das eine Eingabefeld 13 angegeben ist, sind Mehrfachnennungen hier in der Form 100101 in einem Datenfeld möglich: Die Merkmale C4.1, C4.4 sowie C4.6 werden auf 1 gesetzt und die übrigen auf 0.

Beispiel: CSV-Daten mit Schlüsselfeldern

```
INPUT-EXT   FNAME=eingabe.ext     SIZE=1000,EOL    ID=DK( )    SEP=' ; '
...
ADD-PROC
-C3=MK1()
-C4=MK1(frage4)
-C5=MK1(frage4(3))
-C6=UK1(frage5)
```

INPUT-EXT definiert eine externe Eingabedatei als CSV-Datei mit dem Semikolon als Separatorzeichen. Ein Datensatz dieser Eingabedatei könnte folgendermaßen aussehen:

```
c3=1-9; id=0001; frage4=1 3-5 10; frage5=100101; ...
```

`SIZE=1000, EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 1000 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz. Kürzere Datensätze werden mit Leerzeichen auf 1000 Bytes aufgefüllt.

`ID=DK()`
Wegen dieser Angabe in INPUT-EXT wird in allen Datensätzen eine Fall-Identifikation als numerischer Wert mit einem Schlüssel davor erwartet. Da in den Klammen hinter DK kein Schlüssel angegeben ist, wird *id* als Schlüssel verwendet. Zwischen Groß- und Kleinschrift wird dabei nicht unterschieden. Die Reihenfolge der Schlüsselfelder in den Datensätzen ist beliebig. In diesem Beispiel ist der Wert 0001 die Fall-Identifikation.

`-C3=MK1()`
Mit diesem Statement werden die Merkmale der Variablen C3 neu gesetzt.
Da in den Klammern hinter MK kein Schlüssel angegeben ist, wird *c3* als Schlüssel verwendet. Im MK-Format werden immer Mehrfachnennungen in dem gefundenen Feld akzeptiert. Hier enthält der Datensatz die Werte 1-3 9. Entsprechend werden die Merkmale C3.1, C3.2; C3.3 sowie C3.9 auf 1 gesetzt und die übrigen Merkmale auf 0.

`-C4=MK1(frage4)`
Diese Wertezuweisung sucht Merkmalsnummern im Datensatz mit dem Schlüssel *frage4* davor. Im obigen Beispiel sind dies die Angaben 1 3 bis 5 und 10.
Die entsprechenden Merkmale der Variablen C4 werden auf 1 gesetzt, die übrigen auf 0.

`-C5=MK1(frage4(3))`
Dies Statement sucht seine Datenwerte ebenfalls hinter dem Schlüssel *frage4*, jedoch wird nur die dritte Nennung, also das Merkmal 4 eingelesen und der Variablen C5 zugewiesen.

`-C6=UK1(frage5)`
Dieses Statement erwartet seine Daten in einem Feld mit dem Schlüssel *frage5*. Als Datenwerte dienen die Ziffern 1 für erfüllte und 0 für nicht erfüllte Merkmale. In diesem Fall setzt die Angabe 100101 die Merkmale C6.1, C6.4 und C6.6 auf 1 setzen. Die übrigen Merkmale erhalten den Wert 0.

Beispiel: CSV-Daten mit Schlüsseln im ersten Datensatz

```
INPUT-EXT  FNAME=eingabe.ext  ID=DK()  SIZE=1000,EOL  KEYLINE  SEP=';'
...
ADD-PROC
-C3=MK1()
-C4=MK1(frage4)
-C5=MK1(frage4(3))
-C6=UK1(frage5)
```

INPUT-EXT definiert eine externe Eingabedatei als CSV-Datei mit dem Semikolon als Separatorzeichen. Diese Datei könnte mit den beiden folgenden Datensätzen beginnen:

```
   c3;   id;   frage4; frage5; ...
1-3 9; 0001; 10 3-5 1; 100101; ...
```

Die Angabe KEYLINE im Statement INPUT-EXT verlangt, dass der erste Datensatz der Datei Schlüsselbegriffe enthält. Die eigentlichen Daten befinden sich in den folgenden Datensätzen in der gleichen Spalte wie der dazugehörige Schlüssel.

`ID=DK()`
Diese Angabe in INPUT-EXT verlangt in allen Datensätzen eine Fall-Identifikation als numerischen Wert. Da in den Klammen hinter DK kein Schlüssel angegeben ist, wird *id* als Schlüssel verwendet. Zwischen Groß- und Kleinschrift wird dabei nicht unterschieden. Im obigen Beispiel ist der Wert 0001 die Fall-Identifikation.

`SIZE=1000, EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 1000 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz. Kürzere Datensätze werden mit Leerzeichen auf 1000 Bytes aufgefüllt.

`-C3=MK1()`
Mit diesem Statement werden die Merkmale der Variablen C3 neu gesetzt.
Da in den Klammern hinter MK kein Schlüssel angegeben ist, wird c*3* als Schlüssel verwendet. Im MK-Format werden immer Mehrfachnennungen in dem gefundenen Feld akzeptiert. Hier enthält der Datensatz die Werte 1-3 9. Entsprechend werden die Merkmale C3.1, C3.2; C3.3 sowie C3.9 auf 1 gesetzt und die übrigen Merkmale auf 0.

`-C4=MK1(frage4)`
Diese Wertezuweisung sucht Merkmalsnummern im Datensatz unter dem Schlüssel *frage4* . Im obigen Beispiel sind dies die Angaben 1 3 bis 5 und 10.
Die entsprechenden Merkmale der Variablen C4 werden auf 1 gesetzt, die übrigen auf 0.

`-C5=MK1(frage4(3))`
Dieses Statement sucht seine Datenwerte ebenfalls unter dem Schlüssel *frage4,* jedoch wird nur die dritte Nennung, also das Merkmal 4 eingelesen und der Variablen C5 zugewiesen.

`-C6=UK1(frage5)`
Dieses Statement erwartet seine Daten unter dem Schlüssel *frage5*. Im U-Format stehen die Ziffern 1 für erfüllte und 0 für nicht erfüllte Merkmale. In diesem Fall setzt die Angabe 100101 die Merkmale C6.1, C6.4 und C6.6 auf 1. Die übrigen Merkmale erhalten den Wert 0.

Beispiel: Binärdatei

```
INPUT-EXT  FNAME=eingabe.ext  BINARY1  SIZE=80  ID=G(1,2)
...
ADD-PROC
-C1=B1(3)
-C2=B1(3.5)
-C3=B1(3,12)
-C4=I1(5..12)
-C5=I1(5.127..6)
```

INPUT-EXT definiert eine externe Binärdatei, bei der die Bytes wegen BINARY1 einzeln angesprochenen werden können. Bei BINARY2 bilden jeweils zwei Bytes eine Einheit. Ein Datensatz dieser Datei könnte folgendermaßen beginnen:

Inhalt:	11573	10101000	10010000	127	255	1	0	18	0	0	255
Byte:	1 2	3	4	5	6	7	8	9	10	11	12

`ID=G(1,2)`
Diese Angabe in INPUT-EXT verlangt in allen Datensätzen eine Fall-Identifikation in den beiden ersten Bytes als Dualzahl. Im obigen Beispiel ist dies der Wert 11573.

`-C1=B1(3)`
Dies Statement nimmt seine Datenwerte aus Byte 3 der Eingabesätze in Form einzelner Bits. Jedes Byte enthält daher 8 Werte 0 *oder* 1. In diesem Beispiel werden die Merkmale C1.1, C1.3 und C1.5 auf 1 gesetzt und die übrige Merkmale von C1 auf 0 gelöscht.

`-C2=B1(3.5)`
Diese Wertezuweisung arbeitet zunächst genauso wie die vorige. Die Eingabe beginnt jedoch nicht beim ersten sondern beim fünften Bit. Daher wird in diesem Beispiel nur das Merkmal C2.1 auf 1 gesetzt.

`-C3=B1(3,12)`
Dies Statement liest 12 Merkmalswerte 0 und 1 ein, beginnend im Byte 3 bei Bit 1 und endend im Byte 4 beim vierten Bit. In diesem Beispiel werden die Merkmale C3.1, C3.3, C3.5, C3.9 und C3.12 auf 1 gesetzt und die übrige Merkmale von C3 auf 0 gelöscht.

`-C4=I1(5..12)`
Diese Wertezuweisung entnimmt die Daten aus den Bytes 5 bis 12 der Eingabesätze im I-Format. Dabei enthält jedes Byte eine Dualzahl ohne Vorzeichen (Binärzahl, unsigned integer). Jedes Byte kann die Merkmalsnummern 1 bis 255 enthalten. Der Wert 0 wird als *keine Angabe* verstanden. Im obigen Beispiel werden die Merkmale C4.1, C4.18, C4.127 und C4.255 auf 1 gesetzt und die übrigen Merkmale auf 0 gelöscht.

`-C5=I1(5.127..6)`
Dies Statement sucht seine Datenwerte in den Bytes 5 und 6 der Eingabesätze.
Die Angabe .127 sorgt dafür, dass der Wert 127 dem Merkmal C5.1 zugewiesen wird und der Wert 255 dem Merkmal C2.129. Die sonstigen Merkmale werden auf 0 gelöscht.

Beispiel: CSV-Daten ausgeben

```
OUTPUT-EXT  FNAME=ausgabe.ext  ASCII  SEP=';'  SIZE=1000,EOL  ID=DS1(1)
...
-MS1(2)=C7
-US(3)=C8
-MS1(4)=N1+N2+N3
-MS1(5)=N10
-MK1()=C10
-MK1(fr5)=37
-UK1(fr6)=°101
```

OUTPUT-EXT definiert eine externe Ausgabedatei im ASCII-Format.
Durch die Angabe SEP=';' entsteht eine CSV-Datei mit dem Separatorzeichen Semikolon zwischen den Datenfeldern.

Die obigen Wertezuweisungen könnten folgenden Datensatz in diese Datei ausgeben:

```
1124; 1 3-4 10; 100101; 24; ; fr5=37; c10=7 9; fr6=101; ...
```

`SIZE=1000,EOL`
Diese Angabe im Statement OUTPUT-EXT erzeugt Datensätze von maximal 1000 Bytes Länge. EOL stellt ein Zeilenende-Zeichen hinter jeden Datensatz. Leerzeichen am Ende der Sätze werden entfernt, um Speicherplatz zu sparen.

`ID=DS(1)`
Hiermit wird die Fall-Identifikation des laufenden Falles in das erste Feld des Datensatzes ausgegeben. Im obigen Beispiel ist das die Nummer 1224.

`-MS1(2)=C7`
Diese Wertezuweisung stellt die Nummern der Merkmal der Variablen C7, die den Wert 1 besitzen, in das zweite Datenfeld der Ausgabesätze. Im obigen Beispiel besitzen die Merkmale C7.1, C7.3, C7.4 und C7.10 den Wert 1 und die übrigen Merkmale der Variablen den Wert 0.

`-US(3)=C8`
Dieses Statement stellt eine Zeichenkette aus den Werten 0 und 1 der Merkmale der Variablen C8 in das dritte Feld der Ausgabesätze. Im obigen Beispiel besitzen die Merkmale C8.1, C8.4 und C8.6 den Wert 1 und die übrigen den Wert 0.

`-MS1(4)=N1+N2+N3`
In diesem Statement wird zunächst das Ergebnis des Ausdrucks N1+N2+N3 errechnet und als Merkmalsnummer in das dritte Feld der Ausgabesätze gestellt. Hier ist das der Wert 24.

`-MS11(5)=N10`
Hier enthält die Variable N10 eine ungültige Merkmalsnummer (Wert kleiner 1, größer 9999 oder Z0). Dafür wird das fünfte Feld des Datensatzes leer ausgegeben.

`-MK1(fr5)=37`
Diese Wertezuweisung gibt die Merkmalsnummer 37 in das erste noch freie Datenfeld der Datensätze aus, mit dem Schlüssel *fr5=* davor.

`-MK1()=C10`
Mit dieser Wertezuweisung werden die Merkmale mit dem Wert 1 der Variablen C10 in das nächste noch freie Feld der Datensätze gestellt. Da in den Klammern ( ) kein Schlüssel angegeben ist, wird dafür *c10=* verwendet.

`-UK1(fr6)=°101`
Diese Wertezuweisung gibt eine Kette aus Merkmalswerten 0 und 1 in das nächste noch freie Feld der Datensätze aus, mit dem Schlüssel *fr6=* davor.

Abkürzungen

Ein Statement kann gleichzeitig mehreren C-Variablen Werte zuweisen. Durch

```
-C1...5=0
```

werden alle Merkmale der Variablen C1 ... C5 auf 0 gelöscht. Dabei müssen nicht alle Variable zwischen C1 und C5 definiert sein. Die nicht definierten werden einfach ausgelassen.

Im vorigen Beispiel wird allen Variablen der gleiche Wert zugewiesen. Sie können aber auch unterschiedliche Werte erhalten, etwa mit dem Statement:

```
-C1...3=M1(1,2)  INC=4
```

Hier stellt die Angabe INC die Eingabeposition nach jeder Wertezuweisung um vier Stellen weiter. Dieses Statement ist daher gleichwertig mit den folgenden drei Statements:

```
-C1=M1(1,2)
-C2=M1(5,2)
-C3=M1(9,2)
```

Zahlen

Numerische Wertezuweisungen sind Folgestatements zu ADD-PROC, UPD-PROC, MOD-PROC und OUTPUT-EXT. Mit ihrer Hilfe können zu jedem statistischen Fall numerische Werte übertragen werden.
Dies sind positive oder negative Zahlen, die aus maximal neun Dezimalziffern bestehen und bis zu acht Nachkommastellen besitzen können. Zusätzlich gibt es den besonderen Zahlenwert Z0 für keine Angabe.

Der Eingabewert einer solchen Zuweisung kann aus einem Datensatz stammen, der von INPUT-EXT eingelesen wurde, er kann aus einer N-Variablen kommen, die von INPUT-INT eingelesen oder während der laufenden Verarbeitung definiert wurde, er kann während der Verarbeitung durch numerische Ausdrücke errechnet werden aber auch eine konstante Zahl sein, die in den Statements festgelegt ist.

Eine Zuweisung kann einen Wert in eine N-Variable stellen, sie kann eine Zahl in eine externe Datei OUTPUT-EXT ausgeben, und sie kann schließlich einen Wert in einen Datensatz aus INPUT-EXT stellen und dabei die gerade eingelesenen Daten überschreiben. Dieses Überschreiben erfolgt aber nur für die laufende Verarbeitung. Die Datei selbst wird dadurch nicht verändert.

Weitere Angaben zu numerischen Werten finden sich in den Abschnitten *Variable* bei den N-Variablen, *Numerische Ausdrücke* und *Externe Datenfelder*.

Bei der numerischen Wertezuweisung gelten immer folgende Regeln:
Besitzt die zu übertragende Zahl mehr Nachkommastellen als das Zielfeld, so wird der Wert passend für das Zielfeld gerundet.
Ist das Zielfeld auf der linken Seite des Gleichheitszeichens zu klein, um den Wert aufzunehmen, so erscheint eine Fehlermeldung im Zählprotokoll hinter der Statementliste. Es findet dann keine Übertragung statt, das Zielfeld wird nicht verändert.

Folgende Wertezuweisungen sind für numerische Daten möglich:

–Nm	=	k
–Dz ... <d>		numerischer Ausdruck < p > < **BZ** > < **SKB** > < **KKB** >
–DKz ...<d>		
–DSz ...<d>		**N**n < p > < **BZ** > < **SKB** > < **KKB** >
–Fz ... <d>		**T**n < p > < **BZ** > < **SKB** > < **KKB** >
–Gz ... <d>		**ID**i < p >
–Pz ... <d>		
		Dr ...< k > < p > <**INC=**w> < **BZ** > < **SKB** > < **KKB** >
		DKr ...< k > < p > <**INC=**w> < **BZ** > < **SKB** > < **KKB** >
		DSr ...< k > < p > <**INC=**w>< **BZ** > < **SKB** > < **KKB** >
		Fr ...< k > < p > <**INC=**w>
		Gr ...< k > < p > <**INC=**w>
		Pr ...< k > < p > <**INC=**w>

Hierbei bedeutet:

–**N**m Numerische Variable, die einen neuen Wert erhalten soll.

–**D**z –**DK**z –**DS**z –**F**z –**G**z –**P**z
Diesen externen Datenfeldern links vom Gleichheitszeichen wird für jeden statistischen Fall ein neuer Wert zugewiesen. Dies ist nur in den Folgestatements von ADD-PROC, UPD-PROC und OUTPUT-EXT möglich. Gehört die Wertezuweisung zu OUTPUT-EXT, so ist das Zielfeld ein Datenfeld der externen Ausgabedatei. Liegt sie dagegen hinter ADD-PROC oder UPD-PROC, so ist es ein Feld in dem zuletzt aus INPUT-EXT gelesenen statistischen Fall. Die gerade eingelesenen Daten werden dann durch die Wertezuweisung überschrieben, die Eingabedaten selbst aber nicht verändert.

Hinter OUTPUT-EXT sorgt zum Beispiel das Statement

```
-D2(3..4) = N15
```

dafür, dass für jeden statistischen Fall der Wert der Variablen N15 in die Bytes 3 bis 4 des zweiten Datensatzes der auszugebenden statistischen Fälle gestellt wird.
Enthält N15 den Wert Z0 für *keine Angabe*, so wird das Zielfeld mit Leerzeichen gefüllt.
Ein Wert der nicht in das Zielfeld passt nicht übertragen. Dazu werden die Fehlermeldungen 507 oder 522 ausgegeben.

d Anzahl 0 ... 18 Nachkommastellen im Zielfeld.
Numerische Werte werden mit ihren Nachkommastellen und wenn möglich mit Dezimalkomma oder Dezimalpunkt in das Zielfeld gestellt.
Die Angabe *d* hinter dem Zielfeld bewirkt, dass *d* Nachkommastellen im Zielfeld erscheinen.
Besitzt der auszugebende Wert mehr Nachkommastellen als hier gefordert, so wird er gerundet.
Besitzt er weniger, so werden rechts Nullen angefügt.
Im Statement OUTPUT-EXT beeinflusst die Angabe DSTD die Ausgabe der Nachkommastellen.
Zum Beispiel sorgt die Wertezuweisung

```
-D1(20,6)2=N12
```

hinter OUTPUT-EXT dafür, dass für jeden statistischen Fall der Wert aus N12 mit zwei Nachkommastellen in das D-Feld übertragen wird.
Enthält die Variable N12 zum Beispiel die Zahl 3.145, so wird diese als 315 in das Zielfeld gestellt.
Enthält N12 den Wert 3, so wird die Zahl 300 in das D-Feld ausgegeben.

Dr ...
DKr ...
DSr ...
Fr ...
Gr ...
Pr ...

Einem externen Datenfeld rechts vom Gleichheitszeichens wird für jeden statistischen Fall ein Wert entnommen und in die Variable oder das Zielfeld gestellt.
Dies ist nur in den Folgestatements von ADD-PROC, UPD-PROC und OUTPUT-EXT möglich.
Diese Datenfelder stammen stets aus dem zuletzt aus INPUT-EXT gelesenen statistischen Fall.
Hinter ADD-PROC, UPD-PROC und OUTPUT-EXT sorgt zum Beispiel das Statement

```
-N15=D2(3..6)
```

dafür, dass aus dem zweiten Datensatz jedes aus INPUT-EXT gelesenen Falles die Zahl aus den Bytes 3 bis 6 in die numerische Variable N15 gestellt wird.
Besteht das D-Feld nur aus Leerzeichen, so erhält die Variable N15 den Wert Z0 für *keine Angabe*.
Enthält das D-Feld keinen gültigen numerischen Wert, so erscheint eine Fehlermeldung im Zählprotokoll hinter der Statementliste. Es findet dann keine Übertragung statt, das Zielfeld wird nicht geändert.

k Anzahl 0 ... 18 Nachkommastellen im Eingabefeld.
Diese Angabe legt fest, wie viele der Ziffern im Eingabewert als Nachkommastellen zu verstehen sind. Enthält ein Wert im D-Feld bereits ein Dezimalkomma oder einen Dezimalpunkt, so bleibt die Angabe e wirkungslos. Zum Beispiel sorgt die Wertezuweisung

```
-N12=D1(20,6)2
```

dafür, dass die beiden rechten Ziffern jedes Eingabewertes als Nachkommastellen verarbeitet werden.
Enthält das D-Feld die Zahl 354, so wird der Wert 3.54 in N12 gestellt. Enthält das D-Feld dagegen die Zahl 35.4 oder 35,4 so haben die Dezimalzeichen im Eingabefeld Vorrang vor der Angabe e: Der Variablen N12 wird der Wert 35.4 zugewiesen.

Nn Der auszugebende Wert wird der numerischen Variablen Nn entnommen.
Die Nachkommastellen werden dabei automatisch angepasst.
Wenn die Variable N100 den Wert 6.5 enthält und die Variable N20 ohne Nachkommastellen definiert wurde, so stellt das Statement

```
-N20=N100
```

den gerundeten Wert 7 in die Variable N20. Besitzt N20 dagegen zwei Nachkommastellen, so erhält sie den Wert 6.50.

Tn Der auszugebende Wert wird der Textvariablen Tn entnommen.
Die Werte in der Textvariablen müssen dafür den gleichen Regeln folgen wie in D-Feldern.
Zum Beispiel ist

```
26.3
```

ein gültiger numerischer Wert, nicht jedoch

```
26.3%
```

numerischer Ausdruck

Wie im Abschnitt *Numerische Ausdrücke* beschrieben, liefert ein solcher Ausdruck einen numerischen Wert, der anschließend in die Variable oder das Zielfeld auf der linken Seite des Gleichheitszeichens gestellt wird.
Zum Beispiel wertet das Statement

```
-D1(102...105)=(N112 + N120)/2
```

zunächst den numerischen Ausdruck auf der rechten Seite des Gleichheitszeichen aus, und stellt danach das Ergebnis in das Zielfeld. Wenn das Ergebnis des Ausdrucks Nachkommastellen besitzt, werden diese mit Dezimalpunkt oder Dezimalkomma ausgegeben.

k Numerische Konstante, die für jeden statistischen Fall in das Zielfeld zu stellen ist.
Eine numerische Konstante darf maximal neun Dezimalziffern besitzen und bis zu acht Nachkommastellen. Sie kann das Vorzeichen - und einen Dezimalpunkt enthalten.
Zulässig ist auch der Wert Z0 für *keine Angabe*.

IDj Fall-Identifikation ID ... ID4, deren Wert in das Zielfeld zu stellen ist.
(Siehe Angabe IDj= ... im Statement INPUT-EXT).

p eine oder zwei Prüfbedingungen der Form:

```
<v  >v  =v  <=v  >=v  ^=v  ^<v  ^>v
```

mit konstanten numerischen Vergleichswerten v.
Zunächst wird der Eingabewert, falls nötig, gerundet, so dass seine Nachkommastellen in das Zielfeld passen. Danach wird er mit dem Wert v verglichen. Ist eine solche Prüfbedingung nicht erfüllt, so druckt CNTA eine Fehlermeldung, die den falschen Wert enthält.
Dieser Wert wird aber trotzdem in das Zielfeld gestellt.
Zum Beispiel prüft das Statement

```
-N14=D2(22,6)  >0  <=100
```

zunächst den Wert aus dem D-Feld. Er muss größer 0 und gleichzeitig kleiner oder gleich 100 sein.
Enthält das D-Feld den Wert 112, so erscheint im Verarbeitungsprotokoll die Meldung:

```
FEHLER 508: FALSCHER WERT: D2(22,6)='112'
```

zusammen mit der dazugehörigen Fall-Identifikation.

BZ 'blank to zero'
Diese Angabe bewirkt, dass der Eingabewert Z0 (*keine Angabe* oder Leerzeichen) als Null übertragen wird. Ohne BZ bleiben diese Sonderwerte erhalten.
Zum Beispiel stellt das Statement

```
-N112=D1(5,4) BZ
```

eine Null in die Variable N112, wenn das D-Feld nur Leerzeichen enthält.
Ohne die Angabe BZ würde N112 für ein leeres D-Feld den Wert Z0 erhalten.

Hinter einem numerischen Ausdruck sorgt die Angabe BZ dafür, dass für alle numerischen Variablen im Ausdruck der Wert Z0 durch Null ersetzt wird. Ohne BZ liefert ein solcher Ausdruck den Wert Z0, wenn nur eine der beteiligten Variablen Z0 enthält.
In dem Statement

```
-N1=(N10+N11)/2
```

erhält die Variable N1 bereits dann den Wert Z0, wenn N10 den Wert Z0 besitzt und N11 etwa den Wert 10. Durch Angabe von BZ hinter dem Eingabefeld wird der Wert Z0 in N10 als 0 verarbeitet, so dass als Ergebnis die Zahl 5 in die Variable N1 gestellt wird.

SKB 'skip on blank'
Diese Angabe lässt das Zielfeld oder die Zielvariable unverändert, wenn der Eingabewert nur aus Leerzeichen besteht oder nicht vorhanden ist.

KKB 'keep key on blank'
Diese Angabe wirkt nur auf die Zielfelder DK. Sie gibt auch für leere Eingabewerte Schlüssel aus. Für leere Werte werden ohne KKB keine Daten ausgegeben.
Besitzt zum Beispiel die Variable N7 in der Wertezuweisung

```
-DK1()=N7   KKB
```

den Wert Z0 für *keine Angabe,* so wird dafür N7= ausgegeben.

INC=w Schrittweite w der Eingabefelder.
Diese Angabe erlaubt, mit einem Statement mehreren N-Variablen Werte aus verschiedenen externen Feldern vom Typ D, DS, F, G und P zuzuweisen. So überträgt das Statement

```
-N1...5=D1(1,4)    INC=10
```

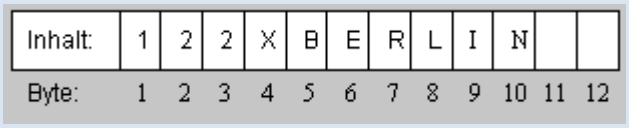
Werte aus D-Feldern in die Variablen N1 ... N5.

Dabei erhält N1 den Wert aus dem Feld D1(1,4). Die Angabe INC=10 verschiebt die Startposition des Eingabefeldes um jeweils 10 Stellen. Die Variable N2 erhält ihren Wert aus dem Feld D1(11,4) usw. Sind nur die Variablen N1 und N5 definiert, so erhält N1 den Wert aus D1(1,4) und N5 den Wert aus D1(11,4).
Die Schrittweite w und die Anzahl der Zielvariablen sind so zu wählen, dass das letzte zu übertragende Feld noch innerhalb des laufenden Datensatzes liegt.
Bei CSV-Dateien ist für w die Anzahl der zu überspringenden Separatorzeichen SEP anzugeben.

Beispiel: ASCII-Daten

```
INPUT-EXT  FNAME=eingabe.ext  ASCII  ID=D(1,4)  SIZE=80,EOL
...
ADD-PROC
-N1=D1(5,4)
-N2=D1(5..8)2
-N5=D1(5)
-N3=D1(9..12)
```

INPUT-EXT definiert eine externe Eingabedatei im ASCII-Format .
Ein Datensatz dieser Datei könnte folgendermaßen beginnen:



`SIZE=80,EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 80 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz.

`ID=D(1,4)`
Hiermit wird in den Bytes 1 bis 4 eines jeden Datensatzes eine numerische Fall-Identifikation angefordert. Im obigen Beispiel ist das die Nummer 1224.

`-N1=D1(5,4)`
Diese Wertezuweisung nimmt den Wert 805 aus den Bytes 5 bis 8 des Eingabesatzes und stellt ihn in die Variable N1. Wurde die Variable N1 mit einer Nachkommastelle definiert, so wird der Eingabewert vorher auf 805,0 erweitert. Ist das Eingabefeld leer, so erhält die Variable N1 den Wert Z0 für *keine Angabe.*

`-N2=D1(5..8)2`
Dieses Statement nimmt den Wert 805 aus den Bytes 5 bis 8 des Eingabesatzes.
Wegen der 2 in dieser Angabe, wird diese Zahl als 8,05 in die Variable N2 gestellt.
Wurde die Variable N2 ohne Nachkommastellen definiert, so wird der Eingabewert vorher auf 8 gerundet.

`-N5=D1(5)`
Hier wird der Wert aus Byte 5 eingelesen. Da dies leer ist, erhält die Variable N5 den Wert Z0 für *keine Angabe.* Mit der zusätzlichen Angabe BZ (Blank to Zero) wird erreicht, dass N5 stattdessen den Wert 0 erhält:

```
      -N5=D1(5)   BZ
```

`-N3=D1(9 . . 12)`
Hiermit wird der Wert -5,6 aus den Bytes 9 bis 12 des Eingabesatzes in die Variable N3 übertragen. Wurde die Variable N3 ohne Nachkommastellen definiert, so wird der Eingabewert vorher auf -6 aufgerundet.

Beispiel: Konstante, Variable, numerische Ausdrücke

```
ADD-PROC
-N1=N20
-N2=N10+N11+N12
-N3=ID
-N4=T22
-N5=17.5
-N6=Z0
```

Diese Folgestatements von ADD-PROC zeigen, wie sich Zahlenwerte aus Variablen, numerischen Ausdrücken und Konstanten übertragen lassen.

`-N1=N20`
Dieses Statement überträgt die Werte der Variablen N20 in die Variable N1.
Besitzt N20 mehr Nachkommastellen als N1, so werden die Werte durch Rundung an die Zielvariable angepaßt. Ist der Wert von N20 zu groß für N1, so wird eine Fehlermeldung in das Zählprotokoll ausgegeben.

`-N2=N10+N11+N12`
Hier wird zunächst der numerische Ausdruck auf der rechten Seite des Gleichheitszeichens ausgewertet und das Ergebnis in die Variable N2 übertragen.
Bei der Auswertung des Ausdrucks wird stets mit einer Genauigkeit von mindestens 16 Nachkommastellen gerechnet. Das Ergebnis wird erst beim Übertragen in die Variable N2 passend gerundet.
Besitzt auch nur eine der Variablen rechts vom Gleichheitszeichen den Wert Z0 für *keine Angabe*, so erhält N2 ebenfalls den Wert Z0 zugewiesen.
Mit der zusätzlichen Angabe BZ (Blank to Zero) erreicht man, dass die Werte Z0 wie 0 behandelt werden:

```
     -N5=D1(5)   BZ
```

`-N3=ID`
Diese Wertezuweisung überträgt die Fall-Identifikation ID des laufenden Falles in die Variable N3. Ist der Wert der Fall-Identifikation ID nicht numerisch oder zu groß für die Zielvariable, so wird eine Fehlermeldung in das Zählprotokoll ausgegeben.

`-N4=T22`
Hiermit wird der Text der Variablen T22 in die numerische Variable N4 übertragen. Ist der Wert von T22 nicht numerisch oder zu groß für die Zielvariable, so wird eine Fehlermeldung in das Zählprotokoll ausgegeben.

`-N5=17.5`
Dieses Statement stellt den konstanten Wert 17,5 in die Variable N5.

`-N6=Z0`
Hiermit erhält die Variable N6 den Wert Z0 für *keine Angabe.*

Beispiel: CSV-Daten einlesen

```
INPUT-EXT  FNAME=eingabe.ext  ID=DS(1)  SIZE=1000,EOL  SEP=';'
...
ADD-PROC
-N1=DS1(2)
-N2=DS1(2) 2
-N3=DS1(3)
-N4=DS1(4)
```

INPUT-EXT definiert eine externe Eingabedatei als CSV-Datei mit dem Semikolon als Separatorzeichen. Ein Datensatz dieser Eingabedatei könnte folgendermaßen aussehen:

```
0001; 805; ; -5,6; ...
```

`SIZE=1000, EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 1000 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz. Kürzere Datensätze werden mit Leerzeichen auf 1000 Bytes aufgefüllt.

`ID=DS(1)`
Wegen dieser Angabe in INPUT-EXT wird eine Fall-Identifikation als numerischer Wert im ersten Datenfeld der Eingabesätze erwartet, also vor dem ersten Semikolon.
In diesem Beispiel ist das der Wert 0001.

`-N1=DS1(2)`
Diese Wertezuweisung überträgt den Wert 805 aus dem zweiten Feld des Eingabesatzes in die Variable N1. Wurde die Variable N1 mit einer Nachkommastelle definiert, so wird der Eingabewert vorher auf 805,0 erweitert.

`-N2=DS1(2) 2`
Dieses Statement nimmt den Wert 805 aus den Bytes 5 bis 8 des Eingabesatzes.
Wegen der zusätzlichen 2 in dieser Angabe, wird diese Zahl als 8,05 in die Variable N2 gestellt. Wurde die Variable N2 ohne Nachkommastellen definiert, so wird der Eingabewert vorher auf 8 gerundet.

`-N3=DS1(3)`
Hier ist das Eingabefeld leer. Die Variable N3 erhält den Wert Z0 für *keine Angabe.*
Die zusätzliche Angabe BZ (Blank to Zero)

```
      -N3=DS1(4) BZ
```

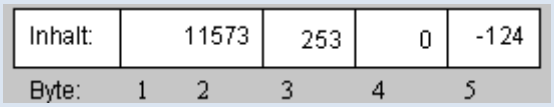
bewirkt, dass N3 stattdessen den Wert 0 erhält.

`-N4=DS1(4)`
Hiermit wird der Wert -5,6 aus den Bytes 9 bis 12 des Eingabesatzes in die Variable N4 übertragen. Wurde die Variable N4 ohne Nachkommastellen definiert, so wird der Eingabewert vorher auf -6 aufgerundet.

Beispiel: Binärdatei

```
INPUT-EXT     FNAME=eingabe.ext    BINARY1    SIZE=80     ID=G(1,2)
...
ADD-PROC
-N1=G1(3)
-N2=G1(3) 2
-N3=G1(4)
-N4=F1(5)
```

INPUT-EXT definiert hie eine externe Binärdatei, bei der die Bytes wegen BINARY1 einzeln angesprochenen werden können. Bei BINARY2 bilden jeweils zwei Bytes eine Einheit. Ein Datensatz dieser Datei könnte folgendermaßen beginnen:



Inhalt:	11573	253	0	-124
Byte:	1 2	3	4	5

ID=G(1,2)
Diese Angabe in INPUT-EXT verlangt in allen Datensätzen eine Fall-Identifikation in den beiden ersten Bytes als Dualzahl. Im obigen Beispiel ist dies der Wert 11 573.

`-N1=G1(3)`
Mit diesem Statement erhält die Variable N1 aus Byte 3 den Wert 253.
Wurde die Variable N1 mit einer Nachkommastelle definiert, so wird der Eingabewert vorher auf 253,0 erweitert.

`-N2=G1(3) 2`
Hiermit wird der gleiche Wert eingelesen wie im vorigen Statement.
Wegen der zusätzlichen 2 in dieser Angabe, wird diese Zahl als 2,53 in die Variable N2 gestellt. Wurde die Variable N2 ohne Nachkommastellen definiert, so wird der Eingabewert vorher auf 3 gerundet.

`-N3=G1(4)`
Diese Wertezuweisung stellt den Wert 0 aus Byte 4 in die Variable N3. In F- und G-Feldern gibt es den Wert Z0 für *keine Angabe* nicht.

`-F5=F1(5)`
Dies Statement überträgt den Wert -124 aus Byte 5 in die Variable N4.
Wenn das Datenfeld einer Binärdatei negative Werte enthalten kann, ist stets das F-Format zu verwenden. Das G-Format würde anstelle des negativen Wertes einen größeren positiven einlesen.

Beispiel: CSV-Daten mit Schlüsselfeldern

```
INPUT-EXT  FNAME=eingabe.ext  SIZE=1000,EOL  ID=DK()  SEP=';'
...
ADD-PROC
-N1=DK1()
-N2=DK1(N1)2
-N3=DK1(frage3)
-N4=DK1(frage5)
```

INPUT-EXT definiert hier eine externe Eingabedatei als CSV-Datei mit dem Semikolon als Separatorzeichen. Ein Datensatz dieser Eingabedatei könnte folgendermaßen aussehen:

```
n1=805; id=0001; frage3= ; frage5=-5,6; ...
```

`SIZE=1000,EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 1000 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz. Kürzere Datensätze werden mit Leerzeichen auf 1000 Bytes aufgefüllt.

`ID=DK()`
Wegen dieser Angabe in INPUT-EXT wird in allen Datensätzen eine Fall-Identifikation als numerischer Wert mit einem Schlüssel davor erwartet. Da in den Klammen hinter DK kein Schlüssel angegeben ist, wird *id* als Schlüssel verwendet. Zwischen Groß- und Kleinschrift wird dabei nicht unterschieden. Die Reihenfolge der Schlüsselfelder in den Datensätzen ist beliebig. In diesem Beispiel ist der Wert 0001 die Fall-Identifikation.

`-N1=DK1()`
Mit diesem Statement erhält die Variable N1 einen neuen Wert. Da in den Klammern hinter DK kein Schlüssel angegeben ist, wird *n1* als Schlüssel verwendet.
Diese Wertezuweisung überträgt den Wert 805 aus dem Eingabesatz in die Variable N1. Wurde die Variable N1 mit einer Nachkommastelle definiert, so wird der Eingabewert vorher auf 805,0 erweitert.

`-N2=DK1(N1)2`
Hiermit wird der gleiche Wert eingelesen wie im vorigen Statement.
Wegen der zusätzlichen 2 in dieser Angabe, wird diese Zahl als 8,05 in die Variable N2 gestellt. Wurde die Variable N2 ohne Nachkommastellen definiert, so wird der Eingabewert vorher auf 8 gerundet.

`-N3=DK1(frage3)`
Hier ist das Eingabefeld leer. Die Variable N3 erhält den Wert Z0 für *keine Angabe.*
Die zusätzliche Angabe BZ (Blank to Zero)

```
-N3=DK1(frage3) BZ
```

bewirkt, dass N3 stattdessen den Wert 0 erhält.

`-N4=DK1(frage5)`
Dieses Statement findet seinen Datenwert -5,6 hinter dem Schlüssel *frage5* und stellt ihn in die Variable N4. Wurde N4 ohne Nachkommastellen definiert, so wird der Eingabewert vorher auf -6 aufgerundet.

Beispiel: CSV-Daten ausgeben

```
OUTPUT-EXT  FNAME=ausgabe.ext  ASCII  SEP=';'  SIZE=1000,EOL
ID=DS1(1)
...
-DS1(2)=N3
-DS1(3) 2=N4
-DS1(4)=N1+N2+N3
-DS1(5)=N10
-DK1( )=N3
-DK1(fr5)=37
```

OUTPUT-EXT definiert eine externe Ausgabedatei im ASCII-Format.
Durch die Angabe SEP=';' entsteht eine CSV-Datei mit dem Separatorzeichen Semikolon zwischen den Datenfeldern.

Die obigen Wertezuweisungen könnten folgenden Datensatz in diese Datei ausgeben:

```
1124; 805; 127; 24,7; ; N3=805; fr5=37; ...
```

`SIZE=1000,EOL`
Diese Angabe im Statement OUTPUT-EXT erzeugt Datensätze von maximal 1000 Bytes Länge. EOL stellt ein Zeilenende-Zeichen hinter jedem Datensatz. Leerzeichen am Ende der Sätze werden abgeschnitten um Speicherplatz zu sparen.

`ID=DS(1)`
Hiermit wird in die Bytes 1 bis 4 eines jeden Datensatzes die Fall-Identifikation des laufenden Falles ausgegeben. Im obigen Beispiel ist das die Nummer 1224.

`-DS1(2)=N3`
Diese Wertezuweisung stellt den Wert der Variablen N3 in das zweite Datenfeld der Ausgabesätze. Hier ist das die Zahl 805.

`-DS1(3) 2=N4`
Hier enthält die Variable N4 den Wert 1,27 mit zwei Nachkommastellen. Dieser wird wegen der Angabe 2 links vom Gleichheitszeichen als 127 in das dritte Feld des Datensatzes gestellt.

`-DS1(4)=N1+N2+N3`
In diesem Statement wird zunächst das Ergebnis des Ausdrucks N1+N2+N3 errechnet und in das vierte Feld der Ausgabesätze gestellt. Hier ist das der Wert 24,7. Die Anzahl ausgegebener Nachkommastellen hängt ab von der Variablen auf der rechten Seite des Gleichheitszeichens, die die größte Anzahl Nachkommastellen besitzt.

`-DS1(5)=N10`
Hier enthält die Variable N10 den Wert Z0 für keine Angabe. Daher bleibt das fünfte Ausgabefeld leer.

`-DK1( )=N3`
Diese Wertezuweisung stellt den Wert 805 der Variablen N3 in das nächste freie Ausgabefeld mit einem Schlüssel davor. Da in den Klammern ( ) kein Schlüssel angegeben ist, wird dafür *N3* verwendet.

`-DK1(fr5)=37`
Dieses Statement überträgt den konstanten Wert 37 in das nächste freie Feld des Ausgabesatzes und stellt den Schlüssel *fr5* davor.

Beispiel: CSV-Daten mit Schlüssel im ersten Datensatz

```
INPUT-EXT  FNAME=eingabe.ext  ID=DK()  SIZE=1000,EOL  KEYLINE  SEP=';'
...
ADD-PROC
-N1=DK1()
-N2=DK1(N1)2
-N3=DK1(frage3)
-N4=DK1(frage5)
```

INPUT-EXT definiert hier eine externe Eingabedatei als CSV-Datei mit dem Semikolon als Separatorzeichen. Diese Datei könnte mit den beiden folgenden Datensätzen beginnen:

```
 n1;   id; frage3; frage5; ...
805; 0001;       ;   -5.6; ...
```

Die Angabe KEYLINE im Statement INPUT-EXT verlangt, dass der erste Datensatz der Datei Schlüsselbegriffe enthält. Die eigentlichen Daten befinden sich in den folgenden Datensätzen in der gleichen Spalte wie der dazugehörige Schlüssel.

`ID=DK( )`
Diese Angabe in INPUT-EXT verlangt in allen Datensätzen eine Fall-Identifikation als numerischen Wert. Da in den Klammen hinter DK kein Schlüssel angegeben ist, wird *id* als Schlüssel verwendet. Zwischen Groß- und Kleinschrift wird dabei nicht unterschieden. Im obigen Beispiel ist der Wert 0001 die Fall-Identifikation.

`SIZE=1000,EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 1000 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz. Kürzere Datensätze werden mit Leerzeichen auf 1000 Bytes aufgefüllt.

`-N1=DK1()`
Mit diesem Statement erhält die Variable N1 einen neuen Wert. Da in den Klammern hinter DK kein Schlüssel angegeben ist, wird *n1* als Schlüssel verwendet.
Diese Wertezuweisung überträgt den Wert 805 aus der ersten Eingabespalte in die Variable N1. Wurde die Variable N1 mit einer Nachkommastelle definiert, so wird der Eingabewert vorher auf 805,0 erweitert.

`-N2=DK1(N1) 2`
Hiermit wird der gleiche Wert eingelesen wie im vorigen Statement.
Wegen der zusätzlichen 2 in dieser Angabe, wird diese Zahl als 8,05 in die Variable N2 gestellt. Wurde die Variable N2 ohne Nachkommastellen definiert, so wird der Eingabewert vorher auf 8 gerundet.

`-N3=DK1(frage3)`
Hier ist das Eingabefeld leer. Die Variable N3 erhält den Wert Z0 für *keine Angabe.*
Die zusätzliche Angabe BZ (Blank to Zero)

```
      -N3=DK1(frage3) BZ
```

bewirkt, dass N3 stattdessen den Wert 0 erhält.

`-N4=DK1(frage5)`
Dieses Statement findet seinen Datenwert -5,6 unter dem Schlüssel *frage5* und stellt ihn in die Variable N4. Wurde N4 ohne Nachkommastellen definiert, so wird der Eingabewert vorher auf -6 aufgerundet.

Abkürzungen

Ein Statement kann gleichzeitig mehreren N-Variablen Werte zuweisen. Durch

```
-N1...5=0
```

werden die Variablen N1 bis N5 mit dem Wert 0 gefüllt. Dabei müssen nicht alle Variable zwischen N1 und N5 definiert sein. Die nicht definierten werden einfach ausgelassen.

Im vorigen Beispiel wird allen Variablen der gleiche Wert zugewiesen. Sie können aber auch unterschiedliche Werte erhalten, etwa mit dem Statement:

```
-N1...3=D1(1,4)   INC=4
```

Hier stellt die Angabe INC die Eingabeposition nach jeder Wertezuweisung um vier Stellen weiter. Dieses Statement ist daher gleichwertig mit den folgenden drei Statements:

```
-N1=D1(1,4)
-N2=D1(5,4)
-N3=D1(9,4)
```

Texte

Wertezuweisungen von Texten sind Folgestatements zu ADD-PROC, UPD-PROC, MOD-PROC und OUTPUT-EXT. Durch sie lassen sich für jeden statistischen Fall Textwerte in Textvariable oder externe Datenfelder vom Typ A, AS oder AK übertragen.

Der Eingabetext einer solchen Wertezuweisung kann aus Datensätzen von INPUT-EXT stammen oder aus T-Variablen von INPUT-INT. Es kann aber auch ein in den Statements direkt angegebener Text sein sowie der Variablen- oder Merkmalstext einer beliebigen Variablen.

Eine solche Wertezuweisung kann sogar Texte in A-, AK- oder AS-Felder von Datensätzen aus INPUT-EXT stellen und dabei die gerade eingelesenen Daten überschreiben. Dieses Überschreiben erfolgt aber nur für die laufende Verarbeitung. Die Datei selbst wird dadurch nicht verändert.

Weitere Angaben zu Textwerten enthalten die Abschnitte Textzeilen und Externe Datenfelder.

Bei der Wertezuweisung von Texten gelten immer folgende Regeln:
Ist das Zielfeld auf der linken Seite des Gleichheitszeichens kürzer als der zu übertragende Text, so wird dieser rechts abgeschnitten. Dazu erscheint keine Fehlermeldung.
Ist das Zielfeld dagegen länger als der Text, so werden rechts Leerzeichen hinzugefügt.
Mit den unten beschriebenen Justagezeichen können diese Verarbeitungsregeln geändert werden.

Folgende Wertezuweisungen sind für Texte möglich:

–**T**m	=	**'**zzz...**'**				<**KKB**>	<**LOW**>	<**UP**>
–**A**...	j =							
–**AS**...		**T**n	<**SKB**>	<**SKB**>		<**KKB**>	<**LOW**>	<**UP**>
–**AK**...		**ID**i	<**SKB**>	<**SKB**>		<**KKB**>	<**LOW**>	<**UP**>
		Nn	<**SKB**>	<**SKB**>		<**KKB**>	<**LOW**>	<**UP**>
		A...	<**SKB**>	<**SKB**>	<**INC=**w>	<**KKB**>	<**LOW**>	<**UP**>
		AS...	<**SKB**>	<**SKB**>	<**INC=**w>	<**KKB**>	<**LOW**>	<**UP**>
		AK...	<**SKB**>	<**SKB**>		<**KKB**>	<**LOW**>	<**UP**>
		UNDO(...**)**				<**KKB**>	<**LOW**>	<**UP**>

Hierbei bedeutet:

= Textübertragung ohne Justage.
Der Eingabetext wird unverändert in das Zielfeld gestellt. Ist er länger als das Zielfeld, wird er rechts abgeschnitten, ist er kürzer, werden rechts Leerzeichen hinzugefügt.

j **=** Textübertragung mit Justage.
Aus dem Eingabetext werden zunächst alle Leerzeichen vor und hinter dem eigentlichen Text entfernt. Ist der verbleibende Rest länger als das Zielfeld, so wird er rechts abgeschnitten und in das Zielfeld gestellt. Eine Fehlermeldung erscheint nicht dazu.
Ist der verbleibende Rest kürzer als das Zielfeld, so hängt die Verarbeitung von dem hier angegebenen Justagezeichen ab:

- L= der Textwert wird linksbündig in das Zielfeld gestellt.
- R= der Textwert wird rechtsbündig in das Zielfeld gestellt.
- M= der Textwert wird in die Mitte des Zielfeldes gestellt.

Der Rest des Zielfeldes wird mit Leerzeichen aufgefüllt.

Das Justagezeichen L kann insbesondere benutzt werden, um führende Leerzeichen aus dem Eingabewert zu entfernen.
Bei Ausgabe in AK- und AS-Felder von CSV-Dateien ist diese Textjustage nicht möglich.

Ar ...
AKr ...
ASr ... Aus einem externen Datenfeld rechts vom Gleichheitszeichen wird für jeden statistischen Fall der zu übertragende Text entnommen. Dies ist immer ein Datenfeld in dem aus der Eingabedatei INPUT-EXT eingelesenen statistischen Falles.
Der Inhalt des Datenfeldes wird durch die Übertragung nicht verändert.

Zum Beispiel sorgt die Angabe:

```
-T7=A2(1,80)
```

hinter ADD-PROC dafür, dass aus dem zweiten Datensatz eines jeden eingelesenen statistischen Falles der Text aus den Positionen 1 ... 80 gelesen und in die Variable T7 gestellt wird.

Bei der Eingabe aus einer CSV-Datei sorgt die Angabe:

```
-T7=AS2(1)
```

hinter ADD-PROC dafür, dass aus dem zweiten Datensatz eines jeden eingelesenen statistischen Falles der Text vor dem ersten Separatorzeichen SEP in die Variable T7 gestellt wird.

Az ...
AKz ...
ASz ... Dem externen Datenfeld links vom Gleichheitszeichen wird für jeden statistischen Fall ein neuer Wert zugewiesen. Dies ist nur in den Folgestatements von ADD-PROC, UPD-PROC und OUTPUT-EXT möglich.
Liegt die Wertezuweisung hinter OUTPUT-EXT, so ist Az, AKz oder ASz ein Datenfeld der externen Ausgabedatei. Liegt sie dagegen hinter ADD-PROC oder UPD-PROC, so ist es ein Feld des aus INPUT-EXT gelesenen statistischen Falles.
Die gerade eingelesenen Daten werden durch die Wertezuweisung überschrieben, die Eingabedatei selbst aber nicht verändert.

Zum Beispiel sorgt die Angabe:

```
-A2(1,80)=T7
```

hinter OUTPUT-EXT dafür, dass der Inhalt der Textvariablen T7 in den zweiten Datensatz eines jeden statistischen Falles der Ausgabedatei an die Position 1 ... 80 gestellt wird. Der Wert aus T7 wird linksbündig in das A-Feld gestellt. Wurde die Variable T7 mit weniger als 80 Positionen definiert, so wird ihr Wert rechts mit Leerzeichen aufgefüllt. Ist T7 dagegen länger als 80 Stellen, so wird der Wert rechts abgeschnitten.

Bei der Ausgabe in eine CSV-Datei sorgt die Angabe:

```
-AS2(3)=T7
```

hinter OUTPUT-EXT dafür, dass der Inhalt der Textvariablen T7 in den zweiten Datensatz eines jeden statistischen Falles der Ausgabedatei hinter das zweite Separatorzeichen SEP gestellt wird. Führende Leerzeichen des Ausgabewertes bleiben erhalten, hintere Leerzeichen werden rechts abgeschnitten.

IDk Fall-Identifikation ID ... ID4.
Der Wert dieser Fall-Identifikation wird für jeden statistischen Fall in das Zielfeld gestellt. Diese Angabe ist nur möglich, wenn die Fall-Identifikation vom alphanumerischen Typ ist. *Siehe Angabe IDk=A ... im Statement INPUT-EXT.*

KKB 'keep key on blank'
Diese Angabe wirkt nur auf die Zielfelder AK. Sie gibt auch für leere Eingabewerte Schlüssel aus.
Für leere Werte werden ohne KKB keine Daten ausgegeben.
Enthält zum Beispiel die Variable T7 in der Wertezuweisung

```
-AK1()=T7  KKB
```

keinen Text, so wird dafür T7= ausgegeben.

SKB 'skip on blank'
Diese Angabe lässt das Zielfeld oder die Zielvariable unverändert, wenn der Eingabewert fehlt oder nur aus Leerzeichen besteht.

INC=w Schrittweite der Eingabefelder.
Mit dieser Angabe können mehreren T-Variablen mit einem Statement Werte aus verschiedenen externen Feldern vom Typ A und AS zugewiesen werden. So überträgt das Statement

```
-T1...5=A1(1,10)    INC=20
```

Werte aus A-Feldern in die Variablen T1 ... T5.

Dabei erhält T1 den Wert aus dem Feld A1(1,10). Die Angabe INC=20 verschiebt die Startposition des Eingabefeldes danach um 20 Stellen. Die Variable T2 erhält ihren Wert aus dem Feld A1(21,10) usw. Sind nur die Variablen T1 und T5 definiert, so erhält T1 den Wert aus A1(1,10) und T5 den Wert aus A1(21,10).
Die Schrittweite w und die Anzahl der Zielvariablen sind so zu wählen, dass das letzte zu übertragende Feld noch innerhalb des laufenden Datensatzes liegt.
Bei CSV-Dateien ist für w die Anzahl der zu überspringenden Separatorzeichen SEP anzugeben.

Tm Textvariable, die einen neuen Wert erhalten soll. Zum Beispiel füllt die Angabe

```
-T20=' '
```

die ganze Textvariable T20 für jeden statistischen Fall mit Leerzeichen. Der alte Inhalt von T20 geht dabei verloren.

Das Statement

```
-T20='FORD'
```

stellt den Text FORD in die Positionen 1 bis 4 der Variablen T20. Der Rest der Variablen wird mit Leerzeichen gefüllt.

Es können auch einzelne Positionen einer T-Variablen verändert werden. Das Statement

```
-T20(3...7)='+'
```

stellt das Pluszeichen + in die Position 3 der Variablen T20. Die Positionen 4 bis 7 werden mit Leerzeichen gefüllt. Der Rest der Variablen bleibt unverändert.

Das Statement

```
-T20(R2)=A1(1,10)
```

stellt den Text aus dem A-Feld nicht in eine feste Position der Variablen T20 sondern relativ an das Ende des alten Wertes. Die Angabe 2 in R2 sorgt für zwei Leerzeichen zwischen dem alten und neuen Text. Enthält die Variable T20 etwa vor der Übertragung den Text

```
ERNST MÜLLER
```

und das A-Feld den Text

```
BERLIN
```

so enthält die Variable T20 nach der Übertragung den Text:

```
ERNST MÜLLER  BERLIN
```

Für diese relativen Positionsangaben sind die Werte R0 ... R3999 möglich. Insbesondere stellt R0 den neuen Text ohne Zwischenraum direkt hinter den alten.

Mit dem Statement

```
-T20(R2,1)=A1(1,10)
```

wird nur das erste Zeichen aus dem A-Feld in die Variable T20 übernommen.

In einem Statement können auch mehrere Textvariable neue Werte erhalten. So füllt das Statement

```
-T100..150=' '
```

die Variablen T100 bis T150 mit Leerzeichen. Dieses Statement kann auch dann verwendet werden, wenn nicht alle Textvariable zwischen T100 und T150 definiert sind.

Nn Der in der Variablen Nn gespeicherte numerische Wert wird als Text aufbereitet und danach in das Ausgabefeld gestellt. Diese Aufbereitung erfolgt genau wie für D-Felder.

Tn Der Textwert wird der Variablen Tn entnommen.
Zum Beispiel stellt die Angabe

```
-A1(1..12)=T30
```

den Textwert aus der Variablen T30 für jeden statistischen Fall in das A-Feld eines externen Datensatzes. Der alte Inhalt dieses A-Feldes geht dabei verloren.

Es besteht auch die Möglichkeit, nicht den ganzen Wert der Variablen Tn zu übertragen, sondern nur einige Zeichen. Durch das Statement

```
-A1(1..12)=T30(3...7)
```

wird nur der Text aus den Positionen 3 bis 7 der Variablen T30 entnommen und in das A-Feld gestellt.

'text' Einzeiliger Text von 1 ... 4 000 Zeichen Länge, die in Hochkommas eingeschlossen sind.
Dieser feste Text wird für jeden statistischen Fall in das Ausgabefeld gestellt.
Zum Beispiel füllt die Angabe

```
-T100='*******'
```

die Textvariable T100 mit sieben Sternen und den Rest mit Leerzeichen.

UNDO(...**)** Die Funktion UNDO aus dem Abschnitt Textzeilen übertragt Variablentexte, Merkmalstexte oder Texte von Textelementen. Wird die Variable C10 folgendermaßen definiert:

```
-C10:5 'Alter'/
       'des Befragten'
         1='18 bis 25 Jahre'
```

so stellt

```
-A1(1..20)=UNDO(C10.1)
```

den Text

```
Alter des Befragen
```

in das Zielfeld A1(1..20). Mit der Angabe

```
-A1(1..20)=UNDO(C10.1)
```

wird der Text

```
18 bis 25 Jahre
```

in die Ausgabedatei gestellt.

UP Der Text wird im Zielfeld in Großschrift übersetzt. ä, ö, ü werden zu Ä, Ö, Ü. ß bleibt unverändert.

LOW Der Text wird im Zielfeld in Großschrift übersetzt. ä, ö, ü werden zu Ä, Ö, Ü. ß bleibt unverändert.

Beispiel: ASCII-Daten

```
INPUT-EXT  FNAME=eingabe.ext  ASCII  ID=A(1,4)  SIZE=80,EOL
...
ADD-PROC
-T1=A1(5..12)
-T2 R =A1(5..12)
-T3(10..20)=A1(5..12)
```

INPUT-EXT definiert eine externe Eingabedatei im ASCII-Format .
Ein Datensatz dieser Datei könnte folgendermaßen beginnen:

Inhalt:	1	2	2	X	B	E	R	L	I	N		
Byte:	1	2	3	4	5	6	7	8	9	10	11	12

`SIZE=80,EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 80 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz.

`ID=A(1,4)`
Hiermit wird in den Bytes 1 bis 4 eines jeden Datensatzes ein Textwert als Fall-Identifikation angefordert. Im obigen Beispiel ist das die Angabe *122A*.

`-T1=A1(5..12)`
Diese Wertezuweisung überträgt den Text *BERLIN* aus den Bytes 5 bis 12 des Datensatzes in die Variable T1. Der Text wird linksbündig in der Variablen ausgerichtet.

`-T2 R =A1(5 . . 12)`
Dieses Statement leistet das Gleiche wie das vorige, jedoch wird der Text wegen der Angabe rechtsbündig in der Variablen T2 gespeichert.

`-T3(10 . . 20)=A1(5 . . 12)`
Diese Wertezuweisung stellt den Text BERLIN linksbündig in die Positionen 10 bis 20 der Variablen T3.

Beispiel: Konstante und Variable

```
...
ADD-PROC
-T1=T20
-T2='Heidelberg'
-T3=ID
```

Diese Folgestatements von ADD-PROC zeigen, wie sich Texte aus Variablen, übertragen lassen.

`-T1=T20`
Dieses Statement überträgt die Werte der Variablen T20 in die Variable T1.
Besitzt T20 längere Texte, als T1 aufnehmen kann, so werden diese rechts abgeschnitten. Ist die Variable T1 länger als T20, so wird rechts mit Leerzeichen aufgefüllt.

`-T2='Heidelberg'`
Hier wird der feste Text *Heidelberg* für alle statistischen Fälle in die Variable T2 gestellt.

`-T3=ID`
Diese Wertezuweisung überträgt die Fall-Identifikation ID des laufenden Falles in die Variable T3.

Beispiel: CSV-Daten einlesen

```
INPUT-EXT  FNAME=eingabe.ext  ID=AS(1)  SIZE=1000,EOL  SEP=';'
...
ADD-PROC
-T1=AS1(2)
-T2=AK1(name)
```

INPUT-EXT definiert eine externe Eingabedatei als CSV-Datei mit dem Semikolon als Separatorzeichen. Ein Datensatz dieser Eingabedatei könnte folgendermaßen aussehen:

```
0001A; Berlin; name=Schmidt; ...
```

`SIZE=1000,EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 1000 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz. Kürzere Datensätze werden mit Leerzeichen auf 1000 Bytes aufgefüllt.

`ID=AS(1)`
Wegen dieser Angabe in INPUT-EXT wird eine Fall-Identifikation als Textwert im ersten Datenfeld der Eingabesätze erwartet, also vor dem ersten Semikolon.
In diesem Beispiel ist das der Wert 0001A.

`-T1=AS1(2)`
Diese Wertezuweisung überträgt den Text *Berlin* aus dem zweiten Feld des Eingabesatzes in die Variable T1.

`-T2=AK1(name)`
Dieses Statement sucht in dem Datensatz nach dem Schlüssel *name=* und überträgt den dahinter stehenden Text *Schmidt* in die Variable T2.

Beispiel: CSV-Daten ausgeben

```
OUTPUT-EXT  FNAME=ausgabe.ext  SEP=';','"'  SIZE=1000,EOL  ID=AS1(1)
-AS1(2)=T1
-AK1(name)=T2
```

OUTPUT-EXT erzeugt die externe Ausgabedatei *ausgabe.ext.* Die obigen Wertezuweisungen könnten folgenden Datensatz in diese Datei ausgeben:

```
1124X;"Berlin";name="Schmidt";...
```

`SEP=';','"'`
Diese Angabe im Statement OUTPUT-EXT erzeugt eine CSV-Datei mit dem Separatorzeichen Semikolon zwischen den Datenfeldern sowie dem Textbegrenzungszeichen ". Ein solches Textbegrenzungszeichen ist nur erforderlich, wenn Texte selbst ein Semikolon enthalten.

`SIZE=1000, EOL`
Diese Angabe im Statement OUTPUT-EXT erzeugt Datensätze von maximal 1000 Bytes Länge. EOL stellt ein Zeilenende-Zeichen hinter jedem Datensatz. Leerzeichen am Ende der Sätze werden abgeschnitten um Speicherplatz zu sparen.

`ID=AS(1)`
Hiermit wird in das erste Datenfeld eines jeden Datensatzes die Fall-Identifikation des laufenden Falles ausgegeben. Im obigen Beispiel ist das der Text 1224X.

`-AS1(2)=T1`
Diese Wertezuweisung stellt den Inhalt *Berlin* der Variablen T1 in das zweite Datenfeld der Ausgabesätze. Wegen des Textbegrenzungszeichens " aus der SEP-Angabe wird der Ausgabewert in Hochkommas eingeschlossen.

`-AK1(name3)=T2`
Mit diesem Statement wird der Text *Schmidt* aus der Variablen T2 in das nächste freie Datenfeld des Ausgabesatzes gestellt, mit dem Schlüssel *name=* davor.

Beispiel: CSV-Daten mit Schlüssel im ersten Datensatz

```
INPUT-EXT  FNAME=eingabe.ext  ID=AK()  SIZE=1000,EOL  KEYLINE
SEP=';'
...
ADD-PROC
-T1=AK1()
-T2=AK1(name)
```

INPUT-EXT definiert eine externe Eingabedatei als CSV-Datei mit dem Semikolon als Separatorzeichen. Diese Datei könnte mit den beiden folgenden Datensätzen beginnen:

```
   t1;    id;    name; ...
Berlin; 0001A; Schmidt; ...
```

Die Angabe KEYLINE im Statement INPUT-EXT verlangt, dass der erste Datensatz der Datei Schlüsselbegriffe enthält. Die eigentlichen Daten befinden sich in den folgenden Datensätzen in der gleichen Spalte wie der dazugehörige Schlüssel.

`ID=AK()`
Diese Angabe in INPUT-EXT verlangt in allen Datensätzen eine Fall-Identifikation als Textwert. Da in den Klammen hinter AK kein Schlüssel angegeben ist, wird *id* als Schlüssel verwendet. Zwischen Groß- und Kleinschrift wird dabei nicht unterschieden. Im obigen Beispiel ist der Wert *0001A* die Fall-Identifikation.

`SIZE=1000, EOL`
Diese Angabe im Statement INPUT-EXT beschränkt die Länge der Datensätze auf maximal 1000 Bytes. EOL verlangt ein Zeilenende-Zeichen hinter jedem Datensatz. Kürzere Datensätze werden mit Leerzeichen auf 1000 Bytes aufgefüllt.

`-T1=AK1( )`
Dieses Statement überträgt den Text *Berlin* aus der Spalte mit der Überschrift *t1* in die Variable T1.

`-T2=AK1(name)`
Hiermit wird der der Text *Schmidt* aus der Spalte mit der Überschrift *name* in die Variable T2 übertragen.

Abkürzungen

Ein Statement kann gleichzeitig mehreren N-Variablen Werte zuweisen. Durch

```
-T1...5=' '
```

werden die Variablen T1 bis T5 mit Leerzeichen gefüllt. Dabei müssen nicht alle Variable zwischen T1 und T5 definiert sein. Die nicht definierten werden einfach ausgelassen.

Im vorigen Beispiel wird allen Variablen der gleiche Wert zugewiesen. Sie können aber auch unterschiedliche Werte erhalten, etwa mit dem Statement:

```
-T1...3=A1(1,4)   INC=4
```

Hier stellt die Angabe INC die Eingabeposition nach jeder Wertezuweisung um vier Stellen weiter. Dieses Statement ist daher gleichwertig mit den folgenden drei Statements:

```
-T1=A1(1,4)
-T2=A1(5,4)
-T3=A1(9,4)
```

FETCH-Funktion

Die Eingabedaten der Wertezuweisungen stammen normalerweise aus Datenfeldern von INPUT-EXT-Dateien oder aus Variablen. Zusätzlich können Eingabewerte aber auch aus Schlüsseldateien entnommen werden.

Beispielsweise lässt sich eine Schlüsseldatei mit dem Statement

```
FETCH-FILE FNAME=GEMEINDE.TBL KEY=D(1,9)
```

einlesen, die in den Positionen 1 ... 9 die Gemeindekennziffer als Schlüssel enthält.
Steht in den Positionen 10 ... 11 der Datei das Bundesland, so kann mit der Wertezuweisung

```
-N10=FETCH(N13 D1(10..11))
```

der Variablen N10 das Bundesland zur Gemeindekennziffer N13 zugewiesen werden.

Nachdem eine Datei mit dem Statement FETCH-FILE ... FETCH-FILE29 eröffnet wurde, können daraus Daten mit den Funktionen FETCH ... FETCH29 eingelesen werden.
Hinter ADD-PROC, MOD-PROC, UPD-PROC und OUTPUT-EXT lässt sich die FETCH-Funktion in allen Wertezuweisungen anstelle externer Datenfelder auf der rechten Seite des Gleichheitszeichens verwenden.

Im Detail erlaubt die FETCH-Funktion folgende Angaben:

FETCHm**(** key ... feld **) < NOM >**

m Mit der Angabe FETCH ... FETCH29 wird die Datei FETCH-FILE ... FETCH-FILE29 ausgewählt, aus der ein Wert zu entnehmen ist.

key ... Hinter der öffnenden Klammer (sind die Schlüsselwerte anzugeben, zu denen der passende Datensatz in der FETCH-Datei zu suchen ist. Aus diesem Satz werden Daten entnommen und in der laufenden Wertezuweisung übertragen. Die Daten stammen aus dem hinter den Schlüsselwerten angegebenen Datenfeld.
Die Schlüssel können als N-Variable, T-Variable, Fall-Identifikationen ID ... ID4 oder als konstante Werte angegeben werden.

Die Anzahl der Schlüssel und ihr Typ muss mit den Angaben KEY ... KEY4 im dazugehörigen Statement FETCH-FILE übereinstimmen.
Wurde im Statement FETCH-FILE der KEY als alphanumerisches Feld definiert, so sind in FETCH nur T-Variable, alphanumerische Fall-Identifikationen oder Textkonstante in Hochkommas zulässig. Für numerische Schlüssel KEY dagegen sind N-Variable, numerische Fall-Identifikationen oder numerische Konstanten anzugeben.

So wäre nach der Definition

```
FETCH-FILE3  FNAME=MARKEN.TBL  KEY=D(6,3)  KEY1=A(1,5)
```

die folgende Wertezuweisung möglich:

```
-N20=FETCH3(N11  T17  D(10,5)2)
```

feld Hier ist das externe Datenfeld anzugeben, aus dem die Datenwerte zu entnehmen sind.
Es sind alle Datenfelder möglich, die im Abschnitt *externe Datenfelder* beschrieben sind.
Dabei legt der Typ der Zielvariablen oder des Zielfeldes links vom Gleichheitszeichen auch den Typ des Eingabefeldes in der FETCH-Funktion fest:

Merkmale
Steht links vom Gleichheitszeichen eine C-Variable oder ein externes Merkmals-Feld so ist in der FETCH-Funktion nur ein Feld vom Typ B, I, K, L, M, MK, MS, U, UK oder US zulässig.
Anstelle des Gleichheitszeichen = ist auch das logische *oder* |= sowie das logische *und* &= möglich.
Hinter der schließenden Klammer der FETCH-Funktion sind auch die Prüfbedingungen und Verarbeitungsregeln für Merkmale erlaubt: Zi, SKE, SH, SL, SR und INC.

Zahlen
Steht links vom Gleichheitszeichen eine N-Variable oder ein externes numerisches Feld so ist in der FETCH-Funktion nur ein Feld vom Typ D, DK, DS, F, G, oder P zulässig.
Hinter der schließenden Klammer der FETCH-Funktion sind auch die Verarbeitungsregeln INC und BZ erlaubt, sowie die Prüfbedingungen < , >, =, ^= usw.

Texte
Steht links vom Gleichheitszeichen eine T-Variable oder ein externes Text-Feld, so ist in der FETCH-Funktion nur ein Feld vom Typ A, AK oder AS zulässig.
Anstelle des Gleichheitszeichen = sind auch die Justagezeichen L=, R= oder M= möglich.
Hinter der schließenden Klammer der FETCH-Funktion sind auch die Verarbeitungsregeln SKB und INC erlaubt.

NOM Falls in der FETCH-Datei zu einem Schlüssel key kein Datensatz gefunden wird, erscheint in der Protokolldatei die Fehlermeldung 518: *Datensatz fehlt ...*
Mit der Angabe NOM hinter der FETCH-Funktion wird die Fehlermeldung für das laufende Statement unterdrückt. Mit der Angabe MSG=FETCH:NO im Statement OPTIONS lässt sich diese Fehlermeldung für alle Statements unterbinden.

Trotz unterdrückter Fehlermeldung wird der Operand ERR auf 518 gesetzt und die Fehler in den Statistiken ESUM und ESUMS aus dem Statements OPTIONS mitgezählt.

Beispiele zur FETCH-Funktion finden sich beim FETCH-FILE-Statement.

Interne Dateien

Interne CNT-Dateien sind spezielle Datenbestände, die die auszuwertenden statistischen Fälle bereits in Form von Variablen enthalten.

Interne Datenbestände werden mit Hilfe des Statements OUTPUT-INT erstellt und enthalten folgende Daten:

- Variablendefinitionen mit den dazugehörigen Texten,
- Textelemente,
- Daten der statistischen Fälle als Variablenwerte,
- Makros,
- Variablengruppen für die Bildschirmanzeige in CNTW.

Die internen Dateien werden automatisch komprimiert und benötigen meist sehr viel weniger Speicherplatz als externe Dateien.

Interne Dateien sind insbesondere dann von Vorteil, wenn ein Datenbestand einmal erstellt und danach häufig ausgewertet werden soll. Sie führen in der Regel zu wesentlich kürzeren Verarbeitungszeiten.

Interne Dateien sind stets nach aufsteigender Fall-Identifikation sortiert. Beim Einlesen werden die statistischen Fälle immer in dieser Reihenfolge verarbeitet.

Transponierte interne Dateien

Internen Dateien sind zunächst fallweise organisiert: Die Variablenwerte eines jeden statistischen Falles sind in einem Datensatz enthalten. Zur Auswertung werden nacheinander alle statistischen Fälle mit allen Variablen eingelesen.

Bei Datenbeständen mit vielen Variablen pro Fall ist diese Vorgehensweise unökonomisch, wenn nur wenige Variable auszuwerten sind. Das Einlesen der nicht benötigten Variablenwerte kann die Verarbeitungsdauer stark erhöhen.

CNTA bietet daher die Möglichkeit, die Daten auch in transponierter Form (gekippt) zu speichern.
Dabei werden die Werte einer jeden Variablen aus allen statistischen Fällen direkt hintereinander gestellt. Sie lassen sich in einem Schritt von der Festplatte einlesen. Die nicht benötigten Variablenwerte werden nicht angefasst.

Die transponierte Speicherungsform hat aber auch Nachteile: Die Erstellung einer solchen Datei erfordert zusätzliche Verarbeitungszeit, die bei großen Datenbeständen fühlbar werden kann. Dieser Zusatzaufwand lohnt sich nur dann, wenn eine interne Datei nach ihrer Fertigstellung häufig ausgewertet werden soll.

Außerdem reduziert sich der Laufzeitvorteil transponierter Dateien, wenn bei umfangreichen Verarbeitungsläufen sehr viele Variable auszuwerten sind. Wird dabei ein hoher Anteil der gespeicherten Variablen benötigt, kann die transponierte Verarbeitungsform sogar zu schlechteren Laufzeiten führen als bei normaler Datenspeicherung.

Im Statement OUTPUT-INT entscheidet die Angabe TRANS über die Speicherungsform der Daten:
Ohne TRANS werden die Daten nur fallweise gespeichert, mit TRANS dagegen sowohl fallweise als auch transponiert. Dies kostet zusätzlichen Speicherplatz auf der Festplatte; das Programm kann aber selbst entscheiden, ob die fallweise oder transponierte Auswertung schneller zum Ergebnis führt.

Beispiel: Interne Datei erstellen

Die folgenden Statements erzeugen eine interne Datei:

```
INPUT-EXT  FNAME=1009.EXT  ASCII  SIZE=80,EOL

DEFS
-C10:2  1='Männer'
        2='Frauen'
-N10:4   'Geburtsjahr des Befragten'
-T10:30  'Wohnort des Befragten'
-C20:3   'Werbung gesehen'
        1='Karte 1'
        2='Karte 2'
        3='Karte 3'
-(10)    'Frage 9:'/
         'Haben Sie zu dieser Marke kürzlich eine Werbung gesehen ?'

ADD-PROC
-C10=L1(5,1)
-N10=D1(6,4)
-T10=A1(10,30)
-C20=L1(40,1)

OUTPUT-INT  FNAME=1009.INT
XTAB  ROW=TOTAL;C20  COL=TOTAL;C10
```

Durch INPUT-EXT wird zunächst eine externe Eingabedatei mit dem Namen *1009.EXT* angefordert. Es handelt sich um eine ASCII-Datei, deren Datensätze maximal 80 Bytes lang und durch Zeilenende-Zeichen abgeschlossen sind.

In den Folgestatements von DEFS werden die Variablen C10, C20, N10 und T10 definiert und mit Texten versehen. Weiter wird dort das Textelement (10) definiert und mit einem festen Fragetext versehen.

Die Folgestatements von ADD-PROC übertragen für jeden statistischen Fall Werte aus den Datensätzen der externen Datei in die neu definierten Variablen.

Das Statement OUTPUT-INT erzeugt nun im gleichen Verarbeitungslauf eine interne Datei mit dem Dateinamen *1009.INT*. In diese Datei werden die Variablendefinitionen übernommen, das Textelement sowie die Daten der Variablen für alle aus der externen Datei eingelesenen statistischen Fälle.

Im gleichen Verarbeitungslauf werden die Variablen mit Hilfe des XTAB-Statements ausgewertet.

Beispiel: Interne Datei auswerten

Die folgenden Statements zeigen Auswertungen aus einer internen Datei:

```
INPUT-INT   FNAME=1009.INT
CODEBOOK
XTAB   COL=TOTAL;C10
       ROW=TOTAL;C20
```

Das Ergebnis von CODEBOOK könnte folgendermassen aussehen:

CODEBOOK

Anzahl Fälle 4594

C10:2			
1	Männer	992	21,6%
2	Frauen	3602	78,4%
Z1	eine Angabe	4594	100,0%
N20:3	Alter		
	Fälle mit Angaben	4591	99,9%
	Fälle ohne Angaben	3	0,1%
	Minimum		19
	Maximum		89
	Mittelwert		35,7
T70:30	Land des Hauptwohnsitzes		
	Fälle mit Angaben	111	2,4%
	Fälle ohne Angaben	4483	97,6%
	kleinste Länge		2
	größte Länge		28
(70)	Frage 12: Wenn Sie im Ausland leben: Bitte geben Sie den Namen des Landes ein		

Hier ist C10 eine kategoriale Variable mit zwei Ausprägungen und N20 eine numerische Variable, die das Alter der Testpersonen enthält. Die Textvariable T70 enthält eine Länderangabe als unverschlüsselten Text und (70) ist ein Textelement, das sich zum Beispiel zur einfachen Beschriftung von Tabellen verwenden lässt.

Aus dem Statement XTAB könnte folgende Tabelle entstehen:

	TOTAL	Männer	Frauen
TOTAL	4594	992	3602
Alter			
Von 18 bis 20 Jahre	3	1	2
Von 21 bis 25 Jahre	238	76	162
Von 26 bis 30 Jahre	1101	329	772
Von 31 bis 35 Jahre	1203	248	955
Von 36 bis 40 Jahre	902	152	750
Von 41 bis 50 Jahre	926	147	779
Von 51 bis 60 Jahre	200	35	165
Über 60 Jahre	18	1	17

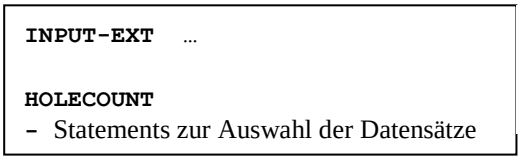
Verarbeitung der statistischen Fälle

Mit den Statements INPUT-EXT und INPUT-INT werden Eingabedateien beschrieben, aus denen CNTA statistische Fälle nacheinander einliest und verarbeitet.

Zu den Dateien aus INPUT-EXT und INPUT-INT lassen sich mit INPUT-SEC weitere interne Dateien und mit FETCH-FILE zusätzliche externe Dateien anfordern. Damit ist eine Vielzahl von Verarbeitungsformen möglich.

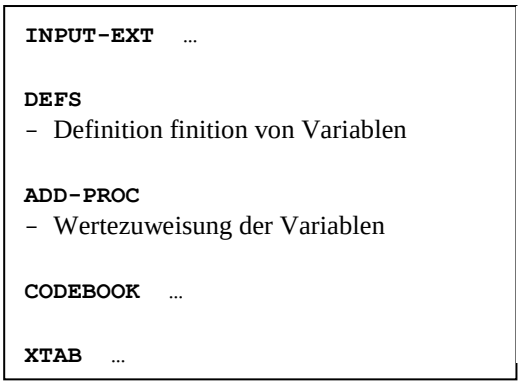
Einige häufig vorkommende Abläufe werden hier mit den dazugehörigen Statements gezeigt.
Die ausführliche Beschreibung dieser Statements befindet sich weiter hinten in diesem Handbuch.

Externe Datei grob auswerten



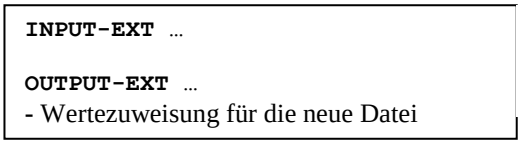
Das Statement INPUT-EXT beschreibt die zu untersuchende externe Datei.
Mit HOLECOUNT lässt sich eine grobe Übersicht über den Inhalt der externen Datei verschaffen.
Die dahinter liegenden Folgestatements beschränken die Auswertung auf einzelne Datensätze oder Positionen innerhalb der Sätze.

Externe Datei detailliert auswerten



INPUT-EXT beschreibt die einzulesende externe Datei.
Mit dem Statement DEFS wird die Definition der Variablen begonnen, die die Werte der externen Datei aufnehmen sollen. Dahinter kann eine unbegrenzte Anzahl von verschiedenen Variablen definiert werden.
Das Statement ADD-PROC leitet die Wertezuweisungen ein, durch die die Variablen ihre Werte aus den Datensätzen der externen Datei erhalten.
CODEBOOK erstellt mit geringem Eingabeaufwand eine Übersichtsauswertung über die definierten Variablen.
Mit den Statements XTAB lassen sich Kreuztabellen zur detaillierten Auswertung der Variablen erzeugen.

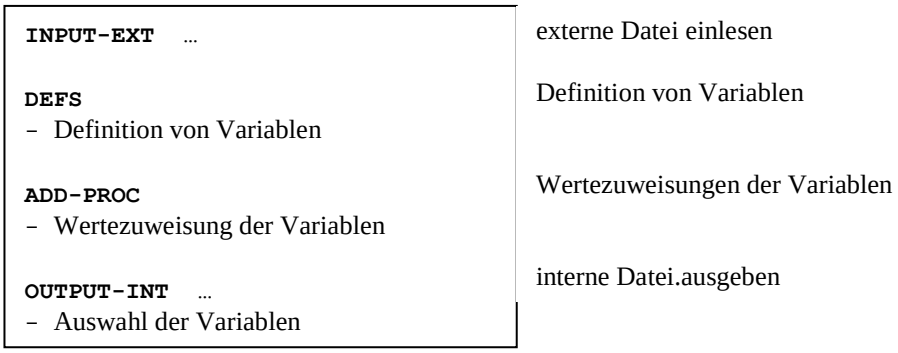
Externe Datei umformen



INPUT-EXT beschreibt die einzulesende externe Datei.

OUTPUT-EXT legt den Dateinamen und die Eigenschaften der neuen externen Datei fest.
Die dahinter liegenden Statements beschreiben, wie die Daten der Eingabedatei in die Ausgabedatei übertragen werden sollen.
Wenn das Statement OUTPUT-EXT die Angabe COPY enthält, können diese Wertezuweisungen auch fehlen. Die Datensätze der Eingabedatei werden in die Ausgabedatei kopiert, zum Beispiel um sie vom ASCII-Format in das ANSI-Format umzuwandeln.

Interne Datei erstellen



INPUT-EXT beschreibt die einzulesende externe Datei.

Mit dem Statement DEFS wird die Definition der Variablen begonnen, die die Werte der externen Datei aufnehmen sollen. Dahinter kann eine unbegrenzte Anzahl von verschiedenen Variablen definiert werden.

Das Statement ADD-PROC leitet die Wertezuweisungen ein, durch die die Variablen ihre Werte aus den Datensätzen der externen Datei erhalten.

OUTPUT-INT erstellt die interne Datei. Die dahinter liegenden Statements wählen die Variablen aus, die in die neue Datei übernommen werden sollen.

Interne Datei auswerten

```
INPUT-INT   ...

DEFS
- Definition von zusätzlichen Variablen

MOD-PROC
- Wertezuweisung der Variablen

CODEBOOK   ...
XTAB   ...
```

interne Datei einlesen

Definition von zusätzlichen Variablen

Wertezuweisungen für die zusätzlichen Variablen

Übersichtsauswertung der Variablen
Kreuztabellen aus den Variablen

INPUT-INT liest die Daten der internen Datei ein.

DEFS erlaubt die Definition von zusätzlichen Variablen und kann die Texte der Variablen aus

INPUT-INT verändern.

Hinter MOD-PROC können die eingelesenen Variablenwerte noch verändert und die neuen Variablen mit Werten versehen werden.

CODEBOOK erstellt eine Übersichtsauswertung der definierten Variablen.

Interne Datei verändern

```
INPUT-INT   ...

DEFS
- Definition von Variablen

MOD-PROC
- Wertezuweisungen der Variablen

OUTPUT-INT   ...
- Auswahl der Variablen
```

interne Datei einlesen

Veränderungen an den Variablen

Veränderungen an den Wertezuweisungen

Ausgabe der neuen internen Datei

INPUT-INT liest die Daten der internen Datei ein.

Mit dem Statement DEFS wird die Definition der Variablen begonnen. Dahinter lassen sich neue Variable definieren. Außerdem können die Texte von bestehenden Variablen aus der internen Datei verändert werden.

Das Statement MOD-PROC leitet die Wertezuweisungen ein. Hier erhalten die neuen Variablen ihre Werte. Es können aber auch die Werte von bestehenden Variablen der internen Datei verändert werden. Diese Änderungen gelten nur für den aktuellen Verarbeitungslauf. Sie werden nicht in die Eingabedatei geschrieben.

OUTPUT-INT erstellt die neue interne Datei. Die dahinter liegenden Statements wählen die Variablen für die neue interne Datei aus.

Externe Daten zu einer internen Datei hinzufüge

INPUT-INT ... **INPUT-EXT** ...	interne Datei einlesen externe Datei mit den zusätzlichen Daten
ADD-PROC - Wertezuweisungen für neue Fälle	Wertezuweisungen für neue Fälle
UPD-PROC - Wertezuweisungen für alte Fälle	Wertezuweisungen für alte Fälle
MOD-PROC - Wertezuweisungen der Variablen	Variablenwerte für alte und neue Fälle verändern
OUTPUT-INT ... - Auswahl der Variablen	erweiterte interne Datei ausgeben

INPUT-INT liest die Daten der internen Datei ein und INPUT-EXT die hinzuzufügenden Daten der externen Datei. CNTA führt dabei die statistischen Fälle aus den beiden Dateien nach folgenden Regeln zusammen:

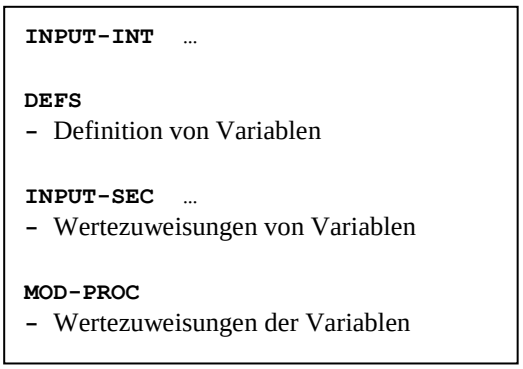
Gibt es zu einem statistischen Fall aus INPUT-EXT einen Fall in INPUT-INT mit gleicher Fall-Identifikation, so handelt es sich um einen *alten Fall*, für den die Wertezuweisungen hinter UPD-PROC ausgeführt werden. Damit werden den Variablen des Falles der internen Datei neue Werte aus dem zugehörigen Fall der externen Datei zugewiesen.

Gibt es zu einem Fall aus INPUT-EXT keinen Fall in INPUT-INT mit gleicher Fall-Identifikation, so handelt es sich um einen *neuen Fall*, für den die Wertezuweisungen hinter ADD-PROC ausgeführt werden. Damit werden den Variablen eines neu angelegten internen Falles Werte aus dem zugehörigen Fall der externen Datei zugewiesen.

OUTPUT-INT erstellt die neue interne Datei, die sowohl die geänderten Variablenwerte als auch die neu angelegten Fälle enthält.

-

Mehrere interne Dateien verarbeiten



INPUT-INT liest die Daten der internen Datei ein.
Mit dem Statement DEFS lassen sich neue Variable definieren. Außerdem können die Texte von bestehenden Variablen aus INPUT-INT verändert werden.
Die Statements INPUT-SEC lesen Daten aus weiteren internen Dateien ein. CNTA führt dabei die statistischen Fälle aus den Dateien nach folgenden Regeln zusammen:

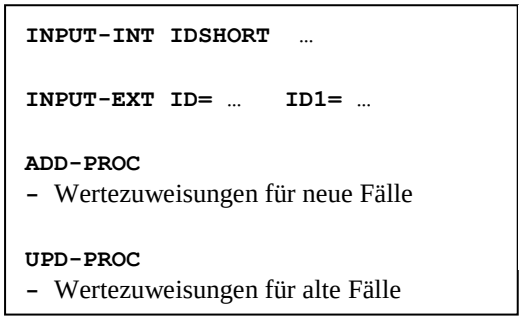
- Die interne Datei INPUT-INT liefert neben ihren Daten vor allem die Variablendefinitionen für die laufende Verarbeitung.
- Die Daten der internen Dateien aus INPUT-SEC stehen ebenfalls in Variablen, lassen sich aber von dort aus nicht direkt auswerten. Ihre Werte müssen erst in die aus INPUT-INT oder DEFS stammenden Variablen übertragen werden.
  Diese Übertragung kann auf zweierlei Weise erfolgen:
- Ist zu INPUT-SEC kein Folgestatement angegeben, so werden die Variablenwerte aus INPUT-SEC automatisch in gleichnamige Variable aus INPUT-INT oder DEFS übertragen. Gibt es zu einer Variablen aus INPUT-SEC kein gleichnamiges Gegenstück, so steht der Variablenwert nicht für die Verarbeitung zur Verfügung.
- Mit Folgestatements hinter INPUT-SEC können die Variablenwerte aus INPUT-SEC andersnamigen Variablen zugewiesen werden. Zum Beispiel sorgt das Statement
  -C27=C1055
  hinter INPUT-SEC dafür, dass die Werte der Variablen C1055 aus INPUT-SEC in die Variable C27 aus INPUT-INT oder DEFS übertragen wird. Der alte Wert von C27 geht dabei verloren.

Für jeden statistischen Fall findet nun folgende Verarbeitung statt:

- Gibt es zu einem statistischen Fall aus INPUT-INT keinen Fall aus INPUT-SEC mit gleicher Fall-Identifikation, so werden die Variablen aus INPUT-INT normal verarbeitet und die mit DEFS neu definierten Variablen gelöscht, also mit den Werten *leer* oder *keine Angabe* versehen.
- Gibt es zu dem statistischen Fall aus INPUT-INT Daten aus INPUT-SEC mit gleicher Fall-Identifikation, so überschreiben die Werte aus INPUT-SEC die Werte aus INPUT-INT. Gibt es in einem Verarbeitungslauf mehrere INPUT-SEC-Statements, so wird dieser Vorgang für jedes INPUT-SEC wiederholt. Das geschieht in der Reihenfolge, in der die INPUT-SEC-Statements angegeben sind.
- Gibt es zu einem statistischen Fall aus INPUT-SEC keinen passenden Fall aus INPUT-INT mit gleicher Fall-Identifikation, so werden alle in INPUT-INT oder DEFS definierten Variablen gelöscht und danach mit den Werten aus INPUT-SEC gefüllt.

Erst nach Übernahme der Daten aus der letzten INPUT-SEC-Datei werden die Statements MOD-PROC ausgeführt. Die Verarbeitung erfolgt dort so, als würden die Daten der Dateien INPUT-INT und INPUT-SEC aus einer einzigen internen Datei stammen.
Anschließend können die Variablen wie gewohnt ausgewertet und in eine neue Datei ausgegeben werden.

Dateien mit unterschiedlicher Anzahl ID-Stufen



Wenn gleichzeitig zwei Eingabedateien zu verarbeiten sind, müssen normalerweise beide Dateien die gleiche Anzahl von Fall-Identifikations-Stufen besitzen, um die Datensätze der Eingabedateien zusammenführen zu können.

Es besteht aber auch die Möglichkeit, zwei Eingabedateien mit unterschiedlicher Anzahl von ID-Stufen zu verarbeiten. Zum Beispiel könnte zu einer Panel-Befragung eine interne Datei mit den Demografie-Daten der Befragten vorliegen. Als Fall-Identifikation wird eine Befragten-Nummer verwendet. In einer externen Datei könnten sich zu jedem Befragten mehrere Datensätze mit Tagebuchdaten befinden. Die erste ID-Stufe wäre wieder die Befragten-Nummer und die Stufe ID1 eine Datumsangabe.

Da der Datei INPUT-INT die Stufe ID1 fehlt, ist dort die Angabe IDSHORT erforderlich.
Zu jedem statistischen Fall aus INPUT-EXT wird der Fall aus INPUT-INT eingelesen, der die gleiche Fall-Identifikation ID besitzt. Seine Variablenwerte stehen dann allen zugehörigen Fällen aus INPUT-EXT zur Verfügung.

Statistische Fälle aus INPUT-EXT zu denen es ein Gegenstück in INPUT-INT gibt, werden normal verarbeitet. Die Wertezuweisungen dazu sind hinter UPD-PROC einzugeben.

Für Statistische Fälle aus INPUT-EXT zu denen es kein Gegenstück in INPUT-INT gibt, enthalten die Variablen aus INPUT-INT keine Werte (keine Angaben, Z0). Die Wertezuweisungen dazu sind hinter ADD-PROC einzugeben.

Fälle aus INPUT-INT ohne Gegenstück in INPUT-EXT werden unverarbeitet übergangen.

Daten aus Tabellendateien einfüge

```
INPUT-INT …

FETCH-FILE …

MOD-PROC
- Variable = FETCH(Schlüssel, Eingabefeld)
```

interne Datei einlesen

Tabellendatei anfordern

Wertezuweisungen mit der FETCH-Funktion

CNTA kann die statistischen Fälle auch mit Daten aus Tabellendateien erweitern:
Eine solche Tabellendatei könnte zum Beispiel eine Gemeindedatei sein, die Gemeindekennziffern als Schlüssel enthält. Zu jeder Gemeindekennziffer liegen dort weitere Daten zur Gemeinde, wie Ortsgröße und Bundesland vor.

Enthalten die statistischen Fälle nun eine Gemeindekennziffer, so kann diese als Schlüssel einer FETCH-Funktion verwendet werden, um die Ortsgröße aus der Gemeindedatei in eine Variable zu stellen.

In diesem Beispiel werden die statistischen Fälle aus einer internen Datei eingelesen.
Es ist aber ebenso möglich, die Daten über INPUT-EXT einzulesen.
Durch das Statement FETCH-FILE wird die Tabellendatei angefordert.

In den Wertezuweisungen wird den Variablen mit der Funktion FETCH ein beliebiges Datenelement aus der Tabellendatei zugewiesen. Dazu ist der Schlüssel in Form einer N- oder T-Variablen anzugeben sowie das Eingabefeld der Tabellendatei, aus dem die Datenelemente zu entnehmen sind.

Mehrere externe Dateien verarbeiten

```
INPUT-EXT …

FETCH-FILE …
FETCH-FILE1 …
FETCH-FILE2 …

ADD-PROC
- Variable = Eingabefeld
- Variable = FETCH(ID, Eingabefeld)
- Variable = FETCH1(ID, Eingabefeld)
- Variable = FETCH2(ID, Eingabefeld)
```

externe Datei einlesen

zusätzliche externe Dateien anfordern

Beginn der Wertezuweisungen
Daten aus der Datei INPUT-EXT einlesen
Daten aus der Datei FETCH-FILE einlesen
Daten aus der Datei FETCH-FILE1 einlesen
Daten aus der Datei FETCH-FILE2 einlesen

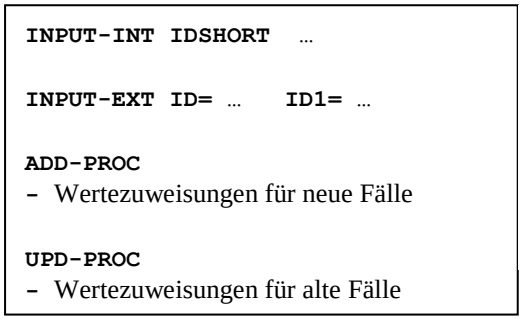
CNTA kann in einem Verarbeitungslauf mehrere externe Dateien einlesen.
Die erste externe Datei wird mit dem Statement INPUT-EXT angefordert.
Die weiteren externen Dateien werden über die Statements FETCH-FILE eingelesen.

Hinter ADD-PROC werden mit einfachen Wertezuweisungen die Daten aus INPUT-EXT eingelesen. Mit den Funktionen FETCH, FETCH1 und FETCH2 in den Wertezuweisungen wird auf die Daten der weiteren externen Dateien zugegriffen.

Dabei ist die Fall-Identifikation ID in den FETCH-Funktionen als Schlüssel zu verwenden.
Zu jedem statistischen Fall aus INPUT-EXT wird dann der zugehörige Fall der FETCH-Datei mit gleicher Fall-Identifikation eingelesen.

Statistische Fälle einer FETCH-Datei, zu der es keinen Fall mit gleicher Fall-Identifikation ID in INPUT-EXT gibt, werden hierbei nicht verarbeitet.

Dateien mit unterschiedlicher Anzahl ID-Stufen



Wenn gleichzeitig zwei Eingabedateien zu verarbeiten sind, müssen normalerweise beide Dateien die gleiche Anzahl von Fall-Identifikations-Stufen besitzen, um die Datensätze der Eingabedateien zusammenführen zu können.

Es besteht aber auch die Möglichkeit, zwei Eingabedateien mit unterschiedlicher Anzahl von ID-Stufen zu verarbeiten. Zum Beispiel könnte zu einer Panel-Befragung eine interne Datei mit den Demografie-Daten der Befragten vorliegen. Als Fall-Identifikation wird eine Befragten-Nummer verwendet. In einer externen Datei könnten sich zu jedem Befragten mehrere Datensätze mit Tagebuchdaten befinden. Die erste ID-Stufe wäre wieder die Befragten-Nummer und die Stufe ID1 eine Datumsangabe.

Da der Datei INPUT-INT die Stufe ID1 fehlt, ist dort die Angabe IDSHORT erforderlich.
Zu jedem statistischen Fall aus INPUT-EXT wird der Fall aus INPUT-INT eingelesen, der die gleiche Fall-Identifikation ID besitzt. Seine Variablenwerte stehen dann allen zugehörigen Fällen aus INPUT-EXT zur Verfügung.

Statistische Fälle aus INPUT-EXT zu denen es ein Gegenstück in INPUT-INT gibt, werden normal verarbeitet. Die Wertezuweisungen dazu sind hinter UPD-PROC einzugeben.

Für Statistische Fälle aus INPUT-EXT zu denen es kein Gegenstück in INPUT-INT gibt, enthalten die Variablen aus INPUT-INT keine Werte (keine Angaben, Z0). Die Wertezuweisungen dazu sind hinter ADD-PROC einzugeben.

Fälle aus INPUT-INT ohne Gegenstück in INPUT-EXT werden unverarbeitet übergangen.

Textzeilen

Texte sind an verschiedenen Stellen möglich. Zum Beispiel als Variablentext, als Blattüberschrift, als Kopf- oder Zeilentext in Tabellen usw. Sie sind stets nach den gleichen Regeln anzugeben:

Einzeilige Texte

Textzeilen sind in Hochkommas ' einzuschließen und dürfen bis zu 500 Bytes lang sein.
Zum Beispiel:

```
'Alter des Befragten'
```

Bei Bedarf werden längere Textzeilen vom Programm in mehrere Zeilen aufgebrochen.

Mehrzeilige Texte

Diese sind mit Schrägstrichen hintereinander einzugeben. Zum Beispiel:

```
'Netto-Einkommen'/'in Euro pro Monat'
```

Dies ist gleichwertig mit der Eingabe:

```
'Netto-Einkommen' /
'in Euro pro Monat'
```

Mehrzeilige Texte dürfen bis zu 255 Zeilen besitzen und jede davon kann bis zu 500 Bytes lang sein.

Leerzeilen

Mehrere direkt hintereinander stehende Schrägstriche erzeugen eine entsprechende Anzahl von Leerzeilen. So ergibt

```
'Anzahl'//'Personen'
```

einen dreizeiligen Text mit einer Leerzeile zwischen den Zeilen *Anzahl* und *Personen*.

Die Eingabe

```
'Anzahl'//
```

erzeugt ebenfalls einen dreizeiligen Text mit der Zeile *Anzahl* und zwei Leerzeilen dahinter.

Schließlich liefert die Angabe

```
///
```

einen dreizeiligen Text, der nur aus Leerzeilen besteht.

Mehrere Eingabezeilen verbinden

Eine Textzeile lässt sich über mehrere Statementzeilen hinweg eingeben, wobei die Textteile mit dem Bindestrich - miteinander zu verbinden sind. So erzeugen die Statementzeilen

```
'Anzahl ' -
'Personen'
```

den einzeiligen Text *Anzahl Personen*.

Hochkommas in den Texten

Soll ein Text selbst ein Hochkomma enthalten, so ist dieses in Form von zwei direkt hintereinander stehenden Hochkommas '' anzugeben, um es von den Hochkommas am Anfang und Ende der Zeile zu unterscheiden. Zum Beispiel erzeugt

```
...'Marke ''VW'''
```

den Text *Marke 'VW'*

Nummernzeichen # in den Texten

Soll eine Textzeile ein Nummernzeichen # enthalten, so sind dafür zwei direkt hintereinander stehenden Nummernzeichen ## anzugeben, um es von Makronamen zu unterscheiden. Zum Beispiel erzeugt das Statement

```
'Marke ##3'
```

den Text *Marke #3*

Unterstreichungsstrich _ als Trennfuge

Mit diesem Zeichen wird eine Soll-Trennstelle im Text für eine eventuelle Silbentrennung vorgegeben.
Ist keine Silbentrennung erforderlich, wird der Unterstreichungsstrich nicht ausgegeben.
Lieferte zum Beispiel das Programm die Trennung

```
Nied-
ersachsen
```

so wird durch Eingabe von

```
'Nieder_sachsen'
```

die Trennung

```
Nieder-
sachsen
```

erreicht. Falls die Spalte breit genug ist, wird das Wort nicht getrennt und ohne _ ausgegeben.
Das Zeichen _ vor einem Wort verhindert die Trennung des Wortes vollständig.

Soll eine Textzeile einen Unterstreichungsstrich _ enthalten, so ist dieses in Form von zwei direkt hintereinander stehenden Unterstreichungsstrichen __ anzugeben, um es von den Soll-Trennstellen _ zu unterscheiden. Zum Beispiel erzeugt das Statement

```
'Frage 1__a'
```

den Text *Frage 1_a*
Soll eine Textzeile einen Unterstreichungsstrich _ enthalten, so sind dafür zwei direkt hintereinander stehende Unterstreichungsstriche __ anzugeben, um es von den Soll-Trennstellen _ für FIT zu unterscheiden. Zum Beispiel erzeugt das Statement

```
'Frage 1__a'
```

den Text *Frage 1_a*

Datum und Uhrzeit

Die Angaben ~DATE~ und ~TIME~ fügen Datum und Uhrzeit des Verarbeitungslaufs an beliebiger Stelle in Textzeilen ein. Zum Beispiel:

```
PAGEP FOOT='Auswertung vom ~DATE~ um ~TIME~ Uhr'
```

Seitennummer

In den Kopftexten HEAD und Fußtexten FOOT der Statements PAGE und PAGEP fügt die Angabe -NR~ die laufende Seitennummer ein. Zum Beispiel:

```
PAGEP FOOTR='Seite ~NR~'
```

Vorhandene Texte durch Klammern (...) übernehmen

Texte aus bereits definierten Variablen und Textelementen lassen sich in neue Texte übernehmen. Sie sind dazu in der Form (10), (N99) oder (C35.1) in Klammern zu stellen. Zum Beispiel wird durch

```
XTAB  TITLE=(C35.1)
```

der Text des Merkmals 1 der Variablen C35 zum Titeltext einer Kreuztabelle.

Diese Klammerausdrücke lassen sich auch mit den Zeichen / und - untereinander und mit Textzeilen kombinieren. Wurde zum Beispiel unter DEFS definiert

```
-(10)  'Anzahl'
-N99   'Personen' / 'im Haushalt'
```

so liefert das CNTA-Statement

```
TITLE='Frage 14: ' - (10) / (N99)
```

die folgenden Textzeilen als Titeltext:

```
Frage 14: Anzahl
Personen
im Haushalt
```

Mehrere Texte in den Klammern (...)

Innerhalb von Klammern lassen sich Variablen- und Merkmalstexte durch die Zeichen / und - mit Zeichenfolgen verbinden.
Eine in Hochkommas gesetzte Zeichenfolge oder eine Leerzeile / wird dabei *nicht* ausgegeben, wenn ein benachbarter Variablen- oder Merkmalstext fehlt.

Wurde zum Beispiel die Variable C35 folgendermaßen definiert:

```
-C35:5 'Berufstätigkeit'/
       'des Befragten'
       1='In Ausbildung'
```

so liefert die Angabe

```
(C35 - ': ' - C35.1)
```

die Textzeilen

Berufstätigkeit
des Befragten: In Ausbildung

Besitzt C35 keinen Variablentext, so entsteht nur eine Zeile ohne Doppelpunkt:

In Ausbildung

Fehlt der Merkmalstext C35.1, so entstehen zwei Zeilen ohne Doppelpunkt:

Berufstätigkeit
des Befragten

Besitzt C35 weder einen Variablen- noch einen Merkmalstext, so wird kein Text ausgegeben.
Mit der Angabe

```
(/C35/)
```

wird für die oben definierte Variable C35 der Text

Berufstätigkeit
des Befragten

erzeugt, mit einer Leerzeile vor und einer hinter den beiden Textzeilen.

Besitzt die Variable C35 keinen Variablentext, so werden weder der Text noch die beiden Leerzeilen ausgegeben.

Übernahme von Variablentexten, Merkmalstexten und Textelementen mit UNDO(...)

Bei der Übernahme von Variablen- und Merkmalstexten mit einfachen Klammern bleiben die Zeilenumläufe und Leerzeilen der Texte erhalten.
Mit UNDO vor der Klammer werden die alten Texte zu einem einzeiligen Text umgeformt.
UNDO entfernt auch Leerzeilen, doppelte Leerzeichen und mehrfache Sonderzeichen, zum Beispiel aus Unterstreichungsstrichen, und macht Silbentrennungen rückgängig.

Wurde zum Beispiel die Variable C35 folgendermaßen definiert:

```
-C35:5 'Berufstätigkeit'/
       'des Befragten  '/
       '---------------'
       1='In Ausbildung,'/
         'Studium'
```

so entsteht durch die Angabe

```
TITLE=UNDO(C35) - ':' / UNDO(C35.1)
```

der Text

Berufstätigkeit des Befragten:
In Ausbildung, Studium

Mehrere Texte in den Klammern hinter UNDO(...)

Auch innerhalb der Klammern von UNDO lassen sich Variablen- und Merkmalstexte durch die Zeichen / und - mit Zeichenfolgen in Hochkommas verbinden.
Wie bei einer einfachen Klammer wird eine in Hochkommas gesetzte Zeichenfolge oder eine Leerzeile / *nicht* ausgegeben, wenn ein benachbarter Variablen- oder Merkmalstext fehlt.
Mit UNDO werden zusätzlich aus den Variablen- und Merkmalstexten Leerzeilen, doppelte Leerzeichen, mehrfache Sonderzeichen und Silbentrennungen entfernt.

Wurde zum Beispiel die Variable C35 folgendermaßen definiert:

```
-C35:5 'Berufstätigkeit'/
       'des Befragten  '/
       '---------------'
       1='In Ausbildung,'/
         'Studium'
```

so liefert die Angabe

```
UNDO(C35 - ': ' - C35.1)
```

die Textzeile

Berufstätigkeit des Befragten: In Ausbildung, Studium

Fehlt für C35 der Variablentext, so wird die Zeile

In Ausbildung, Studium

ohne Doppelpunkt ausgegeben. Fehlt der Merkmalstext C35.1, so entsteht die Zeile

Berufstätigkeit des Befragten

ebenfalls ohne Doppelpunkt. Besitzt C35 weder einen Variablen- noch einen Merkmalstext, so liefert die vorige Eingabe keinen Text.

Mit der Eingabe

```
UNDO(/C35/)
```

wird für die oben definierte Variable C35 der Text

Berufstätigkeit des Befragten

erzeugt, mit einer Leerzeile davor und einer dahinter.
Falls die Variable C35 keinen Variablentext besitzt, werden weder ein Text noch die beiden Leerzeilen ausgegeben.

Schriftarten und Logos

Bei der Textausgabe mit dem Statement PAGE arbeitet das Programm nur mit einer Schrift fester Größe. Logos sind dabei nicht möglich.

Das Statement PAGEP ermöglicht dagegen sehr weitgehende Gestaltungen der Texte:

Mit der Angabe FONTS in den Statements XTAB, CODEBOOK und INDEX lassen sich zentral Regeln für die Schriften festlegen. So kann zum Beispiel eine spezielle Schrift für alle Zeilentexte der Tabellen vorgegeben werden und eine andere für die Spalten.

Abweichend von diesen zentralen Regeln lassen sich einzelne Textzeilen, Worte oder auch Buchstaben besonders gestalten. Es können verschiedene Schriften verwendet, Texte gefettet, kursiv oder unterstrichen ausgegeben und Tabulatoren eingesetzt werden. In jede Textzeile können gescannte Logos oder Bilder eingefügt werden.

Dazu sind entsprechende Schrift-Kommandos in den Textzeilen anzugeben.
Diese beginnen und enden mit einer einzelnen Tilde ~ . Die Kommandos können in Groß- oder Kleinbuchstaben angegeben werden, zum Beispiel ~U~ oder ~u~ für Unterstreichen.
Schrift-Kommandos sind nur wirksam, wenn auf Seitendrucker ausgegeben wird.
Bei der Ausgabe auf Zeilendrucker werden die Kommandos automatisch aus den Texten entfernt.

Folgende Schrift-Kommandos sind an beliebiger Stelle in den Textzeilen möglich:

~B~ Der folgende Text wird in fetter Schrift gedruckt (bold).

~N~ Der folgende Text wird in normaler Schrift gedruckt.

~U~ Der folgende Text wird unterstrichen (underline).

~I~ Der folgende Text wird in kursiver Schrift gedruckt (italic).

~E~ Diese Angabe beendet das vorausgehende Schrift-Kommando und schaltet auf die zuvor wirksame Schrift zurück. `~E~` ist nicht erforderlich, wenn ein Schrift-Kommando bis zum Ende des Textes gelten soll. Ein Schrift-Kommando verliert seine Wirkung immer mit dem Ende der letzten Zeile des Textes.

Es können auch mehrere dieser Schriftattribute gleichzeitig verändert werden. So liefert:
'nur ~BU~eine~E~ Person im Haushalt'
den Text:
nur **eine** Person im Haushalt

Auch besteht die Möglichkeit, solche Schrift-Kommandos in mehreren Stufen zu überlagern.
Zum Beispiel erzeugt die Eingabe:
'nur ~B~~U~eine~E~ Person~E~ im Haushalt'
den Text:
nur **eine Person** im Haushalt

~FTi~ Diese Angabe verändert innerhalb der Zeile die Schriftart.
Es stehen die Schriften FONTi aus dem Statement PAGEP zur Verfügung.
Zum Beispiel könnte die Eingabe:
'nur eine Person ~FT3~im Haushalt'
den Text erzeugen:
nur eine Person im Haushalt

~COi~ Diese Angabe sorgt für Ausgabe des folgenden Textes in der Farbe CO1 ... CO16 aus dem Statement PAGEP. Mit ~CO0~ wird auf schwarze Schrift zurückgeschaltet. In dem Beispiel

```
TEXT '~CO2~Bitte unbedingt beachten:'/
     '~CO0~Die Frage 7 wurde nur von wenigen Befragten beantwortet'
```

wird die erste Zeile rot ausgegeben und die zweite schwarz.

~Gi~ Diese Angabe sorgt für Ausgabe des folgenden Textes in dem Grauton G1 ... G16 aus dem Statement PAGEP. Mit ~G0~ wird auf schwarze Schrift zurückgeschaltet. In dem Beispiel

```
TEXT 'Alter: 14 bis 19 ~G6~Jahre'
```

wird *Alter: 14 bis 19* in schwarzer Schrift ausgegeben und *Jahre* in mittlerem Grau.

~Si~ Diese Angabe legt das absolute Sprungziel für Tabulatorsprünge durch das Kommando ~T~ fest. Dabei gibt i den Abstand des Sprungziels vom Anfang der Zeile an. Er kann in allen Maßeinheiten angegeben werden, die beim Statement PAGEP beschrieben sind.
Zum Beispiel liefert die Eingabe:

```
'~S25~Frage:~T~Es fallen einem nicht immer alle Namen ein.'/
          '~T~Hier habe ich eine Liste ...'
```

die folgenden Textzeilen:

Frage: Es fallen einem nicht immer alle Namen ein.
Hier habe ich eine Liste ...

~S25~ stellt die Tabulatorposition auf 25mm vom Zeilenanfang. Die Tabulator-Kommandos ~T~ dahinter stellen die folgenden Texte dann auf diese Position.
Eine durch ~Si~ festgelegte Tabulatorposition bleibt auch über Textgrenzen hinweg erhalten:
So kann zum Beispiel ~Si~ in einem XTAB-Statement im TITLE-Text ein Sprungziel festlegen. Im BOTTOM-Text des gleichen XTAB oder auch im TITLE-Text eines folgenden XTAB kann durch ~T~ diese Tabulatorposition verwendet werden.
Erst eine neue Angabe ~Si~ überschreibt das vorausgegangene Sprungziel.

~S~ Diese Angabe legt das relative Sprungziel für Tabulatorsprünge ~T~ fest.
Hier wird im Gegensatz zu ~Si~ kein Zahlenwert vorgegeben, sondern der Abstand des Kommandos ~S~ vom Zeilenanfang in der fertigen Druckzeile als Sprungziel verwendet.
Zum Beispiel liefert die Eingabe:

```
'Frage:   ~S~Es fallen einem nicht immer alle Namen ein.'/
     '~T~Hier habe ich eine Liste ...'
```

die folgenden Textzeilen:

Frage: Es fallen einem nicht immer alle Namen ein.
Hier habe ich eine Liste ...

~S~ stellt die Tabulatorposition hinter *Frage:* . Die folgenden Tabulator-Kommandos ~T~ stellen die Texte dann auf diese Position. Eine durch ~S~ festgelegte Tabulatorposition bleibt auch über Textgrenzen hinweg erhalten:
So kann ~S~ in einem XTAB-Statement im TITLE-Text ein Sprungziel festlegen. Im BOTTOM-Text des gleichen XTAB oder auch im TITLE-Text eines folgenden XTAB kann durch ~T~ diese Tabulatorposition verwendet werden.

~T~ Diese Angabe führt einen Tabulatorsprung durch.
Dabei muss das Sprungziel vorher durch das Kommando ~S~ oder ~Si~ vorgegeben werden.
Zum Beispiel liefert die Eingabe:

```
'Frage 3: ~S~Es fallen einem nicht immer alle Namen ein.'/
          '~T~Hier habe ich eine Liste,'/
'~T~auf der verschiedene Produkte aufgeführt sind.'
```

die folgenden Textzeilen:

Frage 3: Es fallen einem nicht immer alle Namen ein.
Hier habe ich eine Liste,
auf der verschiedene Produkte aufgeführt sind.

~Ti~ Diese Angabe führt einen Tabulatorsprung an die Position i in der laufenden Zeile durch.
Dabei gibt i den Abstand des Sprungziels vom Anfang der Zeile an. Er kann in allen Maßeinheiten angegeben werden, die beim Statement PAGEP beschrieben sind.
Zum Beispiel könnte die Eingabe:

```
'Frage 3:~T25~Es fallen einem nicht immer ...'
```

die folgende Textzeile erzeugen:

Frage 3: Es fallen einem nicht immer ...

in der der Tabulatorsprung das Wort 'Es' um 25mm vom Zeilenanfang nach rechts rückt.
Besitzt das Sprungziel i einen zu kleinen Wert, der vor die laufende Position der Zeile zeigt, so findet kein Tabulatorsprung statt.

~ i ~ Mit dieser Angabe lassen sich in jede beliebige Textzeile Logos einfügen.
Dazu muss der Nummer i = 1 ... 999 im Statement PAGEP mit der Angabe

```
LOGOi=dateiname
```

eine Bilddatei im JPEG-Format zugeordnet werden.

Logos erscheinen in den Auswertungen nur, wenn mit dem Statement PAGEP gearbeitet wird.
Ist PAGE aktiv, so werden die Texte ohne Logos ausgegeben.

Zum Beispiel könnte das Statement

```
PAGEP HEAD='~1~ XYZ-Institut'
      LOGO1=d:\logos\kundeXYZ.jpg
```

folgende Kopfzeile auf den Auswertungsseiten erzeugen:

XYZ-Institut

Makros

Makros dienen dazu, häufig wiederkehrende Zeichenfolgen nur einmal in den Statements hinzuschreiben.
Dabei wird der Zeichenfolge ein kurzer Name zugewiesen. In den späteren Statements genügt es dann, anstelle der Zeichenfolge nur noch diesen Namen einzugeben.

Dem Makro ist zunächst ein Text zuzuweisen, beginnend mit dem Zeichen # in der ersten Position einer neuen Zeile, gefolgt von dem Makronamen und dem Text:

#name text...

name Dies ist der Makroname, ein beliebiges Kürzel aus den Buchstaben a ... z, ä, ö, ü, ß, dem Unterstreichungsstrich _ und den Ziffern 0 ... 9. Ein solcher Makroname darf maximal 10 Zeichen lang sein. Auch hier unterscheidet CNTA nicht zwischen Groß- und Kleinbuchstaben.

text... Ein beliebiger Text, der Makrowert.
Erscheint nun in einem späteren Statement der Makroname #name, so wird er automatisch durch den Text *ttt...* ersetzt. Eine solcher Text darf sich über beliebig viele Fortsetzungszeilen erstrecken. Jede Fortsetzungszeile muss mit einem Leerzeichen beginnen. Dieses Leerzeichen wird nicht Bestandteil des Makrowertes.
Der Text *ttt...* kann auch nur aus Leerzeichen bestehen.

Definiert man zum Beispiel die beiden Makros

```
#männer C10.1
#frauen C10.2
```

so kann in späteren Statements anstelle von

```
XTAB ROW=TOTAL;C10.1;C10.2
```

geschrieben werden:

```
XTAB ROW=TOTAL;#männer;#frauen
```

In der Definition eines Makros lassen sich bereits definierte Makros verwenden. So liefern die Statements:

```
#stu   Studie 3874
#kopf  #stu: Käufer von Lackfarben
PAGEP  HEAD='#kopf'
```

das Ergebnis:

```
PAGEP  HEAD='Studie 3875: Käufer von Lackfarben'
```

In einem Verarbeitungslauf kann ein Makro mehrfach neue Texte erhalten. Bei jeder neuen Wertezuweisung geht der vorige Text verloren. Zum Beispiel:

```
#kopf  Studie 3874, Käufer von Lackfarben
PAGE   HEAD='#kopf'
...
#kopf  Studie 3874, Teil 2:
PAGE   HEAD='#kopf'
```

Dabei ist das erste PAGE-Statement gleichwertig mit:

```
PAGE   HEAD='Studie 3874, Käufer von Lackfarben'
```

und das zweite PAGE-Statement liefert das gleiche Ergebnis wie:

```
PAGE   HEAD='Studie 3874, Teil 2:'
```

Ein Makro kann seinen Text auch in mehreren Schritten erhalten. Zum Beispiel:

```
#kopf  Studie 3874
...
#kopf  #kopf, Käufer von Lackfarben
PAGE   HEAD='#kopf'
```

Hierdurch entsteht der Überschriftstext:

```
Studie 3874, Käufer von Lackfarben
```

Bei der Fehlersuche in den Statements kann es hilfreich sein, den aktuellen Wert eines Makros zu kennen.
Im Statement DUMP erlaubt die Angabe MACRO die Anzeige vom Makrowerten im Zählprotokoll.

In einem Makro lassen sich Teile eines Statements speichern, maximal ein ganzes Statement.
Es ist aber nicht möglich, mehrere Statements in ein einzelnes Makro zu stellen.

Makros können auch durch Folgestatements von OUTPUT-INT auf eine interne Datei übernommen werden.
Wird diese interne Datei später mit INPUT-INT eingelesen, so stehen die Makrowerte in der neuen Zählung hinter dem Statement INPUT-INT wieder zur Verfügung.
Mit dem Statement CODEBOOK lassen sich auch die in einer internen Datei gespeicherten Makrowerte ausdrucken.
Makros können auch über das INCLUDE-Statement aus einer anderen Statementdatei eingelesen werden.

Makros mit Argumenten

Makros mit Argumenten werden auf folgende Weise definiert:

#name text... **1** *text...* **2** *text...* **99** *text...*

name Dies ist der Name des Makros, das hier definiert wird.

text... Makrotext, der mit den Argumenten der Form 1 ... 99 in beliebiger Anzahl und Reihenfolge durchsetzt sein kann.

Ein solches Makro ist später in folgender Form aufzurufen:

```
#name(argument1 $$ argument2 $$ ... $$ argument99)
```

Dabei ist *argument1* eine beliebige Zeichenfolge, die das Argument 1 der Makrodefinition ersetzt. Entsprechend ersetzt *argument2* das Argument 2 usw.

Besitzt ein Makro mehrere Argumente, so sind diese bei der Verwendung des Makros durch das Doppelzeichen $$ voneinander zu trennen. Zum Beispiel definiert

```
#rows1  ROW=TOTAL:ABS,$2$;$1$:ABS,$2$
```

das Makro #rows1 mit zwei Argumenten. Dieses Makro kann später in einem XTAB-Statement verwendet werden:

```
XTAB #rows1(C27 $$ 1)...
```

Das ist gleichwertig mit dem Statement:

```
XTAB  ROW=TOTAL:ABS,1;C27:ABS,1 ...
```

Es sind auch leere Argumente möglich. Zum Beispiel:

```
XTAB #cols( )
XTAB #rows1( $$ 1)
XTAB #rows1( $$ )
```

In der Definition eines Makos sind auch bestehende Makros mit Argumenten möglich. Zum Beispiel:

```
#studie Studiennummer: $1$
#kopf #studie($1$).--- $2$
```

Dann liefert das Statement

```
PAGEP HEAD='#kopf(112 $$ Käufer von Lackfarben)
```

das Ergebnis

```
PAGEP HEAD='Studiennummer: 112 --- Käufer von Lackfarben)
```

Makroausdrücke

Aus Makros lassen sich mit logischen und numerischen Operatoren komplexe Ausdrücke bilden.
Diese werden in den unten beschriebenen Wertezuweisungen mit Makroausdrücken sowie in den Statements #IF und PROCESS verwendet.

Zum Beispiel stellt das Statement

```
#c #= #a + #b
```

die Zahl 5 in das Makro #c, sofern Makro #a den Wert 2 und Makro #b den Wert 3 enthält.
Dagegen liefert das Statement

```
#c #a + #b
```

die Zeichenfolge 2 + 3 im Makro #c.

Das Statement

```
#IF (#a + #b)>10
```

sorgt dafür, dass die dahinter stehenden Statements nur dann eingelesen werden, wenn die Summe der Makrowerte aus #a und #b größer als 10 ist.

Makroausdrücke werden aus den folgenden Operanden zusammengesetzt:

#name	Ein beliebiges Makro, zum Beispiel: #anz
k	Numerische Konstante, zum Beispiel: 314.26. Sie kann aus maximal 9 Ziffern bestehen. Als Dezimalzeichen ist der Punkt zu verwenden, das Komma wird nicht akzeptiert. Die Schreibweisen .9 und 9. sind unzulässig, 0.9 und 9.0 sind korrekt.
text	Vergleichswert aus beliebigen Zeichen ohne Leerstellen dazwischen, zum Beispiel: abc1
'text'	Vergleichswert aus beliebigen Zeichen auch mit Leerzeichen dazwischen, in Hochkommas eingeschlossen. Zum Beispiel: `'abc 1234'`

Die folgenden Funktionen sind ebenfalls als Operanden möglich:

#def	Dieser Operand prüft, ob ein Makro definiert ist. So liefert `#def(#xyz)` den Wert 1, wenn das Makro `#xyz` in einem davor liegenden Statement am Anfang einer Zeile steht, andernfalls den Wert 0.
#num	Dieser Operand prüft, ob ein Makro einen numerischen Wert enthält. Für ein beliebiges Makro `#anz` liefert `#num(#anz)` den Wert 1, wenn `#anz` einen numerischen Wert besitzt, dagegen den Wert 0, wenn #anz nicht numerisch oder nicht definiert ist, also nicht in einem davor liegenden Statement am Anfang einer Zeile stand.

Diese Operanden lassen sich durch die folgenden Operatoren miteinander verrechnen:

		Beispiel
+ -	positives und negatives Vorzeichen	`-314` oder `+3.14`
+	Addition	`#anz+75`
-	Subtraktion	`#anz-3`
*	Multiplikation	`#anz*2`
/	Division	`#anz/5`
**	Potenzierung	`#anz**2`

Besitzt ein Makro hierbei einen nicht numerischen Wert, so wird dafür mit dem Wert 0 gerechnet.
Eine Division durch 0 führt zum Ergebnis 0.

Die folgenden Vergleichsoperatoren können in Makroausdrücken verwendet werden:

		Beispiel
=	gleich	`#anz = #max`
<	kleiner	`#ort < kiel`
>	größer	`#name > 'abc 123'`
<=	kleiner oder gleich	`#anz <= 13.4`
>=	größer oder gleich	`#anz >= 13.4`
^= oder ¬=	ungleich	`#anz ^= 0`
¬< oder ^<	nicht kleiner	`#anz ^< 10`
¬> oder ^>	nicht größer	`#anz ^> 10`

Das Ergebnis eines solchen Vergleichs ist die Zahl 1 für *wahr*, wenn die Vergleichsbedingung erfüllt ist oder die Zahl 0 für *falsch*, wenn sie nicht erfüllt ist.

Ist einer der zu vergleichenden Werte nicht numerisch, so werden die beiden Werte zeichenweise verglichen und zwar in der Reihenfolge des ASCII-Codes. Siehe dazu im Abschnitt *Code-Tabellen* die Reihenfolge der Zeichen in der *Zeichenzuordnung im ASCII-Code.*
So liefert der Makroausdruck

```
270<'31'
```

den Wert 1, da das Zeichen 2 aus 270 in der ASCII-Tabelle vor dem Zeichen 3 aus '31' liegt.

Sind beide zu vergleichenden Werte numerisch, so werden die Zahlenwerte miteinander verglichen. Der Makroausdruck

```
170<21
```

liefert den Wert 0, da die Zahl 170 größer als 21 ist.

Die Operanden lassen sich mit den folgenden logischen Operatoren zu komplexen Ausdrücken zusammenfügen:

		Beispiel
&	und	`#anza & #anzb`
| oder !	oder	`#anza | #anzb`
¬ oder ^	nicht	`^#anz`

Das Ergebnis einer solchen logischen Operation ist die Zahl 1, wenn die Bedingung erfüllt ist oder die Zahl 0, wenn sie nicht erfüllt ist.
Besitzt ein Makro einer solchen Operation den Wert 0 oder nur Leerzeichen, so wird dafür *falsch* oder 0 angenommen. Enthält das Makro eine Zahl ungleich 0 oder Textzeichen, so wird dafür *wahr* oder 1 verwendet.

Jeder Makroausdruck kann bis zu 255 Operatoren und Operanden besitzen. Er darf praktisch unbegrenzt viele Klammern (und) enthalten. Die Operatoren eines Ausdrucks werden in dieser Prioritätenfolge abgearbeitet:

hohe Priorität	( )	Klammern um Teilausdrücke
	+ -	Vorzeichen Plus und Minus
	**	Potenzierung
	* /	Multiplikation und Division
	+ -	Addition und Subtraktion
	= < >	Vergleichsoperatoren
	¬ ^	Negation
	&	und
niedrige Priorität	| !	oder

Stehen mehrere Operatoren gleicher Priorität ohne Klammern hintereinander, wird von links nach rechts ausgewertet. So wird der Ausdruck

```
#a1+#a2/10+(#b-10)*#id
```

auf folgende Weise aufgelöst:
Zunächst wird der Quotient #a2/10 ermittelt und zwischengespeichert. Danach wird, der Klammern wegen, die Differenz #b-10 gebildet. Diese wird mit dem Wert aus #id multipliziert und das Ergebnis zwischengespeichert. Schließlich werden die gespeicherten Summanden zum Wert von #a1 addiert

Wertezuweisung mit Makroausdrücken

Die oben beschriebenen Makroausdrücke erlauben eine weitere Form der Wertezuweisung.
Durch die Zeichen #= hinter dem Makronamen wird die dahinter stehende Zeichenfolge als Makroausdruck interpretiert. So stellen zum Beispiel die Statements:

```
#a 3
#b 5
#c #= (#a + #b)*2
```

die Zahl 16 in das Makro #c.

Die Statements

```
#a 3
#b 5
#c (#a + #b)*2
```

stellen dagegen die Zeichenfolge (3 + 5)*2 in das Makro #c.

Anstelle der Wertezuweisung #= sind auch die folgenden Angaben möglich:

```
#+    für Addition
#-    für Subtraktion
#*    für Multiplikation
#/    für Division
```

Diese Angaben verrechnen den Wert des Makroausdrucks mit dem bisherigen Wert des Zielmakros.
So addiert das Statement

```
#c #+ 2
```

die Zahl 2 auf den bestehenden Wert des Makros #c und speichert das Ergebnis im Makro #c.
Als Voraussetzung dafür muss das Makro #c bereits einen numerischen Wert enthalten.
Besitzt #c keinen numerischen Wert, so wird dieses Statement mit einer Fehlermeldung beantwortet.

Ausdruck mit nur einem Makro

Im einfachsten Fall besteht ein Makroausdruck nur aus einem Makro ohne Operatoren.
Ein solcher Ausdruck liefert den Wert 0, wenn das Makro einen nicht numerischen Wert besitzt, sonst aber den numerischen Wert selbst.

So füllt das Statement

```
#x #= #abc
```

das Makro #x mit dem Wert 0, wenn das Makro #abc:
den Wert 0 besitzt,
einen nicht numerischen Wert enthält oder
nicht definiert ist.

Besitzt #abc dagegen einen numerischen Wert, so wird dieser unverändert in das Makro #x übertragen.

Beispiel 1

Folgende Statements definieren die Makros #P1 und #P2:

```
#P1 R(1)ABS (2...16)ABS,1 (17...22)ABS
#P2 R(1)ABS (2...22)PROW1,1,%
```

danach haben die Makroaufrufe:

```
XTAB   PRINT=#P1
       PRINT1=#P2
```

gleiche Wirkung wie:

```
XTAB   PRINT=R(1)ABS (2...16)ABS,1 (17...22)ABS
       PRINT1=R(1)ABS (2...22)PROW1,1,%
```

Beispiel 2

Das folgende Statement definiert das Makro #GB mit einem Argument 1:

```
#GB  Gestützte Bekanntheit und Verwendung von $1$
```

In späteren Statements haben die Makroaufrufe:

```
XTAB  TITLE='#GB(Bondex)'
XTAB  TITLE='#GB(Sikkens)'
```

die gleiche Wirkung wie:

```
XTAB  TITLE='Gestützte Bekanntheit und Verwendung von Bondex'
XTAB  TITLE='Gestützte Bekanntheit und Verwendung von Sikkens'
```

Beispiel 3

Definiert man die beiden Makros:

```
#Männer  C10.1
#Frauen  C10.2
```

so kann später anstelle von

```
XTAB  ROW=TOTAL;C10.1;C10.2
```

geschrieben werden:

```
XTAB  ROW=TOTAL;#Männer;#Frauen
```

Beispiel 4

```
#anz  3
#row1  C10.1;...5;.Z0;.S
#row2  C22.2;...17;.Z0;.S
#row3  C35.1;...4;.Z0;.S
#row4  C51;.Z0;.S
...
#IF   #anz<1  |  #anz>4  |  ^#num( #anz )
Fehler:  Makro  'anz'  falsch
#ELSE
DO  <1>=1...#anz
XTAB  COLS=TOTAL;C1  ROWS=#row<1>
EDO
#EIF
```

Im ersten Statement wird das Makro #anz definiert und mit dem numerischen Wert 3 versehen. Darauf folgt die Definition der Makros #row1 ... #row4 mit verschiedenen Zählausdrücken für ROWS in XTAB-Statements.

Das #IF-Statement prüft den Wert des Makros #anz. Der Makroausdruck in diesem Statement ist erfüllt, wenn #anz kleiner 1, größer 4 oder nicht numerisch ist.
In diesem Fall wird die Zeile Fehler: ... verarbeitet und die Zeilen zwischen #ELSE und #EIF unverarbeitet übersprungen. Da die Zeile Fehler: ... kein gültiges Statement enthält, führt sie zur Fehlermeldung 100 und zum vorzeitigen Ende der Verarbeitung.

Besitzt das Makro #anz jedoch einen numerischen Wert zwischen 1 und 4, so wird die Zeile Fehler: ... unverarbeitet übergangen. Dafür gelangen die Zeilen zwischen #ELSE und #EIF zur Verarbeitung. Im DO-Statement liefert das Makro #anz hier den maximalen Wert 3 für den DO-Index <1>.
Im XTAB-Statement sorgt dieser Index <1> schließlich dafür, dass die Werte der drei Makros #row1 ... #row3 als ROW-Angaben verwendet werden.

Beispiel 5

```
#kont  1  3  5  7  12
#gew  0  0.5  0.8  0.9  1.0

#wk
DO  <1>=#kont  <2>=#gew
#wk  #wk  <1>:<2>
EDO

XTAB  ROW=RCH(N1.5  N5:8  N10:3;  #wk)
```

Hier werden zunächst im Makro #kont die Kontaktklassen für eine Reichweitenzählung angegeben. Das Makro #gew enthält die Gewichte zu diesen Kontaktklassen.

Das Statement

```
#wk
```

vor dem DO-Statement definiert das leere Makro #wk.
Diese Zuweisung eines leeren Wertes kann auch entfallen, wenn das Makro #wk in den davor stehenden Statements noch keinen Wert erhalten hat: Makros ohne Wertezuweisung besitzen stets einen leeren Wert.

Im DO-Statement werden die Makros #kont und #gew durch ihre konkreten Werte ersetzt. In diesem Fall ist es gleichwertig mit:

```
DO <1>=1  3  5  7  12  <2>=0  0.5  0.8  0.9  1.0
```

Direkt hinter dem DO-Statement erscheint das Makro #wk zweimal in einer Zeile.
Das rechts stehende Makro #wk wird zunächst durch seinen bisherigen Wert ersetzt und die dahinter stehende Angabe <1>:<2> angefügt.
Danach erhält das Makro #wk am Anfang der Zeile diesen Wert, der bisherige Inhalt wird durch den rechts stehenden Ausdruck ersetzt.

Bei der Auflösung der DO-Schleife wird das Makro #wk fünfmal mit neuen Werten versehen:

```
#wk  1:0
#wk  1:0 3:0.5
#wk  1:0 3:0.5 5:0.8
#wk  1:0 3:0.5 5:0.8 7:0.9
#wk  1:0 3:0.5 5:0.8 7:0.9 12:1.0
```

Die DO-Schleife hat also die gleiche Wirkung wie die einfache Makrodefinition:

```
#wk  1:0 3:0.5 5:0.8 7:0.9 12:1.0
```

Diese wird dann im Statement XTAB in der Reichweitenfunktion RCH eingesetzt.
Das XTAB ist somit gleichwertig mit folgendem:

```
XTAB ROW=RCH(N1.5 N5:8 N10:3; 1:0 3:0.5 5:0.8 7:0.9 12:1.0)
```

Ein solches Verfahren kann sehr nützlich sein, wenn bestimmte Parameter am Anfang einer Zählung - hier die Werte der Makros #kont und #gew - öfter geändert werden sollen, ohne dabei in die folgenden Statements eingreifen zu müssen.

CNTA.INI CNTW.INI

CNTA.INI ist eine Datei mit individuellen Grundeinstellungen für das Programm CNTA und CNTW.INI die entsprechende Datei für CNTW. Beide lassen sich mit einem beliebigen Texteditor anzeigen und ändern.

Die Datei CNTA.INI wird im Arbeitsverzeichnis CNTAWORK erwartet und CNTW.INI in CNTWWORK. Sind sie nicht vorhanden, so werden sie dort bei Programmbeginn automatisch eingerichtet und mit Standardwerten versehen.

Folgende Eingaben können vom Benutzer nach eigenen Wünschen geändert werden.
Die Reihenfolge spielt keine Rolle, jedoch muss jede Angabe am Anfang einer eigenen Zeile stehen.
Alle hier nicht aufgeführten Angaben sollten nicht verändert werden; sie dienen der Speicherung von Daten zwischen verschiedenen Programmläufen.

CHARSET = | **ASCII** | **ANSI** | **EBCDIC** |

Hiermit wird festgelegt, in welchem Zeichencode die Statements für CNTA vorliegen müssen und das Zählprotokoll CNTLST sowie die Auswertungen über PAGE erstellt werden sollen.
Auf die Ausgabe mit PAGEP hat diese Angabe keinen Einfluss.
Im Abschnitt *Zeichenzuordnung ANSI, ASCII, EBCDIC* im Anhang dieses Handbuches werden diese Zeichencodes ausführlich beschrieben.

Die Angabe CHARSET legt außerdem das Format der externen Dateien INPUT-EXT, OUTPUT-EXT und FETCH-FILE fest, falls sie keine abweichenden Angaben enthalten.

LINES = n Diese Angabe bestimmt die Zeilenanzahl *n = 10 ... 1 000 000* pro Blatt für die Ausgabe der Zählergebnisse mit dem Statement PAGE. Außerdem bestimmt sie die Druckausgabe des Zählprotokolls CNTLST. LINES hat keinen Einfluss auf die Ausgabe über PAGEP.

In den Statementdateien hat die Angabe
```
PAGE LINES=n
```
Vorrang vor der Eintragung `LINES=n` in den INI-Dateien.

LOGOS = Dateiverzeichnis

Die Auswertungen von CNTW können auch Logos enthalten.
Dazu sind entsprechende Bilddateien erforderlich.

Mit LOGOS kann ein spezielles Verzeichnis für diese Logo-Dateien festgelegt werden.
Das Programm sucht die Logos in verschiedenen Verzeichnissen in dieser Reihenfolge:

1. Im Verzeichnis der REP-Datei, wenn eine solche vorhanden ist. Neu vom Programm CNTW angelegte Auswertungen mit dem Namen ~N~.REP befinden sich stets im Verzeichnis *CNTWWORK*.
2. Im Verzeichnis der Statementdatei.
3. Im dem hier durch LOGOS= festgelegten Verzeichnis.
4. Im Programmverzeichnis *CNTW* oder *CNTA*.

MAXWORK = n

Diese Angabe bestimmt die maximal zulässige Größe des Arbeitsbereichs im Hauptspeicher. Für *n* ist die gewünschte Größe zwischen 100 000 bis 2 000 000 Kilobytes anzugeben. Fehlt diese Angabe in der INI-Datei, so wird dafür 1 000 000 angenommen. Mit

```
MAXWORK=100000
```

wird zum Beispiel ein Arbeitsbereich von 100 Megabyte Größe eingerichtet.

Von der Größe des Arbeitsbereichs hängen unter anderem ab:

- Die maximale Größe einer Kreuztabelle.
- Die Anzahl von Wertezuweisungen, die in einer Auswertung möglich sind.
- Die Anzahl von Datendurchläufen für umfangreichere Auswertungen.
- Die Anzahl von Randbedingungen in einem WEIGHT-Statement.

Der auf einer Maschine verfügbare Hauptspeicher wird bei zu groß gewählten Arbeitsbereich vom Betriebssystem durch langsamen Plattenspeicher ergänzt. Es empfiehlt sich, die Angabe MAXWORK etwas unterhalb des tatsächlich verfügbaren Hauptspeichers zu halten. Erzeugen auffällig lang laufende Auswertungen viele Plattenzugriffe, sollte der hier angegebene Wert versuchsweise verkleinert werden.

EDITOR = programm

Diese Angabe ist nur für CNTW wirksam. Sie ermöglicht, aus CNTW heraus einen externen Editor aufzurufen. Gleichzeitig wird der eigene Editor von CNTW deaktiviert. Er zeigt nur noch die Statements an, erlaubt aber keine Eingaben mehr.

Für *programm* ist der volle Programmname des Editors anzugeben, einschließlich Laufwerk und Verzeichnis. Dahinter sind Parameter für den Editor möglich. CNTW stellt zum Programmaufruf des Editors den Namen der Statementdatei hinter diese Angaben. Aus der Eintragung

```
EDITOR = C:\Programme\Edit\Edit.exe -x -a
```

in CNTW.INI könnte so der Programmaufruf

```
C:\Programme\Edit\Edit.exe -x -a auswertung.stm
```

zum Editieren der Statementdatei *auswertung.stm* entstehen.

POS = n

Diese Angabe legt die Zeilenlänge pro Blatt für die Druckausgabe mit dem Statement PAGE fest. Es ist die gewünschte Anzahl von Zeichen pro Zeile anzugeben. Sie muss zwischen 125 und 32 767 liegen. Damit wird die Druckausgabe des Zählprotokolls und der mit dem Statement PAGE erzeugten Auswertungen eingestellt. Diese Angabe hat keinen Einfluss auf die Ausgabe über PAGEP.

Das CNTA-Statement

```
PAGE POS= ...
```

verändert für den aktuellen Verarbeitungslauf die Eintragung `POS=` aus CNTA.INI.

FONT = Schriftname

Schrift für die Bildschirmanzeige.
Zum Beispiel: Arial, Times New Roman, Courier New

FONTSIZE = Punkt

Schriftgröße für die Bildschirmanzeige in Pica-Point mit einer Nachkommastelle. Zum Beispiel: 8.0

FONTTYPE = Typ

Schrifttyp für die Bildschirmanzeige. Auch Kombinationen der Angaben:

N = normal
B = fett
I = kursiv
U = unterstrichen

STATEMENTS = Dateinamen

Diese Angabe dient dazu, Statementdateien schneller am Bildschirm auffinden zu können.

Ohne sie wird das Fenster zur Auswahl von Statementdateien zunächst auf das Verzeichnis CNTWWORK gestellt und als Dateierweiterung die in der jeweiligen Prototypdatei festgelegten Vorgaben verwendet, zum Beispiel STM.

Die Angabe STATEMENTS in der Datei CNTW.INI legt davon abweichende Regeln fest:

Mit der Eintragung

```
STATEMENTS=D:\PROGRAMME\*.PRO
```

wird das Fenster zur Dateiauswahl auf das Verzeichnis D:\PROGRAMME gestellt und nur Dateien mit der Erweiterung PRO angezeigt.

Die Angabe

```
STATEMENTS=D:\PROGRAMME\S*.PRO    *.STM    *.*
```

stellt ebenfalls auf das Verzeichnis D:\PROGRAMME, zeigt aber nur Dateien mit der Erweiterung PRO an, die mit dem Buchstaben S beginnen. Außerdem lassen sich Dateien mit der Erweiterung STM auswählen und durch *.* auch beliebige Dateien.

Die Eintragung

```
STATEMENTS=S*.PRO    *.STM    *.*
```

leistet das Gleiche wie die vorige, jedoch bleibt das Fenster zur Dateiauswahl auf das Verzeichnis CNTWWORK eingestellt.

TRANSLATE i = Dateiname

Diese Angaben sind nur in CNTW.INI sinnvoll. Sie legen für i = 1 ... 5 bis zu fünf Dateien zur maschinellen Sprachübersetzung von Variablen- und Merkmalstexten fest. Zu jedem Sprachpaar ist eine CSV-Datei im ANSI-Code anzugeben, mit dem Semikolon ; als Trennzeichen. Diese Dateien müssen in der ersten Spalte die Texte der Quellsprache und in der zweiten die der Zielsprache enthalten. Außerdem müssen sie mit einer Keyline mit den Namen des Sprachpaares beginnen, gefolgt von dem Bindestrich - und dem zweistelligen Google-Sprachschlüssel.

Für das Sprachpaar Deutsch/Englisch könnte eine solche Datei folgendermaßen beginnen:

```
Deutsch-de          ; Englisch-en
Alter der Befragten ; age of the respondents
```

Texte, die ein Semikolon enthalten, sind in Anführungsstriche " einzuschließen. Anführungsstriche im Text sind als "" anzugeben.

Anstelle von Zahlenwerten lässt sich als Platzhalter das Zeichen * verwenden. Zum Beispiel:

```
Unter * Euro       ; Less than * Euro
* bis unter * Euro ; * to less than * Euro
* Euro und mehr    ; * Euro and more
```

Damit wird der Text

```
1.000 bis unter 2.000 Euro
```

übersetzt in

```
1,000 to less than 2,000 Euro
```

Aufruf des Programms

Start aus der Befehlszeile

Das Programm CNTA kann unter MS/DOS, OS/2 und Windows aus der Befehlszeile (Command Line) durch folgende Eingabe gestartet werden:

cnta name1 name2 name3 

name1 ist dabei der Name der Statementdatei, die die Auswertungsanweisungen enthält.

name2 ist der Name der Druckdatei, in die CNTA seine Druckdaten ausgibt.
In dieser Datei werden zunächst die eingelesenen Statements aufgelistet, zusammen mit eventuellen Fehlermeldungen. Sie enthält weiter Protokolle und Fehlermeldungen zu den verarbeiteten statistischen Fällen und endet mit den eigentlichen Auswertungsergebnissen.

name3 steuert die Bildschirmmeldungen, die Auskunft über den aktuellen Verarbeitungsstand von CNTA geben. Solche Bildschirmausgaben können manchmal störend sein oder die Laufzeiten des Programms verlängern:
Durch Angabe der Zahl `0` für `name3` werden die Meldungen ganz unterdrückt.
name3 kann aber auch ein Dateiname sein. Dann schreibt CNTA seine Meldungen in diese Datei. Während des Programmlaufs gibt diese Datei dann Auskunft über den aktuellen Verarbeitungsstand. Diese Datei enthält nur die neueste Meldung, alte Meldungen werden überschrieben.

Fehlt die Angabe *name3*, so werden diese Meldungen auf den zuständigen Bildschirm ausgegeben. Fehlt zusätzlich die Angabe *name2*, so wird die Druckausgabe in eine Datei mit dem Namen *cntlst* gestellt.
Fehlt die Angabe aller drei Namen, so werden die Statements in einer Datei mit dem Namen *cntipt* erwartet. Wenn *name1* fehlt, besteht keine Möglichkeit *name2* oder *name3* anzugeben.

Alle übrigen Ein- und Ausgabedateien, insbesondere die Dateien mit den auszuwertenden statistischen Fällen, sind in der Statementdatei *cntipt* beziehungsweise *name1* unter INPUT-EXT, INPUT-INT, INPUT-SEC, OUTPUT-EXT, OUTPUT-INT, INCLUDE oder PAGE anzugeben.

Während eines Verarbeitungslaufes legt CNTA noch eine Zwischendatei *strfile* an, die nach der Auswertung überflüssig ist und in der Regel auch von CNTA
selbst wieder gelöscht wird.

Beim Programmende liefert CNTA Return-Codes mit folgender Bedeutung:

0	die Verarbeitung konnte ohne Fehlerabbruch beendet werden.
1	die Verarbeitung wurde wegen eines Fehlers oder eines Benutzereingriffs vorzeitig beendet.

Dieser Return-Code kann in Batch-Prozeduren abgefragt werden.
Unter Windows könnte eine solche Prozedur zum Beispiel so aussehen:

```
@echo off
cnta %1 %2
if errorlevel 1 goto ende
@echo Die Zählergebnisse werden jetzt gedruckt
print cntlst
:ende
```

Start von der Bildschirmoberfläche

Die spezielle Windows-Version von CNTA kann wahlweise aus der Kommandozeile des DOS-Fensters oder von der Bildschirmoberfläche mit der Maus gestartet werden.

Wurde CNTA so installiert, wie oben beschrieben, zum Beispiel mit

```
C:\CNT\CNTA.EXE
```

als Befehlszeile, so meldet sich das Programm nach dem Start mit einem Bildschirmdialog zur Auswahl der Statementdatei.
Als Druckdatei wird dann stets *cntlst* verwendet, und die Bildschirmmeldungen erscheinen in einem eigenen kleinen Fenster.

Es ist auch möglich, den Namen der Statementdatei, der Druckdatei oder der Meldungsdatei bereits in der Befehlszeile vorzugeben, zum Beispiel:

```
C:\CNT\CNTA.EXE cntipt cntlst 0
```

In diesem Fall werden die Statements stets aus der Datei `cntipt` eingelesen, die Druckausgabe erfolgt in `cntlst`, und es werden keine Bildschirmmeldungen ausgegeben.

Beispiel 1

Der Aufruf für einen typischen Verarbeitungslauf mit CNTA könnte folgendermaßen aussehen:

```
cnta  0085.stm  0085.lst  0
```

Die Datei *0085.stm* enthält dabei die zu verarbeitenden Statements.
Die Druckausgabe wird auf die Datei *0085.lst* geschrieben.
Durch die Zahl 0 werden die Konsolmeldungen ganz unterdrückt.

Beispiel 2

Der Aufruf

```
cnta
```

ist gleichwertig mit der Eingabe:

```
cnta  cntipt cntlst
```

Statements

CNTA

25. April 2018

CNTA-Handbuch
Stand 25. April 2018

Eulenkrugstraße 83
22359 Hamburg

Tel: (0 40) 6 03 05 62
E-Mail: info@wesselhoeft.de

Reihenfolge der Statements

Es gibt die folgenden Gruppen von Statements. Die Statements sind stets in der Reihenfolge dieser Gruppen einzugeben. OPTIONS muss zum Beispiel vor allen anderen Statements liegen und INPUT-EXT vor DEFS.

Statements aus der gleichen Gruppe dürfen jedoch beliebig untereinander vertauscht werden:
Es ist einerlei, ob INPUT-EXT vor INPUT-INT liegt oder umgekehrt.
Die Auswertung der Statements erfolgt in der Reihenfolge, in der sie eingegeben werden. Lediglich OUTPUT-EXT mit seinen Folgestatements ganz zum Schluss verarbeitet, egal wo diese Statements eingegeben wurden.

1. Allgemeine Verarbeitungsregeln: OPTIONS [1]

2. Auswahl der Eingabedateien: INPUT-INT [1]
   INPUT-EXT [1]

3. Definition von Variablen und Texten: DEFS [1]

4. Sekundäre interne Dateien: INPUT-SEC

5. Verarbeitungsregeln für Fälle der externen Datei: ADD-PROC [1]
   UPD-PROC [1]

6. Verarbeitungsregeln für alle Fälle: MOD-PROC [1]

7. Fusionieren einer internen Datei: FUSION

8. Auswahl von Fällen: SELECT [1]

9. Wichtung von Fällen: WEIGHT

10. Auswertungen: CODEBOOK
    HOLECOUNT
    PAGE
    PAGEP
    TEXT
    XTAB
    XADD
    INDEX
    OUTPUT-INT [1]
    OUTPUT-EXT [1]

Folgende Statements dürfen an beliebiger Stelle erscheinen:

DO / EDO
FETCH-FILE
REPORT-FILE
INCLUDE
LANG
PRINT
PROCESS
Makrodefinitionen
Kommentare

[1] Dieses Statement ist in jedem Verarbeitungslauf nur einmal möglich.

Regeln zur Beschreibung der Statements

Die Statements für CNTA werden mit einem beliebigen Editor- oder Textprogramm in eine Datei geschrieben.

Ein CNTA-Statement beginnt mit einem Kennwort wie INPUT-EXT, DEFS, XTAB in der ersten Position einer Zeile. Eine Statementzeile darf beliebig lang sein und beliebig viele Fortsetzungszeilen besitzen, die an einer Leerstelle in der ersten Position erkannt werden.

Auf das Kennwort eines Statements können verschiedene andere Angaben folgen. Für diese Angaben ist keine besondere Reihenfolge erforderlich. Sie können auch zur besseren Lesbarkeit durch beliebig viele Leerzeichen voneinander getrennt oder auf verschiedene Fortsetzungszeilen verteilt werden.

Einige Statements, wie zum Beispiel DEFS oder ADD-PROC beginnen mit einem Leitstatement, das das Kennwort enthält, zu dem aber weitere Folgestatements gehören. Jedes Folgestatement muss dabei in einer neuen Zeile beginnen und einen Strich - am Anfang der ersten Zeile besitzen.

Die Statements und ihre Angaben können nach Belieben in Groß- oder Kleinschrift eingegeben werden, auch wenn in diesem Handbuch dafür nur Großbuchstaben verwendet werden.

Die Statements werden in diesem Handbuch nach folgenden Regeln beschrieben:

Angaben in einem Statement, die zwingend erforderlich sind, werden nicht besonders gekennzeichnet. Zum Beispiel:

INPUT-EXT **FNAME =** nnn

Dies bedeutet, dass im Statement INPUT-EXT immer die Angabe FNAME erforderlich ist.

Angaben in fetter Schrift sind unverändert im Statement zu verwenden, wahlweise in Groß- oder Kleinschrift. Angaben in normaler Schrift sind in den Statements durch geeignete Informationen zu ersetzen. Im Statement INPUT-EXT ist *nnn* hinter FNAME durch einen geeigneten Dateinamen zu ersetzen. Zum Beispiel:

```
INPUT-EXT    FNAME=studie087.ext.
```

Angaben eines Statements, die nicht zwingend erforderlich sind, werden in spitze Klammern < und > eingeschlossen. Zum Beispiel:

INPUT-EXT < **NSORT** >

Die Angabe NSORT kann im Statement INPUT-EXT also nach Bedarf angegeben oder weggelassen werden.

Ist in einem Statement unter mehreren Angaben eine Auswahl zu treffen, so werden die zulässigen Alternativen übereinander geschrieben und durch senkrechte Striche begrenzt. Zum Beispiel:

INPUT-EXT < **ASCII** | **ANSI** | **EBCDIC** | **COLBIN** | **QUANTUM** < >

Hier besagen die spitzen Klammern zunächst, dass die Angabe ASCII, ANSI, EBCDIC, COLBIN oder QUANTUM nicht zwingend erforderlich ist. Da diese fünf Angaben übereinander geschrieben und durch senkrechte Striche begrenzt sind, kann eine dieser Angaben ausgewählt werden. Dann darf aber nur diese eine angegeben werden.

Kommentarstatements

***** Statementzeilen mit einem Stern * in der ersten Position sind Kommentarzeilen.
Ihr Inhalt wird nicht verarbeitet, sondern nur im Zählprotokoll ausgedruckt.
Sie dienen zur Dokumentation und besseren Lesbarkeit der Statements.
Liegt hinter einer solchen Kommentarzeile eine Fortsetzungszeile mit einer Leerstelle in der ersten Position, so ist diese Zeile nicht Teil des Kommentars sondern gehört zum davor liegenden Statement.

% Statements mit einem Prozentzeichen % in der ersten Position sind Kommentarstatements.
Neben der ersten Zeile gehören auch alle Fortsetzungszeilen mit einer Leerstelle in der ersten Position zum Kommentar.

%% Schließlich können auch Kommentare in normale Statementzeilen gestellt werden:
Erscheinen direkt hintereinander zwei Prozentzeichen %% in einer Zeile, so versteht CNTA den dahinter stehenden Rest der Zeile als Kommentar, der nicht weiter verarbeitet werden soll.

Beispiel

```
INPUT-INT FNAME=test.int

% XTAB
     COLS=TOTAL;C1.1;...8
     ROWS=TOTAL;C11.1;...20

XTAB
     COLS=TOTAL;C1.1;...8      %% dies sind die Kopfbedingungen
     ROWS=TOTAL;C11.1;...20    %% dies sind die Zeilenbedingungen
* die folgenden Angaben steuern die Druckaufbereitung der Zählergebnisse:
     PRINT=ABS PRINT1=R(2...-1)PROW1
```

Diese Statements bilden einen vollständigen Verarbeitungslauf mit CNTA.

Das Statement *INPUT-INT* beschreibt die Datei mit den auszuwertenden statistischen Fällen.

Das Prozentzeichen in der Zeile *% XTAB* bewirkt, dass dieses Statement zusammen mit seinen beiden Fortsetzungszeilen als Kommentar unverarbeitet übergangen wird.

Die Zeile *XTAB* ohne % wird normal verarbeitet und erzeugt zusammen mit ihren beiden Fortsetzungszeilen eine Kreuztabelle. In diesen Fortsetzungszeilen leiten die Zeichenfolgen *%%* Zeilenkommentare ein: Die dahinter stehenden Texte dienen nur der Erläuterung und werden von CNTA nicht weiter verarbeitet
.
Die Zeile ** die folgenden ...* hinter dem zweiten XTAB beginnt mit einem Stern * in der ersten Position; sie ist eine einzelne Kommentarzeile. Die darauf folgende Zeile *PRINT = ...* ist nicht mehr Teil dieses Kommentars sondern eine Fortsetzungszeile zum davor stehenden XTAB Statement.

ADD-PROC

Das Statement ADD-PROC legt fest, wie statistische Fälle zu verarbeiten sind, für die nur Daten aus INPUT-EXT vorliegen jedoch keine Daten mit gleicher Fall-Identifikation aus INPUT-INT oder INPUT-SEC.
ADD-PROC ist insbesondere dann wirksam, wenn nur ein Statement INPUT-EXT angegeben ist und kein INPUT-INT oder INPUT-SEC.

Für die Bearbeitung statistischer Fälle, zu denen Daten mit gleicher Fall-Identifikation aus INPUT-INT und INPUT-EXT vorliegen, ist das Statement UPD-PROC zuständig.
ADD-PROC und UPD-PROC können niemals wirksam werden.

In einem Verarbeitungslauf darf das Statement ADD-PROC nur einmal vorkommen.
Es besitzt selbst keine weiteren Angaben, dafür aber beliebig viele Folgestatements.
Diese müssen direkt hinter ADD-PROC liegen und mit einem Strich - am Anfang der Zeile beginnen.
Die Folgestatements sind in der Reihenfolge einzugeben, in der sie später ausgeführt werden sollen:

–Cn **=** Merkmalswerte
: Die Bedingungsvariable Cn erhält Merkmalswerte zugewiesen.

–Nn **=** numerischer Wert
: Die numerische Variable Nn erhält einen Wert zugewiesen.

–Tn = Textwert
: Die Textvariable Tn erhält einen Wert zugewiesen.

Genauere Angaben zu diesen Folgestatements befinden sich im Abschnitt Wertezuweisungen.

–DO –EDO
: Beginn und Ende einer Schleife: Die zwischen DO und EDO liegenden Statements werden mehrfach ausgeführt. Siehe DO-Statement.

–IF logischer Ausdruck **–EIF**
: Die Verarbeitung der Statements zwischen IF und EIF hängt ab vom Wert des logischen Ausdrucks. Siehe IF-Statement.

–LANG Sprachschlüssel zur Auswahl von Variablentexten. Siehe LANG-Statement.

–PRT <Variable> <Text>
: Dieses Statement druckt Variablenwerte und feste Texte aus. Siehe PRT-Statement.

–REPORT Ausgabe in Textdatei. Siehe REPORT-Statement.

–CLEAN logischer Ausdruck <**HEAD=**'text'>
: Dieses Statement dient zur Bereinigung fehlerhafter Variablenwerte durch hinter CLEAN liegende Wertezuweisungen. Diese Wertezuweisungen werden nur ausgeführt, wenn der logische Ausdruck erfüllt ist. Im Zählprotokoll gibt eine CLEAN-Statistik eine Übersicht über die ausgeführten Korrekturen.

–TST logischer Ausdruck <Variable> <Text>
: Dieses Statement prüft für die einzelnen statistischen Fälle, ob der logische Ausdruck erfüllt ist. Wenn nicht, wird der Text zusammen mit den Werten der angegebenen Variablen als Fehlermeldung ausgedruckt. Siehe TST-Statement.

–END < **CASE** | **ALL** >

Dieses Statement beendet die Verarbeitung des laufenden Falles in ADD-PROC.
Die hinter END stehenden Folgestatements werden nicht mehr ausgeführt.
Ohne weitere Angaben wird die Verarbeitung des laufenden Falles hinter den Folgestatements von ADD-PROC fortgesetzt.
CASE beendet darüber hinaus die Verarbeitung des laufenden Falles durch alle noch folgenden Statements.
ALL bricht die gesamte Verarbeitung vorzeitig ab.

Darüber hinaus können aber auch die Daten in den gerade eingelesenen Datensätzen durch Folgestatements von ADD-PROC verändert oder überschrieben werden. Siehe Abschnitt Externe Datenfelder und Wertezuweisungen. Die entsprechenden Statements haben folgende Form:

–A ... **=** Textwert
Es wird Text in den gerade eingelesenen Datensatz gestellt.

–AK ... **=** Textwert
Es wird Text in den gerade eingelesenen Datensatz einer CSV-Datei gestellt, mit Schlüsselwort und Gleichheitszeichen davor.

–AS ... **=** Textwert
Es wird Text in den gerade eingelesenen Datensatz einer CSV-Datei gestellt.

–B ... **=** Merkmalswerte
Es werden Merkmalswerte als einzelne Bits in den gerade eingelesenen Datensatz gestellt.

–D ... **=** numerischer Wert
Es wird ein numerischer Wert als Dezimalzahl in den gerade eingelesenen Datensatz gestellt.

–DK ... **=** numerischer Wert
Es wird ein numerischer Wert als Dezimalzahl in den gerade eingelesenen Datensatz einer CSV-Datei gestellt, mit Schlüsselwort und Gleichheitszeichen davor.

–DS ... **=** numerischer Wert
Es wird ein numerischer Wert als Dezimalzahl in den gerade eingelesenen Datensatz einer CSV-Datei gestellt.

–F ... **=** numerischer Wert
Es wird eine Dualzahl mit Vorzeichen (Integer) in den gerade eingelesenen Datensatz gestellt.

–G ... **=** numerischer Wert
Es wird eine Dualzahl ohne Vorzeichen (unsigned Integer) in den gerade eingelesenen Datensatz gestellt.

–I ... **=** Merkmalswerte
Es werden Merkmalswerte als positive Dualzahlen in den gerade eingelesenen Datensatz gestellt.

–K... **=** Merkmalswerte
Es werden Merkmalswerte im Column-Binary-Format in den gerade eingelesenen Datensatz gestellt.

–L ... **=** Merkmalswerte
Es werden Merkmalswerte 1, ... 9, 0, -, & in den gerade eingelesenen Datensatz gestellt.

–M ... **=** Merkmalswerte
Es werden Merkmalswerte in Form von Dezimalzahlen in den gerade eingelesenen Datensatz gestellt.

–MK ... **=** Merkmalswerte
Es werden Merkmalswerte in Form von Dezimalzahlen in den gerade eingelesenen Datensatz einer CSV-Datei gestellt , mit Schlüsselwort und Gleichheitszeichen davor.

–MS ... **=** Merkmalswerte
Es werden Merkmalswerte in Form von Dezimalzahlen in den gerade eingelesenen Datensatz einer CSV-Datei gestellt.

–P ... **=** numerischer Wert
Es wird ein numerischer Wert als gepackte Dezimalzahl in den gerade eingelesenen Datensatz gestellt.

–U ... **=** Merkmalswerte
Es werden Merkmalswerte als Zeichen 0 und 1 in den gerade eingelesenen Datensatz gestellt.

–UK ... **=** Merkmalswerte
Es werden Merkmalswerte als Zeichen 0 und 1 in den gerade eingelesenen Datensatz einer CSV-Datei gestellt, mit Schlüsselwort und Gleichheitszeichen davor.

–US ... **=** Merkmalswerte

Es werden Merkmalswerte als Zeichen 0 und 1 in den gerade eingelesenen Datensatz einer CSV-Datei gestellt

Weitere Angaben über das Zusammenspiel von ADD-PROC mit anderen Statements befinden sich im Abschnitt *Einlesen der statistischen Fälle.*

Beispiel 1

```
INPUT-EXT   FNAME=EINGABE.EXT
DEFS
-C1:7
-C2:4
ADD-PROC
-C1=L1(1..7)
-IF  C1.1
-  C2=K1(8,4)
-EIF
```

Im Statement INPUT-EXT wird eine externe Eingabedatei angefordert. Es liegt kein Statement INPUT-INT vor. Deshalb werden alle statistischen Fälle der Datei durch ADD-PROC und seine Folgestatements verarbeitet.

Das erste Folgestatement weist der Variablen C1 Werte aus einem L-Feld der eingelesenen Datensätze zu. Die weiteren Statements sorgen dafür, dass die Variable C2 genau dann mit Werten versehen wird, wenn das Merkmal C1.1 erfüllt ist.
C2 erhält dann vier Merkmalswerte aus dem K-Feld. Dieses K-Feld befindet sich in den Bytes 8 und 9 der gerade eingelesenen Datensätze.

Beispiel 2

```
INPUT-EXT  FNAME=EINGABE.EXT   ID=D(1..4)
INPUT-INT  FNAME=EINGABE.INT
ADD-PROC
-N7=D1(5..8)
```

Durch das Statement INPUT-EXT wird eine externe Datei und durch das Statement INPUT-INT eine Datei im internen CNT-Format angefordert. Um die Datensätze aus den beiden Dateien verarbeiten zu können, müssen alle Eingabesätze eine Fall-Identifikation besitzen.

Liegt ein Fall aus INPUT-EXT zur Verarbeitung vor, ohne einen passenden Fall aus INPUT-INT mit gleicher Fall-Identifikation, so wird er durch ADD-PROC und seine Folgestatements verarbeitet. Nur dann wird der Variablen N7 ein Wert aus dem D-Feld des eingelesenen Datensatzes zugewiesen.

Wird dagegen ein passender Fall auf INPUT-INT gefunden, so regelt das Statement UPD-PROC die Verarbeitung.

–CLEAN –CLEANN –ECLEAN

–CLEAN | < *logischer Ausdruck* > < **HEAD = '***text***'** >
–CLEANN |

– *Folgestatements*

–ECLEAN

Diese Folgestatements zu ADD-PROC, MOD-PROC, UPD-PROC und OUTPUT-EXT dienen dazu, Variablenwerte zu prüfen, zu bereinigen und dazu eine Statistik in das Zählprotokoll auszugeben.

Dazu lassen sich beliebig komplizierte Prüfbedingungen als logische Ausdrücke formulieren.
Die hinter CLEAN liegenden Folgestatements werden nur ausgeführt, wenn die Prüfbedingung erfüllt ist.
Andernfalls wird die Verarbeitung beim nächsten CLEAN- oder ECLEAN-Statement fortgesetzt.
Die hinter CLEANN liegenden Folgestatements werden nur ausgeführt, wenn die Prüfbedingung *nicht* erfüllt ist.
Das ECLEAN-Statement beendet die CLEAN-Funktionen und führt zur normalen Verarbeitung ohne Protokollierung zurück.

Hinter einem CLEAN- oder CLEANN-Statement sind alle Folgestatements zulässig, ausgenommen Wertezuweisungen mit externen Datenfeldern. Insbesondere sind DO- IF- PRT- und TST-Statements möglich. Nach ECLEAN und weiteren Folgestatements sind neue CLEAN- oder CLEANN-Statements möglich. Das erlaubt zum Beispiel die Bereinigung von Daten direkt nach ihrer Eingabe aus externen Dateien in die Variablen.

Wertezuweisungen hinter den CLEAN-Statements dienen zur Korrektur fehlerhafter Variablenwerte. Zu diesen Korrekturen werden im Zählprotokoll zwei Tabellen ausgegeben. Die erste zeigt die Anzahl der Korrekturen pro CLEAN-Statement mit den dazugehörigen Wertezuweisungen. Sie hat folgenden Aufbau:

```
Statements                                  Anzahl     Korrekturen
-------------------------------------------------------------------
   21  CLEAN C1.Z0...................    27  17.2%    6        3.8%
   22    C1.3=1.......................                    5   3.8%
   23    N1=5.........................                    4   2.5%
-------------------------------------------------------------------
Gesamt                                  27           6
                                                         9
```

Die zweite Tabelle liefert die Anzahl von Korrekturen pro Fall-Identifikation:

```
   Fall-ID   Korrekturen   CLEAN-Statement        Statement
-------------------------------------------------------------------------
       433   3
             2             20 CLEAN C1.1 & C0.1
                                                  21 N1=5
                                                  22 C2.1=1
             1             23 CLEAN ^C1.1 & C0.1
                                                  24 N1=10
                                                  25 C2=1
       434   8
```

---

logischer Ausdruck

Bei CLEAN werden die direkt dahinter liegenden Statements ausgeführt, wenn der logische Ausdruck erfüllt ist, bei CLEANN, wenn er nicht erfüllt ist.
Fehlt dieser logische Ausdruck, so wird lediglich die Angabe HEAD= ... verarbeitet.

HEAD='*text***'** Dieser Text erscheint als Überschrift über der CLEAN-Statistik im Zählprotokoll.
Gleichzeitig beendet diese Angabe die CLEAN-Statistik für davorstehende CLEAN-Statements und beginnt eine neue CLEAN-Statstik mit diesem Text als Überschrift.
Auf diese Weise lassen sich verschiedene Bereinigungsschritte im Zählprotokoll kennzeichnen.
Ohne diese Angabe wird ein Standardtext als Überschrift verwendet und alle CLEAN-Statements erscheinen in derselben CLEAN-Statistik.

Beispiel 1

```
ADD-PROC
-CLEAN C1.1 & C0.1
-  N1=5
-  C2.1=1
-CLEAN ^C1.1 & C0.1
-  N1=10
-  C2=1
-ECLEAN
```

Das erste CLEAN-Statement prüft, ob die beiden Merkmale C1.1 und C0.1 gleichzeitig erfüllt sind. In diesem Fall verlangen die beiden folgenden Wertezuweisungen, dass die Variable N1 den Wert 5 besitzt und C2.1 auf 1 steht. Ist das nicht der Fall, so werden diese Variablenwerte mit entsprechenden Angaben im Zählprotokoll korrigiert.

Anschließend prüft das zweite CLEAN, ob das Merkmal C0.1 erfüllt ist und C1.1 nicht erfüllt ist.
In diesem Fall muß die Variable N1 den Wert 10 besitzen und C2 den Wert 1.

Zu diesen Statements könnte das Zählprotokoll die beiden folgenden Listen der Korrekturen ausweisen.
Die erste zeigt die Anzahl der Korrekturen pro Statement und die zweite die Korrekturen pro Fall-Identifikation.

```
                                  ========================
                                  CLEAN-Protokoll ADD-PROC
                                  ========================
Statements                                                                    Anzahl      Korrekturen
-------------------------------------------------------------------------------------------------------
   18  CLEAN c1.1 & c0.1.....................................     2   3.9%     2       3.9%
   19    n1=5.....................................................                    1   2.0%
   20    c2.1=1...................................................                    2   3.9%
   21  CLEAN ^c1.1 & c0.1.....................................    50  98.0%    49      96.1%
   22    n1=10....................................................                   49  96.1%
   23    c2=1.....................................................                    2   3.9%
-------------------------------------------------------------------------------------------------------
Gesamt                                                                        52          51
                                                                                          54

          ========================
          CLEAN-Protokoll pro Fall
          ========================
   Fall-ID   Korrekturen   Clean-Statement       Statement
-------------------------------------------------------------------------------------------------------
       444   4
             2             18 CLEAN c1.1 & c0.1
                                                 19 n1=5
                                                 20 c2.1=1
             2             21 CLEAN ^c1.1 & c0.1
                                                 22 n1=10
                                                 23 c2=1
       411   2
             2             21 CLEAN ^c1.1 & c0.1
                                                 22 n1=10
                                                 23 c2=1
       433   1
             1             18 CLEAN c1.1 & c0.1
                                                 20 c2.1=1
```

Beispiel 2

```
MOD-PROC
-DO SEP=';' <1>=Schritt 1: Bereinigung ; Schritt 2: Kontrolle
-  CLEAN HEAD='<1>'
-  CLEAN C1.Z0
-    C1.3=1
-    N1=5
-    T1='keine'
-  CLEAN C1.2 & N1^=8
-    N1=8
-  ECLEAN
-EDO
```

Dieses Beispiel zeigt CLEAN-Statements innerhalb einer DO-Schleife, die zweimal durchlaufen wird.
Im ersten Durchlauf erhält der Index <1> als Wert den Text *Schritt 1: Bereinigung* . Daraus erzeugt das erste CLEAN-Statement die Überschrift im CLEAN-Bericht des Zählprotokolls.
Im zweiten Durchlauf erhält <1> den Wert *Schritt 2: Kontrolle*. Das erste CLEAN-Statement erzeugt daraus eine neue Überschrift im Zählprotokoll. Dadurch wird der CLEAN-Bericht des ersten DO-Durchlaufs abgeschlossen und ein neuer Bericht begonnen.

Da der zweite Durchlauf die gleichen Bereinigungsschritte ausführt wie der erste, sollte er keine Korrekturen mehr ausweisen. Ist das doch der Fall, so haben spätere CLEAN-Statements neue Fehler erzeugt.

Das Zählprotokoll dazu könnte so aussehen:

```
                                  ======================
                                  Schritt 1: Bereinigung
                                  ======================
Statements                                                                    Anzahl      Korrekturen
-------------------------------------------------------------------------------------------------------
   19  CLEAN c1.1 & c0.1.....................................     2   3.9%     2       3.9%
   20    n1=5...................................................                    1   2.0%
   21    c2.1=1.................................................                    2   3.9%
   22  CLEAN ^c1.1 & c0.1.....................................    50  98.0%    49      96.1%
   23    n1=10..................................................                   49  96.1%
   24    c2=1...................................................                    2   3.9%
-------------------------------------------------------------------------------------------------------
Gesamt                                                                  52           51
                                                                                        54

                                   ====================
                                   Schritt 2: Kontrolle
                                   ====================
Statements                                                                    Anzahl      Korrekturen
-------------------------------------------------------------------------------------------------------
   19  CLEAN c1.1 & c0.1.....................................     2   3.9%     1       2.0%
   20    n1=5...................................................                    1   2.0%
   21    c2.1=1.................................................                    -       -
   22  CLEAN ^c1.1 & c0.1.....................................    50  98.0%     1       2.0%
   23    n1=10..................................................                    1   2.0%
   24    c2=1...................................................                    -       -
-------------------------------------------------------------------------------------------------------
Gesamt                                                                  52           2
                                                                                        2

          ========================
          CLEAN-Protokoll pro Fall
          ========================
   Fall-ID   Korrekturen   Clean-Statement       Statement
-------------------------------------------------------------------------------------------------------
       444   6
             2             19 CLEAN c1.1 & c0.1
                                                 20 n1=5
                                                 21 c2.1=1
             2             22 CLEAN ^c1.1 & c0.1
                                                 23 n1=10
                                                 24 c2=1
             1             19 CLEAN c1.1 & c0.1
                                                 20 n1=5
```

CODEBOOK

Das Statement CODEBOOK liefert eine leicht lesbare Übersicht über die Variablen, Textelemente, Variablengruppen und Makros eines Verarbeitungslaufs sowie eine Kurzauswertung der Variablenwerte.

CODEBOOK zeigt alle Angaben eines Verarbeitungslaufes, egal ob sie aus INPUT-INT stammen oder hinter DEFS neu definiert wurden. Durch Folgestatements kann die Auswertung auf bestimmte Variable, Textelemente, Variablengruppen oder Makros beschränkt werden.

Eine Statementdatei darf mehrere CODEBOOK-Statements enthalten. Damit ist es möglich, in einem Datendurchlauf verschiedene Übersichtsauswertungen zu erstellen.

Eine solche CODEBOOK-Liste hat einen eigenständigen Wert, erleichtert aber insbesondere die Erstellung weitergehender Auswertungen etwa durch XTAB-Statements.

CODEBOOK ist darüber hinaus nützlich bei der Verarbeitung vertexteter interner Dateien. Alle erforderlichen Angaben sind bereits im internen Datenbestand enthalten, einschließlich der Variablendefinitionen, Texte usw. Wird ein solcher Datenbestand ohne weitere Angaben mit CODEBOOK ausgewertet, so entsteht automatisch eine vollständige Dokumentation der Daten der jeweiligen Untersuchung.

Zu CODEBOOK sind folgende Angaben möglich:

WEIGHT = Nx

Die Auswertung der Variablen erfolgt mit einem Fallgewicht.
Dazu wird für jeden statistischen Fall der Wert der Variablen Nx verwendet.

SCALE = x

Diese Angabe dient der Hochrechnung der Auswertungsergebnisse.
Dazu werden alle Fallzahlen mit dem konstanten Projektionsfaktor x multipliziert.

HEAD = 'text' < **,L**i >

Diese Angabe stellt einen Überschriftstext an den Anfang der CODEBOOK-Auswertung.
Die Überschrift wird zusätzlich zur Blattüberschrift des PAGE-Statements gedruckt, aber nur auf der ersten Seite des Inhaltsverzeichnisses.

'text' Der Text, der als Überschrift gedruckt werden soll.
Er kann auch aus mehreren Zeilen bestehen, wie im Abschnitt *Textzeilen* beschrieben.
Ohne diesen Text wird das Wort *CODEBOOK* als Überschrift ausgegeben.

Li Diese Angaben L1 bis L16 aus dem Statement PAGEP erzeugen verschiedene Linien oberhalb und unterhalb der Überschriften. Ohne Angabe wird die Linie L5 verwendet. Mit L0 lassen sich die Linien entfernen.

SUPPRESS = < **EMPTY** > < **, ITEMS** > < **, VAR** > < **,Z** >

Durch diese Angabe können einzelne Bestandteile aus einer CODEBOOK-Auswertung entfernt werden, um sie kleiner oder übersichtlicher zu machen, wenn nicht alle Angaben benötigt werden.

EMPTY Diese Angabe entfernt alle Zeilen, zu denen keine Zählergebnisse aus dem ausgewerteten Datenbestand vorliegen.

ITEMS Diese Angabe entfernt alle Merkmalstexte aus der CODEBOOK-Liste.

VAR Diese Angabe entfernt alle Variablentexte aus der CODEBOOK-Liste.

Z Diese Angabe entfernt die Anzahlwerte Z0, Z1, Z2 und Z3 aus der Auswertung von C-Variablen.

STYLE = < **HEAD : FT**n> < **,SUMMARY : FTi** > < **,VAR : FT**n > < **,ITEMS : FT**n > < **,NUM : FT**n **>**

Mit diesen Angaben ist es möglich, die Schriftarten der verschiedenen Teile der CODEBOOK-Liste zu verändern. Sie sind aber nur wirksam, wenn diese Liste auf einen Seitendrucker ausgegeben wird.
(Siehe dazu beim PAGEP-Statement).

FTn Schriftart FONTn aus dem Statement PAGEP für die davor angegebenen Texte.

HEAD legt die Schriftart für die Überschrift vor der CODEBOOK-Liste fest.
Ohne diese Angabe wird die Überschrift mit der Schrift FONT5 gedruckt.

SUMMARY legt die Schriftart für die statistischen Werte direkt unter der Überschrift fest.
Ohne diese Angabe wird dafür die Schrift FONT4 verwendet.

VAR legt die Schriftart für die Variablendefinitionen, Variablentexte und die Textelemente fest. Ohne diese Angabe wird dafür die Schrift FONT3 verwendet.

ITEMS legt die Schriftart für die Merkmalstexte fest.
Ohne diese Angabe wird die Schrift FONT2 verwendet.

NUM legt die Schriftart für die Zählergebnisse fest.
Ohne diese Angabe wird die Schrift FONT2 verwendet.

TVAL = < **,MIN** > < **,MAX** > < **,FREQ** >

Diese Angabe wählt die für Textvariable auszuweisenden Werte aus. Dabei wird stets die Anzahl der statistischen Fälle mit einer Angabe ermittelt. Fälle ohne Angabe (mit leerem Text) erscheinen in einer zweiten Zeile.

MIN Zu den T-Variablen wird die Länge des kürzesten Textes ermittelt.
Leere Texte werden dabei nicht berücksichtigt.

MAX Zu den T-Variablen wird die Länge des längsten Textes ermittelt.

FREQ Zu den T-Variablen werden alle vorkommenden Texte mit ihrer Häufigkeit ausgewiesen.
Unterscheiden sich zwei Texte nur in Groß- und Kleinbuchstaben, so werden sie als gleich angesehen.

NVAL = < **MEAN** > < **,MIN** > < **,MAX** > < **,STD** > < **,FREQ** >

Diese Angabe wählt die zu ermittelnden Werte für numerische Variablen aus.
Dabei wird stets die Anzahl der statistischen Fälle mit einem Zahlenwert ausgewiesen. Fälle ohne Angabe (mit dem Wert Z0) erscheinen in einer zweiten Zeile. Der Wert 0 wird als Zahlenwert verstanden.

MEAN Zu den N-Variablen wird der Mittelwert über alle statistischen Fälle ermittelt. Fälle ohne Angabe (mit dem Wert Z0) werden dazu nicht ausgewertet.

MIN Zu den N-Variablen wird der kleinste numerische Wert aus allen statistischen Fällen ermittelt. Fälle ohne Angabe (mit dem Wert Z0) werden dabei nicht berücksichtigt.

MAX Zu den N-Variablen wird der größte numerische Wert aus allen statistischen Fällen ermittelt. Fälle ohne Angabe (mit dem Wert Z0) werden dabei nicht berücksichtigt.

STD Zu den N-Variablen wird die Standardabweichung über alle statistischen Fälle ermittelt. Fälle ohne Angabe (mit dem Wert Z0) werden dazu nicht ausgewertet.

FREQ Zu den N-Variablen werden alle vorkommenden Werte mit ihrer Häufigkeit ausgewiesen.

Fehlt die Angabe NVAL im Statement CODEBOOK, so wird mit NVAL = MEAN,MIN,MAX gearbeitet.

SELECT = *logischer Ausdruck*

Nur Fälle, die den logischen Ausdruck erfüllen, nehmen an der Auswertung teil.
Zum Beispiel wählt

```
SELECT=C10.1
```

nur denjenigen Fälle aus, die die Bedingung C10.1 erfüllen.

Folgestatements

Liegen direkt hinter CODEBOOK keine Folgestatements mit einem Strich am Zeilenanfang, so werden alle Variablen, Textelemente und Variablengruppen der laufenden Verarbeitung ausgewertet. Mit den folgenden Statements lassen sich einzelne Variable, Textelemente, Variablengruppen und Makros zur Auswertung auswählen.

–Cn zur Auswahl einer Bedingungsvariablen Cn.

–Cm **. .** n zur Auswahl aller zwischen Cm und Cn definierten Bedingungsvariablen.

–Cn**[** i **]** zur Auswahl einer Bedingungsvariablen Cn mit Index i.

–Cn**[** i **] . .** j zur Auswahl aller Bedingungsvariablen mit Index zwischen Cn[i] und Cn[j].

–Nn zur Auswahl der numerischen Variablen Nn.

–Nm **. .** n zur Auswahl aller zwischen Nm und Nn definierten numerischen Variablen.

–Nn**[** i **]** zur Auswahl einer numerischen Variablen Nn mit Index i.

–Nn**[** i **] . .** j zur Auswahl aller numerischen Variablen mit Index zwischen Nn[i] und Nn[j].

–Tn zur Auswahl der Textvariablen Tn.

–Tm **..** n zur Auswahl aller zwischen Tm und Tn definierten Textvariablen.

–Tn**[** i **]** zur Auswahl einer Textvariablen Tn mit Index i.

–Tn**[** i **] . .** j zur Auswahl aller Textvariablen mit Index zwischen Tn[i] und Tn[j].

–(n**)** zur Auswahl des Textelements (n).

–(m**) . .** n zur Auswahl aller zwischen (m) und (n) definierten Textelemente.

–Gn zur Auswahl einer Variablengruppe Gn.

–Gm **. .** n zur Auswahl aller zwischen Gm und Gn definierten Variablengruppen.

–MACROS Mit dieser Angabe werden alle Makros des Verarbeitungslaufs mit den Werten ausgedruckt, die sie an diesem Punkt besitzen. Mit dem Statement –MACROS können auch einzelne Makros angefordert werden. So gibt das Statement

```
-MACROS #col3 #col7 #row2..#row12
```

nur die Makros *#col3, #col7* sowie alle zwischen *#row2* und *#row12* definierten Makros aus.

Die Reihenfolge dieser Folgestatements legt auch die Reihenfolge fest, in der die Variablen in der Druckausgabe erscheinen. Liegen solche Folgestatements nicht vor, so werden erst die C-Variablen, danach die N-Variablen, die T-Variablen und schließlich die Textelemente ausgegeben. Diese Ausgabe erfolgt nach aufsteigenden Variablen- und Indexnummern.

Neben der Auswahl von mehreren Variablen in der Form

```
-C1...20
```

sind auch die Wildcard-Zeichen * und ? möglich. Das Statement

```
-NA??
```

gibt alle N-Variablen aus, die vier Zeichen lang sind und mit NA beginnen und

```
-NA*
```

alle N-Variablen, die mit NA beginnen.

Beispiel: Einfache CODEBOOK-Auswertung

```
INPUT-INT  FNAME=testdaten.int
PAGEP PCHAR='.'
CODEBOOK
```

Diese Statements können folgende Auswertung erzeugen:

CODEBOOK

Anzahl Fälle			4594
C10:2			
1	Männer	992	21,6%
2	Frauen	3602	78,4%
Z1	eine Angabe	4594	100,0%
N20:3	**Alter**		
	Fälle mit Angaben	4591	99,9%
	Fälle ohne Angaben	3	0,1%
	Minimum		19
	Maximum		89
	Mittelwert		35,7
T70:30	**Land des Hauptwohnsitzes**		
	Fälle mit Angaben	111	2,4%
	Fälle ohne Angaben	4483	97,6%
	kleinste Länge		2
	größte Länge		28
(70)	**Frage 12: Wenn Sie im Ausland leben: Bitte geben Sie den Namen des Landes ein**		

INPUT-INT fordert einen internen Datenbestand zur Auswertung an.
Ein solcher enthält neben den Daten der statistischen Fälle die Variablendefinitionen mit Texten.

Da CODEBOOK keine Folgestatements besitzt, werden alle Variable der internen Datei ausgewertet.

Hier ist C10 eine kategoriale Variable mit zwei Ausprägungen und N20 eine numerische Variable, die das Alter der Testpersonen als Zahlenwert enthält. Die Textvariable T70 enthält eine Länderangabe als unverschlüsselten Text und (70) ist ein Textelement, das sich zum Beispiel zur Beschriftung von Tabellen verwenden lässt.

Im Statement PAGEP erzeugt PCHAR = '.' die punktierten Linien.

Beispiel: Zweispaltige Auswertung

```
INPUT-INT FNAME=testdaten.int
PAGEP  HOR=*,L3-*
CODEBOOK  WEIGHT=N1
-C10
-C2030
-N20
-T70
-(70)
```

Diese Statements könnten folgende Auswertung erzeugen:

CODEBOOK

Anzahl Fälle			4594
Anzahl Fälle gewichtet			4597
Fallgewicht			N1

C10:2

1	Männer	985	21,4%
2	Frauen	3612	78,6%
Z1	eine Angabe	4597	100,0%

C2030:6 Schulabschluss

1	Volksschule, Hauptschule	106	2,3%
2	Weiterführende Schule ohne Abitur	629	13,7%
3	Abitur	593	12,9%
4	Studium ohne Abschluss	244	5,3%
5	Studium mit Abschluss	2981	64,8%
6	kein Abschluss	46	1,0%
Z1	eine Angabe	4597	100,0%

N20:3 Alter

Fälle mit Angaben	4594	99,9%
Fälle ohne Angaben	4	0,1%
Minimum		19
Maximum		89
Mittelwert		35,7

T70:30 Land des Hauptwohnsitzes

Fälle mit Angaben	107	2,3%
Fälle ohne Angaben	4490	97,7%
kleinste Länge		2
größte Länge		28

(70) Frage 12: Wenn Sie im Ausland leben: Bitte geben Sie den Namen des Landes ein

Im Statement PAGEP sorgt die Angabe HOR=*,L3-* für die Einteilung der Seiten in zwei Spalten mit der Linie L3 dazwischen.

Die Angabe WEIGHT=N1 im Statement CODEBOOK sorgt für eine Auswertung mit Fallgewichten aus der Variablen N1.

Die Folgestatements hinter CODEBOOK wählen einzelne Variable für die Auswertung aus.

Beispiel: Anzeige der vorhandenen Variablenwerte

```
INPUT-INT FNAME=testdaten.int
CODEBOOK NVAL=FREQ TVAL=FREQ
-N20
-T70
```

Diese Statements können folgende Auswertung erzeugen:

CODEBOOK

Anzahl Fälle 1064

N20:3	**Alter**			
	Fälle mit Angaben	1064	100,0%	
	21	8	0,8%	0,8%
	22	22	2,1%	2,8%
	23	35	3,3%	6,1%
	24	62	5,8%	11,9%
	25	111	10,4%	22,4%
	26	147	13,8%	36,2%
	27	186	17,5%	53,7%
	28	217	20,4%	74,1%
	29	276	25,9%	100,0%
T70:30	**Land des Hauptwohnsitzes**			
	Fälle mit Angaben	18	1,7%	
	Fälle ohne Angaben	1046	98,3%	
	Griechenland	1	0,1%	0,1%
	Indien	1	0,1%	0,2%
	Laos	1	0,1%	0,3%
	Luxemburg	1	0,1%	0,4%
	Österreich	5	0,5%	0,8%
	österreich	1	0,1%	0,9%
	Russland	1	0,1%	1,0%
	Schweiz	3	0,3%	1,3%
	Slowenien	1	0,1%	1,4%
	USA	2	0,2%	1,6%
	Vereinigte Arabische Emirate	1	0,1%	1,7%

CODEBOOK kann alle in der Eingabedatei vorhandenen Werte von numerischen Variablen und die vorkommenden Texte von Textvariablen zusammen mit ihrer Häufigkeit ausweisen. Dazu dient die Angabe NVAL=FREQ für numerische Variable und TVAL=FREQ für Textvariable.

DEFS

Das Statement DEFS leitet die Definition von Variablen und Textelementen ein. Es besitzt selbst keine weiteren Angaben, kann jedoch beliebig viele Folgestatements haben.

Die Folgestatements von DEFS
definieren neue Variable,
versehen bereits definierte Variable mit neuen Texten,
definieren feste Texte als Textelemente,
weisen Zähloperanden TOTAL, Z0 usw. Texte zu und
erzeugen Variablengruppen für die Bildschirmanzeige in CNTW.

Jedes Folgestatement beginnt mit einem Strich - in der ersten Stelle der Zeile. Zum Beispiel definiert

```
-C10:2 'Geschlecht'
     1='Männer'
     2='Frauen'
```

eine Bedingungsvariable mit zwei Merkmalen.

Definition von Variablen

Jede Variable darf nur einmal definiert werden. Eine bereits definierte, zum Beispiel aus INPUT-INT eingelesene Variable, kann hinter DEFS jedoch noch mit neuen Texten versehen werden. Die Anzahl der Merkmale, Dezimalziffern, Nachkommastellen und Zeichenanzahl einer alten Variablen kann dann allerdings nicht mehr geändert werden.

Definition von neuen Variablen sind an einem Doppelpunkt hinter dem Variablennamen zu erkennen.
Hinter diesem Doppelpunkt werden die Eigenschaften und Texte der Variablen angegeben:

–Cn**:** m	variablentext	**F**u	var	**DUP**	s **=** merkmalstext	Bedingungsvariable
–Nn**:** d,f	variablentext	**F**u	var	**DUP**		numerische Variable
–Tn**:** z	variablentext	**F**u	var	**DUP**		Textvariable
–Cn**[** i **]:** m	variablentext	**F**u	var	**DUP**	s **=** merkmalstext	Bedingungsvariable mit Index
–Nn**[** i **]:** d,f	variablentext	**F**u	var	**DUP**		numerische Variable mit Index
–Tn**[** i **]:** z	variablentext	**F**u	var	**DUP**		Textvariable mit Index

Bestehende Variable erhalten ohne Doppelpunkt dahinter auf folgende Weise neue Texte:

–Cn	variablentext	**F**u	var	**DUP**	s **=** merkmalstext	Bedingungsvariable
–Nn	variablentext	**F**u	var	**DUP**		numerische Variable
–Tn	variablentext	**F**u	var	**DUP**		Textvariable
–Cn**[** i **]**	variablentext	**F**u	var	**DUP**	s **=** merkmalstext	Bedingungsvariable mit Index
–Nn**[** i **]**	variablentext	**F**u	var	**DUP**		numerische Variable mit Index
–Tn**[** i **]**	variablentext	**F**u	var	**DUP**		Textvariable mit Index

d maximale Anzahl 1 ... 18 von Dezimalziffern, die die Werte der numerischen Variablen insgesamt vor und hinter dem Komma besitzen dürfen. Hier ist zu bedenken, dass Programme wie SPSS und Excel lediglich 16 Dezimalziffern verarbeiten können. Bei der Datenübergabe mit OUTPUT-EXT werden Werte mit mehr als 16 Ziffern von diesen Programme gerundet.

f Anzahl Nachkommastellen 0 ... 18 der Variablenwerte.
Fehlt diese Angabe, so werden die Variablenwerte ohne Nachkommastellen gespeichert.
Die Anzahl Nachkommastellen *f* darf nicht größer als die Anzahl Dezimalstellen *d* sein.

i Nummer 1 ... 99 999 des höchsten zulässigen Index der neu zu definierenden Variablen.

k Nummer 1 ... 99 999 des Index, dessen Variable einen neuen Text erhalten soll.

m Anzahl Merkmale 1 ... 10 000 einer neu definierten C-Variablen.
Dies ist gleichzeitig die höchste zulässige Merkmalsnummer der Variablen.

n Der Name der Variablen, die hier neu definiert oder mit neuen Angaben versehen werden soll. Diese Namen dürfen aus den Buchstaben A....Z, den Ziffern 0 ... 9 sowie dem Zeichen _ bestehen. Sie müssen mit dem Buchstaben C, N oder T beginnen und können bis zu 10 Zeichen lang sein. Endet ein Variablenname mit einer Zahl, so ergeben führende Nullen keine neue Variable. Daher bezeichnen N0001, N001, N01 und N1 dieselbe Variable.

s Nummer 1 ... 10 000 des Merkmals, dem Text zugewiesen werden soll.
Der Text 't' ist hinter s= einzugeben. Durch die Angabe von s= ohne Text dahinter kann ein bereits vorhandener Merkmalstext wieder entfernt werden.

merkmalstext

Text, wie im Abschnitt *Textzeilen* beschrieben, der dem Merkmal s zugewiesen wird.
Ist eine Variable bereits auf INPUT-INT mit Merkmalstexten definiert, so kann ein solcher Merkmalstext durch die Angabe s= ohne Text dahinter wieder entfernt werden.

Fu Fußnotennummer 1 ... 99 999. Durch diese Angabe wird der Variablen oder einem Merkmal das Textelement (u) als Fußnotentext zugeordnet. Vorher muss das Textelement (u) definiert worden sein. Zum Beispiel wird mit den Statements:

```
-(23)  'Keine Vergleichswerte aus dem Vorjahr.'
-N10:3 'Haushaltseinkommen'  F23
```

allen Auswertungen der Variablen N10 der Fußnotentext *Keine Vergleichswerte aus dem Vorjahr*. hinzugefügt.
Ganz entsprechend kann auch den Merkmalen einer C-Variablen ein fester Fußnotentext zugeordnet werden. Zum Beispiel:

```
-C10:6 'Haushaltseinkommen'  1='bis 580 DM' F23
```

In allen durch XTAB erzeugten Kreuztabellen druckt CNTA automatisch den hier zugeordneten Fußnotentext am Ende der Seite, sobald die entsprechende Variable oder das Merkmal in der Tabelle erscheinen.
Ist eine Variable bereits auf INPUT-INT mit einer Fußnote definiert, so kann diese Definition durch die Angabe F0 rückgängig gemacht werden. Die Fußnote wird dann für den aktuellen Verarbeitungslauf aus der Variablen entfernt.

variablentext

Textzeilen, wie im Abschnitt *Textzeilen* beschrieben, die der Variablen als Variablentext zugewiesen werden sollen.

Cx
Nx
Tx Durch diese Angabe werden die Texte der Variablen Cx, Nx oder Tx in die laufende Variable übernommen. Zusätzliche Variablentexte v oder Merkmalstexte t haben Vorrang vor den aus den Variablen Cx, Nx oder Tx kopierten Texten.

z maximale Zeichenanzahl 1 ... 4 000 der Textwerte der neu definierten T-Variablen.

DUP Vor dem Einlesen eines neuen statistischen Falles werden die Werte aller definierten Variablen zunächst gelöscht. Dadurch stehen die Werte des vorigen Falles für den aktuellen Fall nicht mehr zur Verfügung. Durch die Angabe von DUP zu einer Variablen wird dieses Löschen verhindert. Stattdessen behält sie den Wert des vorigen Falles, bis ihr ein neuer Wert zugewiesen wird.
Ein solcher Wert kann daher auch über beliebig viele statistische Fälle erhalten bleiben.
DUP ist nur für den aktuellen Verarbeitungslauf wirksam: Wird eine solche Variable auf eine interne Datei übernommen, so verliert sie dort die Angabe DUP und damit die Befähigung, einen Wert aus dem vorigen statistischen Fall beizubehalten.
DUP kann nur bei der Definition einer neuen Variablen angegeben werden. Es ist nicht möglich, einer von INPUT-INT eingelesenen Variablen diese Eigenschaft zu geben.

Definition von Textelementen

Mit Folgestatements von DEFS können auch Textelemente definiert werden.
Sie bestehen aus den in runde Klammern gesetzten Zahlen (0) ... (99999), denen ein Text zugeordnet wird.
Solche Textelemente sind insbesondere von Nutzen, wenn der gleiche Text mehrmals in einer Auswertung erscheint: Dann genügt es, diesen Text einmal hinter DEFS einem Textelement (n) zuzuordnen und ihn später beliebig oft nur noch durch Angabe der Nummer (n) aufzurufen. Ein bereits definiertes Textelement kann bei der Definition eines anderen Textelements verwendet werden.
Textelemente werden wie Variable über OUTPUT-INT auf interne Datenbestände ausgegeben und später von dort wieder eingelesen.
Ein bereits auf INPUT-INT definiertes Textelement kann später unter DEFS einen neuen Text erhalten; dabei geht der alte Text verloren.

–(n**)** 'text'

Dieses Statement definiert für n = 0 ... 99999 das Textelement (n) und ordnet ihm die Textzeilen *'text'* zu, so wie im Abschnitt *Textzeilen* beschrieben.
Führende Nullen sind dabei möglich: (0001), (001), (01) und (1) bezeichnen dasselbe Textelement.

Zum Beispiel definiert das Folgestatement

```
-(25)  'Frage 25:'/'Wie gefällt Ihnen dieses Plakat?'
```

das Textelement (25) und weist ihm einen zweizeiligen Text zu.
Später kann dieses Textelement verwendet werden:

```
XTAB  TITLE=(25)/'(vorher noch nicht gesehen)'//
```

Der TITLE-Text in diesem XTAB-Statement besteht dann zunächst aus den beiden Zeilen des Textelements (25). Es wird eine dritte Textzeile dahinter gestellt, gefolgt von zwei Leerzeilen.

Texte für Zähloperanden

Verschiedenen Zähloperanden lässt sich ein ein- oder mehrzeiliger Text *'text'* zuweisen, so wie im Abschnitt Textzeilen beschrieben. Diese Texte werden in den Kreuztabellen XTAB an Stelle der Operanden gedruckt.
Zum Beispiel bewirkt das Statement:

```
-TOTAL 'Gesamt'
```

dass in den Kreuztabellen für den Operanden TOTAL stets der Text *Gesamt* ausgegeben wird
In den folgenden Statements kann der Text *'text'* auch weggelassen werden. Damit wird ein bereits vorhandener Text des Operanden entfernt.

Die Texte der Zähloperanden werden über OUTPUT-INT stets auch auf die internen Dateien ausgegeben und stehen später bei der Auswertung solcher Dateien automatisch zur Verfügung.

Folgende Zähloperanden können in Folgestatements von DEFS mit Texten versehen werden:

–CIDi 'text'

Mit dieser Angabe wird der Text der Bedingungen CID.B ... CID4.B für den Anfang einer Gruppe und CID.E ... CID4.E für das Ende einer Gruppe festgelegt.
In den Tabellen werden für den Operanden CID.B folgende Texte ausgegeben:

CID ohne Text	CID mit Text *Gruppen*
CID	***Gruppen***

–IDi 'text'

Die Fall-Identifikation ID ... ID4 erhält einen Text.
In den Tabellen werden für den Operanden ID1 folgende Texte ausgegeben:

ID1 ohne Text	ID1 mit Text *Summe*
ID1	***Summe***

–MAX 'text'

Den Maximalwerten MAX(Cn) und MAX(Nn) wird ein Text zugewiesen.
In den Tabellen werden für den Operanden MAX(C11) folgende Texte ausgegeben:

	MAX ohne Text	MAX mit Text *Maximum*
C11 ohne Variablentext	***MAX(C11)***	***Maximum***
C11 mit Variablentext *Einkommen*	***MAX(C11)***	***Einkommen Maximum***

–MEAN 'text'

Den Mittelwerten MEAN(Cn) und MEAN(Nn) wird ein Text zugewiesen.
In den Tabellen werden für den Operanden MEAN(C11) folgende Texte ausgegeben:

	MEAN ohne Text	MEAN mit Text *Mittelwert*
C11 ohne Variablentext	***MEAN(C11)***	***Mittelwert***
C11 mit Variablentext *Einkommen*	***MEAN(C11)***	***Einkommen Mittelwert***

–MED 'text'

Dem Median MED(Cn) wird ein Text zugewiesen.
In den Tabellen werden für den Operanden MED(C11) folgende Texte ausgegeben:

	MED ohne Text	MED mit Text *Median*
C11 ohne Variablentext	***MED(C11)***	***Median***
C11 mit Variablentext *Einkommen*	***MED(C11)***	***Einkommen Median***

–MIN 'text'

Den Minimalwerten MIN(Cn) und MIN(Nn) wird ein Text zugewiesen.
In den Tabellen werden für den Operanden MIN(C11) folgende Texte ausgegeben:

	MIN ohne Text	MIN mit Text *Minimum*
C11 ohne Variablentext	***MIN(C11)***	***Minimum***
C11 mit Variablentext *Einkommen*	***MIN(C11)***	***Einkommen Minimum***

–MODL 'text'

Den Modalwerten MODL(Cn) und MODL(Nn) wird ein Text zugewiesen.
In den Tabellen werden für den Operanden MODL(C11) folgende Texte ausgegeben:

	MODL ohne Text	MODL mit Text *Modalwert*
C11 ohne Variablentext	***MODL(C11)***	***Modalwert***
C11 mit Variablentext *Einkommen*	***MODL(C11)***	***Einkommen Modalwert***

–MODU 'text'

Den Modalwerten MODU(Cn) und MODU(Nn) wird ein Text zugewiesen.
In den Tabellen werden für den Operanden MODU(C11) folgende Texte ausgegeben:

	MODU ohne Text	MODU mit Text *Modalwert*
C11 ohne Variablentext	***MODU(C11)***	***Modalwert***
C11 mit Variablentext *Einkommen*	***MODU(C11)***	***Einkommen Modalwert***

–PERCTL 'text'

Den Minimalwerten PERCTL(Cn ...) und PERCTL(Nn ...) wird ein Text zugewiesen.
In den Tabellen werden für den Operanden PERCTL(C11,90) folgende Texte ausgegeben:

	PERCTL ohne Text	PERCTL mit Text *Perzentil*
C11 ohne Variablentext	***PERCTL(C11,90)***	***Perzentil***
C11 mit Variablentext *Einkommen*	***PERCTL(C11,90)***	***Einkommen Perzentil***

–STD 't'

Den Standardabweichungen STD(Cn) und STD(Nn) wird ein Text zugewiesen.
In den Tabellen werden für den Operanden STD(C11) folgende Texte ausgegeben:

	STD ohne Text	STD mit Text *Abweichung*
C11 ohne Variablentext	***STD(C11)***	***Abweichung***
C11 mit Variablentext *Einkommen*	***STD(C11)***	***Einkommen Abweichung***

–STE 'text'

Den Standardfehlern STE(Cn) und STE(Nn) wird ein Text zugewiesen.
In den Tabellen werden für den Operanden STE(C11) folgende Texte ausgegeben:

	STE ohne Text	STE mit Text *Fehler*
C11 ohne Variablentext	***STD(C11)***	***Fehler***
C11 mit Variablentext *Einkommen*	***STD(C11)***	***Einkommen Fehler***

–TOTAL 'text'

Durch diese Angabe erhält der Operand TOTAL für die XTAB-Statements einen Text zugewiesen.
In den Tabellen werden für den Operanden TOTAL folgende Texte ausgegeben:

TOTAL ohne Text	TOTAL mit Text *Basis*
TOTAL	***Basis***

–FS 'text'
–FP 'text'

Dem Operanden Cn.F für C-Variable werden Texte zugewiesen.
In den Tabellen werden für den Operanden C11.F folgende Texte ausgegeben:

	FS und FP ohne Text	FS mit Text *Nennung von* FP mit Text *Nennungen von*
C11 ohne Variablentext	***1 Angabe von 6*** ***2 Angaben von 6***	***1 Nennung von 6*** ***2 Nennungen von 6***
C11 mit Variablentext *Einkommen*	***Einkommen*** ***1 Angabe von 6*** ***2 Angaben von 6***	***Einkommen*** ***1 Nennung von 6*** ***2 Nennungen von 6***

–S 'text'

Der Summenoperand Cn.S für C-Variable erhält einen Text zugewiesen.
In den Tabellen werden für den Operanden C11.S folgende Texte ausgegeben:

	S ohne Text	S mit Text *Summe*
C11 ohne Variablentext	***C11.S***	***Summe***
C11 mit Variablentext *Einkommen*	***Einkommen*** ***S***	***Einkommen*** ***Summe***

–Zi 'text'

Jedem der Anzahloperanden Z0, Z1 ... Z123 von C-Variablen lässt sich ein eigener Text zuweisen.
In den Tabellen werden für den Operanden C11.Z0 folgende Texte ausgegeben:

	Z0 ohne Text	Z0 mit Text *keine Angabe*
C11 ohne Variablentext	***C11.Z0***	***keine Angabe***
C11 mit Variablentext *Einkommen*	***Einkommen*** ***Z0***	***Einkommen*** ***keine Angabe***

–KOMP 'text'

Diese Angabe weist dem Operanden KOMP für die XTAB-Statements einen Text zu.
In den Tabellen werden für den Operanden KOMP folgende Texte ausgegeben.

KOMP ohne Text	KOMP mit Text *Sonstige*
KOMP	***Sonstige***

–QUOTED 'text'
–NOTQUOTED 'text'

Diese Angaben ändern die Value-Label von dichotomen Variablen in SPSS-Dateien.
Sie gelten nur für die Ausgabe im U-Format mit den Labeltypen IV und VI.
QUOTED ersetzt den Standardtext *gewählt* und NOTQUOTED den Standardtext *nicht gewählt*. Zum Beispiel:

```
-QUOTED    'genannt'
-NOTQUOTED 'nicht genannt'
```

Definition von Variablengruppen

Variable lassen sich zu Gruppen zusammenfassen und mit einem beschreibenden Gruppentext versehen.
Diese Gruppen gliedern die Anzeige der Variablen in den Variablenfenstern von CNTW, wobei die Gruppentexte als Zwischenüberschriften erscheinen. So fasst das Statement

```
-G1 'Statistik' C1..3 N10
```

die Variablen C1, C2, C3 und N10 zur Gruppe G1 zusammen, die CNTW unter der Überschrift Statistik anzeigt.

Variablengruppen lassen sich hierarchisch gliedern. Zum Beispiel

```
-G1   'Statistik'
-G1.1   'Befragte' C1..3
-G1.2   'Haushalt' C10..20
```

Hier liefert G1 die Hauptüberschrift Statistik, der direkt keine Variablen zugewiesen werden.
Stattdessen erhält sie die Untergruppen G1.1 mit der Überschrift Befragte und G1.2 mit Haushalt, denen hier die Variablen zugeordnet werden.

Die Variablengruppen können mit Folgestatements von OUTPUT-INT in interne Dateien übernommen werden.
Außerdem werden sie in den CODEBOOK-Auswertungen angezeigt.

Eine bereits definierte Variablengruppe lässt sich später in einem neuen G-Statement mit einem anderen Text versehen. Enthält dieses neue G-Statement keine Variablen, so bleiben die alten Variablenzuweisungen erhalten. Enthält es dagegen Variable, so gehen die alten Zuweisungen verloren.

Mit Hilfe der LANG-Statements und solcher Neu-Definitionen lassen sich den Gruppen auch Texte in mehreren Sprachen zuweisen.

Zur Definition der Variablengruppen dienen die Folgestatements von DEFS:

–Gs_1 <**.**s_2 **.**s_3 ... **.**s_9 > **'t'** < **C**m . . . n > < **N**m . . . n> <**T**m . . . n > < **(**m**)** . . . n >

s_1 Es ist eine Zahl 1 ... 255 für die oberste Gruppenstufe anzugeben.
Zur Definition der Gruppe G3 müssen die Gruppen G1 und G2 nicht vorhanden sein.

.s_2 ... **.**s_9 Jede untere Gruppenstufe beginnt mit einem Punkt, gefolgt von einer beliebigen Zahl 1 ... 255.
Untere Gruppenstufen lassen sich hinter DEFS erst definieren, wenn die dazugehörigen oberen Stufen bereits definiert sind:
–G3.2 kann erst dann angegeben werden, wenn –G3 bereits vorliegt, entweder aus INPUT-INT oder aus einer davor liegenden Definition. Zur Definition von –G3 .2 muss –G3.1 nicht vorhanden sein.

't' Der ein- oder mehrzeilige Text, der im Variablenfenster von CNTW als Zwischenüberschrift vor den Variablen dieser Gruppe angezeigt werden soll. Es gelten die im Abschnitt *Textzeilen* beschriebenen Regeln. Textelemente (n) können als Zwischenüberschriften nicht verwendet werden. Gruppen ohne Text sind nicht zulässig.

Cm ... n
Nm ... n
Tm ... n
(m**)** ... n

Es sind die Variablen anzugeben, die der Gruppe angehören sollen. Sie können einzeln oder in Von-Bis-Form aufgeführt werden. Ist C1 eine Variable mit Index, so genügt die Angabe C1, um alle Untervariablen in die Gruppe aufzunehmen.
In der Form C1[7] oder C1[10] ... 40 lassen sich auch einzelne Indizes herausgreifen.
Die Variablen werden in der Reihenfolge in den Variablenfenstern angezeigt, in der sie in den G-Statements aufgeführt sind.
Eine Variable kann mehreren Gruppen angehören.
Gruppen ohne Variable und Textelemente sind zulässig. Sie liefern lediglich Zwischenüberschriften für die Variablenfenster.

Beispiel: Bedingungsvariable

```
DEFS
-C10:3
-C11:3  'Einkommen'  1='bis 2000 EUR'
                     2='2001-4000 EUR'
                     3='über 4000 EUR'
-C12:3  'Einkommen'
-C13:3               1='bis 2000 EUR'
                     2='2001-4000 EUR'
                     3='über 4000 EUR'
```

Diese Statements definieren die Bedingungsvariablen C10 ... C13 mit jeweils drei Merkmalen.

C10 erhält keinen Text; in den Auswertungen wird diese Variable mit ihren Merkmalen in der Form ausgewiesen:

C10.1 C10.2 C10.3

C11 erhält den Variablentext *Einkommen* und zusätzlich Texte zu den Merkmalen. So wird in den Auswertungen das Merkmal C11.1 in der Form ausgewiesen:

Einkommen
bis 2000 EUR

C12 besitzt nur den Variablentext *Einkommen* und keinen Merkmalstext.
Die Auswertungsprogramme drucken für das erste Merkmal dieser Variablen:

Einkommen
1

C13 schließlich besitzt keinen Variablentext, wohl aber Merkmalstexte.
Merkmal 1 dieser Variablen wird in den Auswertungsprogrammen gedruckt als:

bis 2000 EUR

Beispiel: Numerische Variable

```
DEFS
-N0:5
-N1:6,2  'Monatseinkommen'/'netto in EUR'
```

Diese Statements definieren die numerischen Variablen N0 und N1.

N0 ist definiert zur Aufnahme von numerischen Werten mit maximal 5 Dezimalstellen. Die Variable kann also Werte zwischen -99999 und +99999 aufnehmen. Zusätzlich ist bei allen numerischen Variablen der künstliche Wert Z0 für *keine Angabe* möglich. Es sind keine Nachkommastellen definiert. Diese werden beim Füllen der Variablen durch Rundung entfernt. (Kaufmännische Rundung: Ab 0.5 wird auf die nächsthöhere ganze Zahl aufgerundet).

N0 erhält keinen Text zugewiesen. In den Auswertungen erscheint die Variable daher als:

N0

N1 wird mit 6 Dezimalstellen definiert, davon 2 Nachkommastellen. Die Variable kann also Werte zwischen -9999,99 und +9999,99 aufnehmen. N1 wird in den Auswertungen gekennzeichnet durch den Variablentext:

Monatseinkommen
netto in EUR

Beispiel: Textvariable

```
DEFS
-T10:20  'Vorname des Befragten'
-T11:30  'Nachname des Befragten'
-T12:30  'Wohnort'
```

Diese Statements definieren die Textvariablen T10...T12. Sie sollen Name und Wohnort freien Text speichern.

T10 dient zur Speicherung des Vornamens mit maximal 20 Zeichen. T11 kann 30 Zeichen des Nachnamens aufnehmen und T12 bis zu 30 Zeichen für den Wohnort.
T-Variable können alle Arten von Zeichen aufnehmen: Buchstaben, Ziffern und Sonderzeichen.

Beispiel: Textelemente

```
DEFS
-(110) 'FRAGE 11:'/'Können Sie mir sagen, wie hoch das
monatliche'-
      'Nettoeinkommen'/'in diesem Haushalt ist?'/
      'Ich meine damit das Einkommen, das alle im Haushalt' -
      'lebenden'/'Personen zusammen im Monat verdienen.'
```

Dieses Statement definiert das Textelement (110). Es wird auf folgende Weise in den Auswertungen ausgedruckt:

FRAGE 11:
Können Sie mir sagen, wie hoch das monatliche Nettoeinkommen
in diesem Haushalt ist?
Ich meine damit das Einkommen. das alle im Haushalt lebenden

Beispiel: Zähloperanden

```
DEFS
-Z0     'keine Angabe'
-S      'Summe der Angaben'
-TOTAL        'Basis'/'in Mio'
-C11:3 'Einkommen'
        1='bis 2000 €' 2='2001-4000 €' 3='über 4000 €'
```

Diese Statements weisen den Zählausdrücken Z0, S und TOTAL Texte zu.
Anstelle von TOTAL wird dadurch in den Auswertungen ausgewiesen:

Basis
in Mio

Für die Variable C11.Z0 wird ausgegeben:

Einkommen
keine Angabe

und für C11.S:

Einkommen
Summe der Angaben

Beispiel: Texte aus anderen Variablen

Die Texte einer Variablen können auf einfache Weise in andere Variablen übernommen werden. Die Statements:

```
DEFS
-C8..10:3  'Einkommen'  1='bis  2000 €'
                        2='2001-4000 €'
                        3='über 4000 €'
-N10:5  C10
-C11:4  C10  3='4001-5000 €'
             4='über 5000 €'
-C12:3  'Nettoeinkommen'      C10
```

definieren zunächst die Variablen C8, C9 und C10, alle mit dem Variablentext *Einkommen*, alle mit drei Merkmalen und den gleichen Merkmalstexten.
Danach erhält die Variable N10 den Variablentext *Einkommen* aus der Variablen C10.
Die Variable C11 erhält ebenfalls den Variablentext *Einkommen*.
Weiter werden die Texte der Merkmale 1 und 2 aus C10 übernommen, während die Merkmale 3 und 4 neue Texte erhalten.
Für die Variable C12 schließlich werden die Merkmalstexte der Variablen C10 übernommen, jedoch ein neuer Variablentext vergeben.

Beispiel: Variablengruppen

Mit den folgenden Statements werden Variablengruppen definiert und auf die interne Datei BEISPIEL7.INT ausgegeben. Diese Gruppen verbessern die Anzeige der Variablen in den Variablenfenstern von CNTW.

```
DEFS
...
-G1    'Statistik'             C1 .. 99
-G2    'Konsumgewohnheiten'    C1001 .. 1099   C2001   C300   N19
-G3    'Medienverhalten'
-G3.1     'Tageszeitungen'
-G3.1.1      'überregionale Zeitungen'    N9000[1] .. 99
-G3.1.2      'regionale Zeitungen'        N9000[100] .. 199
-G3.2     'Zeitschriften'           N9000[200] .. 299
-G3.5     'Rundfunk'          N9000[300] .. 599
-G3.6     'TV'                N9000[600] .. 999
...
OUTPUT-INT   FNAME = BEISPIEL7.INT
-G1 .. 4
```

Die Variablengruppe G1 erhält den Text *Statistik*. Ihr werden alle zwischen C1 und C99 definierten Variablen zugewiesen.

Die Variablengruppe G3 mit dem Text *Medienverhalten* besitzt keine Variable. Sie dient nur als Zwischenüberschrift in den Variablenfenstern. Erst wenn in CNTW die Gruppe G3 angeklickt wird, werden die untergeordneten Gruppen G3.1, G3.2, G3.5 und G3.6 angezeigt. Um die Variable N9000[100] in CNTW am Bildschirm sichtbar zu machen, sind nach G3 noch die Gruppen G3.1 und G3.1.2 anzuklicken.

Das Folgestatement –G1..4 von OUTPUT-INT übergibt die definierten Variablengruppen einschließlich ihrer Untergruppen und Variablen in die neue interne Datei. Die nicht in den Gruppen enthaltenen Variablen werden nicht ausgegeben. Dafür wären zusätzliche Folgestatements erforderlich.

Abkürzungen

Hinter DEFS können mehrere direkt hintereinander liegende Variable in einer Zeile definiert werden:

```
Statt:      -N1: 6    'Einkommen'
            -N2: 6    'Einkommen'
            -N3: 6    'Einkommen'
genügt:     -N1 .. 3: 6     'Einkommen'
```

Ebenso können mehrere bereits definierte Variable einen neuen Text erhalten:

```
Statt       -C100   1='Männer'  2='Frauen'
            -C101   1='Männer'  2='Frauen'
            -C102   1='Männer'  2='Frauen'
genügt:     -C100 .. 102  1='Männer'  2='Frauen'
```

Es können auch mehrere Variable mit Index in einer Zeile definiert werden:

```
Statt:      -C1[5]:12
            -C2[5]:12
genügt:     -C1..2[5]:12
```

Bereits definierte Variable mit Index können auf ähnliche Weise neue Texte erhalten:

```
Statt       -C1[3]  'Einkommen'
            -C2[3]  'Einkommen'
genügt:     -C1..2[3]  'Einkommen'

Statt       -C1[2]  'Einkommen'
            -C1[3]  'Einkommen'
genügt:     -C1[2]..[3]  'Einkommen'
oder auch:  -C1[2]..3  'Einkommen'
```

Auch die Auswahl von Variablen in den Gruppendefinitionen lässt sich verkürzen:

```
Statt       -G1 'Statistik'  C1  C2  C3  C7  N19[3]  N19[4]  N19[5]  N19[6]
genügt:     -G1 'Statistik'  C1..7  N19[3]..6
```

Dabei wird unterstellt, dass die Variablen C4, C5 und C6 nicht definiert sind.

DO, EDO

Das DO-Statement leitet eine Schleife ein, und das EDO-Statement beendet die Schleife.
Die dazwischen liegenden Statements werden mehrmals ausgeführt, so als hätte man sie mehrmals hintereinander geschrieben.

In den Statements zwischen DO und EDO können einzelne Angaben bei jedem Schleifendurchlauf unterschiedliche Werte erhalten.

DO-Statements sind an jeder beliebigen Stelle zulässig. Innerhalb von Folgestatements sind sie jedoch in der Form –DO und –EDO zu schreiben.

DO < i > **=** a b c ... < **SEP='**s ... **'** >

Dabei ist *i* eine Zahl zwischen 0 und 99, die den DO-Index < 0 > bis < 99 > festlegt.
Für jede der Angaben *a, b, c* ... erfolgt ein Durchlauf durch die hinter DO stehenden Statements, wobei der Index < i > der Reihe nach einen der Werte *a, b, c* ... annimmt.

Zum Beispiel führen die Statements

```
-DO <1>=1, 2, 3
-N<1>=<1>
-EDO
```

das hinter DO liegende Statement drei Mal aus, wobei der Index <1> nacheinander die Werte 1, 2 und 3 annimmt. Daraus entstehen die folgenden Statements:

```
-N1=1
-N2=2
-N3=3
```

Zur Trennung der Indexwerte ist neben den Leerzeichen auch das Komma möglich:

```
-DO <1>=1, 2, 3
```

Mit der Angabe SEP lassen sich auch andere Trennzeichen festlegen.

DO-Statements ganz ohne Werteangaben sind ebenfalls möglich. Zum Beispiel:

```
-DO <1>=
-C100.<1>=<1>
-EDO
```

Hier findet kein Schleifendurchlauf statt, es wird kein Statement -C100 ... erzeugt. Diese Situation entsteht zum Beispiel, wenn die DO-Werte über Makros in das DO-Statement gestellt werden und die Makros keine Angaben enthalten.

Ein DO Statement kann mehrere Indizes < i > gleichzeitig enthalten. Sie müssen sich aber in den Indexnummern < i > unterscheiden und die gleiche Anzahl von Werten besitzen. Beim ersten Durchlauf durch die Statements zwischen DO und EDO sind dann die ersten Werte aller Indizes aktiv, beim zweiten Durchlauf die zweiten Werte usw.
Zum Beispiel erzeugen die Statements

```
-DO  <1>=1  2  3
     <3>=C17.3 C25.9 C12[7].24
-C100.<1>=<3>
-EDO
```

die Wertezuweisungen:

```
-C100.1=C17.3
-C100.2=C25.9
-C100.3=C12[7].24
```

Dieses Beispiel zeigt, dass neben Zahlenwerten auch beliebige Zeichenfolgen als Werte für die DO-Indizes möglich sind. Eine solche Zeichenfolge darf aus maximal 20 Buchstaben, Ziffern und Sonderzeichen bestehen. Kommas und Leerzeichen sind dabei nicht möglich, da sie als Trennzeichen zwischen den verschiedenen Werten dienen.

SEP='s ... **'** Ohne die Angabe SEP gelten Komma und Leerzeichen als Trennzeichen zwischen den Schrittwerten des DO-Statements. Mit `SEP='s...'` können beliebige andere Trennzeichen festgelegt werden. Zum Beispiel dienen nach `SEP=';,.'` das Semikolon, das Komma und der Punkt zur Trennung der Werte. SEP gilt nur für das laufende DO-Statement und nur für die dahinter stehenden Werte.
In einem DO-Statement sind jedoch mehrere SEP-Angaben möglich. Zum Beispiel:

```
DO  SEP=','  <1>=A, B, C  SEP=';'  <2>=1; 2; 3
```

Abkürzungen

Für numerische Indexwerte gibt es eine Reihe von Abkürzungsmöglichkeiten:

Statt:	`DO <1> = 1 1 2.4 2.4 2.4 3`
genügt:	`DO <1> = 1(2) 2.4(3) 3`
Statt:	`DO <3> = 3 4 5 6 21 20 19 18 3.3 4.3 5.3`
genügt:	`DO <3> = 3...6 21...18 3.3...5.3`
Statt:	`DO <9> = 3 5 7 9 11 21 19 17 15 -1.1 -2.2 -3.3 -4.4`
genügt:	`DO <9> = 3...11(2) 21...15(-2) -1.1...-4.4(-1.1)`

Bei nicht numerischen Zeichenfolgen sind solche Abkürzungen jedoch nicht möglich.
So führt zum Beispiel das Statement

```
DO <1>=A...Z
```

nur einen Schleifendurchlauf aus, bei dem der Index <1> den Wert A...Z annimmt.

Führende Nullen des ersten Wertes bleiben auch bei solchen Abkürzungen erhalten.
Die Angaben

```
-DO <1>=08...10
-N1<1>=25
-EDO
```

liefern die Statements

```
-N108=25        -N109=25        -N110=25
```

Schachtelung

DO Statements dürfen auch geschachtelt werden:
Zwischen einem DO Statement und seinem abschließenden EDO Statement dürfen weitere DO und EDO Statements liegen. Dabei laufen die inneren DO-Schleifen 'schneller' als die äußeren.
So erzeugen die folgenden Angaben

```
DO <1> = 1, 2
DO <2>= 05...07
XTAB ROW=TOTAL;C<1><2>
EDO
EDO
```

die Statements:

```
XTAB ROW=TOTAL;C105
XTAB ROW=TOTAL;C106
XTAB ROW=TOTAL;C107
XTAB ROW=TOTAL;C205
XTAB ROW=TOTAL;C206
XTAB ROW=TOTAL;C207
```

Folgende Regeln sind bei geschachtelten DO-Statements zu beachten:

- Es dürfen maximal zehn geschachtelte DO Statements gleichzeitig aktiv sein.
- Jedes EDO Statement beendet das nächstgelegene, noch nicht abgeschlossene DO-Statement.
- In geschachtelten DO-Schleifen laufen die Indices < i > der inneren Schleifen "schneller" als die der äußeren.
- Ein in einer DO-Schleife begonnenes IF- oder IFN-Statement muss auch innerhalb dieser Schleife mit EIF beendet werden.

Beispiel 1

```
ADD-PROC
-DO  <2>=1  1  3
     <5>=7  10  11
     <6>=12  14  12
     <7>=3.21  4.56  99.3
-  C<5>=K<2> (<6>, 12)
-  N<5>=<7>
-  PRT  'Fehler in Satz <2>, Spalte <6>'
-EDO
```

Diese Statements sind gleichwertig mit:

```
ADD-PROC
-C7 = K1(12, 12)
-N7 = 3.21
-PRT  'Fehler in Satz 1, Spalte 12'
-C10 = K1(14, 12)
-N10 = 4.56
-PRT  'Fehler in Satz 1, Spalte 14'
-C11 = K3(12, 12)
-N11 = 99.3
-PRT  'Fehler in Satz 3, Spalte 12'
```

Beispiel 2

```
DO  <1>=1  2  3  4
DO  <2>=10  12
XTAB  COL=C<1>  ROW=C<2>
EDO
EDO
```

Diese Statements sind gleichwertig mit:

```
XTAB  COL=C1  ROW=C10
XTAB  COL=C1  ROW=C12
XTAB  COL=C2  ROW=C10
XTAB  COL=C2  ROW=C12
XTAB  COL=C3  ROW=C10
XTAB  COL=C3  ROW=C12
XTAB  COL=C4  ROW=C10
XTAB  COL=C4  ROW=C12
```

Beispiel 3

```
#T1   'Produkt bekannt, ungestützt'
#T2   'Produkt bekannt, gestützt'
#K1   TOTAL;C1.3;..7;C1.S(3..7)
#K2   TOTAL;C2.1;..9;C2.S(1..9)
#Z1   TOTAL:'Basis';C100.1;..5
#Z2   TOTAL:'Basis';C200.1;..8
DO    <1> = 1  2
DO    <2> = 1  2
XTAB    TITLE=#T<1>    COL=#K<1>    ROW=#Z<2>
EDO
EDO
```

In diesen Statements werden zunächst die Makros #T1 und #T2 für die Titeltexte definiert; danach die Makros K1 und K2 für die Tabellenköpfe und schließlich die Makros #Z1 und #Z2 für die Zeilenangaben. Die beiden DO-Schleifen sind dann gleichwertig mit folgenden Statements:

```
XTAB   TITLE='Produkt bekannt, ungestützt'
       COL=TOTAL;C1.3;..7;C1.S(3..7)
       ROW=TOTAL:'Basis';C100.1;..5
XTAB   TITLE='Produkt bekannt, ungestützt'
       COL=TOTAL;C1.3;..7;C1.S(3..7)
       ROW=TOTAL:'Basis';C200.1;..8
XTAB   TITLE='Produkt bekannt, gestützt'
       COL=TOTAL;C2.1;..9;C2.S(1..9)
       ROW=TOTAL:'Basis';C100.1;..5
XTAB   TITLE='Produkt bekannt, gestützt'
       COL=TOTAL;C2.1;..9;C2.S(1..9)
       ROW=TOTAL:'Basis';C200.1;..8
```

DUMP

Dieses Statement hilft bei der Analyse von Fehlern in den Statements oder im Programm.
DUMP darf an beliebiger Stelle innerhalb der Statements liegen und kann folgende Informationen im Zählprotokoll ausgeben:

- Aktuelle Werte von Makros.
- Dumps aus den Substrings der Arbeitsdatei STRFILE.
- Dumps von Hauptspeicherbereichen.

Pro Statement ist genau einer der folgenden Parameter anzugeben:

MACRO = name

Hinter diesem Statement wird im Zählprotokoll der aktuelle Wert des angegebenen Makros ausgegeben.

name Makroname mit dem Zeichen # davor.
Zum Beispiel könnte im Zählprotokoll direkt hinter dem Statement

```
DUMP MACRO=#rows
```

der Wert des Makros #rows in folgender Form erscheinen:

```
#ROWS: 'ROWS=TOTAL;C117;C128;C195'
```

CSTR0 | . . . | **CSTR31** = **I** | **E**

Es werden vollständige Dumps der Substrings 0 ... 31 der Arbeitsdatei STRFILE erstellt.

I Der Dump wird sofort ausgegeben. Er erscheint im Zählprotokoll direkt hinter diesem Statement.

E Der Dump wird am Ende aller Datendurchläufe ausgegeben. Er erscheint am Ende des Zählprotokolls.

MAINS = **I** | **A** | **E**

Es werden Dumps ausgewählter Hauptspeicherbereiche ausgegeben.

I Der Dump wird sofort, bei Verarbeitung des Statements, ausgegeben.

A Der Dump wird am Anfang jedes Datendurchlaufs ausgegeben. Er erscheint am Ende des Zählprotokolls.

E Der Dump wird am Ende jedes Datendurchlaufs ausgegeben. Er erscheint am Ende des Zählprotokolls.

FETCH-FILE

Die Statements FETCH-FILE öffnen Eingabedateien, aus denen die Funktion FETCH Daten zu Schlüsselwerten oder Keys entnehmen kann. Andere Wege, Daten aus verschiedenen Dateien zusammenzuführen, bieten die Statements FUSION und INPUT-SEC.

So lässt sich eine externe Datei mit dem Statement

```
FETCH-FILE FNAME=GEMEINDE.TBL KEY=D(1,9)
```

einlesen, die in den Positionen 1 ... 9 die Gemeindekennziffer als Schlüssel enthält.
Steht in den Positionen 10 ... 11 dieser Datei das Bundesland als zweistelliger Schlüssel, so kann mit der Wertezuweisung

```
-N10=FETCH(N13 D1(10..11))
```

der Variablen N10 das Bundesland zur Gemeindekennziffer N13 gefunden werden.

Die Statements FETCH-FILE dürfen an beliebiger Stelle angegeben werden, müssen aber vor der ersten Wertezuweisung mit der dazugehörigen FETCH-Funktion liegen. FETCH-FILE darf nicht innerhalb der Folgestatements etwa von DEFS, MOD-PROC usw. angegeben werden.

In einem Verarbeitungslauf können bis zu 30 verschiedene Dateien mit FETCH-FILE, FETCH-FILE1 ... FETCH-FILE29 geöffnet werden. Auf diese Dateien kann anschließend mit den Funktionen FETCH, FETCH1 ... FETCH29 in Wertezuweisungen zugegriffen werden.
Jedes Statement FETCH-FILE ... FETCH-FILE29 darf in einem Verarbeitungslauf nur einmal vorkommen, da die dazugehörigen Dateien bis zum Ende der Auswertung im Hauptspeicher erhalten bleiben. Die Nummern 1 ... 29 müssen nicht lückenlos vergeben werden.

FETCH-Dateien haben den gleichen Aufbau wie externe Dateien in INPUT-EXT oder OUTPUT-EXT.
An die Stelle der Fall-Identifikationen ID ... ID4 treten jedoch die Schlüssel KEY ... KEY4. Sie müssen nicht nach den Schlüsseln vorsortiert sein. Die Daten zu einem Schlüssel können in mehreren Datensätzen enthalten sein.

Die Formate ANSI, ASCII, COLBIN und EBCDIC werden verarbeitet. Neben Datensätzen mit fester und variabler Satzlänge sind auch CSV-Dateien möglich, bei denen die Datenwerte durch ein Separatorzeichen SEP - zum Beispiel Komma oder Semikolon - voneinander getrennt sind.

FETCH-FILE erlaubt folgende Angaben, die weitgehend mit denen aus INPUT-EXT übereinstimmen:

FNAME = dateiname

Hier ist der Name der Eingabedatei anzugeben. Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
FNAME=C:\DATEN\STUDIE27.TAB
FNAME='STUDIE27 TABELLE'
FNAME=\\SERVER\DATEN\GEMEINDEN.TAB
```

Enthält *dateiname* keine Pfadangabe, so erwartet CNTA die Datei im Verzeichnis der Statementdatei. CNTW sucht sie zunächst im Verzeichnis der REP-Datei und danach im Verzeichnis der Statementdatei.
Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\GEMEINDEN.TAB
```

```
KEY  | = | A  | < d > | ( b .. e ) |
KEY1 |   | D  |       | ( b , c )  |
KEY2 |   | F  |
KEY3 |   | G  |
KEY4 |   | P  |

     | AK | < d >  (< 's' >)
     | DK |

     | AS | < d >  ( r )
     | DS |
```

Mit KEY werden Position, Länge und Format der Schlüsselwerte festgelegt.
Liegt nur ein einstufiger Schlüssel vor, so ist dieser mit KEY anzugeben.
Bei mehrstufigen Schlüsseln ist mit der höchsten Stufe bei KEY zu beginnen und lückenlos mit KEY1 ... KEY4 bis zur untersten gewünschten Stufe fortzufahren.

Wird in einem FETCH-FILE-Statement kein KEY angegeben, so führt CNTA selbsttätig einen einstufigen Schlüssel ein, für den alle eingelesenen Fälle durchnumeriert werden: Lückenlos aufsteigend, beginnend bei 1.

Zu jedem Key darf die FETCH-Datei nur einen Fall enthalten. Sie muss aber nicht nach diesen Keys sortiert sein.

Der Zugriff mit der Funktion FETCH auf einen nicht vorhandenen Key führt zu Fehlermeldungen.
Als Keys lassen sich folgende Feldtypen verwenden (siehe Abschnitt *Externe Datenfelder*):

A Der Key besteht aus alphanumerischen Zeichen.

AK In CSV-Dateien: Der Key besteht aus alphanumerischen Zeichen mit Schlüsselwort.

AS In CSV-Dateien: Der Key besteht aus alphanumerischen Zeichen.

D Der Key besteht aus numerischen Zeichen.

DK In CSV-Dateien: Der Key besteht aus numerischen Zeichen mit Schlüsselwort.

DS In CSV-Dateien: Der Key besteht aus numerischen Zeichen.

F Der Key besteht aus einer Dualzahl mit Vorzeichen (Integer).

G Der Key besteht aus einer Dualzahl ohne Vorzeichen (Unsigned Integer).

P Der Key besteht aus einer gepackten Dezimalzahl.
Negative Werte werden vor der Verarbeitung in positive Werte umgesetzt.

d Nummer des Datensatzes, in dem sich der Schlüssel befindet.
Fehlt diese Angabe, so muss der Schlüssel in jedem Datensatz vorliegen, sonst nur in dem hier angegebenen Satz. ACHTUNG: Diese Nummer ist nicht das Satz-Kennzeichen selbst, sondern die Reihenfolgenummer der hinter RC angegebenen Satz-Kennzeichen.

b erstes Byte 1 ... 1 000 000 des Schlüssels in den Datensätzen;
erste Spalte 1 ... 500 000 im COLBIN-Format.

c Länge des Schlüssels in den Datensätzen in Bytes; bei COLBIN in Spalten.
Die maximal zulässigen Längen sind abhängig vom Typ der Datenfelder:
im A-Feld 1 ... 255 Bytes oder Spalten
im D-Feld 1 ... 18 Bytes oder Spalten
im F-Feld 1 ... 8 Bytes
im G-Feld 1 ... 8 Bytes
im P-Feld 1 ... 10 Bytes.

e letztes Byte 1 ... 1 000 000 des Schlüssels in den Datensätzen;
letzte Spalte 1 ... 500 000 im COLBIN-Format.
Die Länge des Schlüssels muss den unter *c* angegebenen Regeln entsprechen.

r Die Angabe r = 1 ... 32 767 ist die Reihenfolgenummer des Wertes im Datensatz d einer CSV-Datei. Leerzeichen vor und hinter den Werten im Datensatz werden ignoriert.

s Diese Angabe ist in Hochkommas einzuschließen. Sie legt fest, welches Schlüsselwort sich in den Datensätzen vor den Key-Werten und dem Gleichheitszeichen befindet.
Ist hier kein Schlüsselwort angegeben, so wird dafür die davor stehende Angabe KEY ... KEY4 verwendet.

SIZE = n < ,**EOL** >

Mit SIZE wird die Länge der Datensätze festgelegt. Ohne diese Angabe wird SIZE=80 angenommen.

n Satzlänge 1 ... 10 000 000. in Bytes, bei COLBIN-Dateien in Spalten.

EOL Die Angabe EOL ist nur bei ANSI-, ASCII-und EBCDIC-Dateien zulässig.
Sie besagt, dass jeder Datensatz durch ein Zeilenende-Zeichen beendet wird (CR=10 oder LF=13 oder CR und LF zusammen).
Die Satzlänge *n* gibt dann die maximal zulässige Länge pro Datensatz an.
Kürzere Datensätze werden beim Einlesen durch Leerstellen aufgefüllt. Längere Datensätze führen zu einer Fehlermeldung, und der über *n* hinausreichende Teil wird nicht verarbeitet.
Die Zeilenende-Zeichen sind bei EOL in der Länge *n* nicht enthalten.
Fehlt die Angabe EOL, so werden Zeilenende-Zeichen wie normale Datenbytes behandelt.

INTEGER = | **NORMAL** | **REVERSE** |

Diese Angabe legt fest, wie die Dualzahlen der externen Datenfelder vom Typ F, G und I in der Eingabedatei gespeichert sind.
Fehlt die Angabe INTEGER, so wird INTEGER=NORMAL angenommen.

NORMAL Die höherwertigen Bits werden in den linken Bytes erwartet.

REVERSE Die höherwertigen Bits werden in den rechten Bytes erwartet.
Dies ist das Format der Windows-PCs.

ASCII
ANSI
UTF8
EBCDIC
COLBIN
BINARY1
BINARY2
XLSX

Diese Angabe legt das Datenformat und den Zeichencode der Datei fest:
Sie entscheidet, welche externen Datenfelder in der Datei vorkommen können und wie diese dort verschlüsselt sind. Insbesondere wird der Zeichencode der Datenfelder vom Typ A, AK, AS, D, DK, DS, L, M, MK, MS, U, UK und US festgelegt.
CSV-Dateien (siehe Angabe SEP) sind nur in den Formaten ASCII, ANSI und EBCDIC möglich.
Weitere Angaben zu diesen Dateiformaten befinden sich im Abschnitt *Externe Dateien* sowie im Anhang *Zeichenzuordnung ANSI, ASCII, EBCDIC* und *Lochkombinationen der COLBIN-Spalten*.
Ohne diese Angabe wird das ANSI-Format verwendet.

ASCII Dies ist das ursprüngliche Datenformat der Personal Computer im MS/DOS. Die Längen und Positionen in den Datensätzen sind in Bytes anzugeben. Alle externen Datenfelder sind möglich.

ANSI Dies ist der von WINDOWS bevorzugte Zeichencode. Der Unterschied zu ASCII beschränkt sich auf die Umlaute, den Buchstaben ß und einige Sonderzeichen. Die Längen und Positionen in den Datensätzen sind in Bytes anzugeben. Alle externen Datenfelder sind möglich.

UTF8 Dieser Zeichencode umfasst alle Zeichen des 16-Bit-UNICODE, also praktisch alle international vorkommenden Schriftzeichen.

COLBIN Die Eingabedaten liegen im Dualkarten-Format (column binary) vor.
In diesem Fall sind alle Positionen und Längen in den Datensätzen nicht in Bytes sondern in Spalten anzugeben. Das gilt sowohl für das Statement FETCH-FILE als auch für die Wertezuweisungen mit der Funktion FETCH. Datenfelder vom Typ P und I sind nicht möglich.

EBCDIC Dies ist das normale Datenformat von Großrechnern. Die Angabe von Längen und Positionen in den Datensätzen erfolgt in Bytes. Alle externen Datenfelder sind möglich.

BINARY1 Diese Dateien bestehen aus ein bis vier Bytes langen Dualzahlen.
Positionen und Längen in den externen Datensätzen sind in Bytes anzugeben.
Es sind nur externe Datenfelder vom Typ B, I, F und G möglich.

BINARY2 Diese Dateien bestehen aus zwei und vier Bytes langen Dualzahlen.
Positionen und Längen in den externen Datensätzen sind in Feldern anzugeben, jedes Feld zu zwei Bytes. Es sind nur externe Datenfelder vom Typ B, I, F und G möglich.

XLSX Diese Angabe liest Daten im XLSX-Format von Excel ein.
Es sind nur externe Datenfelder vom Typ AK, AS, DK, DS, MK, MS, UK und US möglich.
Die Angabe WORKSHEET wählt das gewünschte Arbeitsblatt der Datei aus.

RC = n

n > Durch n = 1 ... 1 000 wird die Anzahl der Datensätze festgelegt, die für jeden Schlüssel einzulesen sind. Diese Datensätze müssen für jeden Schlüssel immer in der gleichen Anzahl und Reihenfolge direkt hintereinander in der Datei liegen.

Im Gegensatz zu der unten beschriebenen Angabe RC = A, D ... kann CNTA hier die Datensätze nicht auf Reihenfolge und Vollständigkeit prüfen. Wegen des erhöhten Fehlerrisikos sollte möglichst mit Satz-Kennzeichen in der Form RC = A, D ... gearbeitet werden.

RC = **A** | **D** | **F** | **G** | **P** | **(** b **..** e **)** | **(** b **,** c **)** | **O:** | **R:** | k**,** k**,** ...

AK | **DK** | **(**< 's' >**)** k**,** k**,** ...

AS | **DS** | **(** r **)** k**,** k**,** ...

Diese Angabe legt ähnlich wie RC = n fest, wie viele Datensätze pro Fall einzulesen sind. Darüber hinaus wird in jedem dieser Datensätze ein eigenes Satz-Kennzeichen k verlangt. Bei dieser Verarbeitungsform kann CNTA die Eingabedatei auf Vollständigkeit prüfen. Die Datensätze eines Schlüssels dürfen ungeordnet hintereinander liegen. CNTA sorgt vor der Verarbeitung für die richtige Reihenfolge: Die Sätze werden in der Reihenfolge angeordnet, in der ihre Satz-Kennzeichen k hier aufgeführt sind. Es sind bis zu 1 000 Datensätze pro Schlüssel möglich.

Datensätze der FETCH-Datei, deren Satz-Kennzeichen hier nicht aufgeführt sind, führen zu Fehlermeldungen und beenden die Verarbeitung vorzeitig.
Unvollständige Fälle, bei denen hier angegebene Satz-Kennzeichen fehlen, werden auf gleiche Weise beantwortet. Die Angaben O / R / S / SR (siehe unten) erlauben abweichende Prüfungsregeln.
Als Satz-Kennzeichen sind die folgenden Feldtypen möglich (siehe *Externe Datenfelder*):

A Die Satz-Kennzeichen sind alphanumerische Zeichen.

AK In CSV-Dateien: Die Satz-Kennzeichen bestehen aus alphanumerischen Zeichen mit Schlüsselwort.

AS In CSV-Dateien: Die Satz-Kennzeichen sind alphanumerische Zeichen.

D Die Satz-Kennzeichen sind numerische Zeichen.

DK In CSV-Dateien: Die Satz-Kennzeichen bestehen aus numerischen Zeichen mit Schlüsselwort.

DS In CSV-Dateien: Die Satz-Kennzeichen sind numerische Zeichen.

F Die Satz-Kennzeichen sind Dualzahlen mit Vorzeichen (Integer).

G Die Satz-Kennzeichen sind Dualzahlen ohne Vorzeichen (Unsigned Integer).

P Die Satz-Kennzeichen sind gepackte Dezimalzahlen. Negative Werte werden vor der Verarbeitung in positive Werte umgesetzt.

b erstes Byte 1 ... 1 000 000 der Satz-Kennzeichen in den Datensätzen der FETCH-Datei;
erste Spalte 1 ... 500 000 im COLBIN-Format.

c Länge der Satz-Kennzeichen in den Datensätzen der FETCH-Datei in Bytes;
bei COLBIN in Spalten. Die Längen sind begrenzt, abhängig vom Datenfeld:
im A-Feld 1 ... 255 Bytes oder Spalten
im D-Feld 1 ... 18 Bytes oder Spalten
im F-Feld und G-Feld 1 ... 8 Bytes
im P-Feld 1 ... 10 Bytes

e letztes Byte 1 ... 1 000 000 der Satz-Kennzeichen in den Datensätzen der FETCH-Datei;
letzte Spalte 1 ... 500 000 im COLBIN-Format.
Die Länge der Satz-Kennzeichen muss den unter c angegebenen Regeln entsprechen.

r Die Angabe r = 1 ... 32 767 ist die Reihenfolgenummer des Wertes im Datensatz d einer CSV-Datei. Leerzeichen vor und hinter dem Satz-Kennzeichen werden ignoriert. Nur bei AS-Feldern gehören die führenden Leerzeichen zum Satz-Kennzeichen. Für die Länge der Werte gelten die unter c angegebenen Regeln für A- und D-Felder.

k Hier sind die zu verarbeitenden Satz-Kennzeichen anzugeben.
Bei D-, DS-, F-, G- und P-Feldern sind dies numerische Werte, bei A- und AS-Feldern sind es in Hochkommas ' eingeschlossene Zeichen oder Zeichenfolgen.

CNTA ordnet die Datensätze eines Schlüssels in der hier angegebenen Reihenfolge an.
Durch die Angabe

```
RC=D(1..2)3,9,1
```

wird zum Beispiel erreicht, dass der Datensatz mit dem Kennzeichen 3 in FETCH-FILE und den zugehörigen FETCH-Funktionen als Satz 1 angesprochen wird. Der Datensatz mit dem Kennzeichen 9 wird entsprechend als Satz 2 und der mit dem Kennzeichen 1 als Satz 3 angesprochen.

s Diese Angabe ist in Hochkommas einzuschließen. Sie legt fest, welches Schlüsselwort sich in den Datensätzen vor den Satz-Kennzeichen und dem Gleichheitszeichen befindet.
Ist hier kein Schlüsselwort angegeben, so wird dafür *RC* verwendet.

Vor jedes Satz-Kennzeichen k kann noch eine der folgenden Angaben gestellt werden.
Damit wird die Prüfung der einzelnen Datensätze eines Schlüssels verändert.

O: k...
R: k...
S: k...
SR

O Die Angabe O = *optional* führt dazu, dass alle dahinter aufgeführten Datensätze als wahlfrei aufgefasst werden. Sie müssen nicht unbedingt vorhanden sein. Fehlen sie, so werden sie ohne Fehlermeldung durch leere Datensätze ersetzt.

R Die Angabe R = *required* führt dazu, dass alle dahinter aufgeführten Datensätze vorhanden sein müssen. Fehlt einer davon, so wird eine Fehlermeldung ausgegeben und die Verarbeitung beendet.

S Die Angabe S = *skip* fasst alle dahinter aufgeführten Datensätze als gültig auf, ohne sie zu verarbeiten. S: ... muss hinter den Angaben O: ... und R: ... stehen.
Die hinter S: ... angegebenen Datensätze werden bei der Nummerierung der Datensätze pro Schlüssel nicht mitgezählt.

SR Diese Angabe SR = *skip rest* ist nur am Ende aller Satz-Kennzeichen zulässig. Sie bewirkt, dass alle nicht aufgeführten Datensätze auch als gültig aufgefasst, aber nicht verarbeitet werden.
Zum Beispiel hat die Angabe

```
RC=D(1,2) 1, O:2,4, R:11,12, O:7, SR
```

folgende Wirkung:

Die Datensätze enthalten den beiden ersten Positionen numerische Satz-Kennzeichen.

Zu jedem statistischen Fall wird ein Datensatz mit dem Kennzeichen 1 erwartet und als Satz 1 verarbeitet. Fehlt dieser Satz, so wird er durch Leerzeichen ersetzt und die Fehlermeldung 505 ausgegeben. Der statistische Fall wird trotzdem verarbeitet.

Sodann können zwei Datensätze mit den Kennzeichen 2 und 4 vorhanden sein, sie werden als Satz 2 und Satz 3 verarbeitet. Wegen der Angabe O: davor dürfen sie auch fehlen. Die fehlenden Datensätze werden durch Leerzeichen ersetzt, ohne Fehlermeldung.

Die Angabe R: verlangt, dass zwei Datensätze mit den Kennzeichen 11 und 12 vorhanden sind, die zur Verarbeitung als Satz 4 und Satz 5 angesprochen werden. Fehlen diese, so wird wieder eine Fehlermeldung ausgegeben und die Verarbeitung beendet.

Der Satz mit dem Kennzeichen 7 ist wieder wahlfrei, er wird als Satz 6 verarbeitet.
Weitere Datensätze sind wegen der Angabe SR zulässig, werden aber nicht verarbeitet.

SKIP = n

Die Angabe SKIP übergeht die ersten n = 1 ... 255 Datensätze der Datei, ohne sie zu verarbeiten. Damit ist es zum Beispiel möglich, Dateien zu verarbeiten, die mit beschreibenden Textsätzen beginnen.

SEP = | 's' | **TAB** | **NUMBER** | < , 't' >

Die Angabe SEP legt fest, dass die Datei im CSV-Format zu verarbeiten ist.
In CSV-Dateien besitzen die einzelnen Datenfelder keine feste Position und Länge, sondern sind durch ein Separatorzeichen (meistens Komma, Semikolon oder Tabulatorzeichen) voneinander getrennt. Der Zugriff auf ein Feld erfolgt über die Reihenfolgenummer 1 ... innerhalb des Datensatzes. Die Länge der Felder ist variabel und vom Datenwert abhängig.
Die CSV-Dateien können zum Beispiel zur Ein- und Ausgabe von externen Dateien an Programme wie Microsoft Excel verwendet werden. Sie sind in den Dateiformaten ASCII, ANSI und EBCDIC möglich, nicht jedoch im COLBIN- und QUANTUM-Format.

's' Hier ist ein beliebiges Zeichen in Hochkommas anzugeben, das als Separatorzeichen zwischen den Datenfeldern dient. Die Zeichen ' und # sind doppelt einzugeben.

TAB Die Angabe TAB macht das Tabulatorzeichen zum Separatorzeichen.

NUMBER Die Angabe NUMBER macht das Nummernzeichen # zum Separatorzeichen.

't' Hier kann ein beliebiges Zeichen in Hochkommas angegeben werden, das als Texterkennungszeichen in alphanumerischen Werten dient (meistens " oder '). Dieses Zeichen kann am Anfang und Ende eines Textwertes für AK- und AS-Felder stehen. Damit wird es möglich, das Separatorzeichen auch innerhalb eines alphanumerischen Wertes zu verwenden. Die Zeichen ' und # sind doppelt einzugeben.

KEYLINE Diese Angabe ist nur zusammen mit SEP möglich. Sie verlangt, dass der erste Datensatz Schlüssel für die Wertezuweisungen durch AK-, DK-, MK- und UK-Felder enthält. Die Datenwerte der Felder müssen in der gleichen Spalte wie ihr Schlüssel stehen. Eine Datei zum Statement

```
FETCH-FILE SEP=';' KEYLINE KEY=DK(fall) …
```

könnte folgendermaßen beginnen:

```
Fall;Geschlecht;Alter
 001;         2;   35
 002;         1;   47
```

Werden durch die Angabe RC mehrere Datensätze pro Fall angefordert, so gilt die Schlüsselzeile für alle Datensätze. Die Datei zum Statement

```
FETCH-FILE SEP=';' KEYLINE KEY=DK(fall) RC=DK(satz)1,2 ...
```

könnte folgendermaßen beginnen:

```
Fall;Satz;Marke;bekannt
 001;   1;   17;      1
 001;   2;  112;      2
 002;   1;   79;      2
```

KEYLINES Wie KEYLINE erklärt diese Angabe die ersten Zeilen der Datei zu Schlüsselzeilen. Sie verlangt jedoch für jede Satzart eine eigene Schlüsselzeile. Die Datei zum Statement

```
FETCH-FILE SEP=';' KEYLINES KEY=DK(fall) RC=DK(satz)1,2...
```

könnte folgendermaßen beginnen:

```
Fall;Satz;Geschlecht;Alter
Fall;Satz;Marke;bekannt
 001;   1;         2;   35
 001;   2;   17;      1
 002;   1;         1;   47
 002;   2;  112;      2
```

DCHAR = ' x '

x ist ein beliebiges Zeichen, das in den Wertezuweisungen mit der FETCH-Funktion aus D-Feldern als Dezimalzeichen gelten soll. Fehlt diese Angabe, so werden Punkt und Komma als Dezimalzeichen verwendet.

TCHAR = ' x '

x ist ein beliebiges Zeichen, das in den Wertezuweisungen mit der FETCH-Funktion aus D-Feldern als Tausenderzeichen gelten soll. Fehlt diese Angabe, so werden keine Tausenderzeichen erkannt.

WORKSHEET = ' name '

Diese Angabe wählt in XLSX-Dateien das gewünschte Arbeitsblatt aus. Fehlt sie, so wird das erste Arbeitsblatt der Datei eingelesen.

Beispiel: Einfache Schlüsseldatei

```
INPUT-INT   FNAME = BERLIN.INT

DEFS
-C2:16  'Bundesland'
-C3:5   'Ortsgröße'
-N2:7   'Anzahl Einwohner'

FETCH-FILE  FNAME=GEMEINDE.TAB  KEY=D(1,5)   ASCII  SIZE=80,EOL

MOD-PROC
-C2=FETCH(N10  M1(10,2,2))
-C3=FETCH(N10  L1(20,1))
-N2=FETCH(N10  D1(30,7))
```

In diesem Beispiel nehmen wir an, dass die interne Datei BERLIN.INT in der Variablen N10 eine Gemeindekennziffer enthält und dass in dieser Datei die Angaben *Bundesland*, *Ortsgröße* und *Anzahl Einwohner* fehlen.

Diese fehlenden Variablen werden in den Folgestatements von DEFS neu definiert.

Mit dem Statement FETCH-FILE wird die Datei GEMEINDE.TAB eingelesen, die zu jeder Gemeindekennziffer eine Zeile mit folgendem Aufbau enthält:

Position 1 bis 5: Gemeindekennziffer wie in N10.

Position 10 bis 11: Bundesland 01 ... 16.

Position 20: Ortsgröße der Gemeinde als Code 1 ... 5.

Position 30 bis 36: Anzahl Einwohner der Gemeinde als numerischer Wert.

Mit der Angabe KEY = D(1,5) im Statement FETCH-FILE wird festgelegt, dass die FETCH-Funktionen in den späteren Wertezuweisungen die Gemeindekennziffer als Schlüssel verwenden.

Die Wertezuweisung

```
-C2=FETCH(N10 M1(10,2,2))
```

sucht für jeden zu verarbeitenden Fall in der Datei GEMEINDE.TAB nach dem Datensatz mit der Gemeindekennziffer N10.
Aus den Positionen 10 bis 11 dieses Datensatzes wird das Bundesland 01 bis 16 im M-Format in die Zielvariable C2 übertragen.

Entsprechend überträgt das Statement

```
-C3=FETCH(N10 L1(20,1))
```

den Code der Ortsgröße aus der Position 20 der FETCH-Datei in die Variable C3.

Das Statement

```
-N2=FETCH(N10 D1(30,7))
```

stellt die Anzahl Einwohner aus den Positionen 30 bis 36 der Gemeindedatei im D-Format in die Variable N2.

Beispiel: Stammdaten zu den statistischen Fällen

```
INPUT-EXT  FNAME=WELLE4.EXT  ID=D(1,5)  ID1=D(6,4)  SIZE=120,EOL

DEFS
-C1:8         'Alter des Befragten'
-C2:2         'Geschlecht des Befragten'

-C100:5       'Wie oft Kaffee gekauft?'
-N100:5,2     'Wieviel kg Kaffee gekauft'

FETCH-FILE  FNAME=PERSONEN.CSV  SEP=';'  KEY=DS( 1 )
            ASCII  SIZE=80,EOL

ADD-PROC
-C1=FETCH(ID MS1(6))
-C2=FETCH(ID MS1(7))

-C100=L1(10,1)
-N100=D1(20,5) 2
```

In diesem Beispiel gibt es eine Personen-Datei PERSONEN.CSV, die zu jedem Befragten die Statistik-Daten enthält, wie Alter und Geschlecht. Die darin gespeicherten Personen werden in verschiedenen Wellen mehrmals nach ihrem Kaffeekonsum befragt, die Statistik-Daten hingegen nur einmal erhoben. Diese Daten sollen aber bei den einzelnen Wellen mit verarbeitet werden.

Mit dem Statement INPUT-EXT wird die Datei WELLE4.EXT mit den Kaffee-Daten zur Auswertung angefordert. Sie besitzt eine zweistufige Fall-Identifikation: ID enthält eine Personen-Nummer, unter der auch in PERSONEN.CSV die Daten der Befragten gespeichert sind. Zusätzlich enthält ID1 eine Fragebogennummer.

Unter DEFS werden die Variablen C1 und C2 zur Aufnahme von Daten aus der Personen-Datei definiert, sowie die Variablen C100 und N100 für Daten aus der externen Datei WELLE4.EXT.

Durch SEP=';' im Statement FETCH-FILE wird festgelegt, dass diese Datei im CSV-Format vorliegt: Die Datenfelder werden nicht über ihre Anfangsposition angesprochen sondern über die Reihenfolgenummer. Diese Datei könnte zum Beispiel aus einem Datenbankprogramm oder aus Microsoft Excel stammen.

Die Angabe KEY=DS(1) legt fest, dass das erste Feld jedes Datensatzes die Personen-Nummer als Schlüssel enthält.

Die Statements

```
-C1=FETCH(ID MS1(6))
-C2=FETCH(ID MS1(7))
```

suchen nun zu jedem Befragten in der Personen-Datei nach dem Datensatz mit der in ID enthaltenen Personen-Nummer. Das erste dieser Statements entnimmt aus dem sechsten Feld des Datensatzes das Alter und das zweite Statement aus dem siebten Feld das Geschlecht des Befragten. Die Daten werden in die Variablen C1 und C2 gestellt.

Die beiden letzten Statements

```
-C100 = L1(10,1)
-N100 = D1(20,5) 2
```

holen die Daten zum Kaffee-Konsum aus der externen Datei WELLE4.EXT und stellen sie in die Variablen C100 und N100.

Beispiel: Zusammenführen zweier externer Dateien

```
INPUT-EXT   FNAME=STATISTIK.EXT     ID=D(1,5)    COLBIN    SIZE=2000

FETCH-FILE  FNAME=MEDIEN.EXT        KEY=D(1,5)   ASCII     SIZE=800,EOL

DEFS
-C1:8         'Alter des Befragten'
-C2:2         'Geschlecht des Befragten'

-N100:3,2  'Frankfurter Allgemeine Zeitung'
-N101:3,2  'Süddeutsche Zeitung'

ADD-PROC
-C1=K1(10,1)
-C2=K1(11,1)

-N100=FETCH(ID F1(20,2) 2)
-N101=FETCH(ID F1(22,2) 2)
```

In diesem Beispiel liegen die zu verarbeitenden Daten in zwei externen Dateien vor.
Die erste Datei im COLBIN-Format enthält unter anderem die Statistik-Daten der Befragten und die zweite im ASCII-Format ihr Medienverhalten in Form von Nutzungswahrscheinlichkeiten. Die Daten eines Befragten besitzen in beiden Dateien die gleiche Paginiernummer.

Mit Hilfe der FETCH-Funktion ist es möglich, die Daten aus beiden externen Dateien in einem Verarbeitungslauf einzulesen. Mit weiteren FETCH-FILE-Statements könnten sogar bis zu 11 externe Dateien in einem Lauf verarbeitet werden.
Weil FETCH-FILE-Dateien vor der Verarbeitung vollständig in den Hauptspeicher eingelesen werden, sind sehr große Dateien für dieses Verfahren ungeeignet.

Die erste externe Datei wird in diesem Beispiel über das Statement INPUT-EXT eingelesen, mit der Fall-Identifikation ID in den Spalten 1 bis 5.
Die Daten aus dieser Datei werden hinter ADD-PROC über die Statements

```
-C1=K1(10,1)
-C2=K1(11,1)
```

den Variablen C1 und C2 zugewiesen.

Die zweite Datei wird über das S112
tatement FETCH-FILE angefordert und in den Hauptspeicher geholt. Die Angabe KEY=D1(1,5) legt fest, dass die Paginiernummer der Befragten als Schlüssel in den FETCH-Funktionen zu benutzen ist.

Die Werte aus dieser Datei werden über die Statements

```
-N100=FETCH(ID F1(20,2)2)
-N101=FETCH(ID F1(22,2)2)
```

in die Variablen N100 und N101 eingelesen. Als Schlüssel oder Key in der Funktion FETCH dient die Fall-Identifikation ID aus INPUT-EXT. Auf diese Weise wird zu jedem Befragten aus INPUT-EXT der dazugehörige Datensatz der FETCH-FILE-Datei angefordert. F1(20,2)2 und F1(22,2)2 sind die Felder in der FETCH-FILE-Datei, aus denen die Daten zu entnehmen sind. Dabei legt die Ziffer 2 hinter dem Feld fest, dass die Eingabewerte zwei Nachkommastellen besitzen.

FUSION

Das Statement FUSION ordnet jedem statistischen Fall der laufenden Verarbeitung (Empfänger; Rezipient) einen Datensatz aus einer zusätzlichen internen Datei (Spender, Donor) zu.
Dies geschieht durch Vergleich von Variablenwerten der Empfänger mit entsprechenden Variablenwerten der Spender, wobei möglichst ähnliche Fälle zusammengeführt werden.
Dabei sind verschiedene Strategien möglich: Gleichmäßige Verwendung der Spenderdaten oder größtmögliche Ähnlichkeit der Spender.

Die Spenderdaten lassen sich in bestehende oder neu definierte Variable übernehmen. Die Regeln dazu liefern die Folgestatements des FUSION-Statements.

Das Statement FUSION führt die Spender und Empfänger folgendermaßen zusammen:
Für jedes Empfänger-Spender-Paar werden aus den vorgegebenen Verbindungsvariablen Ähnlichkeitswerte ermittelt. Ein mathematisches Optimierungsverfahren verbindet danach die Spender und Empfänger so, dass die Summe der Ähnlichkeitswerte für die verbundenen Fälle maximal wird. Das Optimierungsverfahren (erweiterter Kuhn-Munkres-Algorithmus) liefert ohne Iterationen ein optimales Ergebnis: Es ist keine Kombination von Spendern und Empfängern möglich, die alle Randbedingungen erfüllt und eine größere Summe von Ähnlichkeitswerten besitzt.

Die Ähnlichkeit zweier Variablenwerte lässt sich mit verschiedenen Verrechnungsformen ermitteln, die stets Werte zwischen 0 und 100 liefern. Dabei steht 0 für sehr geringe und 100 für sehr große Ähnlichkeit. Sind mehrere Paare von Verbindungsvariablen vorgegeben, so wird aus den Ähnlichkeitswerten der einzelnen Variablen der Mittelwert gebildet, als resultierender Ähnlichkeitswert zwischen Spender und Empfänger. Es ist aber auch möglich, die Verbindungsvariablen unterschiedlich zu gewichten, um ihren Einfluss auf den resultierenden Wert zu variieren.

Andere Wege, Daten aus verschiedenen Dateien zusammenzuführen, bietet das Statement INPUT-SEC und die Wertezuweisung mit der FETCH-Funktion.

In jedem Verarbeitungslauf sind beliebig viele FUSION-Statements möglich.

Das Statement FUSION erlaubt folgende Angaben:

FNAME = dateiname

Hier ist der Name der Spenderdatei anzugeben. Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
FNAME=C:\DATEN\STUDIE27.INT
FNAME='STUDIE27 TABELLE'
FNAME=\\SERVER\DATEN\STUDIE27.INT
```

Enthält *dateiname* keine Pfadangabe, so erwartet CNTA die Datei im Verzeichnis der Statementdatei. CNTW sucht sie zunächst im Verzeichnis der REP-Datei und danach im Verzeichnis der Statementdatei.
Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\STUDIE27.INT
```

Cn=**C**m	,	**EQU**
Nn=**N**m		**MET** < , Gewicht > < , (v ... b) >
Tn=**T**m		**NOM** < , Gewicht >
		NOMM < , Gewicht >

In dieser Form lassen sich bis zu 50 Paare von Verbindungsvariablen angeben, wobei Cm, Nm, Tm aus der Spenderdatei stammen und Cn, Nn, Tn aus INPUT-INT oder DEFS. Aus den Werten dieser Verbindungspaare ermittelt das Programm zu jedem Spender-Empfänger-Paar einen Ähnlichkeitswert, der für die optimale Zuordnung der Spender zu den Empfängern sorgt. Ohne Verbindungsvariable werden die Spender den Empfängern zufällig zugeordnet.

Cn=**C**m zu verbindende Bedingungsvariable
Nn=**N**m zu verbindende numerische Variable
Tn=**T**m zu verbindende Textvariable

Zum Beispiel vergleicht

```
C10=C20,EQU
```

die Werte der Variablen C20 der Spenderdatei mit der Empfängervariablen C10. Durch die Angabe EQU werden nur Spender- und Empfängerfälle miteinander verbunden, die in ihren Verbindungsvariablen gleiche Werte besitzen.

Durch Klammerausdrücke hinter den Variablen lassen sich Variablenwerte vom Vergleich ausschließen. Zum Beispiel:

```
C103(1...6)=C12[2](1...6),MET
```

Hier könnten die beiden Variablen Skalenwerte zwischen 1 und 6 besitzen und Ausprägung 7 für keine Angabe. Durch die Klammerausdrücke (1...6) werden nur die Skalenwerte zur Ermittlung der Ähnlichkeit der Variablenwerte herangezogen. Alle übrigen Werte werden als Z0 (keine Angabe) verstanden. Die Angabe MET ermittelt die Ähnlichkeit zwischen Spender und Empfänger aus der Differenz der Variablenwerte. Besitzen Spender und Empfänger den Wert Z0, so erhalten sie den maximalen Ähnlichkeitswert 100. Besitzt nur einer davon den Wert Z0, so entsteht der minimale Ähnlichkeitswert Z0.

Im Beispiel

```
N10[3](14...)=N12(14...),MET
```

könnten die beiden Variablen Altersangaben enthalten, von denen nur Werte ab 14 in die Ermittlung der Ähnlichkeit eingehen sollen. Kleinere Angaben werden als Z0 (keine Angabe) verstanden.

Textvariablen erlauben nur zwei Vergleichsformen. Die Angabe

```
T103=T12,NOM
```

liefert die maximale Ähnlichkeit 100 wenn beide Texte exakt übereinstimmen, einschließlich Groß- und Kleinschrift. Andernfalls entsteht die Ähnlichkeit 0. Bei

```
T103=T12,MET
```

wird nicht zwischen Groß- und Kleinschrift unterschieden. Alle Werte zwischen 0 und 100 nach den Regeln der Funktion COMP sind möglich (siehe Abschnitt Wertezuweisungen).
Klammerausdrücke zur Selektion bestimmter Variablenwerte sind bei Textvariablen nicht möglich.

EQU Diese Angabe verbindet nur solche Spender und Empfänger miteinander, deren Variablenwerte genau übereinstimmen. Dies gilt auch für die besonderen Werte Z0 (keine Angabe). Kandidaten für EQU-Verbindungen sind zum Beispiel Variable für Geschlecht oder Bundesland. Zum Beispiel

```
C10=C20,EQU
```

Versieht man zu viele Verbindungsvariable mit EQU, so reduziert sich der Spielraum für gute Spender-Empfänger-Kombinationen. Bei einer großen Anzahl von Empfängern und Spendern verbessern die EQU-Angaben hingegen die Laufzeit des Programms, da sie die Zahl der zu untersuchenden Kombinationen von Spendern und Empfängern verkleinert.

NOM Diese Verbindungsform kennt nur die beiden Ähnlichkeitswerte 0 für ungleiche und 100 für gleiche Variablenwerte. Dies gilt auch für den Wert Z0 (keine Angabe). C-Variable dürfen für NOM keine Mehrfachnennungen besitzen. Kandidaten für diese Verbindungsform sind Angaben über Geschlecht oder Wohnort. Zum Beispiel

```
C10=C20,NOM
```

NOM ist bei C-Variablen ohne Mehrfachnennungen sowie N- und T-Variablen zulässig.
Bei Textvariablen führen Unterschiede in Groß- und Kleinschrift zum Ähnlichkeitswert 0 (siehe auch MET).

NOMM Diese Verbindungsform vergleicht C-Variable auch mit Mehrfachnennungen.
Als Ähnlichkeitsmaß wird der M-Koeffizient (Simple Matching) verwendet. (M-Koeffizient; Simple Matching). Der Ähnlichkeitswert zweier Variablenwerte entsteht dabei als Summe der übereinstimmenden Ausprägungen dividiert durch die Gesamtzahl der Ausprägungen. Besitzen die beiden Variablen aus obigem Beispiel fünf Ausprägungen mit den Werten:

```
C10=1,2,5
C20=1,3,5
```

so sind die Ausprägungen 1 und 5 in beiden Variablen erfüllt und Ausprägung 4 ist in beiden nicht erfüllt. Daraus entsteht der Ähnlichkeitswert

```
60=(2+1)/5*100
```

Die beteiligten C-Variablen müssen die gleiche Anzahl von Ausprägungen besitzen oder durch zusätzliche Klammerangaben auf die gleiche Anzahl Ausprägungen korrigiert werden. Zum Beispiel

```
C10=C20(1...5)
```

wenn C10 fünf Ausprägungen besitzt und C20 mehr als fünf.

MET Diese Verbindungsform ermittelt die Ähnlichkeit eines Spenders und Empfängers aus der Differenz der Variablenwerte dividiert durch die maximale Differenz der Variablenwerte der Spender und Empfänger. Kandidaten für MET sind Variable mit Bewertungsskalen aber auch Alters- oder Einkommensangaben. Dabei sind C-Variable ohne Mehrfachnennungen sowie N- und T-Variable zulässig. C-und N-Variable müssen ordinalskalierte Werte besitzen, wie Alter oder Polaritätsskalen.

Besitzt im Beispiel

```
N10=N20,MET
```

N10 den Wert 4, N20 den Wert 8 und ist 10 die größte Differenz zwischen den Werten der beiden Variablen, so entsteht daraus der Ähnlichkeitswert

```
(1-(8-4)/10)*100=40
```

Besitzen beide Variable den Wert Z0 (keine Angabe), so entsteht der Ähnlichkeitswert 100. Hat dagegen nur eine den Wert Z0, wird die Ähnlichkeit zu 0.

Besitzen zwei Werte ungleich Z0 einen Ähnlichkeitswert kleiner als 1, so wird dieser nachträglich auf 1 korrigiert. Damit ist der der Unterschied zwischen Z0 und einer Zahl stets größer der der Unterschied zwischen zwei Zahlen.

Bei Textvariablen in der Form

```
T103=T12,MET
```

werden Ähnlichkeitswerte zwischen 0 und 100 nach den Regeln der Funktion COMP vergeben (siehe Abschnitt Wertezuweisungen).

(v**...** b**)** Die Angabe (v ... b) hinter MET macht unterschiedliche Wertebereiche bei C- und N-Variablen vergleichbar. Im Beispiel

```
N10=N100,MET,(1...6)
```

soll die Variable N10 Wertebereiche 1 bis 3 besitzen und N100 die Werte 1 bis 6.
Die Angabe (1 ... 6) justiert dann die Werte von N10 vor dem Vergleich mit N100 von 1 bis 3 auf 1 bis 6. Umgekehrt justiert die Angabe

```
N10=N100,MET,(1...3)
```

die Werte 1 bis 6 der Variablen N100 auf 1 bis 3.

Gewicht Diese Angabe legt fest, wie stark die Verbindungsvariablen in den Ähnlichkeitswert der Spender-Empfänger-Kombinationen eingehen soll. Dabei sind auch Werte mit Nachkommastellen möglich. Fehlt zu einem Verbindungspaar die Gewichtsangabe, so wird dafür 1 angenommen.

Beim Vergleich eines Spenders mit einem Empfänger entsteht zunächst für jedes Verbindungspaar gemäß MET, NOM und NOMM ein Ähnlichkeitsgewicht zwischen 0 und 100. Aus diesen Einzelgewichten entsteht die Gesamtähnlichkeit als Wert zwischen 1 und 100 durch Addition der Einzelgewichte, multipliziert mit dem Gewicht des Paares und dividiert durch die Summe aller Gewichte.

Enthält das Statement FUSION zum Beispiel die Verbindungsangaben

```
N10=N200,MET,3
N11=N201,MET,1
```

und liefert N10=N200 die Ähnlichkeit 50 sowie N11=N201 den Wert 10, so entsteht eine Gesamtähnlichkeit von

```
(50*3 + 10*1)/4 = 40
```

Ohne die Gewichtsangaben wäre das Ergebnis

```
(50 + 10)/2 = 30
```

BESTFIT = | **ON** | n |

Ohne diese Angabe werden die Spender möglichst gleich oft den Empfängern zugewiesen. Dadurch bleiben die Datenstrukturen des Spenderbestandes bestmöglich erhalten.

ON Diese Angabe weist jedem Empfänger einen Spender mit größtmöglicher Ähnlichkeit der Verbindungsvariablen zu. Die einzelnen Spender können dabei beliebig oft verwendet werden.

n Jeder Spender wird höchstens *n* mal verwendet, gleichmäßige Verwendung der Spender hat Vorrang.

LIMIT = n Mindestwert *n* = 0 ... 100 für die Ähnlichkeit zwischen Spender und Empfänger.
Ohne diese Angabe werden bei schwierigen Zuordnungsproblemen auch Spender und Empfänger mit geringer Ähnlichkeit verbunden. Dabei steht 100 für maximale und 0 für minimale Ähnlichkeit. LIMIT verhindert Verbindungen mit kleineren Ähnlichkeitswerten als *n*. Als Folge können einige Empfänger ohne Spender bleiben.

SELECT = *logischer Ausdruck*

An der Fusion nehmen nur die Fälle der Empfängerdatei teil, die den logischen Ausdruck erfüllen. Alle anderen Fälle erhalten keine neuen Werte. Der logische Ausdruck darf nur Variable aus INPUT-INT oder DEFS enthalten.
Zum Beispiel vergibt

```
SELECT=C10.Z0
```

nur solchen Empfängern neue Werte aus der Spenderdatei FNAME, deren Variable C10 keine Angaben besitzt.

SELECTD = *logischer Ausdruck*

Nur Fälle der Spenderdatei FNAME, die den logischen Ausdruck erfüllen, nehmen an der Fusion teil. Alle anderen Fällen werden nicht zur Vergabe neuer Werte verwendet. Der logische Ausdruck darf nur Variable der Spenderdatei enthalten.
Zum Beispiel wählt

```
SELECT=C20.Z1
```

nur Spender zur Fusion aus, deren Variable C20 genau eine Angabe besitzt.

NEWVARS= | **ALL** | **ALL(**n**)** | **MISSING** |

Diese Angaben sind nur wirksam, wenn keine Folgestatements vorliegen. Sie erzeugen zu den Variablen der Spenderdatei FNAME neue Variable in der Empfängerdatei mit den Variablen- und Merkmalstexten sowie den Variablenwerten der Spender.
NEWVARS=ALL übernimmt alle Variable aus der Spenderdatei in neu definierte Variable. Dazu werden die kleinsten noch freien Variablennummern verwendet.
NEWVARS=ALL(n) vergibt stattdessen die kleinsten noch freien Variablennummern größer oder gleich der Nummer *n.*
NEWVARS=MISSING überträgt zunächst wie bei der normalen Zuordnung die Variablenwerte der Spender in gleichnamige Zielvariable. Zu Spendervariablen ohne gleichnamiges Gegenstück auf der Empfängerseite wird dagegen eine neue Variable mit dem alten Variablennamen angelegt.

VTEXT = 'text'

Diese Angabe stellt die Zeichenfolge *text* vor den Variablentext aller aus der Spenderdatei FNAME stammenden Variablen, die auf der Empfängerseite neu zu definieren sind.
Zum Beispiel stellt

```
VTEXT='Spender'
```

vor alle aus der Spenderdatei übernommenen Variablentexte die Angabe *Spender:* um die Herkunft der Variablen deutlich zu machen.

Folgestatements

Zur Übernahme der Spenderdaten dienen die Folgestatements des Statements FUSION.
Für sie gelten die gleichen Regeln wie beim Statement INPUT-SEC. Dort befindet sich auch die ausführliche Beschreibung.

Normale Zuordnung der Variablenwerte ohne NEWVARS und Folgestatements

Unter diesen Voraussetzungen werden die Datenwerte der Spenderdatei FNAME automatisch in gleichnamige Variable aus DEFS oder INPUT-INT übertragen. Variable der Spenderdatei ohne gleichnamiges Gegenstück werden nicht übernommen. Die Folgestatements verhindern diese automatische Datenübernahme. Sie können die Variablen aus FUSION beliebigen anderen Variablen aus INPUT-INT oder DEFS zuordnen.

Besitzt die Zielvariable aus INPUT-INT oder DEFS mehr Merkmale, Ziffern oder Zeichen als die Spendervariable, so werden die Eingabewerte mit Nullen oder Leerzeichen aufgefüllt. Ist die Zielvariable dagegen kürzer, so werden bei C-Variablen die hohen Merkmalsnummern abgeschnitten und bei T-Variablen die Texte rechts gekürzt, beides ohne Fehlermeldung. Numerische Werte werden bei abweichenden Nachkommastellen gerundet oder erweitert. Passen sie nicht in die Zielvariable, so wird eine Fehlermeldung ausgegeben und kein Wert übertragen.

Zuordnung der Variablen mit NEWVARS

Die Angabe NEWVARS im Statement INPUT-SEC ist nur wirksam, wenn keine Folgestatements zur Auswahl von Variablen vorliegen. Sie erzeugt zu Variablen aus INPUT-SEC auch neue Variable mit den Variablentexten und Datenwerten aus INPUT-SEC.
NEWVARS=ALL übernimmt alle Variable aus INPUT-SEC in neu definierte Variable.
NEWVARS=MISSING überträgt zunächst wie bei der normalen Zuordnung die Variablenwerte aus INPUT-SEC in gleichnamige Zielvariable. Zu Variablen ohne gleichnamiges Gegenstück wird dagegen eine neue Variable mit dem gleichen Variablennamen angelegt.

Zuordnung der Variablen durch Folgestatements

Liegen direkt hinter dem Statement FUSION Folgestatements mit einem Strich – am Zeilenanfang, so werden nur die dort angegebenen Variablen übernommen.
Zum Beispiel überträgt das Statement

```
-C10=C15
```

die Werte der Variablen C15 der Spenderdatei FNAME in die Variable C10 der Empfängerseite. Ist die Variable C10 in INPUT-INT oder DEFS nicht vorhanden, so wird sie durch das Folgestatement neu definiert, einschließlich der Variablen- und Merkmalstexte sowie der Variablenwerte der Spenderdatei.

Beispiel: Gleichmäßige Verwendung der Spender

```
INPUT-INT   FNAME=test100.int

FUSION   FNAME=test10.int
               C400=C500,EQU      %% Geschlecht
               C410=C510,MET      %% Altersgruppen
               C403=C503,NOM      %% Bundesland
               C411=C511,NOMM     %% Kinder im Haushalt
* ----- Verbindungsvariable zur Kontrolle übernehmen
-C1500=C500
-C1510=C510
-C1503=C503
-C1511=C511
* ----- sonstige Spenderdaten übernehmen
-N2000=N100
...

DO <1>=C400    C410    C403    C411
   <2>=C1500  C1510   C1503   C1511
XTAB  ROW=TOTAL; {: 'Empfänger'} {<1>; .Z0}
      COL=TOTAL; {: 'Spender'} {<2>; .Z0}
EDO
```

Das Statement INPUT-INT liest die Empfängerdaten aus der internen Datei test100.int.

FUSION fügt den Fällen der Empfängerdatei die Spenderdaten aus der Datei test10.int hinzu.
Die Spender werden den Empfängern so zugeordnet, dass die Werte der Verbindungsvariablen C400, C410, C403, C411 der Empfängerdatei möglichst wenig von den Werten der Variablen C500, C510, C503, C511 der Spenderdatei abweichen.

Die Angabe EQU verlangt dabei völlige Gleichheit der Werte von C400 und C500.
MET versteht die Werte von C410 und C510 als metrisch skaliert: Die Ähnlichkeit 0 ... 100 zweier Werte wird über ihre Differenz ermittelt.
NOM interpretiert die Werte von C403 und C503 als nominal skaliert: Es gibt nur die beiden Ähnlichkeitswerte gleich (100) und ungleich (0). Mehrfachnennungen sind dabei nicht möglich.
NOMM erlaubt Mehrfachnennungen in den Variablen C411 und C511. Der Ähnlichkeitswert 1 ... 100 ergibt sich aus der Anzahl übereinstimmender Merkmale, sowohl der Werte 1 = genannt als auch 0 = nicht genannt.

Das Folgestatement

```
-C1500=C500
```

übernimmt die Werte der Variablen C500 der Spenderdatei und stellt sie in die Variable C1500 der Empfängerdatei. Existiert die Variable C1500 auf der Empfängerseite noch nicht, so wird sie durch dieses Folgestatement neu definiert, mit den Texten und Werten der Spenderdatei. Gleiches geschieht mit den folgenden Statements.

Das Statement FUSION enthält nicht die Angabe MAXUSE. Dadurch werden die Spenderdaten bei der Fusion möglichst gleich oft verwendet. Die Repräsentativität der Spenderdaten bleibt dabei bestmöglich erhalten, einzelne Verbindungen mit geringer Ähnlichkeit sind dabei in Kauf zu nehmen.

Das Statement

```
-N2000=N100
```

ist ein Beispiel für die Übernahme der eigentlichen Nutzdaten aus der Spenderdatei.

Das Zählprotokoll liefert zum Statement FUSION sehr ausführliche Informationen. Es beginnt mit einer Übersicht über die Eingabedaten:

```
Geschlecht = Männer

Anzahl Empfänger          1408
     ohne Spender            -
Anzahl Spender             148
     nicht verwendet         -
     9 mal verwendet        72
     10 mal verwendet       76
mittlere Übereinstimmung   92%

Spender und Empfänger haben gleiche Werte:
     75% Alter
     88% Bundesländer
     94% Kinder im Haushalt
     61% in allen Verbindungsvariablen gleichzeitig
Prozentuierungsbasis: 1408 Empfänger

Anzahl Empfänger (%)
    |
100 +
    |
    |
    |
    |                                          X
 50 +                                          X       . ~ ...4%
    |                                          X       : ~ 5%...9%
    |                                          X      X ~ 10%
    |                                          X
    |                       . . . X . . . . . X X
  0 --+---+---+---+---+---+---+---+---+---+---+-> Übereinstimmung (%)
      0  10  20  30  40  50  60  70  80  90 100
```

Das Histogramm macht deutlich, wie häufig Ähnlichkeitswerte vom Idealwert 100 abweichen.

Die noch folgenden XTAB-Statements vergleichen die Werte der Verbindungsvariablen aus der Empfängerdatei mit denen der Spenderdatei. Dies liefert zusätzliche Hinweise auf die Qualität der Fusion. Die Tabelle zu den EQU-Variablen C400=C500 zeigt, dass tatsächlich nur Spender und Empfänger mit gleichen Werten zusammengeführt werden:

	Gesamt	Spender		
		Geschlecht		
		Männer	Frauen	keine Angabe
Gesamt	3203	1408	1795	-
Empfänger				
Geschlecht				
Männer	1408	1408	-	-
Frauen	1795	-	1795	-
keine Angabe	-	-	-	-

Die Tabelle der Variablen C410=C510 zeigt einerseits die bevorzugte Verbindung gleicher Variablenwerte,

	Gesamt	Spender							
		Alter							
		14 - 19 Jahre	20 - 24 Jahre	25 - 29 Jahre	30 - 34 Jahre	35 - 39 Jahre	40 - 44 Jahre	45 - 49 Jahre	keine Angabe
Gesamt	3203	218	205	270	243	268	330	320	1349
Empfänger									
Alter									
14 - 19 Jahre	243	170	50	6	12	3	-	-	2
20 - 24 Jahre	243	26	128	68	4	6	5	-	6
25 - 29 Jahre	182	3	14	115	43	4	3	-	-
30 - 34 Jahre	202	13	4	33	104	24	15	8	1
35 - 39 Jahre	247	4	6	18	44	144	26	3	2
40 - 44 Jahre	321	-	1	12	13	72	194	26	3
45 - 49 Jahre	389	-	1	3	9	7	84	281	4
keine Angabe	1376	2	1	15	14	8	3	2	1331

Ein Blick in das obige Zählprotokoll verrät unter Ermittelte Angaben die Ursache für diese Streuung. Da zum Alter die Angabe MET mit den Datenwerten 1 bis 7 vorliegt, führt ein Unterschied der Werte um 1 lediglich zu einer Verschlechterung der Gesamtähnlichkeit von 33.3 / 6 = 5.5%, wie im Protokoll unter Ähnlichkeitsverlust ausgewiesen. Eine Abweichung im Bundesland erzeugt hingegen einen Verlust von 33,3%. Entsprechend akzeptiert der Optimierungsalgorithmus Abweichungen beim Alter leichter als bei den Bundesländern. Ist dieser Effekt unerwünscht, so sind die einzelnen Verbindungsvariablen zu gewichten:

```
FUSION   FNAME=test10.int
                C400=C500,EQU       %% Geschlecht
                C410=C510,6,MET     %% Alter
                C403=C503,NOM       %% Bundesland
                C411=C511,3,NOMM    %% Kinder im Haushalt
```

Hier erhält Alter das Gewicht 6 und Kinder im Haushalt das Gewicht 3. Da die Verbindung C403=C503 kein Gewicht besitzt, wird dafür 1 angenommen. Das führt im Zählprotokoll zu folgendem Ergebnis:

```
Verbindungsvariable:
     C400=C400 EQU       Geschlecht
                           1 = Männer
                           2 = Frauen
     C410=C410 MET,60%   Alter
     C303=C303 NOM,10%   Bundesländer
     C411=C411 NOMM,30%  Kinder im Haushalt

Ermittelte Werte:        Empfänger  Spender  Ähnlichkeitsverlust
     Alter               1...7      1...7    9.9 bei Abweichung um 1
     Bundesländer        1...16     1...16   10.0 wenn ungleich
     Kinder im Haushalt  1...3      1...3    10.0 pro abweichendem Merkmal
```

Damit ist der Ähnlichkeitsverlust in den drei Verbindungsvariablen ungefähr gleich groß.
Bei den Fällen mit gleichen Variablenwerten zeigt das Zählprotokoll eine Verbesserung bei Alter und Kinder im Haushalt und einer Verschlechterung bei Bundesländer, insgesamt aber einer leichte Verbesserung:

```
Spender und Empfänger haben gleiche Werte:
     87% Alter
     77% Bundesländer
     96% Kinder im Haushalt
     64% in allen Verbindungsvariablen gleichzeitig
Prozentuierungsbasis: 1408 Empfänger
```

Diese Verbesserung zeigt sich auch in der Kreuztabelle der Altersgruppen:

	Gesamt	Spender							
		Alter							
		14 - 19 Jahre	20 - 24 Jahre	25 - 29 Jahre	30 - 34 Jahre	35 - 39 Jahre	40 - 44 Jahre	45 - 49 Jahre	keine Angabe
Gesamt	3203	218	205	270	243	268	330	320	1349
Empfänger									
Alter									
14 - 19 Jahre	254	200	36	18	-	-	-	-	-
20 - 24 Jahre	243	18	169	52	4	-	-	-	-
25 - 29 Jahre	187	-	-	180	7	-	-	-	-
30 - 34 Jahre	201	-	-	16	169	16	-	-	-
35 - 39 Jahre	251	-	-	1	49	197	4	-	-
40 - 44 Jahre	330	-	-	-	8	49	271	2	-
45 - 49 Jahre	388	-	-	3	6	6	55	318	-
keine Angabe	1349	-	-	-	-	-	-	-	1349

Beispiel: Größtmögliche Ähnlichkeit der Spender

```
INPUT-INT   FNAME=test100.int

FUSION   FNAME=test10.int
                C400=C500, EQU     %% Geschlecht
                C410=C510, MET     %% Altersgruppen
                C403=C503, NOM     %% Bundesland
                C411=C511, NOMM    %% Kinder im Haushalt
         MAXUSE=1000
```

Dieses Beispiel unterscheidet sich vom vorigen nur durch die Angabe MAXUSE=1000. Diese erlaubt dem Programm, jeden Spender bis zu 1000 mal mit einem Empfänger zu verbinden.
Ohne eine solche Angabe wird für MAXUSE der kleinstmögliche Wert verwendet, bei dem jeder Empfänger einen Spender erhalten kann. Das fördert die gleichmäßige Verwendung der Spender, nimmt aber geringere Ähnlichkeiten zwischen Spendern und Empfängern in Kauf.
Mit höheren Werten von MAXUSE wird dagegen die Ähnlichkeit zwischen Spendern und Empfängern verbessert, die gleichmäßige Verwendung der Spender leidet jedoch. Die Werte aus dem vorigen Beispiel verändern sich dadurch:

```
Geschlecht = Männer

Anzahl Empfänger         1408
     ohne Spender           -
Anzahl Spender             148
     nicht verwendet         -
     3 mal verwendet         3
     4 mal verwendet         9
     5 mal verwendet         8
     6 mal verwendet        29
     7 mal verwendet        12
     8 mal verwendet        16
     9 mal verwendet         9
     10 mal verwendet        7
     11 mal verwendet       16
     12 mal verwendet        7
     13 mal verwendet       11
     14 mal verwendet        6
     15 mal verwendet        1
     16 mal verwendet        2
     17 mal verwendet        2
     18 mal verwendet        3
     19 mal verwendet        1
     21 mal verwendet        2
     22 mal verwendet        2
     23 mal verwendet        1
     36 mal verwendet        1
mittlere Übereinstimmung  96%

Spender und Empfänger mit gleichen Werten:
     79% Alter
     97% Bundesländer
     93% Kinder im Haushalt
     73% in allen Verbindungsvariablen gleichzeitig
Prozentuierungsbasis: 1408 Empfänger

Anzahl Empfänger (%)
    |
100 +
    |
    |
    |                                          X
    |                                          X
 50 +                                          X       . ~ ...4%
    |                                          X       : ~ 5%...9%
    |                                          X       X ~ 10%
    |                                          X
    |                        . . . . . . . . . X X
  0 --+---+---+---+---+---+---+---+---+---+---+-> Übereinstimmung (%)
```

Beispiel: Fehlende Werte durch Fusion ersetzen

```
INPUT-INT   FNAME=Beispiel-OHNE.int

FUSION  FNAME=Beispiel-MIT.int
        C400=C400,EQU    %% Geschlecht
        C410=C410,MET    %% Alter
        C403=C403,MET    %% Haushaltsgröße
-C490=C490

OUTPUT-EXT   FNAME='Beispiel.CSV'
             SIZE=150,EOL        SEP=';'
-DS1(1)=ID
-MS1(2)=C490
```

Dieses Beispiel geht von einem Datenbestand aus, in dem die Variable C490 fehlende Angaben besitzt. Daraus wurde der Datenbestand *BEISPIEL-OHNE.int* selektiert, der die Datensätze mit fehlenden Werten in C490 enthält. Der Datenbestand *BEISPIEL-MIT.int* enthält die restlichen Fälle mit Angaben in der Variablen C490, der als Spenderdatei für die Fusion dienen soll.

Das Statement FUSION ordnet den Fällen ohne Angabe möglichst passende Fälle mit Angaben zu. Als Verbindungsvariable dient zunächst C400 (Geschlecht). Die Angabe EQU dahinter erzwingt Gleichheit der Variablenwerte bei der Fusion. Weiter dienen die Variablen C410 (Alter) und C403 (Haushaltsgröße) als Verbindungsvariable mit der Angabe MET, die metrisch oder ordinal skalierte Variablenwerte anzeigt.

Das Statement

```
-C490=C490
```

übernimmt nun die Werte der Variablen aus der Datei *BEISPIEL-MIT.int* und stellt sie in die Variable C490 der Datei *BEISPIEL-OHNE.int.*

Das Statement OUTPUT-EXT erzeugt eine CSV-Datei aus den Fällen ohne Angabe in C490, die neben der Fall-Idnentifikation die ergänzten Werte der Variablen C490 enthält.

Das Zählprotokoll dazu könnte folgendermaßen aussehen:

```
_____________________________________________________________________________
Gesamt
_____________________________________________________________________________

Anzahl Empfänger           130
     ohne Spender            -
Anzahl Spender             205
     nicht verwendet        75
     1 mal verwendet       130
mittlere Übereinstimmung   83%

Spender und Empfänger mit gleichen Werten:
      66%  Alter
      95%  Haushaltsgröße
      61%  in allen Verbindungsvariablen gleichzeitig
Prozentuierungsbasis: 130 Empfänger

Anzahl Empfänger (%)
    |
100 +
    |
    |
    |
    |                                          X
 50 +                                          X       . ~ ...4%
    |                                          X       : ~ 5%...9%
    |                     X                    X       X ~ 10%
    |                     X                    X
    |                     X                . . X
  0 --+---+---+---+---+---+---+---+---+---+---+-> Übereinstimmung (%)
```

HOLECOUNT

Das Statement HOLECOUNT erzeugt eine Grobauswertung von externen Eingabedateien. Dazu muss ein INPUT-EXT Statement vorliegen, das die auszuwertende externe Datei anfordert. HOLECOUNT erstellt dann eine Auswertung aller Bytes oder Spalten dieser externen Datei (Globalzählung, Grundauswertung, 80/80-Zählung).

Eine Statementdatei darf mehrere HOLECOUNT-Statements enthalten.

Bei der Ausgabe über das Statement PAGEP werden die Zählergebnisse in der Schrift FONT2 und die Überschriften in der Schrift FONT3 aufbereitet. Diese Schriften lassen sich im Statement PAGEP verändern.

Im Statement HOLECOUNT sind folgende Angaben möglich:

SCALE = x

Diese Angabe dient der Hochrechnung der Auswertungsergebnisse. Dazu werden alle Fallzahlen mit dem konstanten Wert *x* multipliziert.

WEIGHT = Nx

Die Auswertung erfolgt mit einem Fallgewicht. Dazu wird für jeden statistischen Fall der Wert der numerischen Variablen *Nx* verwendet.

LCHAR = 'x'>

x ist ein beliebiges Zeichen, das als Lesehilfe in den Auswertungen mit PAGE hinter den Spalten 5, 10 und 12 ausgegeben wird. Fehlt diese Angabe, so wird dafür das Zeichen ¦ verwendet.

HOLECOUNT interpretiert die Eingabedaten als Spalten von Lochkarten und ermittelt für jede Spalte die Anzahl der Lochungen 1 ... 9, 0, - und &.

Dabei werden folgende Angaben gleich behandelt:

0 auch als 10
\- auch als 11 oder X
& auch als 12 oder Y

Im COLBIN- und QUANTUM-Code sind in jeder Spalte beliebige Lochkombinationen als Mehrfachnennungen möglich, die von HOLECOUNT problemlos verarbeitet werden können. Siehe folgendes Beispiel: *Einfache Auswertung* sowie die Codetabellen zum QUANTUM- und COLBIN-Format im Anhang.

Für externe ASCII-, ANSI- oder EBCDIC-Dateien werden die Datensätze zunächst in das COLBIN-Format umgewandelt und danach wie COLBIN-Dateien für HOLECOUNT ausgewertet.

CSV-Dateien mit der Angabe SEP= . . . in INPUT-EXT werden spaltenweise ausgewertet, jedoch nach einem abweichenden Auswertungsschema. Siehe folgendes Beispiel: *Für CSV-Dateien.*

Folgestatements

Besitzt HOLECOUNT keine Folgestatements, so werden alle Bytes oder Spalten der externen Eingabedatei ausgewertet. Durch geeignete Folgestatements kann die Auswertung aber auch auf einzelne Datensätze pro Fall beschränkt werden. Weiter ist es möglich, innerhalb einzelner Datensätze nur bestimmte Bytes oder Spalten auszuwerten. Die Folgestatements müssen direkt hinter dem HOLECOUNT Statement liegen und mit einem Strich am Zeilenanfang beginnen:

–Kn alle Bytes oder Spalten aus dem Datensatz n werden ausgewertet.
Die Nummer n des Datensatzes wird durch die Angabe RC in INPUT-EXT festgelegt.

–Km..n alle Bytes oder Spalten der Datensätze m ... n werden ausgewertet.

–Kn**(**s**)** nur das Byte oder die Spalte s = 1 ... 32 767 des Datensatzes n wird ausgewertet.

–Kn**(**s**..**t**)** die Bytes oder Spalten s ... t = 1 ... 32 767 des Datensatzes n sind auszuwerten.

–Km**..**n**(**s**..**t**)** die Bytes oder Spalten s ... t = 1 ... 32 767 der Datensätze m ... n sind auszuwerten.

Beispiel: Einfache Auswertung

```
INPUT-EXT  COLBIN  FNAME=STUDIE007.CLB
HOLECOUNT
```

Durch das Statement INPUT-EXT wird eine externe Eingabedatei im COLBIN-Format angefordert. Da weder SIZE noch RC angegeben sind, handelt es sich um eine Datei mit einem Datensatz pro Fall von 80 Spalten Länge.

Die Angabe HOLECOUNT wird dann alle 80 Spalten dieses Datensatzes verarbeiten.
Die Druckausgabe dazu könnte folgendermaßen beginnen:

Satz 1 **Anzahl Fälle: 8636**

Satz/Pos	1	2	3	4	5	6	7	8	9	10	11	12	Z0	Z1	Z23	Z123
1/1	166 1,9%	-	-	-	-	-	-	-	-	8470 98,1%	-	-	-	8636 100,0%	-	8636 100,0%
1/2	1375 15,9%	1838 21,3%	1203 13,9%	1000 11,6%	1226 14,2%	112 1,3%	65 0,8%	819 9,5%	832 9,6%	166 1,9%	-	-	-	8636 100,0%	-	8636 100,0%
1/3	-	-	-	-	-	-	-	-	-	4309 49,9%	-	-	4327 50,1%	4309 49,9%	-	4309 49,9%
1/4	-	-	4309 49,9%	-	-	-	-	4309 49,9%	-	4309 49,9%	-	-	4327 50,1%	-	4309 49,9%	4309 49,9%
1/5	-	-	-	-	-	-	-	-	-	4309 49,9%	-	-	4327 50,1%	4309 49,9%	-	4309 49,9%
1/6	-	-	-	-	-	-	-	-	-	4309 49,9%	-	-	4327 50,1%	4309 49,9%	-	4309 49,9%
1/7	-	-	4309 49,9%	-	-	-	-	-	-	-	-	-	4327 50,1%	4309 49,9%	-	4309 49,9%

Es wurden insgesamt 8 636 statistische Fälle verarbeitet. In der ersten Spalte der Datensätze besaßen 166 Fälle oder 1.9% die Angabe 1 und 8 740 Fälle oder 98.1% die Angabe 10. Der Wert 8 636 unter Z1 besagt, da diese Fälle genau eine Angabe in den Spalten 1 oder 10 besaßen. Es wurde kein Fall ohne Angabe (Z0) gefunden und ebenfalls keine Fälle mit Mehrfachnennungen (Z23). Die Anzahl Fälle mit Angaben (Z123) stimmt mit der Anzahl der gesamten Fälle überein.

Zur Spalte 3 gibt es 4 327 Fälle ohne Angabe (Z0) und 4 309 Fälle mit genau einer Angabe 10.

Beispiel: Mit Fallgewicht

```
INPUT-EXT  COLBIN  FNAME=STUDIE007.CLB
DEFS
-N1:3,2  'Fallgewicht'
ADD-PROC
-N1=D1(3,3)2
HOLECOUNT     WEIGHT=N1
```

Durch das Statement INPUT-EXT wird eine externe Eingabedatei im COLBIN-Format angefordert. Da weder SIZE noch RC angegeben sind, handelt es sich um eine Datei mit einem Datensatz pro Fall von 80 Spalten Länge.

In dem Folgestatement von DEFS wird die Variable N1 zur Aufnahme eines Fallgewichts definiert. Das Folgestatement von ADD-PROC liest aus jedem externen Datensatz einen Zahlenwert und stellt ihn in die Variable N1.

Die Angabe HOLECOUNT wird dann alle 80 Spalten dieses Datensatzes verarbeiten. Die Angabe WEIGHT bewirkt, dass die Werte der Variablen N1 bei der Auswertung als Fallgewicht verwendet werden.
Die Druckausgabe dazu könnte folgendermaßen aussehen:

Satz 1 Anzahl Fälle: 4318 Gewichtete Fälle: 4750 Fallgewicht: N1

Satz/Pos	1	2	3	4	5	6	7	8	9	10	11	12	Z0	Z1	Z23	Z123
1/1	91 1,9%	-	-	-	-	-	-	-	-	4659 98,1%	-	-	-	4750 100,0%	-	4750 100,0%
1/2	765 16,1%	1005 21,2%	667 14,0%	560 11,8%	666 14,0%	65 1,4%	37 0,8%	444 9,4%	450 9,5%	91 1,9%	-	-	-	4750 100,0%	-	4750 100,0%
1/3	-	-	-	-	-	-	-	-	-	2410 50,7%	-	-	2340 49,3%	2410 50,7%	-	2410 50,7%
1/4	-	-	2410 50,7%	-	-	-	-	2410 50,7%	-	2410 50,7%	-	-	2340 49,3%	-	2410 50,7%	2410 50,7%

Es wurden insgesamt 4 318 statistische Fälle verarbeitet. Das Fallgewicht N1 erstellt daraus 4 750 gewichtete Fälle. In der ersten Spalte der Datensätze besitzen 91 gewichtete Fälle oder 1.9% die Angabe 1 und 4 659 gewichtete Fälle oder 98.1% die Angabe 10.

Beispiel: Mit Folgestatement

```
INPUT-EXT   ASCII   FNAME=STUDIE007.CLB   RC=D(1,1)1..4   SIZE=100
HOLECOUNT
-K2(3 .. 5)
```

Durch das Statement INPUT-EXT wird eine externe Eingabedatei im ASCII-Format angefordert. Die einzelnen Datensätze sind 100 Bytes lang und zu jedem statistischen Fall sind vier Datensätze mit den Satz-Kennzeichen 1 . . . 4 möglich.

Im Folgestatement von HOLECOUNT wird die Auswertung der Bytes 3 . . . 5 des zweiten Datensatzes eines jeden Falles angefordert. Eine solche Auswertung könnte folgendermaßen aussehen:

Satz 2 **Anzahl Fälle: 4318**

Satz/Pos	1	2	3	4	5	6	7	8	9	10	11	12	Z0	Z1	Z23	Z123
2/3	-	-	-	-	-	-	-	-	-	2118 49,1%	-	-	2200 50,9%	2118 49,1%	-	2118 49,1%
2/4	-	-	2118 49,1%	-	-	-	-	2118 49,1%	-	2118 49,1%	-	-	2200 50,9%	-	2118 49,1%	2118 49,1%
2/5	-	-	-	-	-	-	-	-	-	2118 49,1%	-	-	2200 50,9%	2118 49,1%	-	2118 49,1%

Jedes auszuwertende Byte wird vorübergehend vom ASCII-Code in eine COLBIN-Spalte umgewandelt. Die Lochungen dieser Spalte werden dann für HOLECOUNT gezählt.
Für diese Codeumwandlung werden die Tabellen *Zeichenzuordnung ANSI, ASCII, EBCDIC* und *Lochpositionen der COLBIN-Spalten* aus dem Anhang verwendet.

Enthält ein Byte zum Beispiel den Buchstaben 'A', so wird dieser zunächst in den hexadezimalen Wert 24 00 umgewandelt, die COLBIN-Darstellung dieses Buchstabens. Die entsprechenden Lochungen Y und 1 werden in der HOLECOUNT-Tabelle gezählt.

Dem Sonderzeichen '(' entsprechen die hexadezimalen Bytes 20 12 und die Lochungen Y, 5 und 8. Es wird daher als Kombination dieser drei Angaben ausgewertet.

Beispiel: Für CSV-Dateien

```
INPUT-EXT  FNAME='test6.csv'
           SEP=';'
HOLECOUNT
-K1(2...4)
```

Durch das Statement INPUT-EXT wird eine externe Eingabedatei im CSV-Format angefordert.
Das Folgestatement *–K1(2 . . . 4)* von HOLECOUNT wählt die Spalten 2 bis 4 des ersten Datensatzes eines jeden Falles zur Auswertung aus. Die Ausgabe dazu könnte folgendermaßen aussehen:

Satz 1 Anzahl Fälle: 1204

Satz/Spalte			
1/2			
	keine Angabe	235	19,5%
	eine Angabe	523	43,4%
	zwei Angaben	327	27,2%
	drei oder mehr Angaben	119	9,9%
	1	368	30,6%
	2	216	17,9%
	3	211	17,5%
	4	126	10,5%
	5	99	8,2%
	6	85	7,1%
	7	85	7,1%
	8	80	6,6%
	9	80	6,6%
	10	71	5,9%
	11	66	5,5%
	12	65	5,4%
1/3			
	keine Angabe	1160	96,3%
	eine Angabe	44	3,7%
	-2,17	1	0,1%
	0	1	0,1%
	10	5	0,4%
	20	14	1,2%
	25	2	0,2%
	30	14	1,2%
	35	7	0,6%
1/4			
	eine Angabe	1204	100,0%
	Berlin	128	10,6%
	Bremen	127	10,5%
	Dresden	119	9,9%
	Frankfurt	103	8,6%
	Hamburg	139	11,5%
	Hannover	131	10,9%
	Köln	108	9,0%
	Leipzig	106	8,8%
	München	123	10,2%
	Stuttgart	120	10,0%

Die Mehrfachnennungen in Spalte 3 stammen aus Datenwerten der Form:

```
2 5-7 12
```

die als Angaben 2, 5, 6, 7 und 12 verstanden werden.

–IF –IFN –ELSE –EIF

Dieses sind Folgestatements zu ADD-PROC, MOD-PROC, UPD-PROC und OUTPUT-EXT.
Sie entscheiden anhand von Variablenwerten der einzelnen statistischen Fälle, ob die dahinter liegenden Statements für den jeweiligen Fall verarbeitet oder nicht verarbeitet werden.
Im Gegensatz dazu werten die #IF-Statements Makros aus und entscheiden darüber, ob die dahinter liegenden Statements überhaupt eingelesen oder unverarbeitet übersprungen werden.

Folgende Angaben sind möglich:

–IF logischer Ausdruck < **BZ** >

Dabei können beliebig komplizierte logische Ausdrücke verwendet werden, wie sie im Abschnitt *Logische Ausdrücke* beschrieben sind.
Bei der Verarbeitung jedes statistischen Falles wird der logische Ausdruck ausgewertet.
Ist das Ergebnis *wahr*, so werden die hinter IF liegenden Statements normal verarbeitet.
Ist es hingegen *falsch*, so werden alle Statements, die zwischen dem IF und dem dazugehörigen EIF liegen, nicht ausgeführt.

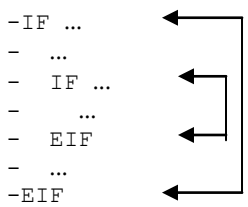
Zum Beispiel:

```
-IF C10.Z23
-   PRT 'Falsche Mehrfachnennungen' C10
-   C10=C10 SR
-EIF
```

Enthält die Variable C10 in diesem Beispiel für einen statistischen Fall Mehrfachnennungen, so wird durch PRT eine Fehlermeldung ausgegeben und der Fehler durch die Wertezuweisung C10=C10 SR korrigiert.

Das zu einem IF gehörige EIF-Statement ist zunächst das erste hinter IF liegende EIF.
Die dazwischen liegenden Statements bilden den Wirkungsbereich des IF-Statements.
IF-Statements lassen sich jedoch auch schachteln:

Im Bereich eines IF kann ein weiteres IF-Statement liegen. Dieses wird ebenfalls durch ein dahinter liegendes EIF beendet. Das EIF zum zweiten IF liegt vor dem EIF zum ersten IF. Zum Beispiel:



Die Statements im Wirkungsbereich eines solchen zweiten, geschachtelten IF-Statements werden nur ausgeführt, wenn die logischen Ausdrücke in beiden IF-Statements *wahr* sind.
Eine solche Schachtelung darf bis zu zehn Stufen tief sein.

Die Angabe BZ (*blank to zero)* hinter dem logischen Ausdruck ersetzt fehlende Werte von N-Variablen im Ausdruck durch Null. In dem Statement

```
-IF (N10+N11)>10
```

ist der logische Ausdruck nicht erfüllt, wenn N10 den Wert Z0 und N11 den Wert 20 besitzt.
Durch BZ hinter dem Ausdruck wird der Wert Z0 in N10 als 0 verarbeitet, so dass der logische Ausdruck wahr wird.

–IFN logischer Ausdruck < **BZ** >

Das Statement IFN arbeitet analog zum IF-Statement, jedoch werden die auf IFN folgenden Statements nur dann ausgeführt, wenn der logische Ausdruck *nicht* erfüllt ist, also den Wert *falsch* besitzt.

–EIF

Dieses Statement beendet vorausgehende IF- oder IFN-Statements.

–ELSE

Dieses Statement darf nur hinter einem IF oder IFN stehen.
Hinter einem IF-Statement beendet ELSE den Wirkungsbereich des IF und sorgt dafür, dass die hinter dem ELSE stehenden Statements immer dann ausgeführt werden, wenn der logische Ausdruck im IF *nicht* erfüllt ist. Der Wirkungsbereich des ELSE wird durch das nächste EIF beendet.
Zum Beispiel:

```
-IF C10.1 | ...10
-   PRT 'Angaben C10.1...10 vorhanden'
-ELSE
-   PRT 'Keine Angaben C10.1...10'
-EIF
```

Ist eines der Merkmale 1 ... 10 der Variablen C10 in diesem Beispiel gesetzt, so wird die Meldung *Angaben C10.1...10 vorhanden* ausgegeben. Ist dagegen keines dieser Merkmale gesetzt, so erscheint die Meldung *Keine Angaben C10.1...10* .

Entsprechend sorgt ein ELSE-Statement hinter einem IFN dafür, dass die hinter ELSE stehenden Statements immer dann ausgeführt werden, wenn der logische Ausdruck aus IFN erfüllt ist.

Beispiel 1

```
-IF   N1>=20  &  N1<=100
-   PRT  N1   'FALSCHER WERT'
-   N1=0
-EIF
```

Der logische Ausdruck in diesem IF-Statement ist erfüllt, wenn die numerische Variable N1 einen Wert zwischen 20 und 100 besitzt.
Für alle statistischen Fälle, bei denen dieser Ausdruck erfüllt ist, werden die beiden darauf folgenden Statements ausgeführt:
Es wird der Wert der Variablen N1 zusammen mit dem Text *FALSCHER WERT* ausgedruckt, und danach wird der Variablen N1 der neue Wert 0 zugewiesen.
In den statistischen Fällen, in denen der logische Ausdruck falsch ist, werden diese Statements übersprungen.

Beispiel 2

```
-IF  N1>20  &  N1<100
-   PRT  N1  'FALSCHER WERT'
-   N1=0
-   IF  C10.1 | ...9
-      N1=10
-   EIF
-EIF
```

Das erste IF-Statement ist identisch mit dem des vorigen Beispiels. Sein Wirkungsbereich enthält ein zweites, geschachteltes IF-EIF-Paar.

Ist der logische Ausdruck in dem ersten IF-Statement erfüllt, so wird zunächst wieder der Variablenwert N1 und die Meldung *FALSCHER WERT* ausgedruckt. Ebenso wird dann der Variablen N1 der Wert 0 zugewiesen. Durch das zweite IF-Statement wird nun zusätzlich geprüft, ob in der Variablen C10 eines der Merkmale 1...9 erfüllt ist.
Wenn dies der Fall ist, wird der Variablen N1 durch das darauf folgende Statement der Wert 10 zugewiesen.

Die Wertezuweisung N1=10 wird also nur in den Fällen ausgeführt, in denen der logische Ausdruck sowohl im ersten IF-Statement als auch im zweiten IF-Statement erfüllt ist.

Zwischen dem Strich in Position 1 und den späteren Angaben können in einem Folgestatement beliebig viele Leerstellen eingefügt werden. Auf diese Weise werden die Statements besser lesbar.

Beispiel 3

```
-IFN  C10.1 | ...9
-  C10.1=1
-EIF
```

Der logische Ausdruck in dem IFN-Statement ist erfüllt, wenn wenigstens eines der Merkmale 1 ... 9 der Bedingungsvariablen C10 erfüllt ist. Im Gegensatz zum IF werden die Statements bis zum abschließenden EIF aber gerade dann ausgeführt, wenn der logische Ausdruck nicht erfüllt ist, also keines der Merkmale 1 ... 9 der Variablen C10 wahr ist.

#IF, #ELSE, #EIF

Die Statements #IF, #ELSE und #EIF entscheiden anhand von Makrowerten, ob die dahinter liegenden Statements eingelesen oder ungelesen übergangen werden.
Im Gegensatz dazu werten die –IF Statements die Variablenwerte der einzelnen statistischen Fälle aus und entscheiden darüber, ob die dahinter liegenden Statements für den jeweiligen Fall verarbeitet oder übersprungen werden.

Folgende Angaben sind möglich:

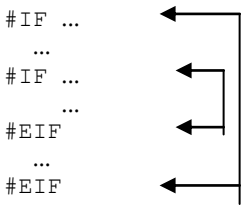
#IF Makroausdruck

Hier sind beliebig komplizierte Ausdrücke möglich, wie sie im Abschnitt *Makros: Makroausdrücke* beschrieben sind.
Beim Einlesen der Statements wird der Makroausdruck ausgewertet.
Ist das Ergebnis *wahr*, so werden die hinter #IF liegenden Statements normal verarbeitet.
Ist das Ergebnis hingegen *falsch*, so werden alle Statements, die zwischen dem #IF und dem dazugehörigen #EIF oder #ELSE liegen, nicht verarbeitet. Zum Beispiel:

```
#IF (#a > 2) & (#b > 10)
DO <1>=1..#a
XTAB COL=C10  ROW=#row<1>
EDO
#EIF
```

Enthält das Makro #a einen Wert größer 2 und das Makro #b einen Wert größer 10, so werden die Statements zwischen #IF und #EIF eingelesen. Ist diese Bedingung dagegen nicht erfüllt, so werden erst wieder die Statements hinter #EIF eingelesen

#IF-Statements lassen sich auch schachteln:
Im Wirkungsbereich eines #IF kann ein weiteres #IF-Statement liegen. Dieses zweite #IF wird ebenfalls durch ein dahinter liegendes #EIF beendet. Zum Beispiel:



Die Statements im Wirkungsbereich eines solchen zweiten, geschachtelten #IF-Statements werden nur ausgeführt, wenn die logischen Ausdrücke in beiden #IF-Statements *wahr* sind.
Eine solche Schachtelung darf bis zu zehn Stufen tief sein.

Das Statement PROCESS hat eine ähnliche Wirkung wie #IF, kann aber nicht geschachtelt werden.

#EIF

Dieses Statement beendet ein vorausgehendes #IF-Statements.

#ELSE

Dieses Statement beendet die Wirkung des vorausgehenden #IF. Die hinter dem #ELSE liegenden Statements werden nur eingelesen, wenn der Makroausdruck im #IF *nicht* erfüllt ist.
Der Wirkungsbereich des #ELSE wird durch das nächste #EIF beendet. Zum Beispiel:

```
#IF #sprache = D
XTAB TITLE='Rangreihe nach Nettoreichweite' ...
#ELSE
XTAB TITLE='Ranking on coverage' ...
#EIF
```

Enthält das Makro #sprache das Zeichen D, so wird das hinter #IF stehende XTAB-Statement mit einem deutschen TITLE-Text eingelesen und verarbeitet, das XTAB-Statement hinter #ELSE jedoch übersprungen.
Enthält das Makro #sprache einen Wert ungleich D, so wird das Statement mit deutschem Text übersprungen und dafür das Statement hinter #ELSE verarbeitet.

#ELSE IF Makroausdruck

Dieses Statement beendet die Wirkung eines davor stehenden #IF oder #ELSE IF.
Die hinter dem #ELSEIF liegenden Statements werden nur dann eingelesen, wenn der Makroausdruck im #ELSEIF erfüllt ist und darüber hinaus die Makroausdrücke der davor stehenden #IF und #ELSE IF *nicht* erfüllt sind. Zum Beispiel:

```
#IF #sprache = D
XTAB TITLE='Rangreihe nach Nettoreichweite' ...
#ELSE IF #sprache = E
XTAB TITLE='Ranking on coverage' ...
#ELSE
Fehler: Makro 'sprache' mit falschem Wert
#EIF
```

Enthält das Makro #sprache das Zeichen D, so wird das hinter #IF stehende XTAB-Statement mit einem deutschen TITLE-Text eingelesen und verarbeitet. Die Statements zwischen #ELSE IF und #EIF werden dagegen ausgelassen.

Enthält das Makro #sprache das Zeichen E, so wird nur das XTAB-Statement mit dem englischen Text eingelesen und die übrigen Statements übersprungen.

Besitzt das Makro #sprache einen Wert ungleich D oder E, so wird die Zeile *Fehler: Makro ...* eingelesen und alle übrigen Statements ausgelassen.
Diese Zeile enthält kein gültiges Statement, produziert also die Fehlermeldung 100 und führt zum vorzeitigen Ende des Programmlaufs.

INCLUDE

Das Statement INCLUDE fügt Statements aus einer anderen Datei in die laufende Auswertung ein.
Die eingefügten Statements werden direkt hinter das INCLUDE gestellt und im Zählprotokoll durch das Zeichen + zwischen Statementnummer und Statementtext gekennzeichnet.

INCLUDE-Statements sind an beliebiger Stelle in der CNTA-Eingabe erlaubt und haben folgenden Aufbau:

INCLUDE dateiname

Dabei ist *dateiname* der Name der Datei mit den einzufügenden Statements. Dateinamen mit Leerzeichen sind in Hochkommas einzuschließen. Zum Beispiel:

```
INCLUDE 'Studie25 Tabellen.stm'
INCLUDE Studie25.stm
INCLUDE \\SERVER\DATEN\Tabellen.stm
```

Enthält der Dateiname keine Pfadangabe, so erwartet CNTA die Datei im Verzeichnis der Statementdatei. CNTW sucht sie zunächst im Verzeichnis der REP-Datei und danach im Verzeichnis der Statementdatei.

Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\STUDIE27.INT
```

Die Dateinamen dürfen Makros enthalten, jedoch keinen DO-Index <1> ... <99>.

INCLUDE-Statements lassen sich schachteln: Eine mit INCLUDE angeforderte Statement-Datei kann selbst wieder INCLUDE-Statements enthalten. Solche Schachtelungen sind bis zu 10 Stufen möglich.

INCLUDE-Statements sind insbesondere dann nützlich, wenn eine Gruppe von Statements in mehreren CNT-Läufen wiederverwendet werden.

Eine INCLUDE-Datei kann in einem CNTA-Lauf beliebig oft aufgerufen werden.
Weiter ist es möglich, dieselbe INCLUDE-Datei in einem CNTA-Lauf beliebig oft einzufügen.

Beispiel

```
INCLUDE   FORMAT1
```

Durch dieses Statement werden Textzeilen aus der Datei FORMAT1 eingelesen und als Statements in die laufende Auswertung eingefügt. Die Datei FORMAT1 könnte dabei folgende Angaben enthalten:

```
PAGE   HEAD='XYZ-Marktforschung'/'Tel: (012) 34 56 78'
       POS=160
       LINES=62
       NODATE
XTAB   SUPPRESS=SPACES     TABS=25, 8
```

Das PAGE-Statement wird dann durch INCLUDE in die laufende Auswertung eingefügt. Auf diese Weise können PAGE-Statements für verschiedene Zwecke gespeichert und bei Bedarf mit INCLUDE aufgerufen werden.
Das ebenfalls in der INCLUDE-Datei enthaltene XTAB enthält keine ROW-, COL- und FILTER-Angabe. Es erzeugt daher auch keine Tabelle, sondern sorgt nur für eine Voreinstellung der Angaben SUPPRESS und TABS für XTAB-Statements, die hinter INCLUDE eingegeben werden.

INDEX

Das Statement INDEX erstellt automatisch ein Inhaltsverzeichnis zu den mit XTAB erzeugten Tabellen.

Für dieses Inhaltsverzeichnisses stehen die Texte der Angaben TITLE, STUBHEAD, FILTER, BOTTOM und ROWS aus XTAB zur Auswahl sowie die Texte HEAD der Statements PAGE und PAGEP.

Es können auch mehrere dieser Textarten gleichzeitig angefordert werden. Sie erscheinen dann eingerückt untereinander in der festen Rangfolge: PAGEHEAD, TITLE, STUBHEAD, FILTER, BOTTOM und ROWS. Leerzeilen und Zeilen mit den Strichzeichen - = und _ werden stets aus den Texten entfernt.

Hinter den Texten werden die Seitennummern der entsprechenden Tabellen ausgegeben.
Mit Hilfe der Angabe PCHAR im Statement PAGE oder PAGEP kann eine punktierte Linie vor den Seitennummern erzeugt oder unterdrückt werden.

Ein solches Inhaltsverzeichnis enthält Angaben zu allen XTAB-Statements, die vor dem INDEX-Statement liegen. In einem Verarbeitungslauf können auch mehrere INDEX-Statements vorkommen. Dann wird zu jedem INDEX ein Inhaltsverzeichnis für die zwischen dem laufenden und dem vorigen INDEX liegenden Tabellen gedruckt.

Wird ein durch INDEX erzeugtes Inhaltsverzeichnis in eine PDF-Datei ausgegeben, so erzeugt das Programm dazu Lesezeichen (Bookmarks). Bei der Ausgabe in Excel-Dateien werden Hyperlinks auf die Tabellen eingerichtet.

INDEX erlaubt folgende Angaben:

HEAD = | **< ' t ' > < ,L**i **> < ,FT**i **>** | **NO**

Diese Angabe erstellt einen Überschriftstext auf der ersten Seite des Inhaltsverzeichnisses.
Fehlt diese Angabe, so wird dafür *Inhaltsverzeichnis* ausgegeben.

't' Text, der als Überschrift auszugeben ist.
Er kann aus mehreren Zeilen bestehen, wie im Abschnitt *Textzeilen* beschrieben.
Fehlt ein solcher Text, wird stattdessen *Inhaltsverzeichnis* verwendet.

Li Diese Angabe beeinflusst die Linien oberhalb und unterhalb der Überschrift. Sie ist nur wirksam, wenn das Statement PAGEP aktiv ist. Mit L0 werden keine Linien ausgegeben, mit L1 bis L16 die entsprechenden Linien aus dem Statement PAGEP. Ohne diese Angabe wird L7 verwendet.

FTi Diese Angabe legt die Schrift für den Überschriftstext fest. Es sind die Schriften FONT1 bis FONT9 aus dem Statement PAGEP möglich. Sie ist nur wirksam, wenn das Statement PAGEP aktiv ist. Ohne die Angabe FTi wird die Schrift FONT5 verwendet.

NO Hiermit wird kein Überschriftstext ausgeben; der Indextext beginnt ganz oben auf dem Blatt.

NUM = FTi

Diese Angabe legt die Schriftart für die Seitennummern im Inhaltsverzeichnis fest. Sie ist nur wirksam, wenn das Statement PAGEP aktiv ist. Ohne die Angabe NUM wird die Schrift FONT2 aus dem Statement PAGEP verwendet.

WIDTH = n Diese Angabe legt die Breite *n* für die Aufbereitung der Texte fest.
Ohne WIDTH ist die Breite abhängig von der verwendeten Schriftgröße.

Ist PAGEP wirksam, so ist die Breite *n* in Millimetern anzugeben; auch die anderen Maßeinheiten aus dem Statement PAGEP sind möglich. Bei PAGE ist als Breite die Anzahl von Zeichen anzugeben. Die Mindestbreite beträgt 5mm oder 5 Zeichen.

PAGEHEADL
PAGEHEADC
PAGEHEADR
< **=** < n > < **, FT**i > < **, UNDO** > >

Diese Angabe stellt HEAD-Texte aus den Statements PAGE und PAGEP in das Inhaltsverzeichnis.

PAGEFOOTL
PAGEFOOTC
PAGEFOOTR
< **=** < n > < **, FT**i > < **, UNDO** > >

Diese Angabe stellt FOOT-Texte aus den Statements PAGE und PAGEP in das Inhaltsverzeichnis.

TITLE < = < n > < **, FT**i > < **, UNDO** > >

Diese Angabe stellt die TITLE-Texte der XTAB-Statements in das Inhaltsverzeichnis.

STUBHEAD < = < n > < **, FT**i > < **, UNDO** > >

Diese Angabe stellt die STUBHEAD-Texte der XTAB-Statements in das Inhaltsverzeichnis.

FILTER < = < n > < **, FT**i > < **, UNDO** > >

Diese Angabe stellt die FILTER-Texte der XTAB-Statements in das Inhaltsverzeichnis.

BOTTOM < = < n > < **, FT**i > < **, UNDO** > >

Diese Angabe stellt die BOTTOM-Texte der XTAB-Statements in das Inhaltsverzeichnis.

ROWS < = < n > < **, FT**i > < **, UNDO** > >

Diese Angabe stellt Zeilentexte der XTAB-Statements in das Inhaltsverzeichnis.
Dabei werden nur die Texte der oberen Textstufe für das Inhaltsverzeichnis ausgewählt.
Liegen zu einer Zeile Variablen- und Merkmalstext vor, so ist dies der Variablentext.
Bei Staffelungen wird nur der Text der obersten Staffelstufe verwendet.
Einstufige Zeilentexte, wie zum Beispiel *Basis*, werden nicht in das Inhaltsverzeichnis übernommen.

---

n Die Angabe n = 1 ... 255 beschränkt die Anzahl von Zeilen, die aus dem jeweiligen Text in das Inhaltsverzeichnis übernommen werden sollen. Besitzt ein Text mehr Zeilen, als hier angegeben sind, so erscheinen die letzten Zeilen nicht im Inhaltsverzeichnis.
Leerzeilen und Zeilen mit den Strichzeichen - = und _ werden stets aus den Texten entfernt und hier auch nicht mitgezählt.
Fehlt die Angabe der Zeilenanzahl, so werden alle Zeilen übernommen.

UNDO Mit dieser Angabe werden die Zeilenumläufe der Texte und die Schriftenkommandos vor der Ausgabe für INDEX entfernt. Anschließend werden die Texte mit neuem Zeilenumlauf in das Inhaltsverzeichnis eingepasst.

FTi Diese Angabe legt die Schrift fest, in der die dazugehörigen Texte zu drucken sind.
Es sind die Angaben FONT1 bis FONT16 aus dem Statement PAGEP möglich. Die Schriftangabe FTi ist nur wirksam, wenn das Statement PAGEP aktiv ist. Ohne diese Angabe wird für die Überschrift FONT5 und sonst FONT2 verwendet.

Beispiel: Index einspaltig

```
XTAB   TITLES='Alle Teilnehmer'
       ROW=TOTAL;C2030   COL=TOTAL;C20
XTAB   ROW=TOTAL;C2030   COL=TOTAL;C20
XTAB   ROW=TOTAL;C2040   COL=TOTAL;C20
XTAB   ROW=TOTAL;C2050   COL=TOTAL;C20
XTAB   ROW=TOTAL;C2060   COL=TOTAL;C20
XTAB   ROW=TOTAL;C2080   COL=TOTAL;C20

PAGEP   PCHAR='.'

INDEX   HEAD='Inhaltsverzeichnis zur Studie 0912'
             TITLE   ROWS
```

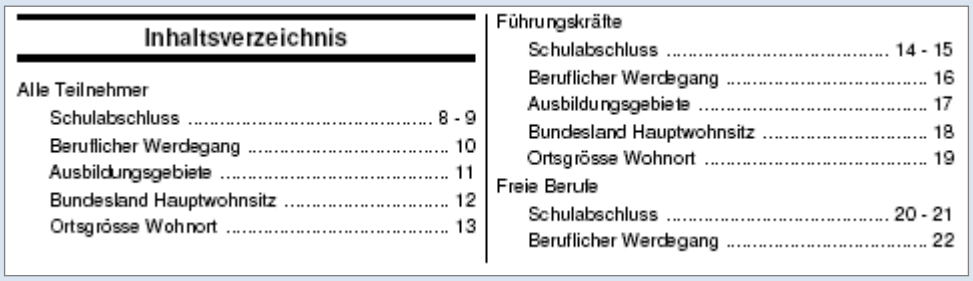
Das INDEX-Statement erzeugt aus den TITLE- und ROW-Angaben der davorliegenden XTAB-Statements das folgende Inhaltsverzeichnis.
Die Angabe PCHAR im Statement PAGEP erzeugt dabei die punktierten Linien.

Beispiel: Index zweispaltig

```
PAGEP    PCHAR='.'    HOR=*,L3-*
INDEX    TITLE    ROWS
```

Im Statement PAGEP teilt die Angabe HOR=*,L3-* die Seiten in zwei gleich breite Spalten mit einer Linie L3 dazwischen.
Die Angabe PCHAR='.' sorgt für punktierte Linien im Inhaltsverzeichnis.

Das Statement INDEX erzeugt aus den TITLE- und ROW-Angaben der davorliegenden XTAB-Statements das folgende Inhaltsverzeichnis:



Inhaltsverzeichnis

INPUT-EXT

Durch das Statement INPUT-EXT wird eine Datei mit statistischen Fällen zur Verarbeitung angefordert, die in dem externen Datenformat ASCII, ANSI, EBCDIC, COLBIN oder QUANTUM vorliegt. Neben Datensätzen mit fester und variabler Satzlänge sind auch CSV-Dateien möglich, bei denen die Datenwerte durch ein Separatorzeichen - zum Beispiel Komma oder Semikolon - voneinander getrennt sind.

Externes Datenformat bedeutet, dass die statistischen Fälle noch nicht in einem früheren CNTA-Lauf durch das Statement OUTPUT-INT in das CNT-interne Datenformat umgewandelt wurden.

Ein statistischer Fall enthält zum Beispiel die Daten zu einem Fragebogen oder Interview. Auf einer externen Datei kann ein solcher Fall aus einem oder mehreren Datensätzen bestehen. Mehrere Datensätze eines Falles müssen direkt hintereinander liegen. Die möglichen Dateiformate und Datenfelder werden in den Abschnitten *Externe Dateien* und *Externe Datenfelder* beschrieben.
In einem Verarbeitungslauf kann das Statement INPUT-EXT nur einmal verwendet werden.

Zu INPUT-EXT gibt es folgende Angaben:

FNAME = dateiname

Hier ist der Name der Eingabedatei anzugeben. Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
FNAME=C:\DATEN\STUDIE27.EXT
FNAME='STUDIE27 EINGABE'
FNAME=\\SERVER\DATEN\STUDIE27.EXT
```

Enthält *dateiname* keine Pfadangabe, so erwartet CNTA die Datei im Verzeichnis der Statementdatei. CNTW sucht sie zunächst im Verzeichnis der REP-Datei und danach im Verzeichnis der Statementdatei.
Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\STUDIE27.EXT
```

ID | **ID1** | **ID2** | **ID3** | **ID4** = **A** | **D** | **F** | **G** | **P** < d > **(a .. e)** | **(a , c)**

AK | **DK** < d > **(< 's' >)**

AS | **DS** < d > **(r)**

Diese Angabe verlangt Fall-Identifikationen in den Datensätzen. Dies können Fragebogennummern sein, Testpersonen-Kennzeichen oder andere Kennzeichen der einzelnen statistischen Fälle.

Fall-Identifikationen dienen dazu, mehrere direkt hintereinander liegende Datensätze zu einem statistischen Fall zusammenzufassen.

Sind in einem Verarbeitungslauf mehrere Dateien zu verarbeiten (INPUT-INT, INPUT-SEC oder INPUT-EXT), so werden die Fall-Identifikationen auch dazu verwendet, die Daten aus den verschiedenen Eingabedateien zu einem statistischen Fall zusammenzuführen.

Weiter werden sie zur Erleichterung der Fehlersuche in Fehlermeldungen ausgedruckt.

Wird keine Fall-Identifikation angegeben, so führt CNTA selbsttätig eine einstufige Fall-Identifikation ID ein, für die alle eingelesenen statistischen Fälle durchnummeriert werden: Lückenlos aufsteigend, beginnend bei 1.
Liegt nur eine einstufige Fall-Identifikation vor, so muss diese mit ID angegeben werden.
Bei mehrstufigen Identifikationen ist mit der höchsten Stufe bei ID zu beginnen und lückenlos mit ID1 ... ID4 bis zur untersten gewünschten Stufe fortzufahren.

Wird eine ID-Angabe gemacht, so müssen die statistischen Fälle in aufsteigender Folge dieser Identifikation sortiert in der Datei enthalten sein. Abweichungen von dieser Regel sind jedoch mit Hilfe der Angaben ASORT, DSORT, NSORT und SORT möglich, die weiter unten beschrieben werden.

Wenn Angaben ID ... ID4 vorliegen, führt CNTA sehr umfangreiche Prüfungen zum korrekten Aufbau der Datenbestände durch; insbesondere wird untersucht, ob alle erforderlichen Datensätze zu der jeweiligen Identifikation vorhanden sind.
Fehlt die Angabe ID und sind mehrere Datensätze pro Fall zu verarbeiten, so werden die eingelesenen Datensätze solange dem laufenden Fall zugeordnet, bis ein bereits vorliegendes Satz-Kennzeichen erneut eingelesen wird. Diese gilt dann als Anfang des nächsten Falles. Bei unsauberen Datenbeständen ist durch diese vereinfachte Verarbeitungsform aber eine Vielzahl von Fehlern möglich, die von CNTA nicht mehr erkannt werden können.

Für die Fall-Identifikationen lassen sich folgende Feldtypen verwenden, so wie im Abschnitt *Externe Datenfelder* beschrieben:

A Die Fall-Identifikation besteht aus alphanumerischen Zeichen.

AK In CSV-Dateien: Die Fall-Identifikation besteht aus alphanumerischen Zeichen mit Schlüsselwort.

AS In CSV-Dateien: Die Fall-Identifikation besteht aus alphanumerischen Zeichen.

D Die Fall-Identifikation besteht aus numerischen Zeichen.

DK In CSV-Dateien: Die Fall-Identifikation besteht aus numerischen Zeichen mit Schlüsselwort.

F Die Fall-Identifikation besteht aus einer Dualzahl mit Vorzeichen (Integer).

G Die Fall-Identifikation besteht aus einer Dualzahl ohne Vorzeichen (Unsigned Integer).

P Die Fall-Identifikation besteht aus einer gepackten Dezimalzahl.
Negative Werte werden vor der Verarbeitung in positive Werte umgesetzt.

d Nummer des Datensatzes, aus dem die Fall-Identifikation entnommen werden soll.
Fehlt diese Angabe, so muss die Fall-Identifikation in jedem Datensatz vorliegen, sonst nur in dem hier angegebenen Satz.
ACHTUNG: Diese Nummer ist nicht das Satz-Kennzeichen selbst, sondern die Reihenfolgenummer der hinter RC angegebenen Satz-Kennzeichen.

a erstes Byte 1 ... 1 000 000 der Fall-Identifikation in den externen Datensätzen;
erste Spalte 1 ... 500 000 im COLBIN-Format.

c Länge der Fall-Identifikation in den externen Datensätzen in Bytes; bei COLBIN in Spalten.
Die maximal zulässigen Längen sind abhängig vom Typ der Datenfelder:
im A-Feld 1 ... 127 Bytes oder Spalten
im D-Feld 1 ... 18 Bytes oder Spalten
im F-Feld 1 ... 8 Bytes
im G-Feld 1 ... 8 Bytes
im P-Feld 1 ... 10 Bytes.

e letztes Byte 1 ... 1 000 000 der Fall-Identifikation in den externen Datensätzen;
letzte Spalte 1 ... 500 000 im COLBIN-Format.
Die Länge der Fall-Identifikation muss den unter *c* angegebenen Regeln entsprechen.

'k' Diese Angabe ist in Hochkommas einzuschließen. Sie legt das Schlüsselwort fest, an dem die ID-Werte in den Datensätzen zu erkennen sind. Fehlt diese Angabe, so wird als Schlüssel *ID ... ID4* verwendet.

r Die Angabe r = 1 ... 32 767 ist die Reihenfolgenummer des Wertes im Datensatz *d* einer CSV-Datei. Leerzeichen vor und hinter der Fall-Identifikation werden ignoriert. Nur bei AS-Feldern gehören die führenden Leerzeichen zur Fall-Identifikation. Für die Länge der Werte gelten die unter *c* angegebenen Regeln für A- und D-Felder.

s Die Angabe A sorgt hier für aufsteigende Sortierung der entsprechenden Fall-Identifikation, die Angabe D für absteigende Sortierfolge.
Eine solche Angabe A oder D hinter ID ... ID4 ist nur möglich, wenn im Statement INPUT-EXT gleichzeitig eine der weiter unten beschriebenen Angaben SORT, ASORT oder DSORT vorliegen.
Die Angaben A und D verändern dabei die Sortierfolge für einzelne Fall-Identifikationen. In dem Statement:

```
INPUT-EXT   SORT   ID=F(1,4)   ID1=F(5,2)D  . . .
```

sorgt SORT zunächst für aufsteigende Sortierung der beiden Fall-Identifikationen ID und ID1. Das D hinter ID1 bewirkt nun eine absteigende Sortierung von ID1. Die aufsteigende Reihenfolge von ID wird dadurch nicht verändert.

MULTIPLE

Die Angabe MULTIPLE erlaubt mehrere direkt hintereinander liegende statistische Fälle mit gleicher Fall-Identifikation. Ohne MULTIPLE führen doppelte Fall-Identifikationen zu Fehlermeldungen, wobei nur der erste Fall verarbeitet wird.

Wird mit MULTIPLE gearbeitet und besitzen die Eingabedaten mehrere Datensätze pro Fall, so ist nur eine beschränkte Datenprüfung möglich: Liegen zwei Fälle mit gleicher Fall-Identifikation direkt hintereinander, so werden alle Datensätze zum ersten Fall gezählt, bis ein Satz-Kennzeichen zum zweiten Mal erscheint. Diese wird dann als Anfang des zweiten Falles angesehen.

ASORT
DSORT
NSORT
SORT

Werden in INPUT-EXT Fall-Identifikationen ID ... ID4 angegeben, so müssen die statistischen Fälle der Datei nach diesen Identifikationen aufsteigend sortiert sein.
Liegen mehrere Identifikationen ID ... ID4 vor, so besitzt ID die höchste Priorität in der Sortierung und ID4 die niedrigste. Falsch platzierte Fälle führen zu einer Fehlermeldung und werden nicht verarbeitet. Für abweichende Verarbeitungsregeln sorgen:

SORT Diese Angabe sortiert die Fälle von INPUT-EXT nach aufsteigenden Fall-Identifikationen.
Die Eingabedatei wird dabei nicht verändert. Es wird nur in einer temporären Zwischendatei ein sortierter Index erstellt. Erst danach werden die statistischen Fälle in der neuen Reihenfolge verarbeitet.
Wird eine sortierte externe Datei gewünscht, so kann diese durch OUTPUT-EXT sehr einfach erstellt werden, indem in INPUT-EXT durch SORT für eine sortierte Eingabe gesorgt wird. Schließlich besteht auch die Möglichkeit, einige Fall-Identifikationen aufsteigend und andere absteigend zu sortieren. Neben der Angabe SORT ist hinter die Angabe IDi der absteigend zu sortierenden Identifikationen der Buchstabe D zu stellen. Weitere Angaben dazu finden sich bei der Beschreibung von ID ... ID4.

NSORT Hierdurch wird die Reihenfolgeprüfung ausgeschaltet. Unsortierte Eingabedaten werden ohne Fehlermeldung verarbeitet. Die gleichzeitige Eingabe von INPUT-INT ist damit nicht möglich.

ASORT ist identisch mit SORT.

DSORT hat die gleiche Wirkung wie SORT, jedoch werden die statistischen Fälle in absteigender Reihenfolge der Fall-Identifikationen verarbeitet.

IDSHORT

Mit der Angabe IDSHORT lassen sich die Daten eines Falles aus INPUT-EXT an alle Fälle gleicher Fall-Identifikation aus INPUT-INT übertragen.

Ohne die Angabe IDSHORT müssen beide Dateien die gleiche Anzahl von ID-Stufen besitzen. Jeder Fall aus INPUT-EXT wird höchstens mit einem passenden Fall aus INPUT-INT zusammengeführt und über UPD-PROC verarbeitet. Weitere dazu passende Fälle aus INPUT-INT erhalten keine Daten aus INPUT-INT zugewiesen. Fälle ohne Gegenstück in der jeweils anderen Datei werden getrennt verarbeitet, Fälle aus INPUT-EXT über ADD-PROC, Fälle aus INPUT-INT nur über MOD-PROC.

Mit der Angabe IDSHORT darf das Statement INPUT-EXT auch weniger ID-Stufen als INPUT-INT besitzen. Jeder Fall aus INPUT-EXT wird mit allen Fällen aus INPUT-INT mit passender Identifikation zusammengeführt und über UPD-PROC verarbeitet. Fälle aus INPUT-EXT ohne Gegenstück in INPUT-INT werden unverarbeitet übergangen und Fälle aus INPUT-INT ohne Gegenstück in INPUT-INT nur über MOD-PROC verarbeitet.

Der Abschnitt *Einlesen der statistischen Fälle* enthält weitere Angaben über die gemeinsame Verarbeitung mehrerer Dateien.

Zum Beispiel sei in INPUT-EXT nur ID mit IDSHORT angegeben und in INPUT-INT sowohl ID als auch ID1 vorhanden. Auf den beiden Dateien sollen nun statistische Fälle mit den folgenden Fall-Identifikationen vorliegen:

```
INPUT-INT      INPUT-EXT

100.1          100
100.2          102
105.0          106
106.2
```

Dann wird zunächst der Fall 100.1 aus INPUT-INT zusammen mit dem Fall 100 aus INPUT-EXT verarbeitet. Für die Wertezuweisungen sind die Statements aus UPD-PROC zuständig.
Danach wird der Fall 100.2 wieder zusammen mit dem Fall 100 aus INPUT-EXT verarbeitet, ebenfalls mit den Statements aus UPD-PROC.
Anschließend wird der Fall 102 aus INPUT-EXT unverarbeitet übergangen, da in INPUT-INT kein passendes Gegenstück vorhanden ist.
Nun wird der Fall 105.0 aus INPUT-INT verarbeitet. Zu ihm existiert auf INPUT-EXT kein Partner: Die Statements aus ADD-PROC und UPD-PROC sind nicht wirksam.
Schließlich wird der Fall 106.2 in UPD-PROC verarbeitet, wieder zusammen mit dem Fall 106 aus INPUT-EXT.

ASCII
ANSI
UTF8
EBCDIC
COLBIN
QUANTUM
BINARY1
BINARY2
XLSX

Diese Angabe legt das Datenformat und den Zeichencode der Datei fest:
Sie entscheidet, welche externen Datenfelder in der Datei vorkommen können und wie diese dort verschlüsselt sind. Insbesondere wird der Zeichencode der Datenfelder vom Typ A, AK, AS, D, DK, DS, L, M, MK, MS, U, UK und US festgelegt.
CSV-Dateien (siehe Angabe SEP) sind nur in den Formaten ASCII, ANSI und EBCDIC möglich.
Weitere Angaben zu diesen Dateiformaten befinden sich im Abschnitt *Externe Dateien* sowie im Anhang *Zeichenzuordnung ANSI, ASCII, EBCDIC* und *Lochkombinationen der COLBIN-Spalten*.
Ohne diese Angabe wird das ANSI-Format verwendet.

ASCII Dies ist das ursprüngliche Datenformat der Personal Computer im MS/DOS.
Die Längen und Positionen in den Datensätzen sind in Bytes anzugeben.
Alle externen Datenfelder sind möglich.

ANSI Dies ist der von WINDOWS bevorzugte Zeichencode.
Der Unterschied zu ASCII beschränkt sich auf die Umlaute, den Buchstaben ß und einige Sonderzeichen. Die Längen und Positionen in den Datensätzen sind in Bytes anzugeben.
Alle externen Datenfelder sind möglich.

UTF8 Dieser Zeichencode umfasst alle Zeichen des 16-Bit-UNICODE, also praktisch alle international vorkommenden Schriftzeichen.

EBCDIC Dies ist das normale Datenformat von Großrechnern. Die Längen und Positionen in den Datensätzen sind in Bytes anzugeben. Alle externen Datenfelder sind möglich.

COLBIN Die Eingabedaten liegen im Dualkarten-Format (column binary) vor.
Die Positionen und Längen in den externen Datensätzen sind nicht in Bytes sondern in Spalten anzugeben. Das gilt sowohl für das Statement INPUT-EXT als auch für die Wertezuweisungen unter ADD-PROC, UPD-PROC und OUTPUT-EXT.
Es sind nur externe Datenfelder vom Typ A, D, F, G, I, K, L, M und U möglich.

QUANTUM Dies ist eine Speicherplatz sparende COLBIN-Variante.
Positionen und Längen in den externen Datensätzen sind in Spalten und nicht in Bytes anzugeben.
Es sind nur externe Datenfelder vom Typ A, D, K, L, M und U möglich.

BINARY1 Diese Dateien bestehen aus ein bis vier Bytes langen Dualzahlen.
Positionen und Längen in den externen Datensätzen sind in Bytes anzugeben.
Es sind nur externe Datenfelder vom Typ B, I, F und G möglich.

BINARY2 Diese Dateien bestehen aus zwei und vier Bytes langen Dualzahlen.
Positionen und Längen in den externen Datensätzen sind in Feldern anzugeben, jedes Feld zu zwei Bytes. Es sind nur externe Datenfelder vom Typ B, I, F und G möglich.

XLSX Diese Angabe liest Daten im XLSX-Format von Excel ein.
Es sind nur externe Datenfelder vom Typ AK, AS, DK, DS, MK, MS, UK und US möglich.
Die Angabe WORKSHEET wählt das gewünschte Arbeitsblatt der Datei aus.

< **RC =** n > Durch n = 1 ... 2 000 wird die Anzahl der Datensätze festgelegt, die für jeden statistischen Fall einzulesen sind. Diese Datensätze müssen für jeden statistischen Fälle immer in der gleichen Anzahl und Reihenfolge direkt hintereinander in der Datei liegen.

Fehlt die Angabe RC, so wird nur ein Datensatz pro Fall verarbeitet.

RC = | **A** / **D** / **F** / **G** / **P** | **(** a **..** e **)** / **(** a , c **)** | **O:** / **R:** | k, k, ...

AK / **DK**	**(**< 's' >**)** k, k, ...

AS / **DS**	**(** r **)** k, k, ...

Diese Angabe verlangt Satz-Kennzeichen in den Datensätzen. Die Sätze eines Falles dürfen in der Datei ungeordnet hintereinander liegen. Das Programm ordnet sie in der Reihenfolge an, in der ihre Satz-Kennzeichen hier aufgeführt sind. Die Angabe

```
RC=D(1,1)5,3
```

verlangt in der Positionen 1 der Datensätze das Satz-Kennzeichen 5 oder 3. Der Datensatz mit dem Kennzeichen 5 wird zum ersten Satz eines Falles und der mit dem Kennzeichen 3 zum zweiten Satz.

Es sind bis zu 2 000 Datensätze pro Fall möglich.
Fehlt die Angabe RC, so wird nur ein Datensatz pro Fall verarbeitet.

Datensätze, deren Satz-Kennzeichen hier nicht angegeben ist, erzeugen eine Fehlermeldung 504. Der falsche Satz wird unverarbeitet übergangen; der statistische Fall aber trotzdem verarbeitet.

Unvollständige Fälle, bei denen hier angegebene Satz-Kennzeichen fehlen, werden mit einer Fehlermeldung 505 beantwortet. Auch diese Fälle werden verarbeitet und die fehlenden Datensätze durch Leerzeichen ersetzt.

Die Angaben O / R / S / SR vor den einzelnen Satz-Kennzeichen erlauben abweichende Prüfungsregeln.

Für die Satz-Kennzeichen lässt sich einer der folgenden Feldtypen verwenden, so wie im Abschnitt *Externe Datenfelder* beschrieben:

A Die Satz-Kennzeichen sind alphanumerische Zeichen.

AK In CSV-Dateien: Die Satz-Kennzeichen bestehen aus alphanumerischen Zeichen mit Schlüsselwort.

AS In CSV-Dateien: Die Satz-Kennzeichen sind alphanumerische Zeichen.

D Die Satz-Kennzeichen sind numerische Zeichen.

DK In CSV-Dateien: Die Satz-Kennzeichen bestehen aus numerischen Zeichen mit Schlüsselwort.

DS In CSV-Dateien: Die Satz-Kennzeichen sind numerische Zeichen.

F Die Satz-Kennzeichen sind Dualzahlen mit Vorzeichen (Integer).

G Die Satz-Kennzeichen sind Dualzahlen ohne Vorzeichen (Unsigned Integer).

P Die Satz-Kennzeichen sind gepackte Dezimalzahlen. Negative Werte werden vor der Verarbeitung in positive Werte umgesetzt.

a erstes Byte 1 ... 1 000 000 der Satz-Kennzeichen in den externen Datensätzen;
erste Spalte 1 ... 500 000 im COLBIN-Format.

c Länge der Satz-Kennzeichen in den externen Datensätzen in Bytes;
bei COLBIN in Spalten. Die Längen sind begrenzt, abhängig vom Datenfeld:
im A-Feld 1 ... 127 Bytes oder Spalten
im D-Feld 1 ... 18 Bytes oder Spalten
im F- und G-Feld 1 ... 8 Bytes
im P-Feld 1 ... 10 Bytes

e letztes Byte 1 ... 1 000 000 der Satz-Kennzeichen in den externen Datensätzen;
letzte Spalte 1 ... 500 000 im COLBIN-Format.
Die Länge der Satz-Kennzeichen muss den unter c angegebenen Regeln entsprechen.

k Hier sind die zu verarbeitenden Satz-Kennzeichen eines jeden Falles anzugeben.
Bei D-, DS-, F-, G- und P-Feldern sind dies numerische Werte, bei A- und AS-Feldern sind es in Hochkommas ' eingeschlossene Zeichen oder Zeichenfolgen.

CNTA ordnet die Datensätze pro Fall in der hier angegebenen Reihenfolge an.
Durch die Angabe

```
RC=D(1..2)3,9,1
```

wird zum Beispiel erreicht, dass der Datensatz mit dem Kennzeichen 3 in ADD-PROC und UPD-PROC als Satz 1 angesprochen wird. Der Datensatz mit dem Kennzeichen 9 wird entsprechend als Satz 2 und der mit dem Kennzeichen 1 als Satz 3 angesprochen.

r Die Angabe r = 1 ... 32 767 ist die Reihenfolgenummer des Wertes im Datensatz *d* einer CSV-Datei. Leerzeichen vor und hinter dem Satz-Kennzeichen werden ignoriert. Nur bei AS-Feldern gehören die führenden Leerzeichen zum Satz-Kennzeichen. Für die Länge der Werte gelten die unter c angegebenen Regeln für A- und D-Felder.

's' Diese Angabe ist in Hochkommas einzuschließen. Sie legt das Schlüsselwort fest, an dem die *RC*-Werte in den Datensätzen zu erkennen sind. Fehlt diese Angabe, so wird als Schlüssel *RC* verwendet.
Diese Angabe ist in Hochkommas einzuschließen. Sie legt fest, welches Schlüsselwort sich in den Datensätzen vor den Satz-Kennzeichen und dem Gleichheitszeichen befindet.
Ist hier kein Schlüsselwort angegeben, so wird dafür *RC* verwendet.

Vor jedes Satz-Kennzeichen *k* kann noch eine der folgenden Angaben gestellt werden.
Damit wird die Prüfung der einzelnen Datensätze eines statistischen Falles verändert.

O: k...
R: k...
S: k...
SR

O Die Angabe O = *optional* führt dazu, dass alle dahinter aufgeführten Datensätze als wahlfrei aufgefasst werden. Sie müssen nicht unbedingt vorhanden sein. Fehlen sie, so werden sie ohne Fehlermeldung durch leere Datensätze ersetzt.

R Die Angabe R = *required* führt dazu, dass alle dahinter aufgeführten Datensätze vorhanden sein müssen. Fehlt einer davon, so wird die Fehlermeldung 505 ausgegeben und der statistische Fall unverarbeitet übergangen.

S Die Angabe S = *skip* fasst alle dahinter aufgeführten Datensätze als gültig auf, ohne sie zu verarbeiten. S: ... muss hinter den Angaben O: ... und R: ... stehen.
Die hinter S: ... angegebenen Datensätze werden bei der Nummerierung der Datensätze pro Schlüssel nicht mitgezählt.

SR Diese Angabe SR = *skip rest* ist nur am Ende aller Satz-Kennzeichen zulässig. Sie bewirkt, dass alle nicht aufgeführten Datensätze auch als gültig aufgefasst, aber nicht verarbeitet werden.

Zum Beispiel hat die Angabe

```
RC=D(1,2) 1, O:2,4, R:11,12, O:7, SR
```

folgende Wirkung:

- Die Datensätze enthalten den beiden ersten Positionen numerische Satz-Kennzeichen.
  Zu jedem statistischen Fall wird ein Datensatz mit dem Kennzeichen 1 erwartet und als Satz 1 verarbeitet. Fehlt dieser Satz, so wird er durch Leerzeichen ersetzt und die Fehlermeldung 505 ausgegeben, der statistische Fall aber trotzdem verarbeitet.
- Sodann können zwei Datensätze mit den Kennzeichen 2 und 4 vorhanden sein, sie werden als Satz 2 und Satz 3 verarbeitet. Wegen der Angabe O: davor dürfen sie auch fehlen. Die fehlenden Datensätze werden ohne Fehlermeldung durch Leerzeichen ersetzt.
- Die Angabe R: verlangt, dass zwei Datensätze mit den Kennzeichen 11 und 12 vorhanden sind, die zur Verarbeitung als Satz 4 und Satz 5 angesprochen werden. Fehlen diese, so wird die Fehlermeldung 505 ausgegeben und der statistische Fall nicht verarbeitet.
- Der Satz mit dem Kennzeichen 7 ist wieder wahlfrei, er wird als Satz 6 verarbeitet.
- Weitere Datensätze sind wegen der Angabe SR zulässig, werden aber nicht verarbeitet.

INTEGER = | **NORMAL** | **REVERSE** |

Mit dieser Angabe wird festgelegt, wie die Dualzahlen der externen Datenfelder vom Typ F, G und I in der Eingabedatei gespeichert sind.
Fehlt die Angabe INTEGER, so wird INTEGER=NORMAL angenommen.

NORMAL Mit dieser Angabe werden die höherwertigen Bits in den linken Bytes erwartet.

REVERSE Mit dieser Angabe werden die höherwertigen Bits in den rechten Bytes erwartet.
Dies ist das Format der Windows-PCs.

SEP = | 's' | **TAB** | **NUMBER** | < , 't' >

Die Angabe SEP legt fest, dass die Datei im CSV-Format zu verarbeiten ist.
In CSV-Dateien besitzen die einzelnen Datenfelder keine feste Position und Länge, sondern sind durch ein Separatorzeichen (meistens Komma, Semikolon oder Tabulatorzeichen) voneinander getrennt. Der Zugriff auf ein Feld erfolgt über die Reihenfolgenummer 1 ... innerhalb des Datensatzes. Die Länge der Felder ist variabel und vom Datenwert abhängig.
Die CSV-Dateien können zum Beispiel zur Ein- und Ausgabe von externen Dateien an Programme wie Microsoft EXCEL verwendet werden. Sie sind in den Dateiformaten ASCII, ANSI und EBCDIC möglich, nicht jedoch im COLBIN- und QUANTUM-Format.

'S' Hier ist ein beliebiges Zeichen in Hochkommas anzugeben, das als Separatorzeichen zwischen den Datenfeldern dient. Die Zeichen ' und # sind doppelt einzugeben.

TAB Die Angabe TAB macht das Tabulatorzeichen zum Separatorzeichen.

NUMBER Die Angabe NUMBER macht das Nummernzeichen # zum Separatorzeichen.

't' Hier ist ein beliebiges Zeichen in Hochkommas möglich, das als Texterkennungszeichen in alphanumerischen Werten dient (meist " oder '). Dieses Zeichen kann am Anfang und Ende eines Textwertes für AK- und AS-Felder stehen und erlaubt, das Separatorzeichen auch innerhalb eines alphanumerischen Wertes zu verwenden. Die Zeichen ' und # sind doppelt anzugeben.

KEYLINE Diese Angabe ist nur zusammen mit SEP möglich. Sie verlangt, dass der erste Datensatz Schlüssel für die Wertezuweisungen durch AK-, DK-, MK- und UK-Felder enthält. Die Datenwerte der Felder müssen in der gleichen Spalte wie ihr Schlüssel stehen.
Erscheint ein Schlüssel mehrfach in einer Schlüsselzeile, so wird die erste Spalte mit diesem Schlüssel verarbeitet. Eine Datei zum Statement

```
INPUT-EXT SEP=';' KEYLINE ID=DK(fall) ...
```

könnte folgendermaßen beginnen:

```
Fall;Geschlecht;Alter
 001;         2;   35
 002;         1;   47
```

Die Wertezuweisung

```
-N10=DK1(Alter)
```

stellt hier die Werte der dritten Spalte in die Variable N10.

Werden durch die Angabe RC mehrere Datensätze pro Fall angefordert, so gilt die Schlüsselzeile für alle Datensätze. Die Datei zum Statement

```
INPUT-EXT SEP=';' KEYLINE ID=DK(fall) RC=DK(satz)1,2...
```

könnte folgendermaßen beginnen:

```
Fall; Satz; Marke; bekannt
 001;    1;    17;       1
 001;    2;   112;       2
 002;    1;    79;       2
```

KEYLINES Wie KEYLINE erwartet diese Angabe Schlüssel in den ersten Zeilen der Datei.
Sie verlangt jedoch für jede Satzart eine eigene Schlüsselzeile. Die Datei zum Statement

```
INPUT-EXT SEP'=';' KEYLINES ID=DK(fall) RC=DK(satz)1,2...
```

könnte folgendermaßen beginnen:

```
Fall; Satz; Geschlecht; Alter
Fall; Satz;      Marke; bekannt
 001;   1;          2;      35
 001;   2;         17;       1
 002;   1;          1;      47
 002;   2;        112;       2
```

SKIP = n

Die Angabe SKIP übergeht die ersten n = 1 ... 255 Datensätze der Datei, ohne sie zu verarbeiten. Damit ist es zum Beispiel möglich, Dateien zu verarbeiten, die mit beschreibenden Textsätzen beginnen.

SIZE = n < ,**EOL** >

Diese Angabe legt die Länge der Datensätze fest. Fehlt sie, so wird SIZE=80 angenommen.

n Satzlänge 1 ... 10 000 000. in Bytes; bei COLBIN-Dateien in Spalten.

EOL Diese Angabe ist nur bei ANSI-, ASCII-, EBCDIC- und QUANTUM-Dateien zulässig.
Bei QUANTUM wird EOL automatisch angenommen und kann daher fehlen.
EOL besagt, dass jeder Datensatz durch ein Zeilenende-Zeichen beendet wird (CR=10 oder LF=13 oder CR und LF zusammen).
Die Satzlänge *n* gibt dann die maximal zulässige Länge pro Datensatz an.
Kürzere Datensätze werden vor der Verarbeitung eines statistischen Falles rechts durch Leerstellen aufgefüllt. Längere Datensätze führen zu einer Fehlermeldung, und der über *n* hinaus reichende Teil wird nicht verarbeitet.
Die Zeilenende-Zeichen sind bei EOL in der Länge *n* nicht enthalten.
Fehlt die Angabe EOL, so werden Zeilenende-Zeichen wie normale Datenbytes behandelt.

DCHAR = ' x **'**

x ist ein beliebiges Zeichen, das in den Wertezuweisungen aus D-Feldern hinter ADD-PROC und UPD-PROC als Dezimalzeichen gelten soll. Fehlt diese Angabe, so werden Punkt und Komma als Dezimalzeichen verwendet.

TCHAR = ' x **'**

x ist ein beliebiges Zeichen, das in den Wertezuweisungen aus D-Feldern hinter ADD-PROC und UPD-PROC als Tausenderzeichen gelten soll. Fehlt diese Angabe, so werden keine Tausenderzeichen erkannt.

WORKSHEET = ' name **'**

Diese Angabe wählt in XLSX-Dateien das gewünschte Arbeitsblatt aus. Fehlt sie, so wird das erste Arbeitsblatt der Datei eingelesen.

Beispiel: Einfache ASCII-Datei

```
INPUT-EXT    FNAME = STUDIE12
             ASCII
             SIZE=80,EOL

ADD-PROC
–N10=D1(8 ... 10)
–PRT  N10
```

Das Statement INPUT-EXT liest eine externe Eingabedatei mit dem Namen STUDIE12. Die Angabe ASCII bestimmt das Dateiformat: Textzeichen im ASCII-Code. SIZE=80,EOL verlangt zu jedem statistischen Fall einen maximal 80 Byte langen Datensatz, wegen EOL mit einem Zeilenende-Zeichen dahinter. Kürzere Datensätze werden vor der Verarbeitung mit Leerzeichen aufgefüllt.

Es ist keine Angabe ID ... vorhanden; daher wird keine Identifikation der statistischen Fälle aus der Datei übernommen. Stattdessen richtet CNTA eine künstliche Identifikation ID ein mit der Reihenfolgenummer der eingelesenen Fälle.

Hinter ADD-PROC erhält die numerische Variablen N10 zu jedem statistischen Fall einen Wert aus dem D-Feld der eingelesenen Datensätze. Dieser wird aus dem ersten und einzigen Datensatz pro Fall aus den Bytes 8 bis 10 entnommen.

Das PRT-Statement stellt danach die eingelesenen Werte der Variablen N10 für jeden der statistischen Fall in das Zählprotokoll, das folgendermaßen beginnen könnte:

```
FALL-ID: 1                                   Stmt: 7
  N10=31

FALL-ID: 2                                   Stmt: 7
  N10=35

FALL-ID: 3                                   Stmt: 7
  N10=36

FALL-ID: 4                                   Stmt: 7
  N10=45

FALL-ID: 5                                   Stmt: 7
  N10=53
```

Beispiel: Statistische Fälle mit mehreren Datensätzen

```
INPUT-EXT   FNAME=STUDIE12
            ASCII
            SIZE=80,EOL
            RC=3
ADD-PROC
-N10=D2(8...10)
```

Es wird wieder eine ASCII-Datei eingelesen mit Datensätzen von 80 Byte Länge und Zeilenende-Zeichen.

Die Angabe RC=3 verlangt zu jedem statistischen Fall drei direkt hintereinander liegende Datensätze.
Wichtig ist dabei, dass die Datensätze der einzelnen Fälle stets in der gleichen Reihenfolge hintereinander liegen.

Hinter ADD-PROC werden der numerischen Variablen N10 aus dem jeweils zweiten Datensatz eines jeden Falles Werte aus den Positionen 8 bis 10 zugewiesen.

```
INPUT-EXT     FNAME=STUDIE12
              ASCII
              SIZE=80,EOL
              ID=(1…4)
              RC=D(5...6)2...4
ADD-PROC
-N10=D2(8...10)
```

Diese Statements lesen wieder eine 80-stellige ASCII-Datei ein.

Die Angabe ID=(1…4) verlangt in allen Datensätzen eines Falles in den Positionen 1 bis 4 die gleiche numerische Fall-Identifikation. Damit ist eine Prüfung der Daten auf korrekte Anordnung der Datensätze zu den einzelnen Fällen möglich. Außerdem dürfen leere Datensätze fehlen. Sie werden bei der Verarbeitung durch Leerzeichen ersetzt.Dazu erscheint jedoch eine Warnung im Zählprotokoll

Die Angabe RC=D(5…6)2…4 verlangt in den Positionen 5 bis 6 der Datensätze numerische Satzarten mit den Werten 2 bis 4. Damit dürfen die Datensätze eines Falles in beliebier Reihenfolge vorliegen. Sie werden vor der Verarbeitung sortiert: Satzart 2 wird zum ersten Datensatz, Satzart 3 zum zweiten und Satzart 4 zum dritten Satz des Falles.

Eine dazu passende Datei könnte aus folgenden Datensätzen bestehen:

```
0002 3 35
0002 2
0002 4
0011 2
0011 4
0011 3 31
0223 3 36
```

Daraus entwickelt das Programm drei statistische Fälle mit den Identifikationen 2, 11 und 223. Da im Fall 223 nur die Satzart 3 vorliegt, werden die Satzarten 2 und 4 durch Leerzeilen ersetzt. Nach Sortierung der Datensätze entnimmt die Wertezuweisung -N10=D2(8…10) Werte aus dem jeweils zweiten Satz jedes Falles, hier also aus Satzart 3. Entsprechend erhält die Variable N10 nacheinander die Werte 35, 31 und 36.

Beispiel: Sortieren einer externen Datei

```
INPUT-EXT  SORT
           FNAME=TEST1.EXT
           COLBIN
           SIZE=80
           ID=D(1..5)
           RC=D(6..7)1...14
OUTPUT-EXT   FNAME=TEST2.EXT
             COPY
```

In diesem Beispiel wird eine externe Datei eingelesen, nach der Fall-Identifikation ID aufsteigend sortiert und danach in der neuen Reihenfolge auf eine zweite externe Datei ausgegeben.

Die Sortierung wird durch die Angabe SORT im Statement INPUT-EXT veranlaßt.
Die Eingabedatei selbst wird durch die Sortierung nicht verändert:

Die Angabe COLBIN legt fest, dass die Eingabedaten im COLBIN-Format vorliegen, also in Form von Lochkartenspalten (siehe Abschnitt *Externe Dateien*).

SIZE=80 verlangt, dass die Datensätze jeweils 80 Spalten lang sind. Da im COLBIN-Format jede Spalte zwei Bytes belegt, sind die Eingabesätze 160 Bytes lang.

Die Angabe ID=D(1..5) verlangt in den Spalten 1 bis 5 eines jeden Eingabesatzes Fall-Identifikationen, nach denen die Eingabedaten zu sortieren sind.

Die Angabe RC besagt, dass jeder statistische Fall aus mehreren hintereinander liegenden Datensätzen besteht. Alle Datensätze eines Falles müssen die gleiche Fall-Identifikation in den Spalten 1 bis 5 besitzen. Die einzelnen Datensätze sind durch die Zahlen 01 bis 14 in den Spalten 6 und 7 gekennzeichnet. Fehlt einer dieser Datensätze zu einer Fall-Identifikation, so gibt CNTA eine Fehlermeldung aus.

Im Statement OUTPUT-EXT bewirkt die Angabe FNAME, dass die umsortierte Datei unter dem Namen TEST2.EXT gespeichert wird.

COPY überträgt die einzelnen statistischen Fälle nach der Sortierung unverändert in die Ausgabedatei. Unabhängig von SORT werden die Datensätze eines jeden statistischen Falles in der Reihenfolge ausgegeben, in der sie hinter RC in INPUT-EXT angegeben sind.
Das Dateiformat COLBIN wird nicht verändert.

INPUT-INT

Durch das Statement INPUT-INT wird eine Datei mit statistischen Fällen zur Verarbeitung angefordert, die im CNT-internen Datenformat vorliegt.

Internes Datenformat bedeutet, dass die statistischen Fälle in einem früheren CNTA-Lauf mit dem Statement OUTPUT-INT in dieses Format überführt wurden. Eine solche interne Datei enthält die Daten der statistischen Fälle in Variablen gespeichert. Weiter enthält sie die Definitionsangaben der Variablen mit ihren Texten sowie Textelemente, Makros und Variablengruppen.

Mit dem Statement CODEBOOK lässt sich eine Übersicht über den Inhalt einer internen Datei gewinnen, einschließlich einer Kurzauswertung der Variablenwerte.

Interne Dateien sind stets nach aufsteigender Fall-Identifikation sortiert. Die statistischen Fälle werden immer in dieser Reihenfolge eingelesen. Weitere Angaben zu den internen Dateien befinden sich im Abschnitt Interne Dateien.

In jedem Verarbeitungslauf kann nur ein Statement INPUT-INT angegeben werden. Für zusätzliche interne Dateien steht das Statement INPUT-SEC zur Verfügung.

Folgende Angaben sind zum Statement INPUT-INT möglich:

FNAME = dateiname

Hier ist der Name der Eingabedatei anzugeben. Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
FNAME=C:\DATEN\STUDIE27.INT
FNAME='STUDIE27 INTERN'
FNAME=\\SERVER\DATEN\STUDIE27.INT
```

Enthält *dateiname* keine Pfadangabe, so erwartet CNTA die Datei im Verzeichnis der Statementdatei. CNTW sucht sie zunächst im Verzeichnis der REP-Datei und danach im Verzeichnis der Statementdatei.
Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\STUDIE27.INT
```

NAME = 'aaa'

Diese Angabe hat lediglich Kontrollfunktion: Sie bewirkt, dass nur ein interner Datenbestand mit dem gespeicherten Kontrollnamen *aaa* zur Verarbeitung zugelassen wird. Der Kontrollname 'aaa' wird bei der Erstellung der internen Datei durch das Statement OUTPUT-INT vergeben und in der internen Datei gespeichert. Als Kontrollname kann zum Beispiel eine Studienbezeichnung oder ein Projektname verwendet werden.
Fehlt die Angabe NAME in INPUT-INT, so wird die unter FNAME angeforderte Datei ungeprüft verarbeitet.

IDSHORT

Mit der Angabe IDSHORT lassen sich die Daten eines Fall aus INPUT-INT an alle Fälle gleicher Fall-Identifikation aus INPUT-EXT übertragen.

Ohne die Angabe IDSHORT müssen beide Dateien die gleiche Anzahl von ID-Stufen besitzen. Jeder Fall aus INPUT-INT wird höchstens mit einem passenden Fall aus INPUT-EXT zusammengeführt und über UPD-PROC verarbeitet. Weitere dazu passende Fälle aus INPUT-EXT werden ohne Daten aus INPUT-INT über ADD-PROC verarbeitet.
Fälle ohne Gegenstück in der jeweils anderen Datei werden getrennt verarbeitet, Fälle aus INPUT-EXT über ADD-PROC, Fälle aus INPUT-INT nur über MOD-PROC.

Mit der Angabe IDSHORT darf die INPUT-INT-Datei auch weniger ID-Stufen als INPUT-EXT besitzen. Jeder Fall aus INPUT-INT wird mit allen Fällen aus INPUT-EXT mit passender Identifikation zusammengeführt und über UPD-PROC verarbeitet. Fälle aus INPUT-INT ohne Gegenstück in INPUT-EXT werden unverarbeitet übergangen und Fälle aus INPUT-EXT ohne Gegenstück in INPUT-INT über ADD-PROC verarbeitet.

Zum Beispiel sei in INPUT-INT nur die Fall-Identifikation ID vorhanden und IDSHORT angegeben. Außerdem soll das Statement INPUT-EXT vorhanden sein und sowohl ID als auch ID1 enthalten. In den beiden Dateien sollen nun statistische Fälle mit den folgenden Fall-Identifikationen vorliegen:

INPUT-EXT INPUT-INT

```
100.1 100
100.2 102
105.0 106
106.2
```

Dann wird zunächst der Fall 100.1 aus INPUT-EXT zusammen mit dem Fall 100 aus INPUT-INT verarbeitet. Für die Wertezuweisungen sind die Statements aus UPD-PROC zuständig.

Danach wird der Fall 100.2 wieder zusammen mit dem Fall 100 aus INPUT-INT verarbeitet, ebenfalls mit den Statements aus UPD-PROC.

Anschließend wird der Fall 102 aus INPUT-INT unverarbeitet übergangen, da in INPUT-EXT kein passendes Gegenstück vorhanden ist.

Nun ist der Fall 105.0 aus INPUT-EXT an der Reihe. Zu ihm existiert auf INPUT-INT kein Partner: Er wird alleine verarbeitet und zwar über die Statements aus ADD-PROC.

Schließlich wird der Fall 106.2 in UPD-PROC verarbeitet, wieder zusammen mit dem Fall 106 aus INPUT-INT.

Der Abschnitt *Einlesen der statistischen Fälle* enthält weitere Angaben über die gemeinsame Verarbeitung mehrerer Dateien.

Beispiel: Interne Datei auswerten

Die folgenden Statements zeigen Auswertungen aus einer internen Datei:

```
INPUT-INT     FNAME=1009.INT
CODEBOOK
XTAB    COL=TOTAL;C10
        ROW=TOTAL;C20
```

Das Ergebnis von CODEBOOK könnte folgendermassen aussehen:

CODEBOOK

Anzahl Fälle 4594

C10:2			
1	Männer	992	21,6%
2	Frauen	3602	78,4%
Z1	eine Angabe	4594	100,0%
N20:3	Alter		
	Fälle mit Angaben	4591	99,9%
	Fälle ohne Angaben	3	0,1%
	Minimum		19
	Maximum		89
	Mittelwert		35,7
T70:30	Land des Hauptwohnsitzes		
	Fälle mit Angaben	111	2,4%
	Fälle ohne Angaben	4483	97,6%
	kleinste Länge		2
	größte Länge		28
(70)	Frage 12: Wenn Sie im Ausland leben: Bitte geben Sie den Namen des Landes ein		

Hier ist C10 eine kategoriale Variable mit zwei Ausprägungen und N20 eine numerische Variable, die das Alter der Testpersonen als Zahlenwert enthält. Die Textvariable T70 enthält eine Länderangabe als unverschlüsselten Text und (70) ist ein Textelement, das sich zum Beispiel zur Beschriftung von Tabellen verwenden lässt.

Aus dem Statement XTAB könnte folgende Tabelle entstehen:

	TOTAL	Männer	Frauen
TOTAL	4594	992	3602
Alter			
Von 18 bis 20 Jahre	3	1	2
Von 21 bis 25 Jahre	238	76	162
Von 26 bis 30 Jahre	1101	329	772
Von 31 bis 35 Jahre	1203	248	955
Von 36 bis 40 Jahre	902	152	750
Von 41 bis 50 Jahre	926	147	779
Von 51 bis 60 Jahre	200	35	165
Über 60 Jahre	18	1	17

INPUT-SEC

Die Statements INPUT-SEC können zusätzlich zu INPUT-INT bis zu 30 weitere interne Dateien einlesen.
- Es lassen sich zusätzliche statistische Fälle einfügen.
- Bestehende statistische Fälle können neue Werte erhalten, die in bestehende oder neu definierte Variable übernommen werden.
- Variablendefinitionen samt Texten lassen sich aus den zusätzlichen Dateien übernehmen.

Dabei werden die Datensätze anhand ihrer Fall-Identifikationen zusammengeführt.
Andere Wege, Daten aus verschiedenen Dateien zusammenzuführen, bietet das Statement FUSION und die Wertezuweisung mit der FETCH-Funktion.

Folgende Angaben sind zu INPUT-SEC möglich:

FNAME = dateiname

Hier ist der Name der Eingabedatei anzugeben. Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
FNAME=C:\DATEN\STUDIE27.INT
FNAME='STUDIE27 INTERN'
FNAME=\\SERVER\DATEN\STUDIE27.INT
```

Enthält der Dateiname keine Pfadangabe, so erwartet CNTA die Datei im Verzeichnis der Statementdatei. CNTW sucht sie zunächst im Verzeichnis der REP-Datei und danach im Verzeichnis der Statementdatei.
Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\STUDIE27.INT
```

NAME = 'aaa'

Diese Angabe hat lediglich Kontrollfunktion: Sie bewirkt, dass nur ein interner Datenbestand mit dem gespeicherten Kontrollnamen *aaa* zur Verarbeitung zugelassen wird. Der Name *aaa* wird bei der Erstellung der internen Datei durch das Statement OUTPUT-INT vergeben und in der internen Datei gespeichert. Als Kontrollname kann zum Beispiel eine Studienbezeichnung oder ein Projektname verwendet werden.
Fehlt die Angabe NAME in INPUT-SEC, so wird die unter FNAME angeforderte Datei ungeprüft verarbeitet.

IDSHORT

Mit der Angabe IDSHORT lassen sich die Daten eines Falles aus INPUT-SEC an alle Fälle gleicher Fall-Identifikation aus INPUT-EXT und INPUT-INT übertragen.

Ohne die Angabe IDSHORT muss die INPUT-SEC-Datei die gleiche Anzahl von ID-Stufen besitzen wie die übrigen Eingabedateien. Dabei wird jedem Fall aus INPUT-SEC nur ein Fall aus INPUT-INT oder INPUT-EXT mit gleicher Identifikation zugewiesen. Weitere Fälle aus INPUT-INT oder INPUT-EXT mit passender Identifikation erhalten keine Daten aus INPUT-SEC.
Aus INPUT-SEC-Fällen ohne passendes Gegenstück werden neue statistische Fälle erstellt.

Mit der Angabe IDSHORT darf die INPUT-SEC-Datei auch weniger ID-Stufen als die übrigen Eingabedateien besitzen. Dabei werden die Daten eines Falles aus INPUT-SEC allen Fällen aus INPUT-INT und INPUT-EXT mit passender Identifikation zugewiesen.
Fälle aus INPUT-SEC ohne passendes Gegenstück in den übrigen Eingabedateien werden unverarbeitet übergangen.

NEWVARS= | **ALL** | **ALL(**n**)** | **MISSING** |

Diese Angaben sind nur wirksam, wenn keine Folgestatements vorliegen. Sie erzeugen zu den Variablen aus INPUT-SEC neue Variable mit den Variablen- und Merkmalstexten sowie den Variablenwerten aus INPUT-SEC.
NEWVARS=ALL übernimmt alle Variable aus INPUT-SEC in neu definierte Variable. Dazu werden die kleinsten noch freien Variablennummern verwendet.
NEWVARS=ALL(n) vergibt stattdessen die kleinsten noch freien Variablennummern größer oder gleich der Nummer *n*.
NEWVARS=MISSING überträgt zunächst wie bei der normalen Zuordnung die Variablenwerte aus INPUT-SEC in gleichnamige Zielvariable. Zu Variablen aus INPUT-SEC ohne gleichnamiges Gegenstück wird dagegen eine neue Variable mit dem gleichen Variablennamen angelegt.

VTEXT = 'text'

Diese Angabe stellt die Zeichenfolge text vor den Variablentext aller aus der INPUT-SEC-Datei stammenden Variablen, die neu zu definieren sind.
Zum Beispiel stellt

```
VTEXT='Neue Variable'
```

vor alle aus INPUT-SEC übernommenen Variablentexte die Angabe *Neue Variable:* um die Herkunft der Variablen deutlich zu machen.

ADD | **ADDP** |

Diese Angaben erzeugen zu jedem Fall aus INPUT-SEC einen neuen Fall, auch wenn in INPUT-INT Daten gleicher Fall-Identifikation vorhanden sind.
Variable aus INPUT-INT oder DEFS, die keine Werte aus INPUT-SEC erhalten, bleiben leer (Z0).

Die Angabe ADDP stellt außerdem für jeden Fall aus INPUT-SEC, zu dem auch Daten aus INPUT-INT mit gleicher Fall-Identifikation vorliegen, eine Fehlermeldung 515 in das Zählprotokoll.

Zusammenführung der statistischen Fälle ohne ADD, ADDP oder IDSHORT

Enthält das Statement INPUT-SEC keine der Angaben ADD, ADDP oder IDSHORT, so werden die Daten aus INPUT-INT und INPUT-SEC anhand ihrer Fall-Identifikationen zusammengeführt.

Gibt es zu einem Fall aus INPUT-SEC in der Datei INPUT-INT einen Fall mit gleicher Identifikation, so werden deren Daten gemeinsam verarbeitet. Die Variablenwerte aus INPUT-SEC überschreiben die Werte aus INPUT-INT. Enthält INPUT-INT keinen zu INPUT-SEC passenden Fall, so wird ein neuer Fall mit den Werten aus INPUT-SEC angelegt.

Gibt es in INPUT-INT mehrere zu INPUT-SEC passende Fälle, so erhält nur der erste davon Werte aus INPUT-SEC.

Liegen zu einer Fall-Identifikation Daten aus mehreren INPUT-SEC-Dateien vor, so werden diese nacheinander in der Reihenfolge der INPUT-SEC-Statements übernommen. Variablenwerte aus einem späteren INPUT-SEC überschreiben die Werte aus den davor liegenden.

Die Daten aus INPUT-EXT werden erst nach Verarbeitung der INPUT-SEC-Statements eingelesen. Die Wertezuweisungen aus ADD-PROC, UPD-PROC und MOD-PROC werden ebenfalls erst danach ausgeführt.

Es ist möglich, ohne das Statement INPUT-INT nur mit INPUT-SEC-Statements zu arbeiten.

Zusammenführung der statistischen Fälle mit ADD und ADDP

Die Angaben ADD oder ADDP im Statement INPUT-SEC erstellen zu jedem Fall aus INPUT-SEC einen neuen Fall, auch wenn INPUT-INT oder INPUT-EXT Fälle mit gleicher Fall-Identifikation enthalten.

Zusammenführung der statistischen Fälle mit IDSHORT

Die Angabe IDSHORT im Statement INPUT-SEC setzt zwei oder mehr ID-Stufen in INPUT-INT voraus. Damit wird jeder Fall aus INPUT-SEC allen Fällen aus INPUT-INT zugewiesen, die in den oberen ID-Stufen mit INPUT-SEC übereinstimmen.

Zum Beispiel sei in INPUT-SEC nur ID mit IDSHORT vorhanden und in INPUT-INT sowohl ID als auch ID1 angegeben. In den Dateien sollen statistische Fälle mit folgenden Fall-Identifikationen vorliegen:

INPUT-EXT	INPUT-SEC
100.1	100
100.2	102
105.0	106
106.2	

Die beiden Fälle 100.1 und 100.2 aus INPUT-INT werden durch den Fall 100 aus INPUT-SEC ergänzt.
Der Fall 102 aus INPUT-SEC wird nicht verarbeitet, da in INPUT-INT kein passendes Gegenstück vorliegt.
Der Fall 105.0 findet kein Gegenstück in INPUT-SEC. Er wird daher nicht verändert.
Schließlich wird der Fall 106.2 mit dem Fall 106 aus INPUT-SEC zusammengeführt.

Zuordnung der Variablen ohne NEWVARS und Folgestatements

Unter diesen Voraussetzungen werden die Datenwerte aus INPUT-SEC automatisch in gleichnamige Variable aus DEFS oder INPUT-INT übertragen. Variablenwerte aus INPUT-SEC ohne gleichnamiges Gegenstück werden nicht übernommen.
Die weiter unten beschriebenen Folgestatements verhindern diese automatische Datenübernahme. Sie können die Variablen aus INPUT-SEC beliebigen anderen Variablen aus INPUT-INT oder DEFS zuordnen.

Besitzt die Zielvariable aus INPUT-INT oder DEFS mehr Merkmale, Ziffern oder Zeichen als die Variable aus INPUT-SEC, so werden die Eingabewerte mit Nullen oder Leerzeichen aufgefüllt. Ist die Zielvariable dagegen kürzer, so werden bei C-Variablen die hohen Merkmalsnummern abgeschnitten und bei T-Variablen die Texte rechts gekürzt, beides ohne Fehlermeldung. Numerische Werte werden bei abweichenden Nachkommastellen gerundet oder erweitert. Passen sie nicht in die Zielvariable, so wird eine Fehlermeldung ausgegeben und kein Wert übertragen.

Zuordnung der Variablen mit NEWVARS

Die Angabe NEWVARS im Statement INPUT-SEC ist nur wirksam, wenn keine Folgestatements zur Auswahl von Variablen vorliegen. Sie erzeugt zu den Variablen aus INPUT-SEC neue Variable mit den Variablentexten und Datenwerten aus INPUT-SEC.
NEWVARS=ALL übernimmt alle Variable aus INPUT-SEC in neu definierte Variable.
NEWVARS=MISSING überträgt zunächst wie bei der normalen Zuordnung die Variablenwerte aus INPUT-SEC in gleichnamige Zielvariable. Zu Variablen aus INPUT-SEC ohne gleichnamiges Gegenstück wird dagegen eine neue Variable mit dem gleichen Variablennamen angelegt.

Zuordnung der Variablen durch Folgestatements

Liegen direkt hinter INPUT-SEC Folgestatements mit einem Strich – am Zeilenanfang, so werden nur die dort angegebenen Variablen übernommen.

Zum Beispiel überträgt das Folgestatement

```
-C10=C15
```

die Werte der Variablen C15 aus INPUT-SEC in die Variable C10 aus INPUT-INT oder DEFS.
Ist die Variable C10 in INPUT-INT oder DEFS nicht vorhanden, so wird sie durch das Folgestatement neu definiert. Die neue Variable C10 erhält dabei die Texte und Werte der Variablen C5 aus INPUT-SEC.

Folgestatements für INPUT-SEC und FUSION

Diese Statements müssen direkt hinter INPUT-SEC oder FUSION liegen und mit einem Strich – am Zeilenanfang beginnen. Sie können die Variablen aus INPUT-SEC oder FUSION den Variablen aus INPUT-INT und DEFS abweichend von den automatischen Regeln zuordnen. Folgende Angaben sind möglich:

–Ca **= C**m
–Ca **= N**m
–Ca **= ID**i

Die Bedingungsvariable Ca aus DEFS oder INPUT-INT erhält einen neuen Wert aus der Bedingungsvariablen Cm, der numerischen Variablen Nm sowie der numerischen Fall-Identifikation IDi, jeweils aus INPUT-SEC oder FUSION.

Besitzt die Zielvariable Ca im Falle Ca=Cm weniger Merkmale als die abgebende Variable Cm, so gehen die überschüssigen Merkmale ohne Fehlermeldung verloren. Besitzt Ca mehr Merkmale, so werden diese auf 0 gesetzt.

Im Fall Ca=Nm erhält das Merkmal, dessen Nummer mit dem Wert von Nm übereinstimmt den Wert 1 = *erfüllt* gesetzt, die übrigen Merkmale den Wert 0 = *nicht erfüllt*. Analog wird im Fall Ca=IDi verfahren.

Ist die Zielvariable Ca nicht definiert, so wird sie durch das Folgestatement mit den Variablen- und Merkmalstexten von Cm neu definiert. Zum Beispiel übertragen die Folgestatements

```
-C1=C17
-C1[2]=C5[1]
```

den Wert der Variablen C17 in die Variable C1 und die von C5[1] in C1[2].

Sind beide Variable in dem Statement

```
-C1=C5
```

mit Index definiert, so sorgt dieses Folgestatement für die Zuweisung aller Indexwerte:
Die Variable C5[1] wird der Variablen C1[1] zugewiesen, die Variable C5[2] der Variablen C1[2] usw. Diese Übertragung wird beim höchsten gemeinsamen Index der beiden Variablen beendet.

–Na **= N**m
–Na **= T**m
–Na **= ID**i

Die numerische Variable Na aus DEFS oder INPUT-INT erhält einen neuen Wert aus der numerischen Variablen Nm, der Textvariablen Tm sowie der numerischen oder alphanumerischen Fall-Identifikation IDi, jeweils aus INPUT-SEC oder FUSION.

Die Zielvariable Na darf im Fall Na=Nm mehr oder weniger Ziffern oder Nachkommastellen besitzen als die abgebende Variable. Die Werte werden bei der Übertragung passend ergänzt oder gerundet.

Ist der zu übertragende Wert nicht numerisch oder zu groß für die Zielvariable Na, so findet keine Übertragung statt. Es wird eine Fehlermeldung ausgegeben und die Variable Na behält ihren alten Wert. Ist die Zielvariable zu klein für einen Variablenwert, so findet keine Übertragung statt. Es erscheint eine Fehlermeldung im Protokoll und die Variable Na behält ihren alten Wert.
Ist die Zielvariable Na nicht definiert, so wird sie durch das Folgestatement mit den Angaben der Variablen Nm neu definiert, einschließlich Variablentext.

–Ta **= N**m
–Ta **= T**m
–Ta **= ID**i

Die Textvariable Ta aus DEFS oder INPUT-INT erhält einen neuen Wert aus der numerischen Variablen Nm, der Textvariablen Tm sowie der numerischen oder alphanumerischen Fall-Identifikation IDi jeweils aus INPUT-SEC oder FUSION. Erlaubt die Zielvariable Ta weniger Zeichen als die abgebende Variable liefert, so wird der Eingabewert rechts ohne Fehlermeldung abgeschnitten. Erlaubt Ta dagegen mehr Zeichen, so wird der Eingabewert rechts mit Leerzeichen aufgefüllt.

Ist die Zielvariable Ta nicht definiert, so wird sie durch das Folgestatement mit der Länge und dem Variablentext von Tm neu definiert. Ansonsten gelten die gleichen Regeln wie bei C-Variablen.

–Ca **...** b **= C**m **...** n

Hiermit werden alle zwischen Cm und Cn definierten Variable aus INPUT-SEC oder FUSION den Variablen Ca bis Cb aus DEFS oder INPUT-INT zugeordnet.

Zum Beispiel überträgt

```
-C1...5=C11...15
```

die Werte der Variablen C11 bis C15 in die Variablen C1 bis C5.
Die beiden Variablen C1 und C5 müssen entweder beide definiert oder beide undefiniert sein.
Sind sie definiert, so werden nur die zwischen C1 und C5 liegenden definierten Variablen mit neuen Werten versehen.
Sind sie undefiniert, so werden alle zwischen C1 und C5 liegenden nicht definierten Variablen neu definiert, passend zu den Variablen C11 bis C15. Nur diese neuen Variablen erhalten Werte zugewiesen.

Ist eine zwischen C11 und C15 liegende Variable mit Index [...] definiert, so muss das Gegenstück zwischen C1 und C5 ebenfalls mit Index definiert sein. Umgekehrt darf eine ohne Index definierte Variable zwischen C1 und C5 nicht einer zwischen C1 und C5 liegenden mit Index zugeordnet werden.

Das Folgestatement

```
-C1[2]...4=C5...7
```

überträgt die Werte der Eingabevariablen C5 bis C7 in die Variablen C1[2] bis C1[4]. Ganz anlog füllt

```
-C5..7=C1[2]..4
```

die Werte der Variablen C1[2] bis C1[4] in die Variablen C5 bis C7.

–Na **...** b **= N**m **...** n

Hiermit werden alle zwischen Nm und Nn definierten Variable aus INPUT-SEC oder FUSION den Variablen Na bis Nb aus DEFS oder INPUT-INT zugeordnet. Es gelten die gleichen Regeln wie bei den C-Variablen.

–Ta **...** b **= T**m **...** n

Hiermit werden alle zwischen Tm und Tn definierten Variable aus INPUT-SEC oder FUSION den Variablen Ta bis Tb aus DEFS oder INPUT-INT zugeordnet. Es gelten die gleichen Regeln wie bei den C-Variablen.

–NEW **–NEW(**n**)**	**=**	**C**a **N**a **T**a **C**a **...** b **N**a **...** b **T**a **...** b

Diese Folgestatements legen zu der rechts vom Gleichheitszeichen angegebenen Variablen aus INPUT-SEC eine neue Variable an, mit den Texten und Werten der abgebenden Variablen.
Für NEW wird die kleinste noch freie Variablennummer verwendet und für NEW(n) mit einer Zahl *n* zwischen 0 und 99 999 die erste freie Nummer größer oder gleich *n*.

Beispiel: Einfache Zusammenführung

```
INPUT-INT  FNAME=studie112.000
DEFS
-C10:5
-N10:3
INPUT-SEC  FNAME=studie112.001
```

Im ersten Statement wird die interne Datei *studie112.000* angefordert. Sie soll die folgenden Variablen enthalten:
C1:10 N1:3 T1:12
Und dazu zwei statistische Fälle mit folgenden Variablenwerten und Fall-Identifikationen:

```
ID  C1   N1   T1
-------------------------
1   4    Z0   -
2   8    222  -
```

Die Folgestatements von DEFS definieren die zusätzlichen, nicht in INPUT-INT enthaltenen Variablen C10 und N10.

Das Statement INPUT-SEC sorgt für die Eingabe der sekundären internen Datei *studie112.001*. Sie soll die folgenden Variablen enthalten:
C1:12 C10:3 C12:10 N1:2 N10:4 T1:14
und dazu die beiden Fälle:

```
ID  C1      C10   C12   N1    N10   T1
-------------------------------------------------------------------------
1   1  12   2     3     82     312  Ford Escort
3   2       3     7     93    1012  Opel Kadett E
```

Es stehen daher die folgenden Variablen aus INPUT-INT und DEFS zur Verfügung:
C1:10 C10:5 N1:3 N10:3 T1:12
Da zum Statement INPUT-SEC keine Folgestatements vorliegen, werden die Variablen automatisch zugeordnet. Dadurch entstehen aus den Daten von INPUT-INT und INPUT-SEC die folgenden Fälle:

```
ID  C1    C10   N1    N10   T1
-----------------------------------------------------------
1   1     2     082   312   Ford Escort
2   8     -     222   Z0    -
3   2     3     093   Z0    Opel Kadett
```

Für den Fall 1 überschreiben und ergänzen die Daten aus INPUT-SEC die Werte aus INPUT-INT, da zu dieser Fall-Identifikation Daten aus beiden Dateien vorliegen. Zunächst überschreibt die Variable C1 aus INPUT-SEC die Variable C1 aus INPUT-INT: Das alte Merkmal 4 wird gelöscht und durch das Merkmal 1 ersetzt. Das Merkmal 12 aus INPUT-SEC geht verloren, da die neue Variable C1 aus INPUT-INT nur 10 Merkmale besitzt. Die Werten von C10 aus INPUT-SEC füllen die neue Variable C10.
Der Wert 82 der Variablen N1 aus INPUT-SEC wird in die jetzt 3-stellige Variable N1 aus INPUT-INT gestellt. Der Wert 312 der Variablen N10 aus INPUT-SEC wird in die neue Variable N10 gestellt. Der Wert Ford Escort der Textvariablen T1 wird durch Entfernen rechter Blanks von 14 auf 12 Stellen gekürzt. Die Variable C12 aus INPUT-SEC geht verloren, da sie weder in INPUT-INT noch in DEFS definiert wurde.

Zum Fall 2 liegen nur Daten aus INPUT-INT vor. Deshalb bleiben seine Werte unverändert; die neuen Variablen C10 und N10 erhalten keine Werte.

Zum Fall 3 schließlich existieren nur Daten aus INPUT-SEC und keine aus INPUT-INT. Hier ist zu beachten, dass der Wert 1012 der Variablen N10 nicht mehr in die neue dreistellige Variable N10 passt. Dafür wird eine Fehlermeldung ausgegeben und der Wert der neuen Variablen nicht verändert, sie behält den Grundstellungswert Z0. Der Wert Opel Kadett E der Textvariablen T1 muß von 14 Stellen auf 12 Stellen verkürzt werden. Dazu wird der Text rechts abgeschnitten, der Buchstabe E geht verloren.

Beispiel: Mit Folgestatements

```
INPUT-INT    FNAME=studie112.000
DEFS
-C8:5
-N8:3
INPUT-SEC   FNAME=studie112.001
-C2..8=C1..10
-N2..8=N1..10
```

Im Statement INPUT-INT wird die interne Datei studie112.000 angefordert. Sie soll die folgenden Variablen enthalten:

C2:10 N2:3 T2:12

und dazu zwei statistische Fälle:

ID	C2	N2	T2
1	4	Z0	-
2	8	222	-

Die Folgestatements von DEFS definieren die zusätzlichen, nicht in INPUT-INT enthaltenen Variablen C8 und N8.

Das Statement INPUT-SEC sorgt für die Eingabe der sekundären internen Datei studie112.001. Sie soll die folgenden Variablen enthalten:

C1:12 C10:3 C12:10 N1:2 N10:4 T1:14

und die beiden Fälle:

ID	C1	C10	C12	N1	N10	T1
1	1 12	2	3	82	0312	Ford Escort
3	2	3	7	93	1012	Opel Kadett E

Die Folgestatements von INPUT-SEC verhindern eine automatische Zuordnung der Variablen. Das erste dieser Folgestatements überträgt die Werte der Variablen C1 und C10 aus INPUT-SEC in die Variablen C2 und C8.
Analog überträgt das zweite Folgestatement die Werte der Variablen N1 und N10 aus INPUT-SEC in die Variablen N2 und N8.
Für die Variablen C12 und T1 stehen hinter INPUT-SEC keine Folgestatements; ihre Werte werden daher nicht verarbeitet.

Für die weitere Verarbeitung durch CNTA stehen danach die folgenden Variablen aus INPUT-INT und DEFS zur Verfügung:

C2:10 C8:5 N2:3 N8:3 T2:12

Aus den Daten von INPUT-INT und INPUT-SEC entstehen dazu die statistischen Fälle:

ID	C2	C8	N2	N8	T2
1	1	2	082	312	-
2	8	-	222	Z0	-
3	2	3	093	Z0	-

LANG ... LANG9

CNTA kann einer Variablen oder einem Textelement gleichzeitig mehrere Texte zuweisen, zum Beispiel in verschiedenen Sprachen oder als Kurz- und Langtexte.

Dazu wird mit den Statements LANG, LANG1 ... LANG9 die Sprache für die dahinter stehenden Statements festgelegt. Diese Sprachschlüssel LANG ... LANG9 können beliebig oft und an jeder Stelle eines Programms erscheinen. Es bleibt dem Anwender überlassen, welche Bedeutung er den einzelnen Sprachschlüsseln geben will.

Innerhalb der Folgestatements von DEFS, ADD-PROC usw. ist die Schreibweise –LANG mit einem Strich davor zu verwenden, außerhalb von Folgestatements ist dieser Strich wegzulassen.

So weisen die Statements

```
DEFS
-LANG
-C1:2  'Geschlecht' 1='männlich' 2='weiblich'
-N1:2  'Alter'
-TOTAL 'Gesamt'
-LANG1
-C1    'sex'        1='male'     2='female'
-N1    'age'
-TOTAL 'total'
```

den Variablen C1, N1 und dem Operator TOTAL unter dem Sprachschlüssel LANG deutsche Texte und unter dem Sprachschlüssel LANG1 englische Texte zu. Da die Statements LANG und LANG1 innerhalb der Folgestatements von DEFS erscheinen, müssen sie mit einem waagerechten Strich beginnen.

In den Auswertungen können mit den Statements LANG ... LANG9 Texte aus unterschiedlichen Sprachen angefordert werden. Nach der obigen Variablendefinition liefern die Statements

```
LANG
XTAB ROW=C1 COL=TOTAL
LANG1
XTAB ROW=C1 COL=TOTAL
```

zunächst eine Tabelle mit deutschen Texten und dahinter die gleiche Tabelle mit englischen Texten.

Bei der Ausgabe auf OUTPUT-INT behalten die Variablen und Textelemente alle unter verschiedenen Sprachschlüsseln vergebenen Texte.

Ist kein Statement LANG ... LANG9 angegeben, so wird mit dem Sprachschlüssel LANG gearbeitet.

MOD-PROC

Das Statement MOD-PROC legt gemeinsame Regeln zur Prüfung und Bearbeitung aller statistischen Fälle fest. Im Gegensatz dazu bearbeitet das Statement ADD-PROC nur 'neue' statistische Fälle und UPD-PROC nur 'alte' Fälle. MOD-PROC wird erst nach der Verarbeitung der Statements ADD-PROC und UPD-PROC wirksam. Das gilt auch dann, wenn ADD-PROC und UPD-PROC hinter MOD-PROC eingegeben werden.

In einem Verarbeitungslauf darf das Statement MOD-PROC nur einmal vorkommen.
Es besitzt selbst keine weiteren Angaben, dafür aber beliebig viele Folgestatements.
Diese müssen direkt hinter MOD-PROC liegen und mit einem Strich - am Anfang der Zeile beginnen.
Die Folgestatements sind in der Reihenfolge einzugeben, in der sie später ausgeführt werden sollen:

Die Folgestatements von MOD-PROC erlauben keinen Zugriff auf Datensätze, die durch INPUT-EXT eingelesen wurden. Dies ist nur in den Folgestatements von ADD-PROC und UPD-PROC möglich.

–Cn = Merkmalswerte
Die Bedingungsvariable Cn erhält Merkmalswerte zugewiesen.

–Nn = numerischer Wert
Die numerische Variable Nn erhält einen Wert zugewiesen.

–Tn = Textwert
Die Textvariable Tn erhält einen Wert zugewiesen.

Genauere Angaben zu diesen Folgestatements befinden sich im Abschnitt Wertezuweisungen .

–DO –EDO
Beginn und Ende einer Schleife: Die zwischen DO und EDO liegenden Statements werden mehrfach ausgeführt. Siehe DO-Statement.

–IF logischer Ausdruck **–EIF**
Die Verarbeitung der Statements zwischen IF und EIF hängt ab vom Wert des logischen Ausdrucks. Siehe IF-Statement.

–LANG Sprachschlüssel zur Auswahl von Variablentexten. Siehe LANG-Statement.

–PRT <Variable> <Text>
Dieses Statement druckt Variablenwerte und feste Texte aus. Siehe PRT-Statement.

–REPORT Ausgabe in Textdatei. Siehe REPORT-Statement.

–CLEAN logischer Ausdruck <**HEAD=**'text'>
Dieses Statement dient zur Bereinigung fehlerhafter Variablenwerte durch hinter CLEAN liegende Wertezuweisungen. Diese Wertezuweisungen werden nur ausgeführt, wenn der logische Ausdruck erfüllt ist. Im Zählprotokoll gibt eine CLEAN-Statistik eine Übersicht über die ausgeführten Korrekturen.

–TST logischer Ausdruck <Variable> <Text>
Dieses Statement prüft für die einzelnen statistischen Fälle, ob der logische Ausdruck erfüllt ist. Wenn nicht, wird der Text zusammen mit den Werten der angegebenen Variablen als Fehlermeldung ausgedruckt. Siehe TST-Statement.

–END < **CASE** | **ALL** >

Dieses Statement beendet die Verarbeitung des laufenden Falles in MOD-PROC.
Die hinter END stehenden Folgestatements werden nicht mehr ausgeführt.
Ohne weitere Angaben wird die Verarbeitung des laufenden Falles hinter den Folgestatements von MOD-PROC fortgesetzt.
CASE beendet darüber hinaus die Verarbeitung des laufenden Falles durch alle noch folgenden Statements.
ALL bricht die gesamte Verarbeitung vorzeitig ab.

Beispiel

```
INPUT-EXT      FNAME=EINGABE.EXT      ID=D(1..4)
INPUT-INT      FNAME=EINGABE.INT
ADD-PROC
-N1=D1(5..8)
UPD-PROC
-N1=D1(10..12)
MOD-PROC
-N1=N1*100
```

Mit dem Statement INPUT-EXT wird eine externe und mit INPUT-INT eine interne Eingabedatei angefordert. Um die Datensätze aus den beiden Dateien zu statistischen Fällen zusammenfügen zu können, müssen alle Eingabesätze eine Fall-Identifikation besitzen.

Die statistischen Fälle auf einer internen Datei besitzen stets eine Fall-Identifikation, zu der keine besondere Angabe erforderlich ist. In INPUT-EXT muß jedoch durch ID Typ und Position dieser Identifikation angegeben werden.

Liegt ein Fall aus INPUT-EXT zur Verarbeitung vor, ohne einen passenden Fall aus INPUT-INT mit gleicher Fall-Identifikation, so wird er durch ADD-PROC und seine Folgestatements verarbeitet. Hier wird der Variablen N1 durch das Folgestatement zu ADD-PROC ein Wert aus den Positionen 5...8 des externen Datensatzes zugewiesen.

Wird dagegen ein passender Fall auf INPUT-INT gefunden, so regelt das Statement UPD-PROC die Verarbeitung. Hier wird der Variablen N1 durch das Folgestatement zu UPD-PROC ein Wert aus den Positionen 10...12 des Datensatzes aus INPUT-EXT zugewiesen.

Das Statement MOD-PROC und sein Folgestatement wird danach in beiden Fällen wirksam: Der Wert der Variablen N1 wird mit 100 multipliziert.

OPTIONS

Mit OPTIONS lassen sich allgemeine Verarbeitungsregeln festlegen, die für den gesamten Programmlauf gültig sind.

OPTIONS muss das erste Statement in der Statementdatei sein, es dürfen höchstens Kommentar-Statements davor liegen.

Folgende Angaben sind möglich:

ELIMIT = | **NO** / n |

NO Es werden beliebig viele Fehlermeldungen zu den Eingabedaten zugelassen und protokolliert. Dies sind die Meldungen aus PRT- und TST-Statements und die Fehlermeldungen 500 ... 599 zu den eingelesenen statistischen Fällen; siehe PRT-Statement, TST-Statement und *Fehlermeldungen*. Der Ausdruck dieser Meldungen kann mit der Angabe MSG=NO verhindert werden.

n Anzahl 1 ... 999 999 999 von Fehlermeldungen, bei der die Verarbeitung vorzeitig abgebrochen werden soll. Wird die Fehlergrenze erreicht, erfolgt der Abbruch mit der Fehlermeldung 003.

Ohne die Angabe ELIMIT wird die Verarbeitung nach 1000 Meldungen *abgebrochen*.

ESUM Diese Angabe stellt am Ende der Datendurchläufe eine Fehlerstatistik über die PRT-Meldungen, die TST-Meldungen und die Fehlermeldungen 500 ... 599 in das Zählprotokoll (siehe PRT-Statement, TST-Statement und *Fehlermeldungen*). Diese könnte folgendermaßen aussehen:

```
STATEMENT  ANZAHL FEHLER
        1  125
       20  12
      109  17
      412  237
```

Diese Fehlerstatistik ist nach aufsteigenden Statementnummern sortiert.

ESUMS Diese Angabe hat die gleiche Wirkung wie ESUM.
Die Fehlerstatistik wird jedoch nicht nach den Statementnummern sortiert sondern nach der Anzahl der gefundenen Fehler.

```
MSG = | NO                                                                                    |
      | < FETCH: | YES | >    < ,MISSING: | YES | >    < ,RANGE: | YES | >                |
      |          | NO  |                  | NO  |                  | NO  |                   |
```

Diese Angaben steuern die Ausgabe der Fehlermeldungen 500 ... im Zählprotokoll:

NO Mit MSG=NO werden die PRT-Meldungen, die TST-Meldungen und die Fehlermeldungen 500 ... 599 zu den eingelesenen statistischen Fällen unterdrückt; siehe PRT-Statement, TST-Statement und im Abschnitt *Fehlermeldungen*.

FETCH Durch MSG=FETCH wird die Ausgabe der Fehlermeldung 518 Datensatz fehlt in FETCH-FILE im Zählprotokoll gesteuert. FETCH:YES sorgt für die Ausgabe der Meldung und FETCH:NO verhindert sie. Fehlt die Angabe FETCH: so wird die Fehlermeldung 518 ausgegeben.

MISSING Durch MSG=MISSING wird die Ausgabe der Fehlermeldung 525 Schlüssel nicht im Datensatz im Zählprotokoll gesteuert. Diese Meldung erscheint, wenn ein Datensatz zu einem Datenfeld vom Typ AK, DK, MK oder UK keinen passenden Schlüssel enthält.
MISSING:YES sorgt für die Ausgabe der Meldung und MISSING:NO verhindert sie.
Fehlt die Angabe MISSING:YES so wird die Fehlermeldung 525 nicht ausgegeben.

RANGE Durch MSG=RANGE wird die Ausgabe der Fehlermeldung 524 Merkmal passt nicht zum Zielfeld im Zählprotokoll gesteuert. RANGE:YES sorgt für die Ausgabe der Meldung und RANGE:NO verhindert sie. Fehlt die Angabe RANGE:YES so wird die Fehlermeldung 524 nicht ausgegeben.

Wenn die Fehlermeldungen 500 ... durch MSG-Angaben unterdrückt werden, enthält der Operand ERR trotzdem die entsprechenden Fehlernummern. Auch die Fehlerstatistiken ESUM und ESUMS werden unverändert ausgegeben.

OLD Beim Übergang von der Version 6.0 zur Version 6.1 von CNTA hat sich die Interpretation von Merkmalswerten in Wertezuweisungen geändert:

In den Versionen vor 6.1 setzte das Statement

```
-C20=101
```

die Merkmale C20.1 und C20.3 auf 1.

Ab Version 6.1 setzt dieses Statement stattdessen das Merkmal C20.101 auf 1.
Um das alte Verfahren zu erhalten, ist nun das Zeichen ° vor die Merkmalswerte 0 und 1 zu stellen:

```
-C20=°101
```

Mit der Angabe OLD im Statement OPTIONS die alten Regeln für ganze Statementdateien reaktivieren.

RSTART = | **n** | **TIME** |

Mit dieser Angabe wird ein Startwert für den Zufallsgenerator von CNTA vergeben.
Der Zufallsgenerator wird bei Wertezuweisungen verwendet, wenn aus Mehrfachnennungen eine Einzelnennung zufällig ausgewählt werden soll. Siehe im Abschnitt *Wertezuweisungen: Merkmale* bei SR . Weiter wird der Zufallsgenerator bei der Erzeugung von Teilstichproben durch das Statement SELECT mit der Angabe SAMPLE verwendet.

Ein Zufallsgenerator erzeugt, vergleichbar mit einem Würfel, eine möglichst regellose Folge von Zahlenwerten. Beginnend mit einem Startwert wird aus jedem so ermittelten Zahlenwert der nächste Zahlenwert errechnet. Anders als beim Würfeln liefert ein Zufallsgenerator bei gleichem Startwert stets die gleiche Zahlenfolge.

Der Zufallsgenerator von CNTA beginnt in der Regel mit dem Startwert 1. Dadurch ist sichergestellt, dass bei zwei verschiedenen Verarbeitungsläufen die gleichen "zufälligen" Ergebnisse erzielt werden. Mit Hilfe der Angabe RSTART kann dieser Startwert verändert werden, um andere Zahlenfolgen zu erzeugen.

n Startwert 1 ... 32 767 für den Zufallsgenerator. Bereits der Übergang vom Startwert 1 zum Startwert 2 liefert das Programm völlig unterschiedliche Zahlenfolgen.

TIME Der Startwert wird aus Datum und Uhrzeit des Computers erzeugt. Diese Angabe liefert ohne Änderungen in den Statements in jedem Verarbeitungslauf eine andere Zufallsfolge.

TEST | **TESTP**

TEST Mit dieser Angabe werden nur Statements gelesen und geprüft.
Datensätze werden nicht gelesen und Auswertungen nicht gedruckt.

TESTP Es werden Statements gelesen und geprüft.
Datensätze werden nicht gelesen, wohl aber die geforderten Auswertungen gedruckt, jedoch nur mit Nullen anstelle der Zahlenwerte.

XREF Diese Angabe erstellt am Ende des Zählprotokolls eine Cross-Reference-Liste.
Sie enthält alle in den Statements neu definierten oder verwendeten Variablen und Textelemente mit den dazugehörigen Statementnummern. Variable und Textelemente aus INPUT-INT und INPUT-SEC, die in den Statements nicht angesprochen werden, erscheinen nicht in der Liste.
Eine Cross-Reference-Liste könnte folgendermaßen aussehen:

```
Definitionen        Referenzen

C1          int     27 34
C10         15      20 27
CALTER      int     27 112 214
N10         16
NGEWICHT    int     27 104
(10)        int     27
```

Unter *Definitionen* werden alle Variablen und Textelemente aufgeführt. Direkt dahinter steht der Ort ihrer Definition. *int* steht hinter Variablen und Textelementen, die aus INPUT-INT oder INPUT-SEC eingelesen wurden. Die Angaben 15 hinter C10 und und 16 hinter N10 sind die Zeilennummern der Definition dieser Variablen in der laufenden Statementdatei. Unter *Referenzen* finden sich die Nummern der Zeilen, in denen die Variablen oder Textelemente verwendet wurden.

OUTPUT-EXT

Das Statement OUTPUT-EXT erzeugt eine Datei mit statistischen Fällen in einem der externen Datenformate, die in den Abschnitten Externe Dateien und Externe Datenfelder beschrieben sind.
Dieses Statement ist in jeder Statementdatei nur einmal möglich.

Aufbau, Verschlüsselung und Format eines solchen Datenbestandes sind frei wählbar. Neben Datensätzen mit fester und variabler Satzlänge sind auch CSV-Dateien möglich, bei denen die Datenwerte durch Separatorzeichen wie Komma oder Semikolon voneinander getrennt sind.

Es gibt zwei unterschiedliche Wege, die Ausgabewerte für OUTPUT-EXT zu bestimmen:
Mit den Angaben ANSI ... QUANTUM lässt sich in den Folgestatements von OUTPUT-EXT für jeden Ausgabewert die Position, die Länge und das Format frei wählen.
Bei AUTOANSI ... SPSS ist lediglich festzulegen, welche Variable auszugeben sind. Die übrigen Angaben legt das Programm automatisch fest.

Folgende Angaben sind im Statement OUTPUT-EXT möglich:

FNAME = dateiname

Hier ist der Name der Ausgabedatei mit den statistischen Fällen anzugeben. Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
FNAME=C:\DATEN\STUDIE27.EXT
FNAME='STUDIE27 AUSGABE'
FNAME=\\SERVER\DATEN\STUDIE27.EXT
```

Eine bereits existierende Datei mit diesem Namen wird vom Programm vor der Ausgabe gelöscht.

Enthält *dateiname* keine Pfadangabe, so stellt CNTA die Datei in das Verzeichnis der Statementdatei und CNTW in das Verzeichnis der REP-Datei.
Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\STUDIE27.EXT
```

VNAME = dateiname

Diese Angabe erstellt zu den Dateiformaten AUTO ... SPSS eine zweite Ausgabedatei mit Beschreibungen der ausgegebenen Variablen und Daten. Es gelten die gleichen Regeln wie bei FNAME. Der Inhalt der Datei VNAME hängt vom gewählten Dateiformat ab:

AUTO…: In diesen Formaten erstellt VNAME die Variablenbeschreibungen im CSV-Format, mit dem Semikolon als Separatorzeichen Siehe *Beispiel 1* zu *OUTPUT-EXT: Folgestatements AUTOANSI ...*
Die Angabe VNAME ist bei den Formaten AUTO ... nicht zwingend erforderlich.

SPSS: Hier ist eine Angabe VNAME mit der Dateierweiterung SPS möglich. Daraus entsteht eine SPSS-Syntax-Datei mit den SPSS-Statements DATA LIST, VARIABLE LABELS und VALUE LABELS. Außerdem wird durch FNAME eine dazu passende Datendatei erzeugt. Mit Hilfe von SPSS lässt sich aus diesen beiden Dateien eine SPSS-SAV-Datei erstellen.
Fehlt die Angabe VNAME, so erzeugt FNAME direkt eine SPSS-SAV-Datei.

TRIPLES: Hier ist die Angabe VNAME zwingend erforderlich. Die Variablendefinitionen werden im Triple-S-Format der Version 1.1 ausgegeben.

ASCII
ANSI
UTF8
COLBIN
QUANTUM
EBCDIC
BINARY1
BINARY2
XLSX
AUTOASCII
AUTOANSI
AUTOUTF8
AUTOCOLBIN
AUTOEBCDIC
AUTOQUANTUM
AUTOXLSX
SPSS
SPSSUTF8
TRIPLES

Diese Angaben legen das Datenformat und den Zeichencode der Datei fest:
Sie entscheidet, welche externen Datenfelder in der Datei vorkommen können und wie diese dort verschlüsselt sind. Insbesondere wird der Zeichencode der Datenfelder vom Typ A, AK, AS, D, DK, DS, L, M, MK, MS, U, UK und US festgelegt.
CSV-Dateien (siehe Angabe SEP) sind nur in den Formaten ASCII, ANSI und EBCDIC möglich.
Weitere Angaben zu diesen Dateiformaten befinden sich im Abschnitt *Externe Dateien* sowie im Anhang *Zeichenzuordnung ANSI, ASCII, EBCDIC* und *Lochkombinationen der COLBIN-Spalten*.
Ohne diese Angabe wird das ANSI-Format verwendet.

ASCII Dies ist das ursprüngliche Datenformat der Personal Computer im MS/DOS.
Die Längen und Positionen in den Datensätzen sind in Bytes anzugeben.
Alle externen Datenfelder sind möglich.

ANSI Dies ist der von WINDOWS bevorzugte Zeichencode.
Der Unterschied zu ASCII beschränkt sich auf die Umlaute, den Buchstaben ß und einige Sonderzeichen. Die Längen und Positionen in den Datensätzen sind in Bytes anzugeben.
Alle externen Datenfelder sind möglich.

UTF8 Dieser Zeichencode umfasst alle Zeichen des 16-Bit-UNICODE, also praktisch alle international vorkommenden Schriftzeichen. Siehe Zeichenzuordnung UTF8.

EBCDIC Dies ist das normale Datenformat von Großrechnern. Die Längen und Positionen in den Datensätzen sind in Bytes anzugeben. Alle externen Datenfelder sind möglich.

COLBIN Die Daten werden im Dualkarten-Format (column binary) erstellt.
Die Positionen und Längen in den externen Datensätzen sind nicht in Bytes sondern in Spalten anzugeben. Das gilt sowohl für das Statement OUTPUT-EXT als auch für die Wertezuweisungen in den Folgestatements.
Es sind nur externe Datenfelder vom Typ A, D, F, G, I, K, L, M und U möglich.

QUANTUM Dies ist eine Speicherplatz sparende COLBIN-Variante. Die Positionen und Längen in den externen Datensätzen sind in Spalten und nicht in Bytes anzugeben.
Es sind nur externe Datenfelder vom Typ A, D, K, L , M und U möglich.

BINARY1 Diese Dateien bestehen aus ein bis vier Bytes langen Dualzahlen.
Positionen und Längen in den externen Datensätzen sind in Bytes anzugeben.
Es sind nur externe Datenfelder vom Typ B, I, F und G möglich.

BINARY2 Diese Dateien bestehen aus zwei und vier Bytes langen Dualzahlen.
Positionen und Längen in den externen Datensätzen sind in Feldern anzugeben, jedes Feld zu zwei Bytes. Es sind nur externe Datenfelder vom Typ B, I, F und G möglich.

XLSX Diese Angabe erzeugt eine Datei im XLSX-Format von Excel.
Es sind nur externe Datenfelder vom Typ AK, AS, DK, DS, MK, MS, UK und US möglich.
Die Angabe WORKSHEET liefert den Namen für das Arbeitsblatt in der Datei.

AUTOANSI
AUTOASCII
AUTOUTF8
AUTOCOLBIN
AUTOEBCDIC
AUTOQUANTUM
AUTOXLSX

Diese Angaben unterscheiden sich von ASCII, ANSI, COLBIN, EBCDIC, QUANTUM und XLSX dadurch, dass die Ausgabedatei automatisch organisiert wird: Position, Länge und Typ der Datenfelder legt das Programm selbsttätig fest.

Mit Folgestatements hinter OUTPUT-EXT lassen sich Reihenfolge und Auswahl der Variablen sowie die externen Feldtypen verändern. Die Parameter CSTD, DSTD, NSTD und VSTD verändern die Ausgaberegeln.

In die durch PAGE oder PAGEP vorgegebenen Ausgabedateien stellt das Programm einen Codeplan, der den Dateiaufbau mit den vorhandenen Variablen- und Merkmalstexten beschreibt. Mit Folgestatements hinter OUTPUT-EXT lassen sich auch Textelemente in diesen Codeplan ausgeben. Die Angabe HEAD in OUTPUT-EXT liefert eine Überschrift für diese Beschreibung und VNAME stellt den Codeplan zusätzlich in eine CSV-Datei.

SPSS Diese Angabe erstellt eine Datendatei FNAME und eine SPSS-Syntax-Datei VNAME, die direkt von SPSS ab Version 6.0 zu verarbeiten sind. Die Organisation der Ausgabedatei erfolgt automatisch: Position, Länge und Typ der Datenfelder legt das Programm selbsttätig fest. Dabei werden die Werte der vorhandenen Variablen in die externe Datei ausgegeben.

Die Syntax-Datei VNAME definiert zu den ausgegebenen CNT-Variablen entsprechende SPSS-Variable einschließlich der zugehörigen Texte.

Mit Folgestatements hinter OUTPUT-EXT lassen sich Reihenfolge und Auswahl der Variablen sowie externe Feldtypen verändern. Die Parameter CSTD, DSTD, NSTD und VSTD verändern die Ausgaberegeln. Die Angabe HEAD in OUTPUT-EXT erzeugt ein TITLE-Statement in der Datei VNAME.

SPSS–UTF8 Diese Angabe unterscheidet sich von SPSS nur dadurch, dass die Datei VNAME mit den Label-Texten im UTF8-Code erstellt wird. Die Datendatei FNAME wird weiterhin im ASCII-Code erstellt.
Enthält eine externe Datendatei Werte im UTF8-Code, die an SPSS weitergereicht werden soll, so empfiehlt es sich diese mit INPUT-EXT ASCII einzulesen. Auf diese Weise lassen sich UTF8-Texte über T-Variable ohne Code-Umwandlung in die SPSS-Datei ausgeben.

TRIPLES Diese Angabe erstellt eine Datendatei FNAME und eine Datei VNAME mit einer Beschreibung der ausgegebenen Variablen im Triple-S-Format. Siehe *www.triple-s.org*.

Die Organisation der Ausgabedatei erfolgt automatisch: Position, Länge und Typ der Datenfelder legt das Programm selbsttätig fest. Dabei werden die Werte der vorhandenen Variablen in die externe Datei ausgegeben.

Mit Folgestatements hinter OUTPUT-EXT lassen sich Reihenfolge und Auswahl der Variablen sowie externe Feldtypen verändern. Die Parameter CSTD, DSTD, NSTD und VSTD verändern die Ausgaberegeln. Die Angabe HEAD in OUTPUT-EXT erzeugt eine <TITLE>-Angabe in der Datei VNAME.

Weitere Angaben zu den Dateiformaten befinden sich im Abschnitt Externe Dateien und im Anhang Zeichenzuordnung ANSI, ASCII, EBCDIC und Lochpositionen der COLBIN-Spalten

SIZE = n < **,EOL** | **,EOLF** >

Diese Angabe legt die Länge der einzelnen Datensätze fest.
Fehlen die Angaben SIZE oder *n* so wird

- bei COPY die Angabe SIZE aus INPUT-EXT übernommen
- für COLBIN-, QUANTUM- und EBCDIC-Dateien SIZE=80 angenommen
- bei ASCII- und ANSI-Dateien SIZE=80, EOL gesetzt
- für SPSS-Dateien SIZE=40000,EOL gesetzt
- für TRIPLES-Dateien wird SIZE=100000,EOL angenommen.
- bei Dateien vom Typ AUTO ... wird für SIZE die effektiv benötigte Satzlänge verwendet

n Satzlänge 1 ... 10 000 000 in Bytes; bei COLBIN-Dateien in Spalten.

EOL Diese Angabe ist nur bei externen Dateien im ASCII-, ANSI- und EBCDIC-Format zulässig.
Sie beendet die Datensätze mit den beiden Zeilenende-Zeichen CR=10 und LF=13.
Die Satzlänge *n* gibt dabei die maximal zulässige Länge der Datensätze an.
Die Zeilenende-Zeichen selbst sind in dieser Länge *n* nicht enthalten. Leerstellen am Ende der Datensätze werden vor der Ausgabe entfernt.
SPSS-, TRIPLES- und QUANTUM-Dateien werden auch ohne diese Angabe mit Zeilenende-Zeichen erstellt.

EOLF Diese Angabe bewirkt das Gleiche wie EOL, beendet die die Datensätze jedoch nur mit dem Zeichen LF=13.

COPY Diese Angabe kopiert die aus INPUT-EXT eingelesenen Datensätze auf OUTPUT-EXT.
Folgestatements sind dazu nicht erforderlich, können die kopierten Datensätze aber noch verändern.

Für die Ausgabeformate AUTOASCII, AUTOEBCDIC, AUTOCOLBIN, AUTOQUANTUM, SPSS und TRIPLES ist die Angabe COPY nicht möglich.

INPUT-EXT und OUTPUT-EXT müssen für COPY gleiche Satzlängen und gleiche Anzahl Datensätze pro Fall besitzen, dürfen sich aber in der Angabe EOL unterscheiden.

In OUTPUT-EXT können die Angaben zum Dateiformat ASCII ... COLBIN, zu SIZE und zu RC fehlen. Sie werden dann aus INPUT-EXT übernommen.

Unterschiedliche Datenformate der Ein- und Ausgabedatei wandelt COPY automatisch um, zum Beispiel von COLBIN nach ASCII. Dies ist jedoch nur sinnvoll, wenn die Eingabedatei keine B-, K-, I-, F- oder G-Felder enthält; deren Werte werden bei der Umwandlung zerstört.

Die Übertragung der Datensätze von INPUT-EXT nach OUTPUT-EXT erfolgt nach eventuellen Veränderungen der Eingabesätze durch Folgestatements von ADD-PROC und UPD-PROC. Danach werden Angaben ID und RC aus OUTPUT-EXT in die Ausgabesätze gestellt und schließlich eventuelle Folgestatements von OUTPUT-EXT ausgeführt.

Besitzt die Eingabedatei INPUT-EXT mehrere Datensätze pro Fall und sind einige davon ptional (Angabe *O:* hinter RC), so können solche Sätze in der Eingabe auch fehlen. Sie werden durch Datensätze ersetzt, die nur Leerzeichen enthalten, auch keine ID- noch RC-Angaben. COPY überträgt sie ebenso unvollständig nach OUTPUT-EXT. Durch Angabe von ID und RC in OUTPUT-EXT und Folgestatements lassen sich eventuelle Mängel beheben.

ID **ID1** **ID2** **ID3** **ID4**	=	**A** **D** **F** **G** **P**	< d >	**(** a **..** e**)** **(** a , c **)**	< **,** 'name' >
		AK **DK**	< d >	**(**< 's' >**)**	< **,** 'name' >
		AS **DS**	< d >	**(** r **)**	< **,** 'name' >

Durch diese Angaben werden die entsprechenden Fall-Identifikationen in jeden Datensatz der Ausgabedatei gestellt. Eine solche Fall-Identifikation kann im Statement INPUT-EXT definiert sein oder auf einer internen Eingabedatei INPUT-INT vorliegen. Gibt es weder in INPUT-EXT noch in INPUT-INT eine Fall-Identifikation, so definiert CNTA automatisch ID und nummeriert dafür alle statistischen Fälle durch, beginnend mit der Nummer 1. Auch diese automatisch vergebene Identifikation kann hier ausgegeben werden.

Liegt in OUTPUT-EXT die Angabe COPY vor, so werden zunächst die Eingabedaten von INPUT-EXT in die Ausgabesätze übertragen. Erst danach werden die hier angegebenen Fall-Identifikationen in die Ausgabesätze gestellt. Sie überschreiben dabei in dem Feld (a..e) die Werte aus den Eingabesätzen.

Erst nach der Übertragung der ID-Werte in die Ausgabesätze werden die Wertezuweisungen aus den Folgestatements von OUTPUT-EXT ausgeführt.

Anders als bei INPUT-INT müssen diese ID-Angaben nicht vollständig vorliegen: Sind zum Beispiel ID und ID1 aus INPUT-EXT definiert, so ist es möglich, in OUTPUT-EXT lediglich ID1 anzugeben.

Für die Fall-Identifikationen lassen sich folgende Feldtypen verwenden, so wie im Abschnitt *Externe Datenfelder* beschrieben:

A Die Fall-Identifikationen werden als alphanumerische Zeichen ausgegeben.

AK In CSV-Dateien: Der Key besteht aus alphanumerischen Zeichen mit Schlüsselwort.

AS In CSV-Dateien: Die Fall-Identifikationen werden als alphanumerische Zeichen ausgegeben.

D Die Fall-Identifikationen werden als numerische Zeichen ausgegeben.

DK In CSV-Dateien: Der Key besteht aus numerischen Zeichen mit Schlüsselwort.

DS In CSV-Dateien: Die Fall-Identifikationen werden als numerische Zeichen ausgegeben.

F Die Fall-Identifikationen werden als Dualzahlen mit Vorzeichen ausgegeben (Integer).

G Die Fall-Identifikationen werden als Dualzahlen ohne Vorzeichen ausgegeben (Unsigned Integer).

P Die Fall-Identifikationen werden als gepackte Dezimalzahlen ausgegeben.

d Nummer des Datensatzes, in den die Fall-Identifikation gestellt werden soll.
Fehlt diese Angabe, so wird die Fall-Identifikation in jeden Datensatz ausgegeben, sonst nur in den hier angegebenen Satz.
ACHTUNG: Diese Nummer ist nicht das Satz-Kennzeichen selbst, sondern die Reihenfolgenummer des hinter RC angegebenen Kennzeichens.

a erstes Byte 1 ... 1 000 000 der Fall-Identifikation in den externen Datensätzen;
erste Spalte 1 ... 500 000 im COLBIN-Format.

c Länge der Fall-Identifikation in den externen Datensätzen in Bytes; bei COLBIN in Spalten.
Die maximale Länge hängt vom Datenfeld ab:

im A-Feld:	1 ... 127 Bytes oder Spalten
im D-Feld:	1 ... 16 Bytes oder Spalten
im F-Feld:	1 ... 8 Bytes
im G-Feld:	1 ... 8 Bytes
im P-Feld:	1 ... 10 Bytes.

Sind die auszugebenden Werte größer als das hier festgelegte Zielfeld, so werden sie abgeschnitten: Im A-Feld rechts, ohne Fehlermeldung; sonst links und mit Fehlermeldung.

e letztes Byte 1 ... 1 000 000 der Fall-Identifikation in den externen Datensätzen;
letzte Spalte 1 ... 500 000 im COLBIN-Format.
Die Länge der Fall-Identifikation muss den unter c angegebenen Regeln entsprechen.

r Die Angabe r = 1 ... 32 767 ist die Reihenfolgenummer des Wertes im Datensatz d einer CSV-Datei. Leerzeichen vor und hinter der Fall-Identifikation werden ignoriert. Nur bei AS-Feldern gehören die führenden Leerzeichen zur Fall-Identifikation. Für die Länge der Werte gelten die unter c angegebenen Regeln für A- und D-Felder.

's' Diese Angabe legt das Schlüsselwort fest, mit dem die ID-Werte in den Datensätzen zu kennzeichnen sind. Fehlt sie, so wird als Schlüssel ID ... ID4 verwendet.

'name' Diese Angabe vergibt bei SPSS-Dateien *name* als Variablenname anstelle von ID … ID4.
Bei Dateien in den Formaten AUTO… wird ID … ID4 im Codeplan durch *name* ersetzt.

Weitere Angaben zu den Datenfeldern finden sich im Abschnitt Externe Datenfelder.

HEAD = 'text' < **,L**i >

Diese Angabe stellt einen Überschriftstext an den Anfang der Dateibeschreibung für die Dateiformate AUTOASCII, AUTOCOLBIN, AUTOEBCDIC und AUTOQUANTUM.
Im Dateiformat SPSS erzeugt HEAD ein TITLE-Statement in der Datei VNAME und im Dateiformat TRIPLES eine <TITLE>-Angabe.

'text' Der Text, der als Überschrift ausgegeben werden soll.
Er kann auch aus mehreren Zeilen bestehen, wie im Abschnitt *Textzeilen* beschrieben.
Fehlt ein solcher Text, wird statt dessen der Text *Dateibeschreibung* ausgegeben.

Li Mit dieser Angabe können die Linien oberhalb und unterhalb der Überschrift beeinflusst werden.
Mit L0 werden diese Linien entfernt; eine Angaben L1 bis L16 wählt die Linie aus, so wie sie im Statement PAGEP definiert wurde.
Diese Angabe ist nur wirksam, wenn eine Dateibeschreibung auf einen Seitendrucker ausgegeben wird. (Siehe dazu beim PAGEP-Statement).

Mit der Angabe RC wird die Anzahl und die Reihenfolge der Datensätze festgelegt, die zu jedem statistischen Fall auszugeben sind. Fehlt die Angabe RC, so gelten folgende Regeln:
Beim Ausgabeformat SPSS und TRIPLES legt CNTA gerade so viele Datensätze pro Fall an, wie zur Aufnahme der Variablenwerte erforderlich sind.
Bei OUTPUT-EXT mit COPY werden die RC-Angaben aus INPUT-EXT übernommen.
Ansonsten wird nur ein Datensatz pro Fall ausgegeben.

RC = n Durch n = 1 ... 2 000 wird die Anzahl der Datensätze festgelegt, die für jeden statistischen Fall auszugeben ist.
Beim Ausgabeformat SPSS legt diese Angabe die Mindestanzahl auszugebender Sätze fest, wobei CNTA den Wert n automatisch erhöht, wenn die Variablenwerte mehr Ausgabesätze benötigen.

Mit der Angabe RC=n erhalten die Datensätze kein Satz-Kennzeichen. Dazu ist RC=A, D ... erforderlich.

RC = **A** | **D** | **F** | **G** | **P** [**(** a **..** e **)** | **(** a , c **)**] [**O:** | **R:**] k**,** k**,** ...

AK | **DK** **(**< 's' >**)** k**,** k**,** ...

AS | **DS** **(** r **)** k**,** k**,** ...

Diese Form der RC-Angabe legt fest, wie viele Datensätze pro Fall auszugeben und dass jeder Ausgabesatz mit einem Satz-Kennzeichen k zu markieren ist.
Es sind 1 ... 2 000 Datensätze pro Fall möglich.

Ist in OUTPUT-EXT auch COPY angegeben, so müssen beide externe Dateien die gleiche Anzahl von Datensätzen besitzen. Durch die RC-Angabe können die Ausgabesätze neue Satz-Kennzeichen erhalten.

OUTPUT-EXT füllt die Ausgabesätze in dieser Reihenfolge:

- Übertragung der Eingabedaten von INPUT-EXT in die Ausgabesätze, sofern COPY angegeben ist.
- Ausgabe der hier angegebenen Satz-Kennzeichen in die Datensätze.
  Durch COPY übertragene Werte können dabei überschrieben werden.
- Ausgabe von externen Datenfeldern durch Wertezuweisungen aus den Folgestatements. Die in den vorigen Schritten übertragenen Werte können dabei überschrieben werden.

Für die Satz-Kennzeichen lässt sich einer der folgenden Feldtypen verwenden, so wie im Abschnitt *Externe Datenfelder* beschrieben:

A Die Satz-Kennzeichen sind alphanumerische Zeichen.

AK In CSV-Dateien: Die Satz-Kennzeichen bestehen aus alphanumerischen Zeichen mit Schlüsselwort.

AS In CSV-Dateien: Die Satz-Kennzeichen sind alphanumerische Zeichen.

D Die Satz-Kennzeichen sind numerische Zeichen.

DK In CSV-Dateien: Die Satz-Kennzeichen bestehen aus numerischen Zeichen mit Schlüsselwort.

DS In CSV-Dateien: Die Satz-Kennzeichen sind numerische Zeichen.

F Die Satz-Kennzeichen sind Dualzahlen mit Vorzeichen (Integer).

G Die Satz-Kennzeichen sind Dualzahlen ohne Vorzeichen (Unsigned Integer).

P Die Satz-Kennzeichen sind gepackte Dezimalzahlen.

a erstes Byte 1 ... 1 000 000 der Satz-Kennzeichen in den externen Datensätzen;
erste Spalte 1 ... 500 000 im COLBIN-Format.

c Länge der Satz-Kennzeichen in den externen Datensätzen in Bytes; bei COLBIN in Spalten.
Die maximale Länge hängt vom Datenfeld ab:
im A-Feld: 1 ... 127 Bytes oder Spalten
im D-Feld: 1 ... 16 Bytes oder Spalten
im F-Feld: 1 ... 8 Bytes
im G-Feld: 1 ... 8 Bytes
im P-Feld: 1 ... 10 Bytes.

e letztes Byte 1 ... 1 000 000 der Satz-Kennzeichen in den externen Datensätzen;
letzte Spalte 1 ... 500 000 im COLBIN-Format.
Die Länge der Satz-Kennzeichen muss den unter c angegebenen Regeln entsprechen.

r Die Angabe r = 1 ... 32 767 ist die Reihenfolgenummer des Wertes im Datensatz d einer CSV-Datei. Leerzeichen vor und hinter dem Satz-Kennzeichen werden ignoriert. Nur bei AS-Feldern gehören die führenden Leerzeichen zum Satz-Kennzeichen. Für die Länge der Werte gelten die unter c angegebenen Regeln für A- und D-Felder.

s Diese Angabe ist in Hochkommas einzuschließen. Sie legt das Schlüsselwort fest, mit dem die RC-Werte in den Datensätzen zu kennzeichnen sind. Fehlt diese Angabe, so wird dafür *RC* verwendet.

O: Die Angabe O=optional führt dazu, dass alle dahinter aufgeführten Datensätze nicht ausgegeben werden, wenn sie nur Leerzeichen enthalten. Die Fall-Identifikationen IDi und das Satz-Kennzeichen RC zählen dabei nicht zum Datenteil. Ein Datensatz ist leer, bevor ihm durch ein Folgestatement von OUTPUT-EXT Daten zugewiesen werden, und er bleibt leer, wenn durch ein Folgestatement Leerzeichen ausgegeben werden. Zum Beispiel sorgt

```
RC=D(1,1) 2, O:3,4
```

dafür, dass der erste Datensatz eines Falles mit dem Satz-Kennzeichen 2 immer ausgegeben wird, auch wenn er leer ist. Die beiden weiteren Datensätze mit den Kennzeichen 2 und 3 werden nur ausgegeben, wenn sie nicht leer sind.

R: Die Angabe R=required führt dazu, dass alle dahinter aufgeführten Datensätze in jedem Fall ausgegeben werden, auch wenn sie im Datenteil nur Leerzeichen enthalten. Mit R: kann ein vorausgehendes O: wieder rückgängig gemacht werden und umgekehrt.
Zum Beispiel hat die Angabe

```
RC=D(1,1) 2, O:3,4, R:6
```

folgende Wirkung:
Der erste Datensatz eines jeden statistischen Falles wird immer auf OUTPUT-EXT ausgegeben und erhält das Satz-Kennzeichen 2.
Der zweite und der dritte Datensatz erhalten die Satz-Kennzeichen 3 und 4. Sie werden nur dann ausgegeben, wenn sie zusätzlich zu den Fall-Identifikationen IDi und zum Satz-Kennzeichen RC noch Werte enthalten, die vom Leerzeichen verschieden sind. Der letzte Datensatz eines statistischen Falles wird immer ausgegeben und erhält das Satz-Kennzeichen 6.

k Hier sind alle gewünschten Satz-Kennzeichen anzugeben.
Im D-, DS-, F-, G- und P-Feld sind dies numerische Werte, im A- und AS-Feld sind es in Hochkommas eingeschlossene Zeichen oder Zeichenfolgen. Zum Beispiel:

```
RC=D(1..2)1,2,2
RC=A(1..2)'XX','YY','ZZ'
```

CNTA ordnet die Datensätze eines Falles in der hier vorgegebenen Reihenfolge an. Durch die Angabe

```
RC=D(1..2)3,9,1
```

wird zum Beispiel erreicht, dass der Datensatz mit dem Kennzeichen 3 in den Wertezuweisungen als Satz 1 anzusprechen ist. Der Datensatz mit dem Kennzeichen 9 wird entsprechend als Satz 2 und der mit dem Kennzeichen 1 als Satz 3 angesehen.

Direkt hintereinander liegende numerische Satz-Kennzeichen müssen nicht einzeln aufgeführt werden:

Statt	`RC=D(1..1)1,2,3,4,9,8,7,6`
genügt	`RC=D(1..1)1..4,9..6`

SEP = | 's' | **TAB** | **NUMBER** | < , 't' >

Die Angabe SEP legt fest, dass die Datei im CSV-Format auszugeben ist.
In CSV-Dateien besitzen die einzelnen Datenfelder keine feste Position und Länge, sondern sind durch ein Separatorzeichen (meistens Komma, Semikolon oder Tabulatorzeichen) voneinander getrennt. Der Zugriff auf ein Feld erfolgt über die Reihenfolgenummer 1 ... innerhalb des Datensatzes. Die Länge der Felder ist variabel und vom Datenwert abhängig.

Die CSV-Dateien können zum Beispiel zur Ein- und Ausgabe von externen Dateien an Programme wie Microsoft EXCEL verwendet werden.

's' Hier ist ein beliebiges Zeichen in Hochkommas anzugeben, das als Separatorzeichen zwischen den Feldern dient. Die Zeichen ' und # sind doppelt einzugeben.

TAB Die Angabe TAB macht das Tabulatorzeichen zum Separatorzeichen.

NUMBER Die Angabe NUMBER macht das Nummernzeichen # zum Separatorzeichen.

' t ' Hier kann ein beliebiges Zeichen in Hochkommas angegeben werden, das als Texterkennungszeichen in alphanumerischen Werten dient (meistens " oder '). Dieses Zeichen kann am Anfang und Ende eines Textwertes für AK- und AS-Felder stehen. Damit wird es möglich, das Separatorzeichen auch innerhalb eines alphanumerischen Wertes zu verwenden. Die Zeichen ' und # sind doppelt einzugeben.

INTEGER = | **NORMAL** | **REVERSE** |

Durch diese Angabe wird festgelegt, wie die Dualzahlen der externen Datenfelder vom Typ F, G und I in der Ausgabedatei zu speichern sind.
Fehlt die Angabe INTEGER, so wird dafür INTEGER=NORMAL angenommen.

NORMAL Mit dieser Angabe werden die höherwertigen Bits in die linken Bytes gestellt.

REVERSE Mit dieser Angabe werden die höherwertigen Bits in die rechten Bytes gestellt.

CSTD = | **B**
I
K
L
M
U < , label >
UM < , label > |

Diese Angabe legt zu den Dateiformaten AUTOASCII, AUTOANSI, AUTOEBCDIC, AUTOCOLBIN, AUTOQUANTUM, SPSS und SPSSUTF8 den Typ der externen Datenfelder für die C-Variablen fest. Ohne die Angabe CSTD verwendet das Programm das M-Format.

Abhängig vom Dateiformat sind folgende Angaben möglich:

SPSS SPSSUTF8	M, U und UM M erzeugt kategoriale SPSS-Variable mit den Werten 1, 2, 3 ... und U dichotome mit den Werten 0 und 1. UM macht die Ausgabe einer C-Variablen abhängig von der Anzahl ihrer Nennungen: Mit Mehrfachnennungen entstehen dichotome und ohne Mehrfachnennungen kategoriale SPSS-Variable.
AUTOCOLBIN	I, K, L, M und U
AUTOQUANTUM	K, L, M und U
Sonstige Dateien mit SEP	M und U
Sonstige Dateien ohne SEP	I, K, L, M und U

label Diese Angabe legt die Labeltexte für U-Felder der SPSS-Ausgabe fest.
Für eine Variable mit dem Variablentext *Geschlecht* und dem Merkmalstext *Männer* gilt:

label	**Variable Label**	**Value Label**
I	Männer	1 = Geschlecht
IV	Männer | Geschlecht	1 = gewählt 0 = nicht gewählt
VI	Geschlecht | Männer	1 = gewählt 0 = nicht gewählt
V	Geschlecht	1 = Männer

Ohne diese Angabe wird der Labeltyp I verwendet. Die Value-Label der Labeltypen IV und VI lassen sich durch beliebige andere Texte ersetzen. Dazu dienen hinter dem Statement DEFS die Folgestatements QUOTED für g*ewählt* und NOTQUOTED für *nicht gewählt.*
Für einzelne Variable ist dies auch mit der Angabe *s = 'valuelabel'* in den Folgestatements von OUTPUT-EXT möglich.

In den Folgestatements von OUTPUT-EXT lassen sich einzelnen Variablen von CSTD abweichende Feldtypen zuweisen.

In K-Feldern werden nur die Positionen 1 ... 10 verwendet.

In den Datensätzen der externen Datei wird Platz für alle vorhandenen Datenwerte reserviert einschließlich der Mehrfachnennungen. Dabei werden zunächst nur die in den Daten vorhandenen Merkmale berücksichtigt. Die zusätzliche Angabe VSTD in OUTPUT-EXT reserviert dagegen Platz für alle definierten Merkmale der C-Variablen.

In den Folgestatements lässt sich für einzelne Variable eine feste Anzahl auszugebender Nennungen vorgeben.

DSTD = < **LB** | **LZ** > <, '.' | ',' | ' ' > <,n>

Diese Angabe verändert die Ausgabe numerischer Werte in die D- DS- und DK-Felder der externen Datei. Ohne DSTD bleiben die Nachkommastellen der Eingabewerte erhalten und werden mit Dezimalzeichen ausgegeben. In D-Feldern erscheinen die Zahlen mit führenden Nullen.

LB Hiermit werden die Zahlen in D-Feldern mit führenden Leerzeichen ausgegeben.
Diese Angabe wirkt auf alle auszugebenden D-Felder in den Folgestatements von OUTPUT-EXT und auf die Angaben ID = (...) in OUTPUT-EXT selbst.
Die Satz-Kennzeichen RC = (...) werden dagegen stets mit führenden Nullen ausgegeben.

LZ Diese Angabe gibt die Zahlen in D-Feldern mit führenden Nullen aus.

`'.' ',' ''` Diese Angabe entscheidet über das Dezimalzeichen Punkt oder Komma, mit dem Nachkommastellen ausgegeben werden. Mit `''` werden die Nachkommastellen ohne Dezimalzeichen ausgegeben.

n Die Nachkommastellen von numerischen Werten werden normalerweise unverändert in die D-, DS- und DK-Felder übernommen. Hier lässt sich eine davon abweichende Anzahl 0 ... 17 von Kommastellen für alle Wertezuweisungen in diese Felder vorgeben.
Nachkommastellen in einzelnen Wertezuweisungen haben Vorrang vor der DSTD-Angabe; folgendes Statement gibt unabhängig von DSTD stets zwei Nachkommastellen aus:

```
-D1(10,5)2=N1
```

NSTD = **D** | **F** | **G** | **P**

Diese Angabe legt für die Dateiformate AUTOANSI, AUTOASCII, AUTOEBCDIC, AUTOCOLBIN, AUTOQUANTUM, SPSS, SPSSUTF8 und TRIPLES den Typ der externen Datenfelder für die N-Variablen fest. (siehe Abschnitt *Externe Datenfelder*).
Ohne diese Angabe wird das D-Format verwendet.

In den Folgestatements von OUTPUT-EXT lassen sich für einzelne Variable von NSTD abweichende Formate wählen. Dabei sind folgende Angaben möglich:

Bei AUTOQUANTUM, SPSS, TRIPLES oder SEP:	D-Felder.
Bei AUTOCOLBIN:	D-, F- und G-Felder.
Sonst:	D-, F-, G- und P-Felder.

MSTD = NS

Diese Angabe beeinflusst die Ausgabe von Mehrfachnennungen in MS- und MK-Feldern.
Ohne MSTD=NS erscheinen Mehrfachnennungen mit Von-Bis-Abkürzungen in der Form:

```
... ;1 3-7 10; ...
```

Mit MSTD=NS werden alle Nennungen einzeln aufgeführt:

```
... ;1 3 4 5 6 7 10; ...
```

VSTD Ohne diese Angabe reserviert das Programm in den Formaten SPSS, SPSSUTF8, TRIPLES und AUTOASCII ... zu jeder Variablen in den Ausgabesätzen so viel Platz, wie die vorgefundenen Datenwerte benötigen.

Mit der Angabe VSTD erfolgt die Platzeinteilung nicht anhand der Datenwerte sondern nach der Variablendefinition: Die Datenfelder werden so gewählt, dass auch der größtmögliche Wert Platz findet. Damit wird der Aufbau der Ausgabedatei unabhängig von den zufälligen Eingabewerten.

Bei C-Variablen sorgt VSTD jedoch nur für Einfachnennungen. In den Folgestatements von OUTPUT-EXT ist für jede Variable mit Mehrfachnennungen die gewünschte Anzahl von Nennungen vorzugeben. Zum Beispiel reserviert das Folgestatement von OUTPUT-EXT

```
-C10   M=3
```

Platz für drei Nennungen der Variablen C10 im M-Format.

KEYLINE
KEYLINES Diese Angaben sind nur zusammen mit SEP und in XLSX-Dateien möglich. Sie stellen Schlüssel für die Wertezuweisungen durch AK-, DK-, MK- und UK-Felder in die ersten Datensätze .

Die Schlüssel der Datenfelder erscheinen durch diese Angaben nur in den ersten Datensätzen der Ausgabedatei und nicht mit Gleichheitszeichen vor den Datenwerten. Schlüssel und zugehörige Datenwerte stehen in derselben Spalte.

Eine Datei zum Statement

```
OUTPUT-EXT SEP=';' KEYLINE ID=DK(Fall) ...
```

könnte folgendermaßen beginnen:

```
Fall; Satz; Marke; bekannt
 001;    1;    17;       1
 001;    2;   112;       2
 002;    1;    79;       2
```

Werden mehrere Datensätze pro Fall ausgegeben, so erhält jeder Datensatz mit KEYLINES eine eigene Schlüsselzeile. Die Datei zum Statement

```
OUTPUT-EXT SEP';' KEYLINES ID=DK(Fall) RC=DK(Satz)1,2
```

könnte folgendermaßen beginnen:

```
Fall; Satz; Geschlecht; Alter
Fall; Satz;      Marke; bekannt
 001;   1;          2;      35
 001;   2;         17;       1
 002;   1;          1;      47
 002;   2;        112;       2
```

WORKSHEET = ' name **'**

Diese Angabe legt für XLSX-Dateien den Namen des Arbeitsblattes für die Daten fest. Bei fehlender Angabe wird als Name *Tabelle1* verwendet.

LIST = | **ON** | **OFF** |

Das Programm kann zu jeder Ausgabedatei OUTPUT-EXT einen Codeplan erstellen, der angibt, in welche Datensätze, Positionen und Formate die Variablen ausgegeben wurden.
Mit der Angabe LIST=ON wird ein solcher Codeplan erzeugt, mit LIST=OFF wird seine Ausgabe unterdrückt. Die Voreinstellung des Programms ist LIST=ON.

Folgestatements für ASCII … XLSX

Dies sind die Folgestatements für die Datenformate ASCII, ANSI, EBCDIC, BINARY1, BINARY2, COLBIN, QUANTUM und XLSX: Das Statement OUTPUT-EXT allein erstellt zunächst nur leere Ausgabesätze. Die Angaben COPY, ID und RC versehen diese Datensätze mit Werten. Die weitere Füllung mit Daten erfolgt durch Folgestatements, wobei Position und Länge der Felder vorzugeben sind.

Diese Folgestatements liegen direkt hinter OUTPUT-EXT und beginnen mit einem Strich – am Anfang der Zeile. Sie sind in der Reihenfolge einzugeben, in der sie ausgeführt werden sollen. Dabei befinden sich die Eingabedaten aus INPUT-EXT und die Variablen in dem Zustand, den sie nach allen Folgestatements von ADD-PROC, UPD-PROC, MOD-PROC und den WEIGHT-Statements besaßen. Siehe in den Abschnitten *Externe Datenfelder, Wertezuweisungen* und *Verarbeitung einzelner statistischer Fälle.*

Dateien mit der Angabe SEP sowie XLSX-Dateien verlangen die Feldtypen AS, DS, MS, US, AK, DK, MK oder UK. Alle anderen Dateien benötigen die Feldtypen A, D, F, G, P, B, I, K, L, M und U.

–A ... **=** Textwert
Es wird Text in den Datensatz gestellt.

–AK ... **=** Textwert
Es wird Text in den Datensatz gestellt, mit Schlüsselwort und Gleichheitszeichen davor.

–AS ...= Textwert
Es wird Text in den Datensatz.

–B ... **=** Merkmalswerte
Es werden Merkmalswerte als einzelne Bits in den Datensatz gestellt.

–D ... **=** numerischer Wert
Es wird ein numerischer Wert als Dezimalzahl in den Datensatz gestellt.

–DK ... **=** numerischer Wert
Es wird ein numerischer Wert als Dezimalzahl in den Datensatz gestellt, mit Schlüsselwort und Gleichheitszeichen davor.

–DS ... **=** numerischer Wert
Es wird ein numerischer Wert als Dezimalzahl in den Datensatz gestellt.

–F ... **=** numerischer Wert
Es wird eine Dualzahl mit Vorzeichen (Integer) in den Datensatz gestellt.

–G ... **=** numerischer Wert
Es wird eine Dualzahl ohne Vorzeichen (unsigned Integer) in den Datensatz gestellt.

–I ... **=** Merkmalswerte
Es werden Merkmalswerte als positive Dualzahlen in den Datensatz gestellt.

–K ... **=** Merkmalswerte
Es werden Merkmalswerte im Column-Binary-Format in den Datensatz gestellt.

–L ... **=** Merkmalswerte
Es werden Merkmalswerte 1, ... 9, 0, -, & in den Datensatz gestellt.

–M ... **=** Merkmalswerte
Es werden Merkmalswerte in Form von Dezimalzahlen in den Datensatz gestellt.

–MK ... **=** Merkmalswerte
Es werden Merkmalswerte in Form von Dezimalzahlen in den Datensatz gestellt , mit Schlüsselwort und Gleichheitszeichen davor.

–MS ... **=** Merkmalswerte
Es werden Merkmalswerte in Form von Dezimalzahlen in den Datensatz gestellt.

–P ... **=** numerischer Wert
Es wird ein numerischer Wert als gepackte Dezimalzahl in den Datensatz gestellt.

–U ... **=** Merkmalswerte
Es werden Merkmalswerte als Zeichen 0 und 1 in den Datensatz gestellt.

–UK ... **=** Merkmalswerte
Es werden Merkmalswerte als Zeichen 0 und 1 in den Datensatz gestellt, mit Schlüsselwort und Gleichheitszeichen davor.

–US ... = Merkmalswerte
Es werden Merkmalswerte als Zeichen 0 und 1 in den Datensatz gestellt.

Zur Ermittlung von Zwischenergebnissen können auch Variablen in den Folgestatements von OUTPUT-EXT neue Werte zugewiesen werden. Anders als unter ADD-PROC, MOD-PROC und UPD-PROC hat diese Zuweisung nur vorübergehende Wirkung: Sie gilt nur für die Folgestatements von OUTPUT-EXT. Liegt zum Beispiel hinter OUTPUT-EXT noch ein XTAB oder CODEBOOK, so werden dort die Werte der Variablen vor dieser Wertezuweisung verwendet. Siehe Abschnitt Wertezuweisungen .

–Cn = Merkmalswerte
Die Bedingungsvariable Cn erhält Merkmalswerte zugewiesen.

–Nn = numerischer Wert
Die numerische Variable Nn erhält einen Zahlenwert zugewiesen.

–Tn = Textwert
Die Textvariable Tn erhält einen Wert zugewiesen.

Wie unter ADD-PROC, UPD-PROC und MOD-PROC stehen die folgenden Statements zur Steuerung der Verarbeitung zur Verfügung:

–DO –EDO
Beginn und Ende einer Schleife: Die zwischen DO und EDO liegenden Statements werden mehrfach ausgeführt. Siehe DO-Statement.

–IF logischer Ausdruck **–EIF**
Die Verarbeitung der Statements zwischen IF und EIF hängt ab vom Wert des logischen Ausdrucks. Siehe IF-Statement.

–LANG Sprachschlüssel zur Auswahl von Variablentexten. Siehe LANG-Statement.

–PRT <Variable> <Text>
Dieses Statement druckt Variablenwerte und feste Texte aus. Siehe PRT-Statement.

–REPORT Ausgabe in Textdatei. Siehe REPORT-Statement.

–CLEAN logischer Ausdruck <**HEAD=**'text'>
Dieses Statement dient zur Bereinigung fehlerhafter Variablenwerte durch hinter CLEAN liegende Wertezuweisungen. Diese Wertezuweisungen werden nur ausgeführt, wenn der logische Ausdruck erfüllt ist. Im Zählprotokoll gibt eine CLEAN-Statistik eine Übersicht über die ausgeführten Korrekturen.

–TST logischer Ausdruck <Variable> <Text>
Dieses Statement prüft für die einzelnen statistischen Fälle, ob der logische Ausdruck erfüllt ist. Wenn nicht, wird der Text zusammen mit den Werten der angegebenen Variablen als Fehlermeldung ausgedruckt.
Siehe TST-Statement.

–END < **CASE** | **ALL** >
Dieses Statement beendet die Verarbeitung des laufenden Falles in OUTPUT-EXT.
Die hinter END stehenden Folgestatements werden nicht mehr ausgeführt.
Ohne weitere Angaben wird die Verarbeitung des laufenden Falles hinter den Folgestatements von OUTPUT-EXT fortgesetzt.
CASE beendet darüber hinaus die Verarbeitung des laufenden Falles durch alle noch folgenden Statements.
ALL bricht die gesamte Verarbeitung vorzeitig ab.

Beispiel: Externe Datei ausgeben

```
INPUT-EXT  FNAME=STUDIE12
           ASCII
           SIZE=80,EOL
           ID=D(2..2)
           ID1=D(5..9)
           RC=D(1..1)2,3,5...7

OUTPUT-EXT   FNAME=STUDIE12.NEU
             ANSI
             SIZE=100
             ID=D(1..1)
             ID1=D(2..6)
             RC=2
-D1(10..18)=D2(30..38)   BZ
-M1(20..26,2)=C1
-D1(27,9)=(N100+N101)/100
-A2(10..80)=A3(10..80)
```

Das Statement INPUT-EXT liest eine externe Eingabedatei mit dem Dateinamen STUDIE12 ein. Sie besitzt eine zweistufige Fall-Identifikation und fünf Eingabesätze pro Fall. Der erste davon wird an dem Satz-Kennzeichen 2 erkannt, der zweite am Kennzeichen 3 und die folgenden an den Kennzeichen 5 ... 7.

Die Angabe ASCII legt das Datenformat der Datei fest und SIZE verlangt 80 Bytes lange Datensätze. Wegen EOL muß jeder Satz durch ein Zeilenende-Zeichen abgeschlossen sein. Die Sätze dürfen kürzer als 80 Bytes sein, sie werden dann rechts mit Leerzeichen auf 80 Bytes aufgefüllt.

Das Statement OUTPUT-EXT erzeugt eine externe Ausgabedatei mit dem Dateinamen STUDIE12.NEU im ANSI-Format. RC=2 erzeugt zu jedem statistischen Fall zwei Datensätze. Wegen SIZE=100 besitzt jeder davon eine feste Satzlänge von 100 Bytes ohne Zeilenende-Zeichen.

Durch die Angabe von ID und ID1 werden die Fall-Identifikationen an den Anfang der beiden auszugebenden Datensätze gestellt.

Die Folgestatements von OUTPUT-EXT füllen die Datensätze mit Werten:

`-D1(10..18)=D2(30..38)   BZ`
Dieses Statement Überträgt numerische Werte aus den Positionen 30...38 der jeweils zweiten Eingabesätze aus INPUT-EXT in die Positionen 10...18 der ersten Ausgabesätze. Die Angabe BZ gibt Eingabewerte, die nur aus Leerzeichen bestehen, als Null aus.

`-M1(20..26,2) = C1`
Dieses Folgestatement gibt Merkmalswerte aus der Variablen C1 als zweistellige Zahlenwerte 00 ... 99 aus und stellt sie in ein M-Feld in den Bytes 20 bis 27 des zweiten Ausgabesatzes. Damit sind bis zu drei nebeneinander stehende Mehrfachnennungen möglich.

`-D1(27,9) =(N100+N101)/100`
Hier wird der Wert eines numerischen Ausdrucks ermittelt und das Ergebnis in die Positionen 27...36 des ersten Ausgabesatzes gestellt.

`-A2(10..80) = A3(10..80)`
Dieses Statement übertrgt einen Text, zum Beispiel aus einer offenen Frage, aus den Positionen 10...80 des jeweils dritten Eingabesatzes und stellt ihn in die Positionen 10...80 des zweiten Ausgabesatzes eines jeden Falles. Dabei wird der Text vom ASCII-Format ins ANSI-Format umgesetzt, sodass Umlaute korrekt in der ANSI-Datei erscheinen.

Beispiel: Column-Binary-Datei ergänzen

```
INPUT-EXT   FNAME=EINGABE
            COLBIN
            SIZE=80
            ID=D(1..5)
            RC=D(6..6)1..4
DEFS
-C100:12
-N100...125:2

ADD-PROC
-C100=K2(3,12)
-N100...125=D2(7,2)  INC=2

OUTPUT-EXT    FNAME=STUDIE83
              COPY
-K1(10,12)=C100
-D2(10..18)=N100 + ..125
```

Das Statement INPUT-EXT bewirkt, dass eine externe Eingabedatei mit dem Dateinamen EINGABE verarbeitet wird. Sie besitzt eine Fall-Identifikation und vier Datensätze pro Fall. Die Datei liegt im COLBIN-Format vor und besteht aus 80 Spalten langen Sätzen.

In den Folgestatements von DEFS werden die Variablen C100 sowie N100...N125 definiert. Diese erhalten in den Folgestatements von ADD-PROC Werte zugewiesen.

Das Statement OUTPUT-EXT erzeugt eine externe Ausgabedatei mit dem Dateinamen STUDIE83. Die Angabe COPY bewirkt, dass diese Ausgabedatei ebenfalls im COLBIN-Format erstellt wird und auch aus 4 Datensätzen pro Fall besteht. Jeder dieser Sätze ist wieder 80 Spalten lang.

Weiter sorgt COPY dafür, dass die Eingabesätze aus INPUT-EXT zunächst unverändert in OUTPUT-EXT übertragen werden.

Die Folgestatements zu OUTPUT-EXT füllen die externen Ausgabesätze zusätzlich mit Datenwerten. Dabei werden die durch COPY übertragenen Daten teilweise überschrieben. Das erste Folgestatement stellt die 12 Merkmale der Bedingungsvariablen C100 in die Spalte 10 des jeweils ersten Ausgabesatzes von OUTPUT-EXT.

Danach wird die Summe aus den Werten der Variablen N100 ... N125 ermittelt und in die Spalten 10...18 des jeweils zweiten Datensatzes von OUTPUT-EXT gestellt. Auch hierbei werden die vorherigen Werte der Spalten 10...18 überschrieben.

Folgestatements für AUTOASCII ... SPSS TRIPLES

Diese Folgestatements von OUTPUT-EXT dienen zur Füllung der externen Datensätze mit Datenwerten.
Sie gelten für die Dateiformate AUTOANSI, AUTOASCII, AUTOEBCDIC, AUTOCOLBIN, AUTOXLSX, SPSSUTF8, SPSS und TRIPLES. CNTA organisiert Position und Länge der Datenfelder automatisch.

Das Statement OUTPUT-EXT allein erzeugt zunächst nur leere Ausgabesätze. Die Angaben ID und RC stellen erste Werte in die Datensätze.

Zur Auswahl der Variablen, deren Werte in die externe Datei auszugeben sind, dienen neben den DO- und IF-Statements die folgenden Auswahlstatements hinter OUTPUT-EXT.
Ohne Auswahlstatements werden alle definierten Variablen ausgegeben, nicht jedoch die Textelemente.
Die Reihenfolge dieser Statements legt auch die Reihenfolge der Variablenwerte in der externen Datei fest.

–Cn < | **(**a **...** b**)** | **(**a **,** c**)** | **(**a**)** | > < | **B** | **I** < **=** m > | **L** < **=** m > | **M** < **=** m > | **K** | **U** < **,** label > | > < 'text' > <**VAR=**v> < **Z0=**z > < **MV=**n > < s **=** 'valuelabel' >

Dieses Statement wählt eine C-Variable für die Ausgabedatei FNAME aus. Zum Beispiel:

```
-C117
-C2000[25]    M=3
-C1...18
-C2[10]...20
-CA*
```

–Nn < | **D** < **=** z > | **F** < **=** z > | **G** < **=** z > | **P** < **=** z > | > < 'text' > < s **=** 'valuelabel' > <**VAR=**v> < **Z0=**z > < **MV=**n > < **DATE** >

Dieses Statement wählt eine N-Variable für die Ausgabedatei FNAME aus. Zum Beispiel:

```
-N117
-N2000[25]
-N1...18
-N2[10]...20
-NA*
```

–Tn < **A=** a > < 'text' > <**VAR=**v> < **Z0=**z >

Dieses Statement wählt eine T-Variable für die Ausgabedatei FNAME aus. Zum Beispiel:

```
-T117
-T2000[25]
-T1...18
-T2[10]...20
-TA*
```

–**(**n**)** Diese Angabe ist nur zu den Dateiformaten AUTOANSI ... möglich, nicht jedoch bei SPSS, SPSSUTF8 und TRIPLES. Sie stellt den Text des Textelements (n) lediglich in die Codeplandatei VNAME und in die Beschreibung der Ausgabedatei, jedoch nicht in die Datendatei.
Zum Beispiel sorgt

```
-(117)
```

für die Ausgabe des Textelements (117) und

```
-(0)...9999
```

gibt alle zwischen (0) und (9999) definierten Textelemente aus.

B I L M K U

Durch eine solche Angabe kann für die ausgewählte C-Variable ein von CSTD im Statement OUTPUT-EXT abweichendes Format vorgegeben werden. Sie legt fest, in welchem Format die Merkmale aus C-Variablen in die externe Datei übernommen werden sollen.
Siehe auch Abschnitt *Externe Datenfelder*.
Abhängig vom Dateiformat sind folgende Angaben möglich:
SPSS und SPSSUTF8: Die Format M für kategoriale Variable und U für dichotome.
TRRIPLES: Das Format M.
AUTOASCII und AUTOEBCDIC: Alle diese Formate.
AUTOCOLBIN und AUTOQUANTUM: Nur die Formate I, K, L, M und U.

D F G P Durch eine solche Angabe wird für die N-Variable dieses Statements ein von der Angabe NSTD im Statement OUTPUT-EXT abweichendes Format bestimmt.
Abhängig vom Dateiformat sind folgende Angaben möglich:

AUTOASCII und AUTOEBCDIC: Alle Formate D, F, G und P.
SPSS, SPSSUTF8: Nur das Format D.
AUTOCOLBIN und AUTOQUANTUM: Nur die Formate D, F und G.

Zum Beispiel wählt das Statement

```
-N1...10  F
```

die Variablen N1 bis N10 zur Ausgabe in die Datei OUTPUT-EXT aus und speichert ihre Werte dort im F-Format.

A= a Diese Angabe weist den Texten der T-Variablen in den Datensätzen von OUTPUT-EXT Platz für a = 1 ... 4 000 Zeichen zu.
Texte mit mehr als *a* Zeichen werden bei der Datenausgabe abgeschnitten.
Ohne die Angabe A wird Platz für den längsten in den Daten enthaltenen Text reserviert.

m Dieser Wert bestimmt die Anzahl von Nennungen der C-Variablen, die in jedem Datensatz von OUTPUT-EXT Platz finden sollen. Im I-Format sind das bis zu 200 Nennungen, im L-Format bis zu 12 und in den sonstigen Formaten bis zu 9 999 Nennungen.
C-Variable mit mehr als *m* Nennungen führen zu Fehlermeldungen bei der Datenausgabe.
Ohne die Angabe *m* wird Platz für alle in den Daten enthaltenen Merkmale reserviert.

z Dieser Angabe weist den Werten der N-Variablen Platz für mindestens z = 1 ... 16 Ziffern zu.
N-Variable mit Werten über *z* Ziffern führen zu Fehlermeldungen bei der Datenausgabe.
Ohne die Angabe *z* wird Platz für den größten in den Daten enthaltenen Wert reserviert.

'text' Die hier eingegebenen Textzeilen ersetzen für OUTPUT-EXT den Variablentext der auszugebenden Variablen. (Siehe Abschnitt Textzeilen).
In den Formaten AUTOASCII, AUTOEBCDIC, AUTOCOLBIN und AUTOQUANTUM dient dieser Text zur Beschreibung der Datenfelder in der ausgedruckten Dateibeschreibung.
Im Format SPSS und SPSSUTF8 liefert diese Angabe den Text für VARIABLE LABELS und im Format TRIPLES für <LABEL> in der Datei VNAME.

(a **...** b **)** Diese Angabe stellt nur die zwischen den Merkmalsnummern *a* und *b* liegenden Werte in die Ausgabedatei.

(a **,** c **)** Diese Angabe überträgt *c* Merkmale, beginnend bei der Merkmalsnummer *a* in die Ausgabedatei.

(a **)** Diese Angabe überträgt alle Merkmale zwischen der Nummer *a* und dem letzten definierten Merkmal der Variablen in die Ausgabedatei.

label Diese Angabe legt die Label-Texte für U-Felder der SPSS-Ausgabe abweichend von der Angabe CSTD im Statement OUTPUT-EXT fest:
Für eine Variable mit dem Variablentext *Geschlecht* und dem Merkmalstext *Männer* gilt:

label	Variable Label	Value Label
I	Männer	1 = Geschlecht
IV	Männer | Geschlecht	1 = gewählt 0 = nicht gewählt
VI	Geschlecht | Männer	1 = gewählt 0 = nicht gewählt
V	Geschlecht	1 = Männer

Ohne diese Angabe wird der Labeltyp I verwendet. Die Value Label der Labeltypen IV und VI lassen sich durch beliebige andere Texte ersetzen. Dazu dienen hinter dem Statement DEFS die Folgestatements QUOTED für g*ewählt* und NOTQUOTED für *nicht gewählt.*
Für einzelne Variable ist dies auch mit der Angabe *s = 'valuelabel'* in diesem Statement möglich.

VAR= In den Ausgabedateien und Codeplänen verwendet CNT zunächst den Namen der CNT-Variablen zur Beschreibung der Variablenwerte.
Mit der Angabe VAR lassen sich davon abweichende Namen vorgeben. Zum Beispiel:

```
-C125 VAR=Frage7
```

Wird zum Beispiel eine C-Variable für SPSS mit Mehrfachnennungen ausgegeben, so ergänzt CNTA für jede vorgesehene Nennung den Variablennamen um eine Nummer 1 ....
Zu dem Folgestatement

```
-C125 M=3
```

erstellt CNTA die drei SPSS-Variablen: C125_1, C125_2 und C125_3.
Aus dem Folgestatement

```
-C125 VAR=FRAGE7 M=3
```

entstehen die drei SPSS-Variablen: FRAGE7_1, FRAGE7_2 und FRAGE7_3

Z0=z In den Dateiformaten SPSS und SPSSUTF8 wird der Wert Z0 durch den hier vorgegebenen Wert *z* ersetzt. Bei C-Variablen ist diese Angabe nur im M-Format möglich; anstelle von Z0 erscheint dann die beliebige Zahl *z* in der SPSS-Datei. Für N-Variable wird anstelle von Z0 die beliebige Zahl *z* ausgegeben und für T-Variable mit leerem Text die in Hochkommas eingeschlossene Zeichenfolge *z*.
Zum Beispiel:

```
-C125 Z0=-999
-T10  Z0='keine Angabe'
```

MV=n In den Dateiformaten SPSS und SPSSUTF8 lassen sich hier für N- und C-Variable Missing-Values deklarieren. Es sind bis zu drei, durch Kommas getrennte, Einzelwerte möglich aber auch eine Von-Bis-Angabe mit einem weiteren Einzelwert.
Zum Beispiel:

```
-C125 MV=9
-N10  MV=0,3
-N11  MV=1...5
-N12  MV=5...7,12
```

s **=** 'valuelabel'
Dieser Text aus 1 ... 127 Bytes erstellt für die SPSS-Ausgabe die Value Label.
Bei N-Variablen zum Beispiel:

```
-N10 -1.5='klein'
        0='mittel'
      +1.5='groß'
```

Bei der Ausgabe von C-Variablen für SPSS mit U-Feldern und den Label-Angaben VI oder IV lassen sich hiermit die Standardtexte *gewählt* und *nicht gewählt* ersetzt. Zum Beispiel:

```
-C10  U,VI  0='not quoted' 1='quoted'
```

DATE Bei der Ausgabe von N-Variablen in SPSS-Dateien sorgt diese Angabe dafür, dass die daraus erstellte SPSS-Variable ihre Werte als Datum anzeigt. Dazu muss die N-Variable das Datum in der Form JJJJ MM TT enthalten. Es darf nicht vor dem 1. 1. 1970 liegen. Zum Beispiel:

```
-N22 DATE
```

Mit den üblichen Abkürzungsregeln können in einem Statement auch mehrere Variable zur Ausgabe auf OUTPUT-EXT ausgewählt werden. Zum Beispiel:

```
OUTPUT-EXT
-C1...17
-N25[3]...50
-(0)...99999
```

Das erste der Folgestatements wählt alle zwischen C1 und C17 definierten Variablen aus, das zweite alle Variablen zwischen N25[3] und N25[50] und das dritte alle überhaupt definierten Textelemente.

Innerhalb der Folgestatements sind auch –DO und –EDO zulässig, nicht dagegen –IF, –EIF, –PRT und –TST.

Beispiel: ASCII-Datei automatisch erstellen

```
INPUT-INT     FNAME=berlin.int

OUTPUT-EXT  AUTOASCII
            FNAME=output.ext
            HEAD='Projekt Berlin'
            VNAME=codeplan.csv
-C10
-C2030   VAR=Frage3   'Schulabschluss'
-N20
-T70
-(70)
```

Mit dem Statement INPUT-INT wird die interne Datei *berlin.int* eingelesen.
Diese enthält einige Variable, deren Werte in eine externe ASCII-Datei auszugeben sind.

Im Statement OUTPUT-EXT erzeugt AUTOASCII die externe ASCII-Datei, deren Aufbau CNTA automatisch erzeugt. Es sind lediglich in den Folgestatements von OUTPUT-EXT die auszugebenden Variablen in der gewünchten Reihenfolge aufzuführen.

Das Programm stellt stets einen Codeplan in die Ergebnisliste. Diese könnte hier so aussehen:

Projekt Berlin

Länge pro Datensatz 33 EOL
Anzahl Datensätze pro Fall 1

	Typ	Satz	Position	Länge	
C10	M	1	1	1	
					1 Männer
					2 Frauen
Frage3	M	1	2	2,1	Schulabschluss
					1 Volksschule, Hauptschule
					2 Weiterführende Schule ohne Abitur
					3 Abitur
					4 Studium ohne Abschluss
					5 Studium mit Abschluss
					6 kein Abschluss
N20	D	1	4	2	Alter
T70	A	1	6	28	Land des Hauptwohnsitzes
(70)					Frage 12: Wenn Sie im Ausland leben: Bitte geben Sie den Namen des Landes ein

Die Überschrift dieser Liste stammt aus der Angabe HEAD in OUTPUT-EXT.

In den Folgestatements von OUTPUT-EXT befinden sich zusätzliche Angaben zur Variablen C2030:
VAR=Frage3 bewirkt, dass der Text *Frage3* anstelle des Variablennamens im Codeplan erscheint und *‚Schulabschluss'* ersetzt den Variablentext von C2030.

Die Angabe *VNAME=berlin.csv* erzeugt eine CSV-Datei mit diesem Codeplan, die sich zum Beispiel mit Excel anzeigen und bearbeiten lässt.

Beispiel: Datenübergabe an SPSS

```
INPUT-INT    FNAME=berlin.int
OUTPUT-EXT   SPSS
             FNAME=berlin.dat
             VNAME=berlin.sps
             HEAD='Projekt Berlin'
             ID=A(1-4)
-C10
-C2030   VAR=Frage3   'Schulabschluss'
-N20
-T70
-(70)
```

Mit dem Statement INPUT-INT wird die interne Datei *berlin.int* eingelesen.
Diese enthält einige Variable, deren Werte, Definitionen und Texte so ausgegeben werden sollen, dass das Programm SPSS sie direkt verarbeiten kann.

Im Statement OUTPUT-EXT bereitet die Angabe SPSS die Variablenwerte für SPSS auf und stellt sie in die externe Datei *berlin.dat.* Die Organisation dieser Datei wird von CNTA automatisch durchgeführt. Es sind lediglich in den Folgestatements von OUTPUT-EXT die auszugebenden Variablen in der gewünschten Reihenfolge aufzuführen.

Die Angabe *VNAME = berlin.sps* erzeugt eine SPSS-Syntax-Datei mit den zur Datenübergabe erforderlichen SPSS-Statements. Diese könnten in unserem Beispiel folgendermaßen aussehen:

```
DATA LIST FILE='berlin.dat' RECORDS=1 /ID    1-4 (A)
  C10                     5
  Frage3                  6
  N20                     (T7,F2.0)
  T70                     9-36 (A).
TITLE='Projekt Berlin'.
VARIABLE LABELS
 Frage3                   'Schulabschluss'
 N20                      'Alter'
 T70                      'Land des Hauptwohnsitzes'.
VALUE LABELS
 /C10                        1 'Männer'
                             2 'Frauen'
 /Frage3                     1 'Volksschule, Hauptschule'
                             2 'Weiterführende Schule ohne Abitur'
                             3 'Abitur'
                             4 'Studium ohne Abschluss'
                             5 'Studium mit Abschluss'
                             6 'kein Abschluss'.
SAVE OUTFILE='berlin.SAV'.
```

Die Syntax-Datei beginnt mit dem SPSS-Statement DATA LIST mit dem Dateinamen aus FNAME geeigneten SPSS-Variablennamen und der Fall-Identifikation aus ID.
Darauf folgen die Variablendefinitionen für SPSS mit den Positionsangaben in den Datensätzen. Als Variablenname für SPSS wird der CNT-Variablenname verwendet, außer wenn mit VAR ein abweichender Name vorgegeben ist, wie hier mit *VAR=Frage3*.

Die Angabe HEAD in OUTPUT-EXT erzeugt das SPSS-Statement TITLE.

Als Variable_label werden die Variablentexte der CNT-Variablen verwendet und als Value-Label die Merkmalstexte, außer wenn wie hier mit *‚Schulabschluss'* ein abweichender Text vorgegeben wird.

Die SPSS-Datei endet mit dem SAVE-Statement für Daten-Datei *berlin.dat.* Dadurch genügt es zu Datenübernahme, in SPSS die Syntax-Datei *berlin.sps* aufzurufen und dort auszuführen.

OUTPUT-INT

Dieses Statement erzeugt einen internen Datenbestand, wie im Abschnitt Interne Dateien beschrieben. Er enthält:
Variablendefinitionen aus der laufenden Verarbeitung,
Variablentexte und Textelemente mit den Sprachangaben LANG ... LANG9,
Makros der laufenden Verarbeitung,
Variablengruppen für die Bildschirmanzeige in CNTW,
Daten aller verarbeiteten statistischen Fälle in Form von Variablenwerten.

Der Aufbau der Datei erfolgt völlig selbsttätig durch CNTA im Gegensatz zu den externen Dateien aus OUTPUT-EXT. Ein interner Datenbestand lässt sich daher viel leichter erstellen und auswerten. Er wird auch bei der Auswertung durch CNTA wesentlich schneller verarbeitet.
Die statistischen Fälle einer internen Datei werden stets nach aufsteigenden Fall-Identifikationen sortiert.
Jeder Fall erhält die aus INPUT-INT oder INPUT-EXT stammenden Fall-Identifikationen, falls nicht durch die Angabe IDSHORT oder IDi im Statement OUTPUT-INT andere Fall-Identifikationen vergeben werden.
Wenn keine Fall-Identifikationen vorhanden sind, werden die Ausgabesätze beginnend bei 1 durchnummeriert.

Die Datenspeicherung in einer internen Datei erfolgt normalerweise fallweise: Alle Variablenwerte eines statistischen Falles werden hintereinander in einen Datensatz gestellt. Die Angabe TRANS speichert die Daten sowohl fallweise als auch transponiert (gekippt): Die hintereinander stehenden Werte einer Variablen bilden einen Datensatz.

Interne Datenbestände werden von CNTA automatisch komprimiert, um Speicherplatz auf den Festplatten und Verarbeitungszeit zu sparen.

Mit Folgestatements von OUTPUT-INT lässt sich der interne Datenbestand auf ausgewählte Variable, Textelemente und Variablengruppen beschränken.

Zu OUTPUT-INT sind folgende Angaben möglich:

FNAME = dateiname

Hier ist der Name der internen Ausgabedatei anzugeben. Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
FNAME=C:\DATEN\STUDIE27.INT
FNAME='STUDIE27 INTERN'
FNAME=\\SERVER\DATEN\STUDIE27.INT
```

Eine bereits existierende Datei mit diesem Namen wird vom Programm vor der Ausgabe gelöscht.

Enthält *dateiname* keine Pfadangabe, so stellt CNTA die Datei in das Verzeichnis der Statementdatei und CNTW in das Verzeichnis der REP-Datei.
Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\STUDIE27.INT
```

DROP Mit der Angabe DROP werden alle Variablen und Textelemente übergangen, die in den Folgestatements aufgeführt sind. Es werden nur die nicht aufgeführten Variablen in die interne Datei übernommen.
Ohne die Angabe DROP werden nur die in den Folgestatements enthaltenen Variablen und Textelemente in die Ausgabedatei übernommen.
Folgestatements –Gs ... zur Selektion von Variablengruppen dürfen nicht zusammen mit DROP angegeben werden.
DROP bleibt wirkungslos, wenn keine Folgestatements vorliegen.

ID	**=**	**ID**
ID1		**ID1**
ID2		**ID2**
ID3		**ID3**
ID4		**ID4**
		Nn
		Tn

Diese Angaben vergeben die Fall-Identifikationen auf OUTPUT-INT abweichend vom Standardverfahren: Ohne sie übernehmen die statistischen Fälle auf OUTPUT-INT die aus INPUT-EXT oder INPUT-INT stammenden Fall-Identifikationen.

Durch die Angabe von

```
ID=ID1  ID1=N17  ID2=T25
```

wird zum Beispiel erreicht, dass die Werte von ID1 aus den Eingabedateien in OUTPUT-INT zur Fall-Identifikation ID werden. Die Werte der numerischen Variablen N17 werden in OUTPUT-INT zur Fall-Identifikation ID1 und die Werte der Textvariablen T25 schließlich zur Fall-Identifikation ID2.

Liegt in OUTPUT-INT auch nur eine einzige solche Angabe vor, werden alle automatischen ID-Zuordnungen außer Kraft gesetzt. Auch wenn im obigen Beispiel in INPUT-EXT etwa noch ID3 angegeben war, wird trotzdem keine Fall-Identifikation ID3 auf OUTPUT-INT ausgegeben.

Die statistischen Fälle werden in der Ausgabedatei aufsteigend nach diesen neuen Fall-Identifikationen sortiert. Dabei dürfen auch mehrere Fälle die gleichen Fall-Identifikationen besitzen. Sie werden ohne Fehlermeldung direkt hintereinander einsortiert.

IDSHORT

Durch diese Angabe können die Fall-Identifikationen auf OUTPUT-INT abweichend vom Standardverfahren gesetzt werden: Ohne diese Angaben erhalten die statistischen Fälle auf OUTPUT-INT automatisch die aus INPUT-EXT und INPUT-INT vorgegebenen Fall-Identifikationen. IDSHORT ist hier nur sinnvoll, wenn die beiden Statements INPUT-INT und INPUT-EXT gleichzeitig vorhanden sind und eines davon auch die Angabe IDSHORT enthält. Wird bei einem solchen Verarbeitungslauf eine interne Datei ausgegeben, so erhält diese normalerweise die vollständigen Fall-Identifikationen und nicht die durch IDSHORT verkürzten.
Durch die Angabe IDSHORT in OUTPUT-INT wird erreicht, dass die Datei mit verkürzten Fall-Identifikationen erstellt wird. Sie erhält dann die gleiche Anzahl von ID-Stufen wie die Eingabedatei, zu der IDSHORT angegeben wurde.

NAME='aaa'

Diese Angabe hat lediglich Kontrollfunktion. Sie bewirkt, dass im internen Datenbestand der Kontrollname *aaa* gespeichert wird. Dies kann zum Beispiel eine Studienbezeichnung oder ein Projektname sein, der maximal 20 Stellen lang ist.

Mit der entsprechenden Angabe NAME in INPUT-INT kann dann sichergestellt werden, dass der richtige Datenbestand verarbeitet wurde.
Außerdem wird dieser Kontrollname in verschiedenen Auswertungen zur Identifikation der internen Datei angedruckt.

TRANS
TRANSONLY

Diese Angaben speichern die Daten der statistischen Fälle transponiert (gekippt).
Normalerweise speichert CNTA alle Variablenwerte eines statistischen Falles direkt hintereinander in einen Datensatz. Bei der transponierten Speicherung stehen die Werte einer Variablen aus allen statistischen Fällen direkt hintereinander. Diese transponierte Speicherungsform führt zu wesentlich schnelleren Auswertungen, wenn aus einem großen Datenbestand nur ein kleiner Teil der darin enthaltenen Variablen zu verarbeiten ist. Siehe dazu auch den Abschnitt *Interne Dateien.*

Die Angabe TRANS speichert alle Variablenwerte neben der normalen Form ein zweites Mal transponiert in der Datei. Bei der Verarbeitung einer mit TRANS erstellten internen Datei erkennt CNTA selbsttätig, ob die Auswertung über die normalen oder die transponierten Daten schneller durchzuführen ist.

TRANSONLY speichert die Variablenwerte nur einmal in der transponierten Form.
Das reduziert die Größe der internen Datei, ist aber nur zu empfehlen, wenn niemals ein größerer Teil der Variablen gleichzeitig auszuwerten ist.

Beispiel: Ohne Folgestatements

```
INPUT-INT  FNAME=DATEI1.INT  NAME='STUDIE009-1'
DEFS
-C10:3 'Werbung gesehen'
         1='Karte 1'
         2='Karte 2'
         3='Karte 3'
-(100) 'Frage 85:'/
       'Haben Sie zu dieser Marke kürzlich eine Werbung gesehen ?'

OUTPUT-INT  FNAME=DATEI2.INT  NAME='STUDIE009-2'
```

Durch INPUT-INT wird zunächst eine bereits vorhandene interne Datei mit dem Dateinamen DATEI1.INT angefordert. Diese enthält Variablendefinitionen, Textelemente und die Daten zu den statistischen Fällen als Variablenwerte.
Diese interne Datei enthält auch den Kontrollnamen STUDIE009-1. Aufgrund dieser Angabe bei NAME führt CNTA eine Prüfung durch und verweigert die Verarbeitung einer internen Datei ohne diesen Kontrollnamen.

Durch DEFS und seine Folgestatements wird die neue Variable C10 definiert sowie das Textelement (100).

Falls die Variable C10 bereits in der Datei INPUT-INT definiert ist, wird eine Fehlermeldung ausgegeben und die Verarbeitung nicht durchgeführt: Bestehende Variable dürfen nicht mit einem Doppelpunkt dahinter neu definiert werden, wohl aber ohne Doppelpunkt neue Texte erhalten.
Wenn das Textelement (100) schon in INPUT-INT existiert, erhält es hier ohne Warnung oder Fehlermeldung einen neuen Text zugewiesen.

Mit dem Statement CODEBOOK kann ein Inhaltsverzeichnis der internen Datei erstellt werden, das alle darin enthaltenen Variablen mit ihren Texten aufführt.
Darüber hinaus wird eine Kurzauswertung aller Variablenwerte erstellt.

Das Statement OUTPUT-INT erzeugt nun eine neue interne Datei. Diese übernimmt alle Variablendefinitionen und Textelemente aus der Eingabedatei INPUT-INT. Weiter wird die Definition für C10 und (100) ausgegeben und die Werte aller statistischen Fälle zu diesen Variablen. Bei späteren Auswertungen müssen weder die Variablen neu definiert noch die Texte neu eingegeben werden.

Die Ausgabedatei erhält den Namen DATEI2.INT und den Kontrollnamen STUDIE009-2.

Folgestatements zur Auswahl von Variablen, Gruppen und Makros

Stehen hinter OUTPUT-INT keine Folgestatements, so werden alle Variable, Textelemente und Variablengruppen der laufenden Verarbeitung in die interne Datei ausgegeben, jedoch keine Makros.
Sollen nur einige Variable, Textelemente, Variablengruppen oder Makros in die neue Datei übernommen werden, so lassen diese durch die folgenden Statements auswählen. Als Folgestatements von OUTPUT-INT müssen sie immer mit einem waagerechten Strich am Zeilenanfang beginnen.

–Cn zur Auswahl einer Bedingungsvariablen Cn.

–Cm **..** n zur Auswahl aller Bedingungsvariablen von Cm bis Cn, die definiert sind.

–Nn zur Auswahl der numerischen Variablen Nn.

–Nm **..** n zur Auswahl aller numerischen Variablen von Nm bis Nn, die definiert sind.

–Tn zur Auswahl der Textvariablen Tn.

–Tm **..** n zur Auswahl aller Textvariablen von Tm bis Tn, die definiert sind.

–(n**)** zur Auswahl des Textelements (n).

–(m**) ..** n zur Auswahl aller Textelemente von (m) bis (n), die definiert sind.

–Gs zur Auswahl der Variablengruppe Gs. Alle darin enthaltenen Untergruppen, Gruppentexte und Variablen werden automatisch in die interne Datei ausgegeben. Für *s* sind die Werte 1 ... 255 möglich.

–Gs **..** z zur Auswahl aller zwischen Gs und Gz definierten Variablengruppen. Für s und z sind die Werte 1 ... 255 möglich. Alle dazugehörigen Untergruppen, Gruppentexte und Variablen werden automatisch in die interne Datei übernommen.

–LANGn Mit dieser Angabe werden nur die Texte mit dem Sprachschlüssel LANGn für die Variablen und Textelemente in die interne Datei ausgegeben. Ohne eine solche Angabe werden die Texte für alle Sprachschlüssel ausgegeben.

–LANGm **..** n Mit dieser Angabe werden nur die Texte mit dem Sprachschlüssel LANGm ... LANGn für die Variablen und Textelemente in die interne Datei ausgegeben. Ohne eine solche Angabe werden die Texte für alle Sprachschlüssel ausgegeben.

–MACROS Mit dieser Angabe werden alle Makros des Verarbeitungslaufs in die interne Datei ausgegeben, mit den Werten, die sie an diesem Punkt besitzen. Wird diese Datei später mit INPUT-INT zur Verarbeitung angefordert, so stehen diese Makros mit ihren Werten wieder zur Verfügung. Es lassen sich aber auch einzelne Makros für die Ausgabe auf OUTPUT-INT auswählen. So gibt das Statement

```
-MACROS #col3 #col7 #row2..#row12
```

nur die Makros *#col3, #col7* sowie alle zwischen *#row2* und *#row12* definierten Makros aus.

Die Texte für TOTAL, Z0 ... werden in jedem Fall in OUTPUT-INT übernommen.

Die Angabe DROP im Statement OUTPUT-INT entscheidet, ob die Variablen und Textelemente der Folgestatements in die Ausgabedatei übernommen werden sollen oder von der Übernahme auszuschließen sind. G-Statements zur Auswahl von Variablengruppen vertragen sich nicht mit DROP.

Variable mit Index können entweder ganz oder gar nicht in OUTPUT-INT übernommen werden. Folgestatements mit Indexangaben etwa in der Form -C7[3] oder gar -C7[3]...20 sind daher nicht zulässig.

Neben der Auswahl von mehreren Variablen in der Form

```
-C1...20
```

sind auch die Wildcard-Zeichen * und ? möglich. Das Statement

```
-NA??
```

gibt alle N-Variablen aus, die vier Zeichen lang sind und mit NA beginnen und

```
-NA*
```

alle N-Variablen, die mit NA beginnen.

Beispiel: Folgestatements mit Variablen und Textelementen

```
OUTPUT-INT    FNAME=DATEI2.INT
              ID=N25
-C100
-C200..810
-N200..520
-(300)..(350)
```

Hier werden wegen der Folgestatements nicht mehr alle Variablen und Textelemente ausgegeben sondern nur noch:
die Bedingungsvariable C100, sofern sie definiert ist.
alle zwischen C200 und C810 definierten Bedingungsvariablen. C200 und C810 selbst müssen dabei nicht definiert sein.
alle zwischen N200 und N520 definierten numerischen Variablen.
alle zwischen (300) und (350) definierten Textelemente.

Zu den statistischen Fällen werden auch nur die Werte der hier angegebenen Variablen in den internen Datenbestand übernommen.

Die Angabe ID=N25 im Statement OUTPUT-INT sorgt dafür, dass nicht die Fall-Identifikationen aus INPUT-INT oder INPUT-EXT in die statistischen Fälle der Ausgabedatei übernommen werden. Statt dessen werden dort die Werte der numerischen Variablen N25 als Fall-Identifikation ID verwendet und die statistischen Fälle automatisch aufsteigend danach sortiert.
Besitzt die Variable N25 für zwei statistische Fälle den gleichen Wert, so erhalten beide Fälle in der Ausgabedatei die gleiche Fall-Identifikation, dies ohne Fehlermeldung oder Warnung.

Beispiel: Folgestatements mit DROP

```
OUTPUT-INT  FNAME=DATEI2.INT
            DROP
-C200..300
-N0..30
-(10)..(20)
```

Nach der Angabe DROP im Statement OUTPUT-INT werden die in den Folgestatements aufgeführten Variablen und Textelemente nicht in die Ausgabedatei übernommen:
Alle zwischen C200 und C300 liegenden Bedingungsvariablen werden übergangen.
Alle zwischen N0 und N30 liegenden numerischen Variablen werden übergangen.
Alle zwischen (10) und (20) liegenden Textelemente werden übergangen.

Alle übrigen Variablen und Textelemente werden in die interne Datei OUTPUT-INT ausgegeben.

Beispiel: Folgestatements mit Variablengruppen

Mit den folgenden Statements werden Variablengruppen definiert und auf die interne Datei BEISPIEL4.INT ausgegeben. Diese Gruppen verbessern die Anzeige der Variablen in den Variablenfenstern von CNTW.

```
DEFS
...
-G1    'Statistik'            C1 .. 99
-G2    'Konsumgewohnheiten'   C1001 .. 1099   C2001   C300   N19
-G3    'Medienverhalten'
-G3.1     'Tageszeitungen'
-G3.1.1      'überregionale Zeitungen'    N9000[1] .. 99
-G3.1.2      'regionale Zeitungen'        N9000[100] .. 199
-G3.2     'Zeitschriften'            N9000[200] .. 299
-G3.5     'Rundfunk'                 N9000[300] .. 599
-G3.6     'TV'                       N9000[600] .. 999
...
OUTPUT-INT   FNAME = BEISPIEL4.INT
-G1
-G3.1
-G3.2 .. 6
```

Das Folgestatement –G1 von OUTPUT-INT stellt die Gruppe G1 mit dem Gruppentext *Statistik* in die interne Datei sowie alle zwischen C1 und C99 definierten Variablen.

Durch das Folgestatement –G3.1 werden die Variablengruppen G3.1, G3.1.1 und G3.1.2 ausgegeben. Da indizierte Variable nur vollständig in eine interne Datei übernommen werden können, sorgt –G3.1 für die Ausgabe aller Variablen N9000[i].
Die Variablengruppe G3 und ihr Gruppentext *Medienverhalten* gelangen nicht in die interne Datei.

Das Statement –G3.2 .. 6 gibt nur die Variablengruppen G3.2, G3.5 und G5.6 aus, da die Gruppen G3.3 und G3.4 nicht definiert sind.

Die nicht in den Gruppen enthaltenen Variablen werden nicht ausgegeben. Dafür wären zusätzliche Folgestatements erforderlich, zum Beispiel

```
-C1 .. 9999
-N1 .. 9999
```

Folgestatements für die Datenerfassung mit CNTD

Die statistischen Fälle einer internen Datei lassen sich mit dem Erfassungsprogramm CNTD anzeigen, bearbeiten und erfassen:
Die Dateneingabe erfolgt direkt in die Variablen.
Regeln zur Prüfung der Eingabewerte bei der Erfassung können vorgegeben werden.
Sprungbedingungen ermöglichen eine von den Eingabewerten abhängige Filtersteuerung.
Automatische und halbautomatische Vercodung offener Texte.
CNTD kann auch ohne CNTW-Dongle arbeiten. Dann ist jedoch nur die Neuerfassung für die hinter START-CNTD aufgeführten Variablen möglich.

Die Regeln zur Datenerfassung werden durch Folgestatements von OUTPUT-INT vorgegeben, die mit folgendem Statement einzuleiten sind:

–START–CNTD < | **ID** | **ID1** | **ID2** | **ID3** | **ID4** | **=** nr > <**AUTO**> < **LANG**m > < **STATUS =** | **N**n | **C**n | >

nr Anzahl Zeichen der ID-Stufe für die Erfassung. Numerische ID-Stufen erlauben bis zu 18 Zeichen, alphanumerische so viele, wie bei der Definition der ID-Stufe in OUTPUT-INT festgelegt.
Fehlt diese Angabe für eine ID-Stufe, so wird die größtmögliche Zeichenanzahl verwendet.

AUTO Mit dieser Angabe werden die Fall-Identifikationen in CNTD durch automatisches Weiterzählen zu Beginn jedes neuen Falles ermittelt. Bei mehrstufigen Fall-Identifikationen gilt dies nur für die unterste Stufe.

LANGm Mit der Angabe m wird die Sprache der Variablentexte ausgewählt, die in CNTD anzuzeigen sind. Damit ist es möglich, den Variablen unterschiedliche Texte für die Datenerfassung und Auswertung zu geben. Fehlt die Angabe LANG, so werden die Texte zu der niedrigsten vorhandenen Nummer *m* in CNTD angezeigt.

Cn Diese Variable nehmen den Erfassungsstatus der Fälle auf:
Nn Sie erhalten den Wert 1, wenn der Fall vollständig mit CNTD erfasst wurde und den Wert Z0, wenn die Erfassung vorzeitig beendet wurde.

Hinter START-CNTD werden mit weiteren Folgestatements die Variablen und ihre Prüfungsregeln für die Datenerfassung ausgewählt. Die hier angegebenen Variablen werden in die Datei OUTPUT-INT übernommen. Ihre Reihenfolge hinter START-CNTD legt auch die Reihenfolge für die Erfassung fest.

Auch vor dem Statement START-CNTD können Variable zur Ausgabe in die interne Datei angegeben werden. Für die Neuerfassung sind diese jedoch nur verfügbar, wenn sie noch einmal hinter START-CNTD erscheinen.

Die Variablen für die Datenerfassung sind in folgender Form auszuwählen:

–Cn < v < **…** b> > < m **:** z > < **(** p < **…** q> **)** > < **R** > < **DUP** >

Dieses Statement wählt eine C-Variable zur Erfassung aus.

v oder v **…** b
: Diese Angabe bestimmt die Anzahl zulässiger Nennungen der Variablen *Cn*:
v ist die Mindestanzahl Nennungen und *b* die Höchstanzahl.
Wird nur *v* angegeben, so sind genau *v* Merkmalsnummern einzugeben.
Fehlen beide Angaben, so ist nur ein Merkmal erlaubt.

m **:** z
: Mit dieser Angabe lässt sich die Erfassung der direkt hinter diesem Statement liegenden Variablen überspringen. *m* ist dabei eine Merkmalsnummer von *Cn* und *z* ein Sprungziel hinter dem laufenden Statement. Die Erfassung verzweigt zum Sprungziel *z*, wenn das Merkmal *m* für die Variable *Cn* eingegeben wurde. Siehe unten *Filter und Sprünge*.

(p oder p **…** q **)**
: Diese Angabe legt die für die Erfassung zulässigen Merkmalswerte fest. Bis zu 255 solcher Angaben sind möglich. Fehlen diese Angaben, so sind alle definierten Merkmalsnummern von Cn zulässig.

R
: Diese Angabe erzwingt die Eingabe von Werten (required).
Ohne *R* kann die Variable leer bleiben, auch wenn Mindestnennungen mit *v* vorgegeben sind.

DUP
: Nur im ersten erfassten Fall ist eine Eingabe möglich. In alle folgenden Fälle wird der Wert automatisch aus dem Vorgänger kopiert.

Zum Beispiel:

`-C900 4:C9`	Es kann nur ein Merkmal erfasst werden, keine Angabe ist möglich. Nach Eingabe von Merkmal 4 wird mit Variable C9 fortgefahren.
`-C80[3] 1..6`	1 bis 6 Nennungen oder keine Angabe sind zulässig.
`-C19 3`	3 Nennungen oder keine Angabe sind erlaubt.
`-C9 3 R`	Es sind genau 3 Merkmale einzugeben.
`-C1..5 R`	Alle zwischen C1 und C5 definierten Variablen verlangen genau eine Angabe.
`-C10[3]..4`	Die Variablen C[3] und C10[4] werden mit einer oder keiner Angabe erfasst.
`-C12 (3 5...7 17)`	Es lassen sich nur die Merkmalswerte 3, 5, 6, 7 und 17 eingeben.

–Cn **=** < v < **…** b> > . . .

Dieses Statement mit Gleichheitszeichen füllt eine C-Variable mit fest vorgegebenen Merkmalswerten.
Bis zu 255 Angaben sind möglich, entweder Einzelwerte *v* oder Von-Bis-Werte *v* ... *b*. Zum Beispiel:

```
-C10=1 5 7...9 24
```

–Nn < **(** p < **…** q> **)** > < **R** > < **DUP** >

Dieses Statement wählt eine N-Variable zur Erfassung aus.

(p oder p **…** q **)**
Diese Angabe legt die zulässigen Eingabewerte für die Variable *Nn* fest:
p gibt den kleinsten Wert an und *q* den größten. Wird nur *p* angegeben, so ist nur dieser Wert zulässig. Bis zu 255 solcher Angaben sind möglich. Fehlen diese Angaben, so sind Eingabewerte zwischen 1 und dem größtmöglichen Wert der Variablen erlaubt.

R Diese Angabe erzwingt die Eingabe von Werten (required).
Ohne *R* kann die Variable auch leer bleiben.

DUP Nur im ersten erfassten Fall ist eine Eingabe möglich. In alle folgenden Fälle wird der Wert automatisch aus dem Vorgänger kopiert.

In diesem Beispiel soll die Variable N10 unter DEFS mit -N10:3 definiert sein:

`-N10`	Werte zwischen 1 und 999 oder keine Eingabe sind möglich.
`-N10 -10...10`	Werte zwischen -10 und +10 oder keine Eingabe sind zulässig.
`-N10 987`	Nur der Wert 987 oder keine Eingabe sind erlaubt.
`-N10 R`	Es muss ein Wert zwischen 1 und 999 eingegeben werden.
`-N10 (3 5...7 -17)`	Es lassen sich nur die Werte 3, 5, 6, 7 und −17 eingeben.

–Nn **=** k

Dieses Statement mit Gleichheitszeichen füllt eine N-Variable mit einem konstanten Wert. Vorzeichen und Nachkommastellen sind dabei möglich. Zum Beispiel:

```
-N10[2] = -1.5
```

–Tn < v < **...** b> > < **R** > < **DUP** >

Dieses Statement wählt eine T-Variable zur Erfassung aus.

v oder v **...** b
: Diese Angabe legt die Länge der Eingabetexte für die Variable *Tn* fest.
v ist die Mindestlänge der Eingabewerte und *b* die Höchstlänge.
Ist nur *v* angegeben, so müssen alle Eingabewerte genau diese Länge besitzen. Ohne diese Angaben muss die Eingabelänge zwischen einem Zeichen und der definierten Länge der Variablen liegen.

R
: Diese Angabe erzwingt die Eingabe von Werten (required).
Ohne *R* kann die Variable auch leer bleiben. Zum Beispiel:

`-T101`	Eingaben bis zur Länge der Variablen sind möglich oder keine Eingabe.
`-T101 2..10`	Die Eingaben müssen aus 2 bis 10 Zeichen bestehen oder leer sein.
`-T101 5`	Genau fünf Zeichen oder keine Eingabe sind zulässig.
`-T101 4 R`	Eine Eingabe aus vier Zeichen ist zwingend erforderlich.

DUP
: Nur im ersten erfassten Fall ist eine Eingabe möglich. In alle folgenden Fälle wird der Wert automatisch aus dem Vorgänger kopiert.

–Tn **=** | '*text*' | **USER** |

Dieses Statement mit Gleichheitszeichen füllt eine T-Variable mit einem festen Wert *text* oder dem User-Namen, unter dem der Erfasser in Windows angemeldet ist. Zum Beispiel:

```
-T10='Studie132'
-T11=USER
```

DO-Statements

DO-Statements hinter START-CNTD dienen in erster Linie der Erfassung von Gitterfragen.
Eine solche Frage könnte im Fragebogen so aussehen:

Kinder im Haushalt

	männlich	weiblich	Alter
1. Kind	1	2	∴ ∴
2. Kind	1	2	∴ ∴
3. Kind	1	2	∴ ∴
4. Kind	1	2	∴ ∴
5. Kind	1	2	∴ ∴

Die folgenden CNTA-Statements erzeugen eine dazu passende Erfassungsdatei:

```
DEFS
-C1[5]:2  'Geschlecht' 1='männlich'  2='weiblich'
-N1[5]:2  'Alter'

OUTPUT-INT  ...
-START-CNTD
-DO <1>=1..5
-C1[<1>]
-N1[<1>]
-EDO
-C2
```

Die Dateneingabe zu dieser Frage erfolgt zeilenweise in der Reihenfolge
C1[1] N1[1] C1[2] N1[2] ...
Die Taste ÷ im numerischen Block beendet dabei die DO-Schleife vorzeitig und fährt mit der Variablen *C2* fort. Die Taste Enter beendet die laufende Zeile vorzeitig und fährt mit dem Anfang der nächsten Zeile fort.

Filter und Sprünge

Abhängig von bereits eingegebenen Werten, kann die Erfassung für weitere Variable übersprungen werden.
Dazu lassen sich Sprungbedingungen und Sprungziele in den Folgestatements hinter START-CNTD angeben.
Zum Beispiel:

```
OUTPUT-INT FNAME=1022.INT
-START-CNTD
-C1 1:C3                    Die Angabe 1:C3 bewirkt nach Eingabe des Merkmals 1 den
                            Sprung zur Variablen C3.
-GOTO END IF C1.Z0          Beendet die Erfassung des laufenden Falles, wenn keine Eingabe
                            zur Variablen C1 vorliegt.
-C2
-C3
```

GOTO-Statements

GOTO-Statements lassen sich überall in die Folgestatements von START-CNTD einfügen.
Sie beginnen am Zeilenanfang mit der Angabe:

`-GOTO` sprungziel

Ohne zusätzliche Angaben führt dieses Statement zu einem unbedingten Sprung zum *sprungziel* sobald die Erfassung dieses Statement erreicht.

Die GOTO-Statements können auch mit einer Bedingung versehen werden in der Form:

`-GOTO` sprungziel `IF` logischer Ausdruck

In diesem Fall wird nur dann beim *sprungziel* fortgefahren, wenn die Bedingung im *logischer Ausdruck* erfüllt ist. Hier sind alle Angaben möglich, die im Abschnitt *Logische Ausdrücke* beschrieben sind.
Zum Beispiel:

```
OUTPUT-INT FNAME=1022.INT
-START-CNTD
-C1 1:C3
-GOTO C3 IF C1.1|..4
-GOTO C5 IF C1.5|..8
-C2                         Diese Variable wird nur erfasst, wenn C1 nicht die Werte 1 ... 8 besitzt.
-GOTO C6
-C3                         Diese Variable wird nur erfasst, wenn C1 die Werte 1 ... 4 besitzt.
-GOTO C6
-C5                         Diese Variable wird nur erfasst, wenn C1 die Werte 5 ... 8 besitzt.
-C6                         Hier wird in jedem Fall fortgesetzt.
```

Sprungziele

Sprungziele müssen immer hinter der Zeile liegen, von der aus gesprungen werden soll.
Als Sprungziele sind folgende Angaben möglich:

- Name einer Variablen, die in den Folgestatements hinter der laufenden Zeile erscheint.
- END oder ENDE zum vorzeitigen Beenden der Eingabe eines statistischen Falles.
- Eine Sprungmarke, bestehend aus bis zu 10 beliebigen Zeichen ungleich END oder ENDE. Sprungmarken lassen sich überall in die Folgestatements einfügen, müssen aber am Zeilenanfang mit den Zeichen –> beginnen, in der Form –>frage4.

Zum Beispiel:

```
-C1   1:C3
-GOTO   frage4   IF  C1.1|..4
-C7
->frage4
-C4
```

Beispiel: Leere Erfassungsdatei für CNTD erstellen

```
DEFS
-C1:2  'Geschlecht'   1='Männer'  2='Frauen'
-N1:2  'Alter'
-T1:30 'Name des Befragten'

OUTPUT-INT   FNAME=demo.int
-START-CNTD   ID=4   AUTO
-T1   R
-N1   14 .. 99
-C1
```

Das Statement DEFS und seine Folgestatements definieren verschiedene Variable.

Das Statement OUTPUT-INT erzeugt die interne Datei für die Datenerfassung.
Da kein Statement INPUT-EXT und INPUT-INT vorliegt, enthält die Erfassungsdatei Variablendefinitionen und Erfassungsregeln aber keine statistischen Fälle.

Das Statement START-CNTD stellt Angaben zur Datenerfassung mit CNTD in die Ausgabedatei. Die Angabe ID=4 verlangt maximal vier Zeichen lange Fall-Identifikationen. AUTO sorgt bei der Erfassung für automatische Vergabe der Fall-Identifikation.

Die Folgestatements hinter START-CNTD wählen die Variablen T1, N1 und C1 für die Neuerfassung in der hier angegebenen Reihenfolge aus.
Hinter -T1 sorgt R dafür, dass der Name des Befragten zwingend einzugeben ist.
Zu -N1 sind nur Altersangaben zwischen 14 und 99 möglich.
Da bei N1 und C1 die Angabe R fehlt, können diese Variablen auch leer bleiben.

PAGE

Das Statement PAGE sorgt für die Ausgabe der Zählergebnisse im einfachen Textmodus. Besondere Schriften, Linien, Tonflächen, Logos usw. sind nicht möglich.

PAGE legt die Eigenschaften der Ausgabeseiten fest, wie Breite und Höhe, Blattüberschriften, Seitennummerierung usw. Diese Angaben gelten für alle folgenden Statements, bis sie durch weitere PAGE- oder PAGEP-Statements verändert werden.

Im Gegensatz zu PAGE erlaubt das Statement PAGEP eine anspruchsvollere Gestaltung der Zählergebnisse: Verschiedene Schriftarten und Schriftgrößen, Linien unterschiedlicher Stärke und Farbe sind möglich sowie Tonflächen und Logos. In einem Verarbeitungslauf dürfen PAGE- und PAGEP-Statements gemischt vorkommen.

PAGEP beschränkt die Ausgabe der Auswertungen auf das DIN-A4-Format. Größere Tabellen werden automatisch auf mehrere Seiten verteilt. PAGE kann hingegen sehr große Tabellen erstellen, ohne sie zu zerlegen. Dadurch ist es zum Beispiel möglich, umfangreiche Tabellen unzerlegt an Excel zu übergeben. Außerdem kann PAGE die Zählergebnisse als TXT-Datei ausgeben, die sich von Textprogrammen anzeigen und verändern lässt.

Fehlen beide Statements PAGE und PAGEP, so arbeitet das Programm im PAGEP-Format.

Folgende Angaben sind für PAGE möglich:

POS = n

n = 40 ... 32 767 legt die Anzahl Druckzeichen pro Zeile eines *Blattes* fest.
Ohne die Angabe POS wird mit der Eintragung POS = ... aus den Dateien CNTA.INI oder CNTW.INI gearbeitet. Fehlt diese ebenfalls, so werden 125 Zeichen pro Zeile verwendet.

LINES = n

n = 10 ... 1 000 000 legt die Anzahl der Zeilen pro Blatt fest. Die höchstmögliche Zeilenanzahl wird zusätzlich durch die Eintragung MAXWORK in den Dateien CNTA.INI oder CNTW.INI begrenzt: Die durch LINES und POS vorgegebene Anzahl von Zeichen pro Blatt darf nur die Hälfte des durch MAXWORK vorgegebenen Arbeitsbereichs verbrauchen.
Ohne die Angabe LINES wird mit der Eintragung LINES = ... aus den Dateien CNTA.INI oder CNTW.INI gearbeitet. Fehlt diese ebenfalls, so werden 65 Zeilen pro Blatt verwendet.

MARGINS = < **L** < **:** n> > < **, R** < **:** n> > < **, T** < **:** n> > < **, B** < **:** n> >
Diese Angabe legt die Randbreiten und Randlinien der Ausgabeseiten fest. Ohne MARGINS wird ohne Randbreiten. Zum Beispiel erzeugt

```
MARGINS=T:3, L:10
```

einen oberen Rand von 3 Zeilen Höhe und einen linken Rand von 10 Zeichenpositionen Breite.

- **L** Angaben zum linken Rand des Blattes.
- **R** Angaben zum rechten Rand des Blattes.
- **T** Angaben zum oberen Rand des Blattes.
- **B** Angaben zum unteren Rand des Blattes.
- n Hinter T und B ist dies die Anzahl Zeilen über und unter den Ausgabedaten.
  Hinter L und R ist es die Anzahl Zeichenpositionen rechts und links der Ausgabedaten.
  Die Texte HEADL, HEADC, HEADR, FOOTL, FOOTC, FOOTR werden *innerhalb* des oberen und unteren Randes gedruckt. Reicht der hier angegebene Rand nicht für diese Texte, so vergrößert das Programm die Ränder automatisch.

HOR = < r– > b , < r– > b ...

HOR unterteilt die Ausgabeseiten in mehrere nebeneinander liegende Spalten. Jede dieser Spalten wird bei der Ausgabe wie eine eigene Seite behandelt. Die Angabe

```
HOR=75,5-75
```

erzeugt auf jedem Blatt zwei nebeneinander liegende Spalten von 50 Zeichen Breite mit einem Abstand von 5 Zeichen.

```
HOR=*,6-*,6-*
```

unterteilt jedes Blatt in drei nebeneinander liegende Spalten gleicher Breite mit Abständen von 6 Zeichen dazwischen.

Die Kopf- und Fußtexte aus HEAD und FOOT werden nur pro Blatt ausgegeben und nicht pro Spalte.
Jedes Komma hinter HOR führt zu einer weiteren Spalte. Maximal 20 Spalten sind auf einem Blatt möglich.

r– Diese Angabe erzeugt einen Abstand zwischen der laufenden und der vorigen Spalte.
Werte zwischen 0 und 99 Zeichen sind möglich.

b Breite der Spalte ohne den Abstand davor. Werte zwischen 40 und 255 Zeichen sind möglich.
Anstelle dieser Breite kann auch ein Stern * angegeben werden. Dann wird die verfügbare Breite des Blattes gleichmäßig auf alle mit einem Stern versehenen Spalten verteilt.

VER = < r– > h , < r– > h ...

VER unterteilt die Ausgabeseiten in mehrere übereinander liegende Teilseiten. Jedes Komma hinter VER führt zu einer weiteren Teilseite auf dem Blatt. Maximal 20 Teilseiten sind auf einem Blatt möglich.
Die Kopf- und Fußtexte aus HEAD und FOOT werden nur einmal pro Blatt ausgegeben und nicht pro Teilseite.

r– Anzahl 0 ... 99 Leerzeilen oberhalb der Teilseite.

h Höhe der Teilseite ohne die Leerzeilen davor.
Werte zwischen 10 und 99 Zeilen sind möglich.
Anstelle dieser Höhe kann auch ein Stern * angegeben werden. Dann wird die verfügbare Höhe des Blattes gleichmäßig auf alle mit einem Stern versehenen Teilseiten verteilt.

Die Angabe

```
VER=60,4-60
```

teilt jedes Blatt in zwei übereinander liegende Teilseiten von 60 Zeilen Höhe.
Dazwischen werden vier Leerzeilen ausgegeben.

HEADL **HEADC** **HEADR**	**=**	**NO** <,' t '>

Diese Angaben erzeugen Kopftexte oberhalb der Auswertungsergebnisse auf jedem Blatt.
Ohne sie erscheint der Programmname, die laufende Versionsnummer und der Name des Anwenders als Blattüberschrift links oben.

HEADL Kopftext links oben auf dem Blatt.
HEADC Kopftext oben mittig auf dem Blatt.
HEADR Kopftext rechts oben auf dem Blatt.

NO Nach dieser Angabe wird der Kopftext nicht mehr ausgegeben.

't' Ein- oder mehrzeiliger Text, wie im Abschnitt Textzeilen beschrieben.
Die Angaben ~DATE~ und ~TIME~ stellen Datum und Uhrzeit des Auswertungslaufs an beliebiger Stelle in die Textzeilen. ~NR~ setzt die Seitennummer des laufenden Blattes ein.

Zum Beispiel:

```
PAGEP  HEADL='Studie 325'/'Welle 2'  HEADR='Seite ~NR~'
```

FOOTL **FOOTC** **FOOTR**	**=**	**NO** <,'t'>

Diese Angaben erzeugen Fußtexte unterhalb der Auswertungsergebnisse auf jedem Blatt.
Ohne die Angabe FOOTR erscheint rechts unten auf jedem Blatt die Seitennumerierung*: Seite 1*...

FOOTL Fußtext links unten auf dem Blatt.
FOOTC Fußtext unten mittig auf dem Blatt.
FOOTR Fußtext rechts unten auf dem Blatt.

NO Nach dieser Angabe wird der Fußtext nicht mehr ausgegeben.
Ohne die Angabe FOOTR erscheint rechts unten auf jedem Blatt die Seitennumerierung*: Seite 1*

't' Ein- oder mehrzeiliger Text, wie im Abschnitt Textzeilen beschrieben.
Die Angaben ~DATE~ und ~TIME~ stellen Datum und Uhrzeit des Auswertungslaufs an beliebiger Stelle in die Textzeilen. ~NR~ setzt die Seitennummer des laufenden Blattes ein.

Zum Beispiel:

```
PAGEP  FOOTL='Studie 325, ~DATE~ ~TIME~'  FOOTR='Seite ~NR~'
```

NR = n

Die nächste zu druckende Seitennummer *n* = 1 ... 10 000. Bei dieser beginnend werden alle folgenden Seiten fortlaufend numeriert. Die S*e*itennummern lassen sich mit der Angabe ~NR~ an beliebiger Stelle in die Randtexte HEAD... und FOOT...einfügen. Ohne die Angabe . FOOTR erscheint rechts unten auf jedem Blatt die Seitennummerierung *Seite 1*... Mit FOOTR=NO wird dieser Text entfernt.

ALTER	**=**	**YES** **NO**

Die Angabe ALTER=YES spiegelt die Kopftexte HEAD und Fußtexte FOOT auf Seiten mit gerader Seitennummer: HEADL und HEADR werden vertauscht, ebenso FOOTL und FOOTR. ALTER=NO macht diese Regel rückgängig.

LCHARS = 'x ...'

Dies ist eine Kette aus 1 ... 4 in Hochkommas eingeschlossene Zeichen, aus denen Linien und Tabellenränder gebildet werden. Die gewünschten Zeichen sind in folgender Reihenfolge einzugeben:

senkrechter Strich | für senkrechte Linien
waagerechter Strich – für Linien über und unter den Tabellen sowie unterhalb der Kopftexte
Kreuzungspunkt zweier Linien +
waagerechter Strich – für Linien innerhalb der Kopftexte

Werden weniger als 4 Zeichen angegeben, so wird das letzte Zeichen auch anstelle der noch fehlenden verwendet. Fehlt die Angabe LCHARS ganz, so wird mit

```
LCHARS='|-+-'
```

gearbeitet.

DCHAR = 'x'

x ist ein beliebiges Zeichen, das in den Auswertungen (XTAB, CODEBOOK, HOLECOUNT) als Dezimalzeichen (zum Beispiel Punkt oder Komma) verwendet wird. Bei der Bestellung von CNTA und CNTW kann festgelegt werden, welches Zeichen zu verwenden ist, wenn die Angabe DCHAR fehlt.

NCHAR = 'x'

x ist ein beliebiges Zeichen, das in allen Auswertungen für das Zählergebnis Null zu drucken ist. Bei der Bestellung von CNTA kann festgelegt werden, welches Zeichen zu verwenden ist, wenn die Angabe NCHAR fehlt. Meist wird dafür der Strich - verwendet.

PCHAR = 'x'

x ist ein beliebiges Zeichen, das als Lesehilfe in XTAB, CODEBOOK und INDEX zwischen den Zeilentexten und den Zahlenwerten gedruckt werden soll.
Wird hier ein Punkt angegeben, so entsteht eine punktierte Linie. Ein Leerzeichen verhindert die Ausgabe einer solchen Linie. Bei der Bestellung von CNTA kann festgelegt werden, welches Zeichen zu verwenden ist, wenn die Angabe PCHAR fehlt.

RCHAR = 'x'

x ist ein beliebiges Zeichen, das in allen Auswertungen für Werte zu drucken ist, die zwar von Null verschieden sind, aber für die Druckausgabe auf Null gerundet werden.
Bei der Bestellung von CNTA kann festgelegt werden, welches Zeichen zu verwenden ist, wenn die Angabe RCHAR fehlt. Meist wird dafür `0` verwendet.

TCHAR = 'x'

x ist ein beliebiges Zeichen, das in den Auswertungen die Tausenderstelle von Zahlen markieren soll. Zum Beispiel `TCHAR='.'` für 1.000 oder `TCHAR=' '` für 1 000.
Fehlt die Angabe TCHAR, so wird kein Tausenderzeichen ausgegeben: 1000.

CSV **DIF** **XLSX** **PDF** **TXT**	=	dateiname < **,KEEP** > **NO**

Diese Angaben stellen die Auswertungsergebnisse der dahinter stehenden Statements in den Formaten CSV, DIF, XLSX, PDF oder TXT in die Datei *dateiname*. Zum Beispiel erzeugt die Angabe

```
PDF=STUDIE27.PDF
```

die PDF-Datei STUDIE27.PDF.

Mehrere dieser Angaben sind gleichzeitig in einem Statement möglich. Sie bleiben gültig, bis sie in einem späteren Statement PAGE oder PAGEP durch neue Angaben ersetzt werden.

Bei der Ausgabe im XLSX- und PDF-Format kann das Programm automatisch Lesezeichen (Bookmarks) erzeugen. Dazu ist in der Auswertung mit dem Statement INDEX ein entsprechendes Inhaltverzeichnis zu erstellen.

Für CSV-Dateien legt die Angabe DCHAR im Statement PAGEP das Dezimalzeichen fest und SEP bestimmt das Separator- und das Texterkennungszeichen der Datei. Der Zeichencode ANSI, UTF8 usw folgt der Angabe CHARSET aus CNTA.INI oder CNTW.INI.

Bei XLSX -Dateien beginnt nach jeweils 100 Ausgabeseiten ein neues Arbeitsblatt. Die Arbeitsblätter werden vom Programm dabei mit Standardtexten versehen. Mit der Angabe WORKSHEET lassen sich dafür auch eigene Texte vorgeben.

Die Angabe TXT stellt die Zählergebnisse im TXT-Format in die angegebene Datei. Fehlt die Angabe TXT, so erscheinen die Auswertungsergebnisse in der Datei *cntlst* hinter der Statementliste. Diese Datei *cntlst* wird in das Verzeichnis der Eingabestatements ausgegeben.

dateiname Name der Datei, die die Auswertungsergebnisse der folgenden Statements aufnehmen soll. Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
TXT=C:\LISTEN\STUDIE95.TXT
CSV='STUDIE95 AUSWERTUNGEN.CSV'
PDF=\\SERVER\DATEN\STUDIE95.PDF
```

Enthält *dateiname* keine Pfadangabe, so stellt CNTA die Datei in das Verzeichnis der Statementdatei und CNTW in das Verzeichnis der REP-Datei.

Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
TXT=%STUDIEN%\NR%KUNDE%\STUDIE27.TXT
```

Eine bereits existierende Datei mit dem hier angegebenen Namen wird zu Beginn der Verarbeitung gelöscht. Erscheint der Dateiname in einem späteren PAGE- oder PAGEP-Statement noch einmal, so wird die bis dahin erstellte Datei fortgeschrieben und nicht gelöscht.

Enthält der Dateiname das Zeichen &, so wird für jede Tabelle eine neue Datei erstellt und das Zeichen & durch eine laufende Nummer 1, 2, ... ersetzt. Mehrere zusammenstehende &-Zeichen erzeugen dabei eine laufende Nummer mit einer entsprechenden Anzahl Ziffern. Die Angabe

```
PAGEP CSV=C:\TAB&&.CSV
```

stellt die erste Tabelle in die Datei *TAB01.CSV*, die zweite Tabelle in *TAB02.CSV*, die dritte in *TAB03.CSV* usw.

KEEP Diese Angabe ist nur für CSV, DIF und XLSX wirksam:
Normalerweise werden die Zählerwerte ohne die durch PRINT=' ... & ... ' hinzugefügten Zeichen in die Ausgabedatei gestellt; auch werden die Blattüberschriften HEAD... und Fußtexte FOOT... nicht ausgegeben.
Die Angabe KEEP übernimmt alle diese Angaben in die Ausgabedatei. Zum Beispiel:

```
PAGE CSV='STUDIE 105.CSV',KEEP
```

NO setzt die Angabe CSV ... PDF aus früheren PAGE- oder PAGEP-Statements außer Kraft.

SEP = | 's' | **TAB** | **NUMBER** | < , 't' >

Diese Angabe bestimmt das Separatorzeichen *s* und das Texterkennungszeichen *t* für die CSV-Ausgabe.
Mit TAB wird das Tabulatorzeichen zum Separatorzeichen und mit NUMBER das Nummernzeichen #.
Für *t* ist ein beliebiges Zeichen in Hochkommas anzugeben, das als Texterkennungszeichen vor und hinter alphanumerische Werte gestellt wird. Meistens wird dafür " oder ' verwendet. Die Zeichen ' und # sind doppelt einzugeben.

Fehlt die Angabe SEP so wird dafür

```
SEP=';','"'
```

angenommen.

Die Angabe

```
SEP=':'
```

macht den Doppelpunkt zum Separatorzeichen. Es werden keine Texterkennungszeichen ausgegeben.

WORKSHEET = | 'name' | **TITLE** | **BOTTOM** | **STUBHEAD** | **FILTER** | **NO**

Diese Angabe versieht die Arbeitsblätter der Excel-Ausgabe mit eigenen Namen zur Anzeige in den Excel-Tabs.
Alle folgenden Auswertungsseiten werden in dieses Arbeitsblatt gestellt bis hin zum nächsten PAGE- oder PAGEP-Statement mit einer neuen WORKSHEET-Angabe.

Ohne diese Angabe erhalten die Arbeitsblätter Standardnamen wie *Tabellen, Index ...* wobei nach jeweils 100 Seiten ein neues Arbeitsblatt beginnt

name Der hier eingegebene Text wird bei der Excel-Ausgabe als Name für die Arbeitsblätter verwendet. Sobald dieser Text wechselt, wird ein neues Arbeitsblatt begonnen.

TITLE Die TITLE-Texte aus XTAB dienen bei der Excel-Ausgabe als Namen der Arbeitsblätter.

BOTTOM Die BOTTOM-Texte aus XTAB dienen bei der Excel-Ausgabe als Namen der Arbeitsblätter.

STUBHEAD Die STUBHEAD-Texte aus XTAB dienen bei der Excel-Ausgabe als Namen der Arbeitsblätter.

FILTER Die FILTER-Texte aus XTAB dienen bei der Excel-Ausgabe als Namen der Arbeitsblätter.

NO Die WORKSHEET-Angaben aus früheren PAGE- oder PAGEP-Statements werden rückgängig gemacht. Das Programm versieht die Arbeitsblätter wieder mit den Standardnamen.

Beispiel

```
PAGE   LINES=10000
       POS=5000
       TXT=Tabelle.txt
       XLSX=Tabelle.xlsx

XTAB   ROW=TOTAL;C10;C20; ...
       COL=TOTAL;C2030; ...
```

Im Statement PAGE sorgen die Angaben. LINES und POS für Ausgabeseiten mit 10 000 Zeilen und 5 000 Zeichen pro Zeile, geeignet für sehr große Tabellen.

Mit TXT werden die Zählergebnisse in die Datei *Tabelle.txt* im TXT-Format ausgegeben, geeignet für die Anzeige und Änderung durch Textprogramme. Das Ergebnis könnte folgendermaßen aussehen:

TOTAL 1204

	TOTAL	Schulabschluß		
		Volksschule, Haauptschule	mittlere oder höhere Schule ohne Abitur	Abitur, Universität
TOTAL	1204	240	521	443
Geschlecht				
Männlich	1041	218	449	374
Weiblich	163	22	72	69
Alter				
bis 29 Jahre	226	48	82	96
30 bis 39 Jahre	341	53	157	131
40 bis 49 Jahre	316	61	144	111
50 bis 59 Jahre	219	53	97	69
60 Jahre und älter	102	25	41	36

Die Angabe XLSX gibt die Zählergebnisse zusätzlich .im XLSX-Format aus, geeignet für die Anzeige und Weiterverarbeitung durch Excel. Dies könnte folgendermaßen aussehen:

	A	B	C	D	E
1	TOTAL 1204				
2		TOTAL	Schulabschluß		
3			Volksschule, Haauptschule	mittlere oder höhere Schule ohne Abitur	Abitur, Universität
7	TOTAL	1204	240	521	443
8	Geschlecht				
9	Männlich	1041	218	449	374
10	Weiblich	163	22	72	69
11	Alter				
12	bis 29 Jahre	226	48	82	96
13	30 bis 39 Jahre	341	53	157	131
14	40 bis 49 Jahre	316	61	144	111
15	50 bis 59 Jahre	219	53	97	69
16	60 Jahre und älter	102	25	41	36

PAGEP

Das Statement PAGEP erlaubt sehr anspruchsvolle Aufbereitungen der Zählergebnisse.
Verschiedene Schriftarten und Schriftgrößen, Linien unterschiedlicher Stärke und Farbe sind möglich sowie Tonflächen und Logos. Die Ausgabeseiten sind allerdings auf das DIN-A4-Format beschränkt. Sehr große Tabellen werden jedoch automatisch auf mehrere Seiten verteilt.

PAGEP legt die Eigenschaften der Druckseiten fest, wie Breite und Höhe, Blattüberschriften, Seitennummerierung usw. Diese Angaben gelten für alle folgenden Statements, bis sie durch weitere PAGE- oder PAGEP-Statements verändert werden. In einem Verarbeitungslauf dürfen PAGE- und PAGEP-Statements gemischt vorkommen.

Im Gegensatz zu PAGEP liefert das Statement PAGE die Zählergebnisse im einfachen Textmodus. Dafür kann PAGE sehr große Tabellen erstellen, ohne sie zu zerlegen. Dadurch ist es zum Beispiel möglich, umfangreiche Tabellen unzerlegt an Excel zu übergeben. Außerdem kann PAGE die Zählergebnisse als TXT-Datei ausgeben, die sich von Textprogrammen anzeigen und verändern lässt.

Fehlen beide Statements PAGE und PAGEP, so arbeitet das Programm im PAGEP-Format.

Maßeinheiten

Größen, Breiten, Höhen und Positionen im Statement PAGEP werden von CNTA als Werte in Millimetern verstanden, zum Beispiel 3.2 als Schriftgröße von 3,2 mm. Dabei sind bis zu drei Nachkommastellen möglich; als Dezimalzeichen ist der Punkt zu verwenden.
Alle Maße können aber auch in anderen Einheiten angegeben werden. Dazu sind entsprechende Buchstaben direkt hinter den Zahlenwert zu stellen, wahlweise in Groß- oder Kleinschrift:

d = Didot Punkt,	zum Beispiel 10d	1d = 0.376mm
p = Pica Punkt,	zum Beispiel 9p	1p = 0.3528mm
c = Cicero,	zum Beispiel 2.5c	1c = 4.512mm

cm = Zentimeter, zum Beispiel 5.25cm
mm = Millimeter, zum Beispiel 24mm; dies ist gleichwertig mit der Angabe 24
1mm = 2.660d = 2.835p = 0.222c
Windows misst die Schriftgrößen in Pica Punkt.

Folgende Angaben sind im PAGEP-Statement möglich:

FORM = | **LANDSCAPE** | **PORTRAIT** | **MIXED** | **EXCEL**

Diese Angabe bestimmt die Größe und Orientierung der Druckseiten. Fehlt sie, so wird FORM=PORTRAIT angenommen.

LANDSCAPE DIN A4, Querformat

PORTRAIT DIN A4, Hochformat.

MIXED DIN A4, Blattüberschrift und Seitennummern im Hochformat, Daten und Tabellen im Querformat

EXCEL 250 cm Breite bei 150 cm Höhe,: Zur Ausgabe sehr großer Tabellen ohne Umbruch für EXCEL-Dateien.

HEADL **HEADC** **HEADR**	**=**	**NO** <,' t '> <, **FT**i > < , **K** >

Diese Angaben erzeugen Kopftexte oberhalb der Auswertungsergebnisse auf jedem Blatt.
Ohne sie erscheinen Name und laufende Versionsnummer des Programms mit dem Namen des Anwenders als Blattüberschrift links oben.

HEADL Kopftext links oben auf dem Blatt.
HEADC Kopftext oben mittig auf dem Blatt.
HEADR Kopftext rechts oben auf dem Blatt.

NO Nach dieser Angabe wird der Kopftext nicht mehr ausgegeben.

't' Ein- oder mehrzeiliger Text, wie im Abschnitt *Textzeilen* beschrieben.
Die Angaben ~DATE~ und ~TIME~ stellen Datum und Uhrzeit des Auswertungslaufs an beliebiger Stelle in die Textzeilen. ~NR~ setzt die Seitennummer des laufenden Blattes ein.

FTi Schrift FONTi aus dem Statement PAGEP für den Kopftext.
Fehlt diese Angabe, so wird die Schrift FONT4 verwendet.

K Werden mit der Angabe LOGOS oben rechts oder oben links Logos auf das Blatt gestellt, so verschieben diese Logos die Kopftexte. Mit der Angabe K überschreiben die Kopftexte die Logos..

Zum Beispiel:

```
PAGEP  HEADL='Studie 325'/'Welle 2'  HEADR='Seite ~NR~'
```

FOOTL **FOOTC** **FOOTR**	**=**	**NO** <,'t'> <, **FT**i > < , **K** >

Diese Angaben erzeugen Fußtexte unterhalb der Auswertungsergebnisse auf jedem Blatt.
Ohne die Angabe FOOTR erscheint rechts unten auf jedem Blatt die Seitennumerierung*: Seite 1*...

FOOTL Fußtext links unten auf dem Blatt.
FOOTC Fußtext unten mittig auf dem Blatt.
FOOTR Fußtext rechts unten auf dem Blatt.

NO Nach dieser Angabe wird der Fußtext nicht mehr ausgegeben.

't' Ein- oder mehrzeiliger Text, wie im Abschnitt *Textzeilen* beschrieben.
Die Angaben ~DATE~ und ~TIME~ stellen Datum und Uhrzeit des Auswertungslaufs an beliebiger Stelle in die Textzeilen. ~NR~ setzt die Seitennummer des laufenden Blattes ein.

FTi Schrift FONTi aus dem Statement PAGEP für den Fußtext.
Fehlt diese Angabe, so wird die Schrift FONT2 verwendet.

K Werden mit der Angabe LOGOS unten rechts oder unten links Logos auf das Blatt gestellt, so verschieben diese Logos die Fußtexte. Mit der Angabe K überschreiben die Fußtexte die Logos..

Zum Beispiel:

```
PAGEP   FOOTL='Studie 325, ~DATE~ ~TIME~'   FOOTR='Seite ~NR~'
```

ALTER = | **YES** | **NO** |

Die Angabe ALTER=YES spiegelt die Kopftexte HEAD und Fußtexte FOOT auf Seiten mit gerader Seitennummer: HEADL und HEADR werden vertauscht, ebenso FOOTL und FOOTR. ALTER=NO macht diese Regel rückgängig.

HOR = < | **Li–** | r– | > b , < | **Li–** | r– | > b ...

HOR unterteilt die Ausgabeseiten in mehrere nebeneinander liegende Spalten. Jede dieser Spalten wird bei der Ausgabe wie eine eigene Seite behandelt. Zum Beispiel

```
HOR=90,10-90
```

erzeugt auf jedem Blatt zwei nebeneinander liegende Spalten von 90mm Breite mit einem Abstand von 10mm.

```
HOR=*,L3-*,L3-*
```

unterteilt jedes Blatt in drei nebeneinander liegende Spalten gleicher Breite mit senkrechten Linien L3 dazwischen.

Die Kopf- und Fußtexte aus HEAD und FOOT werden nur pro Blatt ausgegeben und nicht pro Spalte.
Jedes Komma hinter HOR führt zu einer weiteren Spalte. Maximal 20 Spalten sind auf einem Blatt möglich.

Li– Senkrechte Linie L1 ... L16 zwischen der laufenden und der vorigen Spalte.

r– Diese Angabe erzeugt einen Abstand zwischen der laufenden und der vorigen Spalte.
Werte zwischen 0mm und 50mm sind in allen oben beschriebenen Maßeinheiten möglich.

b Breite der Spalte ohne den Abstand oder die Breite der Linie davor.
Werte zwischen 20mm und 200mm sind in allen oben beschriebenen Maßeinheiten möglich.
Anstelle dieser Breite kann auch ein Stern * angegeben werden. Dann wird die verfügbare Breite des Blattes gleichmäßig auf alle mit einem Stern versehenen Spalten verteilt.

VER = < | **Li–** | r– | > b , < | **Li–** | r– | > b ...

VER unterteilt die Ausgabeseiten in mehrere übereinander liegende Teilseiten. Jedes Komma hinter VER führt zu einer weiteren Teilseite auf dem Blatt. Maximal 20 Teilseiten sind auf einem Blatt möglich.
Die Kopf- und Fußtexte aus HEAD und FOOT werden nur einmal pro Blatt ausgegeben und nicht pro Teilseite.

Li– Waagerechte Linie L1 ... L16 zwischen der laufenden und der vorigen Teilseite.

r– Breite des Randes oberhalb der Seite in den oben beschriebenen Maßeinheiten.
Werte zwischen 0mm und 50mm sind in allen oben beschriebenen Maßeinheiten möglich.

h Höhe der Teilseite ohne den Rand oder die Linie davor.
Werte zwischen 10mm und 500mm sind in allen oben beschriebenen Maßeinheiten möglich.
Anstelle dieser Höhe kann auch ein Stern * angegeben werden. Dann wird die verfügbare Höhe des Blattes gleichmäßig auf alle mit einem Stern versehenen Seiten verteilt.

Die Angabe

```
VER=130,L3-130
```

teilt jedes Blatt in zwei übereinander liegende Teilseiten von 130mm Höhe. Dazwischen wird die waagerechte Linie L3 ausgegeben.

LOGOS = < **TL:** n > < **, TC:** n > < **, TR:** n >
< **, CL:** n > < **, CC:** n > < **, CR:** n >
< **, BL:** n > < **, BC:** n > < **, BR:** n >

Durch diese Angabe können Logos frei auf das Druckblatt gestellt werden. Sie erscheinen auf jeder durch PAGEP gedruckten Seite, wie auf einem vorgedruckten Formular.

n Dies ist die Nummer 1 ... 999 des Logos, das an der davor festgelegten Position ausgegeben werden soll. Dazu muss der Nummer *n* im Statement PAGEP mit der Angabe

```
LOGOn=dateiname
```

eine Bilddatei im JPEG-Format zugeordnet werden. Die Nummer n = 0 bewirkt, dass ein Logo aus einem früheren PAGEP Statement nicht mehr ausgegeben wird.

TL Das Logo n wird oben links ausgegeben.
TC Das Logo n wird oben in der Mitte ausgegeben.
TR Das Logo n wird oben rechts ausgegeben.
CL Das Logo n wird in der Mitte links ausgegeben.
CC Das Logo n wird in der Mitte des Blattes ausgegeben.
CR Das Logo n wird in der Mitte rechts ausgegeben.
BL Das Logo n wird unten links ausgegeben.
BC Das Logo n wird unten in der Mitte ausgegeben.
BR Das Logo n wird unten rechts ausgegeben.

Diese Logos verschieben normalerweise die Überschriften, Fußtexte, Seitennummern sowie Datum und Uhrzeit in den Randtexten. Durch zusätzliche Angabe von K zu HEAD, FOOT oder NR ist es möglich, diese Texte auch in die Logos hinein zu drucken.

Die Rand-Logos werden normalerweise in einem Abstand von 6 mm zur Blattgrenze platziert. Wird jedoch mit MARGINS ein Rand von weniger als 6 mm gefordert, so rücken die Logos entsprechend näher an die Blattgrenze.

Tabellen und andere Auswertungen weichen den Logos nicht aus. Damit lassen sich beispielsweise Logos in der Art von Wasserzeichen verwenden. Auch kann ein Logo rechts oben auf den Seiten in einem Bereich erscheinen, in den normalerweise keine Tabellen hineinreichen: Die Tabellen rutschen nicht nach unten, es wird kein zusätzlicher Platz für das Logo benötigt. Ist dieser Effekt nicht gewünscht, so lassen sich die Tabellen durch geeignete MARGIN-Angaben in PAGEP aus den Logo-Bereichen fernhalten.

MARGINS = < **L** < **:** b> < **: L**i> > < **, R** < **:** b> < **: L**i> > < **, T** < **:** b> < **: L**i> > < **, B** < **:** b> < **: L**i> >

Diese Angabe legt die Randbreiten und Randlinien der Ausgabeseiten fest. Ohne MARGINS wird eine Randbreite von 6 mm ohne Randlinien verwendet. Die Angabe ALTER=YES vertauscht auf Seiten mit geraden Seitennummern die rechten und linken Ränder miteinander. Zum Beispiel erzeugt

```
MARGINS=L:15:L3, R:15:L3
```

einen linken und einen rechten Rand von jeweils 15mm Breite, beide durch eine Linie L3 begrenzt.

L Angaben zum linken Rand des Blattes.
R Angaben zum rechten Rand des Blattes.
T Angaben zum oberen Rand des Blattes.
B Angaben zum unteren Rand des Blattes.
b Breite des Randes in den oben beschriebenen Maßeinheiten.
Dies ist der Abstand der Texte und Zählergebnisse von den Seitengrenzen.
Die Text HEADL, HEADC, HEADR, FOOTL, FOOTC, FOOTR werden *innerhalb* des oberen und unteren Randes gedruckt. Reicht der hier angegebene Rand nicht für diese Texte, so vergrößert das Programm die Ränder automatisch.
Fehlt diese Angabe, so wird als Breite 6mm angenommen.
Li Linie L1 ... L16, die an diesem Rand ausgegeben werden soll.

CSV **DIF** **XLSX** **PDF**	=	dateiname < **,KEEP** > **NO**

Diese Angaben stellen die Auswertungsergebnisse der hinter PAGEP stehenden Statements in den Formaten CSV, DIF, XLSX oder PDF in die Datei *dateiname*. Zum Beispiel stellt die Angabe

```
PDF=STUDIE27.PDF
```

die Auswertungsergebnisse in eine PDF-Datei.

Mehrere dieser Angaben sind gleichzeitig in einem Statement möglich. Sie bleiben gültig, bis sie in einem späteren Statement PAGE oder PAGEP durch neue Angaben ersetzt werden.

Bei der Ausgabe im PDF-Format kann das Programm automatisch Lesezeichen (Bookmarks) erzeugen.
Dazu ist in der Auswertung mit dem Statement INDEX ein entsprechendes Inhaltverzeichnis zu erstellen.

Bei der Ausgabe von XLSX-Dateien wird nach jeweils 100 Ausgabeseiten ein neues Arbeitsblatt begonnen. Die Arbeitsblätter werden vom Programm dabei mit Standardtexten versehen. Mit der Angabe WORKSHEET lassen sich dafür auch eigene Texte vorgeben.

Für CSV-Dateien legt die Angabe DCHAR im Statement PAGEP das Dezimalzeichen fest und SEP bestimmt das Separator- und das Texterkennungszeichen der Datei. Der Zeichencode ANSI, UTF8 usw folgt der Angabe CHARSET aus CNTA.INI oder CNTW.INI.

dateiname — Name der Datei, die die Auswertungsergebnisse der folgenden Statements aufnehmen soll.
Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
XLSX=C:\LISTEN\STUDIE95.XLSX
CSV='STUDIE95 AUSWERTUNGEN.CSV'
PDF=\\SERVER\DATEN\STUDIE95.PDF
```

Enthält *dateiname* keine Pfadangabe, so stellt CNTA die Datei in das Verzeichnis der Statementdatei und CNTW in das Verzeichnis der REP-Datei.
Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\STUDIE27.TXT
```

Eine bereits existierende Datei mit dem hier angegebenen Namen wird zu Beginn der Verarbeitung gelöscht. Erscheint der Dateiname in einem späteren PAGE- oder PAGEP-Statement noch einmal, so wird die bis dahin erstellte Datei fortgeschrieben und nicht gelöscht.

Enthält der Dateiname das Zeichen &, so wird für jede Tabelle eine neue Datei erstellt und das Zeichen & durch eine laufende Nummer 1, 2, ... ersetzt. Mehrere zusammenstehende &-Zeichen erzeugen dabei eine laufende Nummer mit einer entsprechenden Anzahl Ziffern. Die Angabe

```
PAGEP   CSV=C:\TAB&&.CSV
```

stellt die erste Tabelle in die Datei *TAB01.CSV*, die zweite Tabelle in *TAB02.CSV*, die dritte in *TAB03.CSV* usw.

KEEP — Diese Angabe ist nur für CSV, DIF und XLSX wirksam:
Normalerweise werden die Zählerwerte ohne die durch PRINT= '... & ... ' hinzugefügten Zeichen in die Ausgabedatei gestellt; auch werden die Blattüberschriften HEAD... und Fußtexte FOOT... nicht ausgegeben.
Die Angabe KEEP übernimmt alle diese Angaben in die Ausgabedatei. Zum Beispiel:

```
PAGEP CSV='STUDIE 105.CSV',KEEP
```

NO — setzt die Angabe CSV ... PDF aus früheren PAGE- oder PAGEP-Statements außer Kraft.

SEP = | 's' | < , 't' >
TAB
NUMBER

Diese Angabe bestimmt das Separatorzeichen *s* und das Texterkennungszeichen *t* für die CSV-Ausgabe.
Mit TAB wird das Tabulatorzeichen zum Separatorzeichen und mit NUMBER das Nummernzeichen #.
Für *t* ist ein beliebiges Zeichen in Hochkommas anzugeben, das als Texterkennungszeichen vor und hinter alphanumerische Werte gestellt wird. Meistens wird dafür " oder ' verwendet. Die Zeichen ' und # sind doppelt einzugeben.

Fehlt die Angabe SEP so wird dafür
```
SEP=';','"'
```
angenommen.

Die Angabe
```
SEP=':'
```
macht den Doppelpunkt zum Separatorzeichen. Es werden keine Texterkennungszeichen ausgegeben.

Durch
```
SEP=TAB,''''
```
wird das Tabulatorzeichen zum Separatorzeichen und das Hochkomma ' zum Texterkennungszeichen.

WORKSHEET = | 'name' |
TITLE
BOTTOM
STUBHEAD
FILTER
NO

Diese Angabe versieht die Arbeitsblätter der Excel-Ausgabe mit eigenen Namen zur Anzeige in den Excel-Tabs. Alle folgenden Auswertungsseiten werden in dieses Arbeitsblatt gestellt bis hin zum nächsten PAGE- oder PAGEP-Statement mit einer neuen WORKSHEET-Angabe.

Ohne diese Angabe erhalten die Arbeitsblätter Standardnamen wie *Tabellen, Index* ... wobei nach jeweils 100 Seiten ein neues Arbeitsblatt beginnt

name — Der hier eingegebene Text wird bei der Excel-Ausgabe als Name für die Arbeitsblätter verwendet. Sobald dieser Text wechselt, wird ein neues Arbeitsblatt begonnen.

TITLE — Die TITLE-Texte aus XTAB dienen bei der Excel-Ausgabe als Namen der Arbeitsblätter.

BOTTOM — Die BOTTOM-Texte aus XTAB dienen bei der Excel-Ausgabe als Namen der Arbeitsblätter.

STUBHEAD — Die STUBHEAD-Texte aus XTAB dienen bei der Excel-Ausgabe als Namen der Arbeitsblätter.

FILTER — Die FILTER-Texte aus XTAB dienen bei der Excel-Ausgabe als Namen der Arbeitsblätter.

NO — Die WORKSHEET-Angaben aus früheren PAGE- oder PAGEP-Statements werden rückgängig gemacht. Das Programm versieht die Arbeitsblätter wieder mit den Standardnamen.

DCHAR = 'x'

x ist ein beliebiges Zeichen, das in den Auswertungen (XTAB, CODEBOOK, HOLECOUNT) als Dezimalzeichen (zum Beispiel Punkt oder Komma) verwendet wird.
Bei der Bestellung von CNTA kann festgelegt werden, welches Zeichen zu verwenden ist, wenn die Angabe DCHAR fehlt.

NCHAR = 'x'

x ist ein beliebiges Zeichen, das in allen Auswertungen für den Wert Null zu drucken ist.
Fehlt diese Angabe, so wird dafür der Strich. - verwendet. Bei der Bestellung von CNTA kann festgelegt werden, welches Zeichen zu verwenden ist, falls die Angabe NCHAR fehlt.

PCHAR = 'x'

x ist ein beliebiges Zeichen, das als Lesehilfe in XTAB, CODEBOOK und INDEX zwischen den Zeilentexten und den Zahlenwerten gedruckt werden soll.
Wird hier ein Punkt angegeben, so entsteht eine punktierte Linie. Ein Leerzeichen verhindert die Ausgabe einer solchen Linie. Bei der Bestellung von CNTA kann festgelegt werden, welches Zeichen zu verwenden ist, falls die Angabe PCHAR fehlt.

RCHAR = 'x'

x ist ein beliebiges Zeichen, das in allen Auswertungen für Werte zu drucken ist, die zwar von Null verschieden sind, aber für die Druckausgabe auf Null gerundet werden.
Fehlt diese Angabe, so wird dafür 0 verwendet. Bei der Bestellung von CNTA kann festgelegt werden, welches Zeichen zu verwenden ist, falls die Angabe RCHAR fehlt.

TCHAR = 'x'

x ist ein beliebiges Zeichen, das in den Auswertungen die Tausenderstelle von Zahlen markieren soll.
Zum Beispiel TCHAR=' . ' für 1.000 oder TCHAR=' ' für 1 000.
Fehlt die Angabe TCHAR, so wird kein Tausenderzeichen ausgegeben: 1000.

NR = *n*

Die nächste zu druckende Seitennummer *n* = 1 ... 10 000. Bei dieser beginnend werden alle folgenden Seiten fortlaufend numeriert. Die Seitennummern lassen sich mit der Angabe ~NR~ an beliebiger Stelle in die Randtexte HEAD... und FOOT...einfügen. Ohne die Angabe . FOOTR erscheint rechts unten auf jedem Blatt die Seitennummerierung *Seite 1* .. Mit FOOTR=NO wird dieser Text entfernt.

FONTi = | **H** | **T** | **L** | **N** | *name* | , | **N** | **I** | **B** | **BI** | , g < , z > < , | **CO**i | **G**i | >

Diese Angaben definieren die Schriften FT1 ... FT16 zur Verwendung in den Auswertungsstatements XTAB, CODEBOOK, INDEX usw.

H	Schriftart *Helvetica* ähnlich *Arial.*
T	Schriftart *Times* ähnlich *Times New Roman.*
L	Schriftart *Line Printer* ähnlich *Courier New.*
N	Schriftart *Helvetica Narrow* ähnlich *Arial Narrow.*
name	Name einer Schriftfamilie, so wie etwa von Microsoft Word zur Schriftauswahl angezeigt. z.B. *Arial* oder *Times New Roman.* Namen mit Leerzeichen, sind in Hochkommas ' einzuschließen. Zulässig sind alle True-Type- und Open-Type-Schriften. Diese sollten in Windows installiert sein oder im Programmverzeichnis von CNTW oder CNTA zu stehen.

N	Normalschrift: Schriftstil weder fett noch kursiv.
I	Kursive Schrift (italic).
B	Fette Schrift (bold).
BI	Fette und kursive Schrift (bold italic).

g	Schriftgröße in einer der oben beschriebenen Maßeinheiten. Diese Angabe legt gleichzeitig den Zeilenabstand für mehrzeilige Texte fest.
z	Fehlt diese Angabe, so gilt der durch die Schriftart selbst vorgegebene Zeilenabstand. Excel kennt keine von der Schriftart abweichende Zeilenabstände, sodass der hier angegebene Abstand bei der Excel-Ausgabe nicht wirksam wird.
Gi **CO**i	Diese Angaben geben die Schrift nicht schwarz sondern im Grauton G1 ... G16 oder der Farbe CO1 ... CO16 aus. Im Statement PAGE sind diese Angaben wirkungslos.

Liegen keine Angaben zu den Schriften FONTi vor, so gelten folgende Voreinstellungen:

FONT1 = H, N, 6d	Helvetica normal, 6 Punkt
FONT2 = H, N, 8d	Helvetica normal, 8 Punkt
FONT3 = H, B, 8d	**Helvetica halbfett, 8 Punkt**
FONT4 = H, N, 10d	Helvetica normal, 10 Punkt
FONT5 = H, B, 10d	**Helvetica halbfett, 10 Punkt**
FONT6 = H, B, 12d	**Helvetica halbfett, 12 Punkt**
FONT7 = T, N, 8d	Times normal, 8 Punkt
FONT8 = T, B, 8d	**Times halbfett, 8 Punkt**
FONT9 = T, N, 10d	Times normal, 10 Punkt
FONT10 = T, N, 10d	**Times halbfett, 10 Punkt**
FONT11 = N, N, 8d	Helvetica schmal normal, 8 Punkt
FONT12 = N, B, 8d	**Helvetica schmal halbfett, 8 Punkt**
FONT13 = N, B, 10d	**Helvetica schmal halbfett, 10 Punkt**
FONT14 = H, N, 8d, G1	Helvetica normal, 8 Punkt, weiß für dunklen Hintergrund
FONT15 = H, N, 8d, G1	**Helvetica halbfett, 8 Punkt, weiß für dunklen Hintergrund**
FONT16 = H, B, 10d, G1	**Helvetica halbfett, 10 Punkt, weiß für dunklen Hintergrund**

Enthalten die Auswertungsstatements keine Angaben zu den Schriften, so gelten folgende Voreinstellungen:

Schrift	XTAB	CODEBOOK	INDEX	TEXT	HOLECOUNT	OUTPUT-EXT	PAGEP
FT2	Grundschrift	Grundschrift	Grundschrift	Grundschrift	Grundschrift	Grundschrift	FOOT
FT3		Variablendefinitionen			Kopftexte		
FT4		Auswertung im Kopf					HEAD
FT5		Überschrift	Überschrift			Überschrift	

LINEi = s **,** d < | **,** g | **, CO**i | **, G**i | > < **,**t >

Diese Angaben definieren die Linien L1 ... L16 zur Verwendung in den Auswertungsstatements XTAB, CODEBOOK, INDEX usw.

s Stärke der Linie in einer der oben beschriebenen Maßeinheiten.

d Abstand zwischen der Linie und den Texten oder Zählergebnissen in einer der oben beschriebenen Maßeinheiten. Fehlt diese Angabe, so gilt ein Abstand von 1.5mm.

g Grauton 0 ... 100 der Linie. Die Angabe 100 erzeugt schwarze und 0 weiße Linien.
Werte dazwischen liefern verschiedene Grautöne.

Gi Grauton G1 ... G16 aus diesem Statement PAGEP , in der die Linie ausgegeben werden soll.

COi Farbe CO1 ... CO16 aus diesem Statement PAGEP in der die Linie ausgegeben werden soll.

t Typ 1 oder 2 der Linie. Typ 1 sorgt für einfache Linien und Typ 2 für doppelte.
Bei doppelten Linien besitzen die beiden Teillinien je ein Drittel der unter *s* angegebenen Stärke der Gesamtlinie. Ohne diese Angabe werden einfache Linien erzeugt.

Enthält PAGEP keine Angaben zu den Linien, so wird folgende Voreinstellung verwendet:

```
L1 = 0.05, 1.5, 100        L9 = 1.50, 1.5, 100
L2 = 0.10, 1.5, 100        L10 = 0.20, 1.5, 0
L3 = 0.20, 1.5, 100        L11 = 0.30, 1.5, 0
L4 = 0.30, 1.5, 100        L12 = 0.40, 1.5, 0
L5 = 0.40, 1.5, 100        L13 = 0.50, 1.5, 0
L6 = 0.50, 1.5, 100        L14 = 0.75, 1.5, 0
L7 = 0.75, 1.5, 100        L15 = 1.00, 1.5, 0
L8 = 1.00, 1.5, 100        L16 = 1.50, 1.5, 0
```

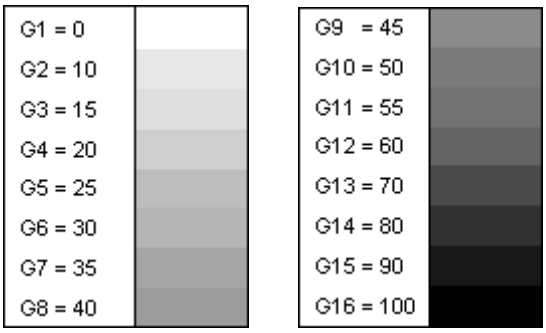
Im Statement XTAB wird ohne die Angabe LINES die Linie L3 verwendet.

Gi = p

Diese Angaben ordnen den Kürzeln G1 ... G16 Grautöne zur Einfärbung von Linien, Schriften und Tonflächen zu. Im Statement XTAB lassen sich damit zum Beispiel Zeilen oder Spalten unterlegen.

p Grauton 0 ... 100. Die Angabe 100 liefert schwarz und 0 weiß. Werte dazwischen erzeugen abgestufte Grautöne.

Ohne Angaben zu den Grautönen Gi in PAGEP gelten folgende Voreinstellungen:

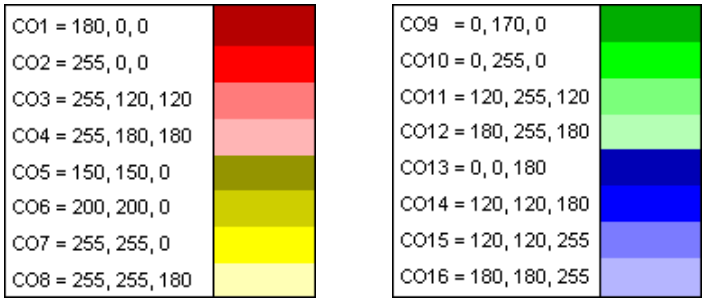


COi = r **,** g **,** b

Dieser Angaben ordnen den Kürzeln CO1 ... CO16 Farben im RGB-System zu, zum Einfärben von Schriften, Tonflächen und Linien.

r Farbwert 0 ... 255 für rot.
g Farbwert 0 ... 255 für grün.
b Farbwert 0 ... 255 für blau.

Liegen keine Angaben zu den Farben vor, so wird folgende Voreinstellung verwendet:



LOGO i = dateiname < **,** zoomfaktor >

Mit dieser Angabe wird der Zahl i eine Bilddatei im JPEG-Format zugewiesen. Mit der Angabe Li lässt diese Bilddatei in jede Textzeile als Logo einfügen oder im Statement PAGEP mit der Angabe LOGOS frei auf den Seiten der Auswertungen platzieren.

Bei der Arbeit mit PAGE lassen sich Logos angeben ohne in den Auswertungen zu erscheinen.

i Nummer 1 ... 999 unter der das Logo in den Statements aufgerufen werden soll.

dateiname Name einer Bilddatei im JPEG-Format mit der Dateierweiterung JPG.
Fehlt die Dateierweiterung, so wird JPG angenommen.
Der Dateiname kann auch ohne Pfad angegeben werden. Dann sucht das Programm die Logo-Datei in den folgenden Verzeichnissen:
Im Verzeichnis der REP-Datei. Neu angelegte Auswertungen mit dem Namen ~N~.REP befinden sich stets im Verzeichnis CNTWWORK oder CNTAWORK.
Im Verzeichnis der Statementdatei.
In dem durch die Angabe LOGOS= in der Datei CNTA.INI oder CNTW.INI festgelegten Verzeichnis.
Im Programmverzeichnis CNTW oder CNTA.

zoomfaktor Vergrößerungsfaktor 1 ... 999 als Prozentwert. Fehlt diese Angabe, so wird 100 angenommen und das Logo in Originalgröße ausgegeben. Werte unter 100 verkleinern und Werte über 100 vergrößern das Logo.

Zum Beispiel könnte das Statement

```
PAGEP LOGO1=d:\logos\kundeXYZ.jpg,200
      HEAD='~1~ XYZ-Institut'
```

folgende Kopfzeile auf den Auswertungsseiten erzeugen:

XYZ-Institut

Die Angabe 200 vergrößert dabei das Logo auf 200% seiner Originalgröße.

Beispiel

```
DEFS
-C1:8  1='~11~ aaa' 2='~12~ bbb' 3='~13~ ccc' 4='~14~ ddd'
       5='~15~ eee' 6='~16~ fff' 7='~17~ ggg' 8='~18~ hhh'
PAGEP   LOGO10=c:\cntw\cnt.jpg
        LOGO11=shampooa
        LOGO12=shampoob
        LOGO13=shampooc
        LOGO14=shampood
        LOGO15=shampooe
        LOGO16=shampoof
        LOGO17=shampoog
        LOGO18=shampooh
        L3=0.5,1,CO16           L4=1,2,CO16
        MARGINS=T:22:L4
        LOGOS=TL:10           HEAD='~B~Produktvergleich'
XTAB   ROW=TOTAL:'Gesamt';;MEAN(C2);MEAN(C3);MEAN(C4)
       COL=C1
       LINES=L3
```

Diese Statements könnten folgende Tabelle erzeugen:

Produktvergleich

	 aaa	 bbb	 ccc	 ddd	 eee	 fff	 ggg	 hhh
Gesamt	551	538	549	545	532	557	531	515
Preis	5,81	5,82	5,83	5,74	5,84	5,88	5,95	6,04
Qualität	1,39	1,43	1,41	1,46	1,36	1,39	1,40	1,43
Sympathie	3,46	3,44	3,46	3,46	3,48	3,59	3,62	3,45

Im Statement PAGEP werden zunächst die Logos 10 ... 18 definiert: Der JPEG-Datei *c:\cntw\cnt.jpg* mit dem CNT-Logo wird die Nummer 10 zugewiesen und der Datei *shampooh.jpg* schließlich die Nummer 18.

Die Angabe *LOGOS=TL:10* im Statement PAGEP stellt das Logo 10 links oben auf jede Seite der Auswertungen. Die übrigen Logos werden durch die Angaben *~11~* bis *~18~* in die Merkmalstexte der Variablen C1 übernommen. Durch *COL=C1* im Statement XTAB gelangen sie in die Köpfe der Tabelle.

Die Angabe *L3=0.5,1,CO16* in PAGEP erzeugt eine blaue Linie, die über die Angabe *LINES=L3* in XTAB für die Linien in der Tabelle verwendet wird. Mit *L4=1,2,CO16* wird die breite blaue Linie erzeugt, die durch die Angabe *MARGINS=T:22:L4* den oberen Rand von 22mm Höhe abschließt.

PRINT

PRINT steuert die Ausgabe der Statements in das Zählprotokoll. Dieses Statement darf nicht innerhalb der Fortsetzungszeilen eines anderen Statements liegen.

Folgende Angaben sind möglich:

PRINT OFF

Alle folgenden Statementzeilen werden normal verarbeitet, aber nicht mehr in das Zählprotokoll ausgegeben. Nur fehlerhafte Zeilen werden zusammen mit der entsprechenden Fehlermeldung ausgegeben. CNTA beginnt die Verarbeitung stets mit der Voreinstellung PRINT ON.

PRINT ON

Diese Angabe macht ein vorausgehendes PRINT OFF wieder rückgängig. Es werden wieder alle Statementzeilen in der Liste ausgegeben.

PROCESS

PROCESS steuert die Verarbeitung der dahinter stehenden Statements. Dieses Statement darf nicht innerhalb der Fortsetzungszeilen eines anderen Statements liegen.

PROCESS OFF

Alle folgenden Statements werden weder gedruckt noch verarbeitet, sondern unverarbeitet und ohne Fehlerprüfung übergangen.
CNTA beginnt die Verarbeitung stets mit der Voreinstellung PROCESS ON.

PROCESS ON

Diese Angabe sorgt dafür, dass alle folgenden Statementzeilen verarbeitet werden.
PROCESS ON macht insbesondere ein vorausgehendes PROCESS OFF wieder rückgängig.

PROCESS Makroausdruck

Dieses Statement wirkt wie ein PROCESS ON, wenn der Makroausdruck erfüllt ist, und wie ein PROCESS OFF, wenn er nicht erfüllt ist. Damit besteht die Möglichkeit, die Ausführung bestimmter Statements von Bedingungen abhängig zu machen. Eine vollständige Beschreibung der Makroausdrücke befindet sich im Abschnitt *Makros*. Zum Beispiel:

```
PROCESS #var > 0
XTAB  ROW=TOTAL;C1  COL=TOTAL;C2
PROCESS ON
XTAB  ROW=TOTAL;C22  COL=TOTAL;C92  FILTER=N37
```

Hier wird das erste XTAB-Statement nur dann ausgeführt, wenn das Makro `#var` vorher einen Wert größer als 0 erhalten hat. Das zweite XTAB-Statement wird in jedem Fall wirksam.

```
PROCESS #filter
```

Dieser Ausdruck ist erfüllt und führt zu PROCESS ON, wenn das Makro `#filter` vorher definiert wurde, und zu PROCESS OFF, wenn es nicht definiert wurde.

```
PROCESS #arows >= #frows
```

Dieser Ausdruck ist erfüllt und führt zu PROCESS ON, wenn der Wert des Makros `#arows` größer oder gleich dem Wert des Makros `#frows` ist. Andernfalls wirkt dies Statement wie PROCESS OFF. Insbesondere wird dies Statement als PROCESS OFF verstanden, wenn eines der beiden Makros vorher nicht definiert wurde.

Die Werte von Makros können alphanumerische Zeichenfolgen oder auch Zahlenwerte sein. Stehen auf beiden Seiten einer solchen Vergleichsrelation Zahlenwerte, so wird der Vergleich nach numerischer Größe durchgeführt.
Steht auf einer oder auf beiden Seiten ein alphanumerischer Wert, so wird nach der alphabetischen Sortierfolge verglichen:

```
#arows    2.57
#frows    13
PROCESS   #arows <= #frows
```

Hier stehen auf beiden Seiten des Vergleichsoperators `<=` numerische Werte. Der linke davon ist numerisch kleiner als der rechte, also ist der Makroausdruck erfüllt, es wird PROCESS ON wirksam.

```
#arows   2.57
#frows     13a
PROCESS    #arows <= #frows
```

Hier steht auf der rechten Seite des Vergleichsoperators <= ein alphanumerischer Wert. Daher wird der Vergleich zeichenweise nach alphabetischer Sortierfolge durchgeführt, und der linke Wert ist größer als der rechte. Der Makroausdruck ist also nicht erfüllt, es wird PROCESS OFF wirksam.

–PRT

PRT-Statements dienen dazu, Fehlermeldungen und Variablenwerte der einzelnen statistischen Fälle in das Zählprotokoll CNTLST auszugeben. Sie sind Folgestatements zu ADD-PROC, UPD-PROC, MOD-PROC und OUTPUT-EXT.

PRT druckt frei wählbare Texte und Variablenwerte hinter der Statementliste. Die Druckaufbereitung ist dabei fest vorgegeben. Mit Hilfe von IF-Statements kann die Ausgabe von Bedingungen abhängig gemacht werden.

Das Statement TST arbeitet ähnlich. Es druckt Meldungen jedoch nur aus, wenn eine im Statement selbst enthaltene Bedingung nicht erfüllt ist. Die Werte der an der Prüfbedingung beteiligten Variablen werden bei TST automatisch ausgedruckt.

Die Anzahl der PRT- und TST-Meldungen ist Teil der Fehlerstatistik, die am Ende der Datendurchläufe ausgedruckt werden kann. Siehe dazu im Statement OPTIONS die Angaben ESUM und ESUMS.

Das Statement REPORT gibt ebenfalls Werte der einzelnen statistischen Fälle aus. Im Gegensatz zu PRT und TST erfolgt die Ausgabe in eigene Dateien und in frei wählbarer Druckaufbereitung.

Folgende Angaben sind hinter –PRT möglich:

< 'ttt...' >

Hiermit wird zu jedem statistischen Fall eine Textzeile *ttt...* aus 1...255 Zeichen in das Zählprotokoll ausgegeben. Sie wird rechts neben eventuelle Variablenwerte gestellt.

UNDO Mit der Funktion UNDO aus dem Abschnitt Textzeilen lassen sich Variablen- und Merkmalstexte sowie Textelemente ausgeben. Besitzt das Merkmal 1 der Variablen C10 zum Beispiel den Text *18 bis 25 Jahre*, so wird dieser mit

```
-REPORT UNDO(C10.1)
```

in das Zählprotokoll ausgegeben.

Cn
Cm . . . n
Cn**[** i **]**
Cn**[** i **]** . . . j

Die Werte der hier angegebenen Bedingungsvariablen werden zu jedem statistischen Fall ausgedruckt. In einem PRT-Statement sind mehrere dieser Angaben möglich, auch kombiniert mit N- und T-Variablen.

Nn
Nm . . . n
Nn**[** i **]**
Nn**[** i **]** . . . j

Genau wie bei den C-Variablen können hiermit die Werte von numerischen Variablen ausgedruckt werden. In einem PRT-Statement sind mehrere dieser Angaben möglich, auch kombiniert mit C- und T-Variablen.

Tn **T**m . . . n **T**n**[** i **]** **T**n**[** i **]** . . . j

Genau wie bei den C-Variablen können die Werte von Textvariablen ausgedruckt werden. In einem PRT-Statement sind mehrere dieser Angaben möglich, auch kombiniert mit C- und N-Variablen.

ERR Fehlernummer 500 ... der zuletzt ausgeführten Wertezuweisung.
Der Operand ERR wird zu Beginn jeder Wertezuweisung auf 0 gesetzt. Läuft die Wertezuweisung fehlerfrei ab, so behält ERR den Wert 0. Wird hingegen ein Fehler gefunden, so enthält ERR anschließend die Nummer 500 ... 599 der entsprechenden Fehlermeldung. Mit den Statements

```
-PRT ERR
```

lässt sich der Wert des Operanden ERR im Zählprotokoll ausdrucken.

Beispiel

```
INPUT-INT  FNAME=EINGABE.INT
MOD-PROC
-PRT  C1..2
-IF N1<N2
-  PRT N1..2  'N1 ist kleiner als N2'
-EIF
-PRT T7
```

Dabei druckt das erste Folgestatement von MOD-PROC für jeden statischen Fall aus der Datei INPUT-INT die Werte der Variablen C1 und C2 aus. Ist zum Beispiel der Fall mit der Identifikation 1017 in Arbeit, so könnte das folgende Druckausgabe erzeugen:

```
FALL-ID: 1017
   C1 = 1 12
   C2 =
```

Das heißt, dass in der Variablen C1 hier die Merkmale 1 und 12 gesetzt sind. In der Variablen C2 sind keine Merkmale erfüllt.

Die nächsten Statements stellen immer, wenn der Wert der Variablen N1 kleiner als der von N2 ist, eine Meldung in das Zählprotokoll. In unserem Fall 1017 könnte das folgendermaßen aussehen:

```
FALL-ID: 1017
   N1 = 10.17    N1 ist kleiner als N2
   N2 = 20.3
```

Anstelle der drei Statements

```
-IF N1<N2
-  PRT  N1..2  'N1 ist kleiner als N2'
-EIF
```

könnte auch das Statement

```
-TST  N1<N2  'N1 ist kleiner als N2'
```

verwendet werden.

Das letzte Statement druckt die in der Textvariablen T7 zu den einzelnen statistischen Fällen gespeicherten Text aus. Dies könnte etwa ein Markenname sein:

```
FALL-ID: 1017
   T7 = 'Opel'
```

–REPORT

Die Statements REPORT ... REPORT9 geben frei wählbare Texte und Variablenwerte zu jedem statistischen Fall in Textdateien aus. Diese Dateien dienen zum Beispiel der Prüfung eines Datenbestandes und der Fehlersuche, lassen sich aber auch als neue Datenbestände zur weiteren Verarbeitung verwenden. Der Aufbau der Textzeilen lässt sich frei gestalten.

REPORT ... REPORT9 sind Folgestatements zu ADD-PROC, UPD-PROC, MOD-PROC und OUTPUT-EXT. Vor ihrer Verwendung sind mit den Statements REPORT-FILE ... REPORT-FILE9 entsprechende Dateien einzurichten.

Mit REPORT-Statements können zu jedem statistischen Fall beliebig viele Textzeilen ausgegeben werden.
Ein neues REPORT-Statement beginnt dabei nicht automatisch mit einer neuen Zeile, sondern setzt die Zeile des vorigen Statements fort. Mit Hilfe von IF-Statements lassen sich die Ausgaben auch von Bedingungen abhängig machen.

Die Statements TST und PRT haben eine ähnliche Funktion wie REPORT. Sie geben ihre Daten jedoch nur in das Zählprotokoll CNTLST aus, die Gestaltung der Druckzeilen ist vom Programm fest vorgegeben und mit jedem PRT und TST wird eine neue Druckzeile begonnen.

Die folgenden Angaben sind hinter –REPORT in beliebiger Anzahl und Reihenfolge möglich. Sie sind durch ein Komma oder Leerzeichen voneinander zu trennen. Die dazugehörigen Daten werden stets in die nächste freie Position der laufenden Zeile ausgegeben. Leerzeichen vor oder hinter den Werten werden entfernt. Ist eine Zeile länger als SIZE im Statement REPORT-FILE erlaubt, so werden die überschüssigen Zeichen abgeschnitten.

' ttt ...**'** Dies ist ein einzeiliger Text aus 1 ... 500 Zeichen, der zu jedem statistischen Fall ausgegeben wird.
Zum Beispiel:

```
-REPORT 'keine Angabe'
```

'~DATE~'
'~TIME~'

Dies Angaben stellen das Datum oder die Uhrzeit des Programmlaufs in die REPORT-Datei.
Zum Beispiel:

```
-REPORT 'Datum: ~DATE~  Uhrzeit: ~TIME~'
```

UNDO Mit der Funktion UNDO aus dem Abschnitt Textzeilen lassen sich Variablen- und Merkmalstexte sowie Textelemente ausgeben. Besitzt das Merkmal 1 der Variablen C10 zum Beispiel den Text *18 bis 25 Jahre*, so wird dieser mit

```
-REPORT UNDO(C10.1)
```

als einzeiliger Text in die Report-Datei gestellt.

ID < **:** p >
ID i

Durch Angabe einer Fall-Identifikation ID ... ID4 wird ihr jeweiliger Wert für jeden statistischen Fall ausgegeben. Eine Zahl : *p* stellt die Fall-Identifikation in ein Feld von *p* Zeichen Breite.
Durch

```
-REPORT ID:8
```

werden numerische Werte rechtsbündig und alphanumerische Werte linksbündig in ein acht Zeichen breites Feld ausgegeben. Zu große Werte werden ohne Fehlermeldung abgeschnitten, Texte rechts und Zahlenwerte links.

< n > NL Mit der Angabe NL wird die laufende Druckzeile beendet und eine neue Zeile begonnen.
Durch eine Zahl n vor NL lassen sich zusätzliche Leerzeilen ausgeben.
So erzeugt

```
-REPORT 3NL
```

zum Beispiel zwei Leerzeilen vor der nächsten Druckzeile.

Cn **< : p >**
Cn **(v .. b)**

Durch Angabe einer C-Variablen werden die Nummern 1 ... 9 999 der erfüllten Merkmale ausgegeben. Mehrfachnennungen erscheinen hintereinander durch Kommas getrennt. Durch

```
-REPORT C1 (5..9)
```

wird die Ausgabe der Variablen C1 auf die Merkmale 5 bis 9 beschränkt.
In der Form C1 .. 2 oder C7 [2] .. 5 lassen sich gleichzeitig mehrere Variable ausgeben.
Mit der Zahl *:p* hinter der Variablen werden die Merkmalswerte linksbündig in ein Feld von p Zeichen Breite gestellt. Zum Beispiel gibt

```
-REPORT C7:12
```

die erfüllten Merkmale der Variablen C7 linksbündig in ein 12 Zeichen langes Feld der Ausgabezeile und füllt rechts mit Leerzeichen auf. Zu lange Ausgaben werden ohne Fehlermeldung rechts abgeschnitten.

Nn **< : p >**

Durch Angabe einer N-Variablen werden die Werte der Variablen ausgegeben.
Für *keine Angabe* wird Z0 ausgegeben. Werte mit Nachkommastellen erscheinen mit Dezimalpunkt.
In der Form N1 .. 2 oder N7 [2] .. 5 lassen sich gleichzeitig mehrere Variable ausgeben.
Mit der Zahl *: p* wird der Variablenwert rechtsbündig in ein Feld von *p* Zeichen Breite gestellt.
Zum Beispiel stellt

```
-REPORT N17:5
```

den Wert der Variablen N17 rechtsbündig in ein fünf Zeichen langes Feld der Ausgabezeile und füllt links mit Leerzeichen auf. Zu große Werte werden ohne Fehlermeldung links abgeschnitten.

Tn **< : p >**
Tn **(v .. b)**

Hiermit werden die Werte einer T-Variablen ausgegeben. Das Statement

```
-REPORT T1(5...9)
```

gibt die Zeichen 5 bis 9 der Variablen T1 aus und

```
-REPORT T1(2...-1)
```

das zweite bis letzte Zeichen.
Die Angabe ***: p*** stellt den Variablenwert linksbündig in ein Feld von *p* Zeichen Breite. Zum Beispiel gibt das Statement

```
-REPORT T17:12
```

den Text aus der Variablen T17 linksbündig in ein zwölf Zeichen langes Feld der Ausgabezeile und füllt rechts mit Leerzeichen auf. Erst dahinter wird mit der Ausgabe fortgefahren.
Zu lange Texte werden ohne Fehlermeldung rechts abgeschnitten.
In der Form T1 .. 2 oder T7 [2] .. 5 lassen sich gleichzeitig mehrere Variable ausgeben.

---

p **T** Diese Angabe rückt die Ausgabe für den nächsten Wert an die Position p der laufenden Zeile.
So wird durch

```
-REPORT 20T,N17
```

der Wert der Variablen N17 ab Position 20 der laufenden Zeile ausgegeben.
Steht p hinter dem letzten bisher ausgegebenen Zeichen, so wird mit Leerzeichen aufgefüllt.
Steht p vor dem letzten Zeichen, so wird der alte Text hinter der Position *p* nicht gedruckt.

< n > **X** Es wird ein Leerzeichen in die laufende Zeile gestellt. Mit der Angabe n davor lassen sich gleichzeitig mehrere Leerzeichen ausgeben. Zum Beispiel stellt

```
-REPORT X N1 3X
```

ein Leerzeichen vor den Wert der Variablen N1, gefolgt von drei weiteren Leerzeichen.

< n > **TAB** Es wird ein Tabulatorzeichen in die laufende Zeile gestellt. Mit der Angabe n davor lassen sich gleichzeitig mehrere Tabulatorzeichen ausgeben. Zum Beispiel stellt

```
-REPORT TAB N1 3TAB
```

ein Tabulatorzeichen vor den Wert der Variablen N1, gefolgt von drei weiteren Tabulatorzeichen.

Beispiel

```
INPUT-INT       FNAME=EINGABE.INT
REPORT-FILE     FNAME=PROTOKOLL
MOD-PROC
-REPORT '===== ' ,ID:3, ' ======================================='
-REPORT NL 'Marke: ' C1:10 X  ' Ausgabe: ' N1:7 ' EUR'
-REPORT NL 5X 'C1: ' C1 ' N1: ' N1
-IF LAST
-  REPORT NL '================================================'
-  EIF
```

Diese Statements könnten die folgende Report-Datei erzeugen:

```
=====   1 =======================================
Marke: 1,10       Ausgabe:  2372 EUR
     C1: 1,10 N1: 2372
=====   2 =======================================
Marke: 3,9        Ausgabe: 78200 EUR
     C1: 3,9 N1: 78200
=====   3 =======================================
Marke: 6,9        Ausgabe:    Z0 EUR
     C1: 6,9 N1: Z0
=====   4 =======================================
Marke: 1,11       Ausgabe:  2372 EUR
     C1: 1,11 N1: 2372
=====   5 =======================================
Marke: 2,11       Ausgabe:  1560 EUR
     C1: 2,11 N1: 1560
=====   6 =======================================
Marke: 3,8        Ausgabe: 78200 EUR
     C1: 3,8 N1: 78200
================================================
```

Das Statement REPORT-FILE erstellt dabei eine Textdatei mit dem Namen PROTOKOLL.

Das erste Statement REPORT gibt zu jedem statistischen Fall eine Linienzeile aus, die die Fall-Identifikation enthält.

Das zweite Statement REPORT gibt die Werte der Variablen C1 und N1 aus, jeweils mit einem Text davor. Durch die Angaben :10 hinter C1 und :7 hinter N1 werden die Werte in feste Spalten von 10 und 7 Zeichen Breite gestellt. Wegen der Angabe NL zu Beginn dieses Statements erscheinen die Daten am Anfang einer neuen Zeile.

Das dritte Statement REPORT zeigt die gleichen Werte mit anderen Texten und ohne Spalteneinteilung.

Das letzte Statement REPORT wird wegen –IF LAST nur für den letzten eingelesenen Fall wirksam. Es gibt eine abschließende Linienzeile aus.

REPORT-FILE

Die Statements REPORT-FILE erzeugen Textdateien mit Informationen zu den einzelnen statistischen Fällen. Im Gegensatz zu PRT und TST können die Ausgaben in die Report-Dateien frei gestaltet werden.

Die Datenausgabe selbst erfolgt mit den Statements –REPORT hinter ADD-PROC, MOD-PROC, UPD-PROC und OUTPUT-EXT. Damit können feste Texte, Variablenwerte und Fall-Identifikationen ausgegeben und beliebig innerhalb einer oder mehrerer Zeilen positioniert werden.

Im Beispiel am Ende dieses Abschnitts wird eine einfache Anwendung der Statements REPORT-FILE und REPORT gezeigt. Ausführliche Angaben befinden sich in der Beschreibung des Statements REPORT.

Die Statements REPORT-FILE dürfen an beliebiger Stelle angegeben werden, müssen aber vor dem ersten REPORT-Statement stehen. REPORT-FILE darf nicht innerhalb der Folgestatements etwa von DEFS, MOD-PROC usw. angegeben werden.
In einem Verarbeitungslauf können gleichzeitig bis zu zehn verschiedene Dateien REPORT-FILE, REPORT-FILE1 ... REPORT-FILE9 erstellt werden. Jedes diese Statements darf in einem Verarbeitungslauf nur einmal vorkommen.

REPORT-FILE erlaubt folgende Angaben, die weitgehend mit denen aus INPUT-EXT übereinstimmen:

FNAME = dateiname

Hier ist der Name der Ausgabedatei anzugeben. Dateinamen mit Leerzeichen oder Kommas sind in Hochkommas ' einzuschließen. Zum Beispiel:

```
FNAME=C:\DATEN\STUDIE27.AUS
FNAME='STUDIE27 REPORT'
```

Enthält *dateiname* keine Pfadangabe, so stellt CNTA die Datei in das Verzeichnis der Statementdatei und CNTW in das Verzeichnis der REP-Datei.
Eine bereits existierende Datei mit diesem Namen wird überschrieben.
Der Dateiname darf auch in Prozentzeichen % eingeschlossene Umgebungsvariable enthalten, sofern diese im Betriebssystem definiert sind. Zum Beispiel:

```
FNAME=%STUDIEN%\NR%KUNDE%\STUDIE27.PS
```

ANSI
ASCII
UTF8
COLBIN
EBCDIC

Diese Angaben legen den Zeichencode fest, in welchem die Texte in die Report-Datei auszugeben sind. Ohne diese Angaben wird mit dem normalen Format des verwendeten Computers gearbeitet: Weitere Angaben zu den Codes befinden sich im Anhang *Zeichenzuordnung ANSI, ASCII, EBCDIC* und *Lochpositionen der COLBIN-Spalten*.

ANSI Dies ist der von MS-Windows bevorzugte Zeichensatz.

ASCII Dies ist der vom MS-DOS eingeführte Zeichensatz. Der Unterschied zum ANSI-Code beschränkt sich auf die Umlaute, den Buchstaben ß und einige Sonderzeichen.

UTF8 Die Datei wird im UTF8-Code erstellt. Dieser umfasst praktisch alle weltweit vorkommenden Schriftzeichen.

COLBIN Die Daten sind im Dualkarten-Format (column binary) auszugeben.

EBCDIC Dies ist das normale Datenformat von Großrechnern.

SIZE = n <,**EOL**>

Mit SIZE wird die Länge der Datensätze festgelegt. Fehlt diese Angabe, so wird SIZE=80,EOL angenommen. Textzeilen, die über das Ende der Datensätze hinausreichen, werden ohne Fehlermeldung abgeschnitten.

n Satzlänge 1 ... 10 000 000. in Bytes, bei COLBIN-Dateien in Spalten.

EOL Diese Angabe ist bei COLBIN-Dateien nicht möglich.
Sie besagt, dass jeder Datensatz durch ein Zeilenende-Zeichen beendet wird (CR=10 oder LF=13 oder CR und LF zusammen).
Die Satzlänge n gibt dann die maximal zulässige Länge pro Datensatz an.
Die Zeilenende-Zeichen sind bei EOL in der Länge *n* nicht enthalten.

APPEND Diese Angabe erhält bestehende REPORT-Dateien. Die mit -REPORT ausgegebenen Daten werden an das Ende der Datei gestellt. Ohne APPEND werden bestehende Dateien vor dem Programmlauf gelöscht.

Beispiel

```
INPUT-INT     FNAME=BERLIN.INT
REPORT-FILE   FNAME=FEHLER.LST

MOD-PROC
-IF C17.Z23
-REPORT  'ID: '  ID  3X 'Mehrfachnennungen in C17:'  C17
-EIF
```

Das Statement REPORT-FILE erzeugt die Protokolldatei FEHLER.LST.
Das Statement REPORT wird wegen IF C17.Z23 immer wirksam, wenn ein statistischer Fall in der Variablen C17 Mehrfachnennungen besitzt. Daraus könnte in der Datei FEHLER.LST folgendes Protokoll entstehen:

```
ID: 1543   Mehrfachnennungen in C17: 2,3,9
ID: 1595   Mehrfachnennungen in C17: 3,7
ID: 1812   Mehrfachnennungen in C17: 1,4,8,9
ID: 1992   Mehrfachnennungen in C17: 2,7
ID: 2012   Mehrfachnennungen in C17: 3,4,8
ID: 2075   Mehrfachnennungen in C17: 1,3
ID: 2134   Mehrfachnennungen in C17: 1,2,3,4,5,7
ID: 2209   Mehrfachnennungen in C17: 2,3,7
```

Hinter REPORT sorgt die Angabe 'ID: ' für die Ausgabe des Textes am Zeilenanfang.
Mit ID wird die Fall-Identifikation des gerade bearbeiteten Falles ausgegeben.
3X sorgt für drei Leerzeichen vor dem Text 'Mehrfachnennungen...' und mit C17 werden die erfüllten Merkmale der Variablen C17 ausgegeben, mit Kommas als Trennzeichen zwischen den Nummern der erfüllten Merkmale

SELECT

Das Statement SELECT wählt aus den Eingabedaten bestimmte statistische Fälle zur weiteren Verarbeitung aus. SELECT wird wirksam, direkt nachdem die Statements aus ADD-PROC, MOD-PROC und UPD-PROC ausgeführt sind, jedoch vor dem Statement WEIGHT.
Die nicht von SELECT ausgewählten Fälle werden unverarbeitet übergangen.

SELECT ermöglicht folgende Angaben:

---

logischer Ausdruck

wie im Abschnitt *Logische Ausdrücke* beschrieben.
Es werden nur die Fälle verarbeitet, für die der logische Ausdruck erfüllt ist.

SAMPLE = n

Durch diese Angabe wird aus den Eingabedaten eine Zufallsstichprobe erzeugt. Nur die dazugehörigen Fälle werden verarbeitet. Dabei wird ein Zufallsgenerator verwendet, der in Verarbeitungsläufen mit gleichen Daten stets auch die gleiche Teilstichprobe liefert.
Ist dieses nicht gewünscht, so können im Statement OPTIONS mit der Angabe RSTART aus gleichen Daten unterschiedliche Teilstichproben erzeugt werden.

n = 0 ... 100 legt fest, wie viel Prozent der Eingabedaten ungefähr in die Stichprobe aufzunehmen sind. Ist gleichzeitig ein logischer Ausdruck angegeben, so wird die Prozentzahl n nur auf die Fälle angewendet, die den logischen Ausdruck erfüllen.

MAX = m

Diese Angabe sorgt dafür, dass nur die ersten m = 0 ... 999 999 999 Fälle aus der Eingabe verarbeitet werden. Ist zusätzlich noch ein logischer Ausdruck oder die Angabe SAMPLE vorhanden, so werden für MAX nur diejenigen Fälle gezählt, die die anderen Bedingungen bereits erfüllt haben.

Beispiel

```
INPUT-INT   FNANE=STUDIE2.INT       ID=D(1,4)
INPUT-EXT   FNAME=STUDIE2.NEU
UPD-PROC

SELECT      ID>=100 & <=9000     SAMPLE=60
```

Hier werden gleichzeitig Daten aus INPUT-INT und INPUT-EXT eingelesen und zu statistischen Fällen zusammengefügt. UPD-PROC mit den hier ausgelassenen Folgestatements enthält die dazu erforderlichen Wertezuweisungen.

SELECT sorgt dann dafür, dass nur die Fälle mit einer Fall-Identifikation zwischen 100 und 9000 zur Verarbeitung zugelassen werden. Von diesen wählt SAMPLE eine Teilstichprobe von etwa 60% aus.

TEXT

Dieses Statement fügt freien Text in die Auswertungen ein. Dies kann auf eigenen Seiten erfolgen, zum Beispiel für Titelseiten oder Zwischenblätter, oder auf Seiten, die bereits Auswertungsergebnisse aus XTAB enthalten.
Folgende Angaben sind in TEXT-Statements möglich:

' text ' Der auszugebende Text als eine oder mehrere Textzeilen, wie im Abschnitt Textzeilen beschrieben. Der Text kann über mehrere Seiten hinweg reichen: Sobald die letzte Zeile der alten Seite gefüllt ist, druckt CNTA auf der nächsten Seite weiter.
Der hier vorgegebene Zeilenlauf bleibt erhalten. Es findet kein automatischer Zeilenumlauf statt.

Die Druckausgaben von TEXT-Statements können durch die Angabe CONT zusammen mit Tabellen aus XTAB-Statements auf eine Seite gestellt werden, aber nicht zusammen mit anderen Auswertungen wie CODEBOOK und HOLECOUNT.

Bei Druckausgabe mit dem Statement PAGEP wird die Schrift FONT2 verwendet.

CONT Die Angabe CONT verhindert den Blattvorschub am Textende.
Die Druckausgabe des nächsten TEXT- oder XTAB-Statements wird auf die gleiche Seite gedruckt wie der Text des laufenden TEXT-Statements. Durch die Angabe von START in den TEXT- und XTAB-Statements kann die Lage der Texte und Tabellen auf der Seite beeinflusst werden.

Die Fußnotentexte mehrerer Tabellen werden gesammelt am Ende jeder Seite ausgegeben.

WIDTH = n Diese Angabe legt die Breite *n* für die Aufbereitung des Textes fest.
Ohne diese Angabe wird der Text bis zum rechten Blattrand ausgegeben. Die Höhe der Textausgabe lässt sich nicht vorgeben, sie wird durch die Textlänge und die Ränder aus MARGINS bestimmt.

Ist PAGEP wirksam, so ist die Breite *n* in Millimetern anzugeben; auch die anderen Maßeinheiten aus dem Statement PAGEP sind möglich. Bei PAGE ist als Breite die Anzahl von Zeichen anzugeben. Die Mindestbreite beträgt 1mm oder 1 Zeichen.

STYLE = < **FT**i > < | **,G**i | **,CO**i | > < | **,L** | **,R** | **,C** | >

FTi Diese Angabe legt die Schriftart FT1 ... FT16 aus PAGEP für den Text fest.
Ohne sie wird die Schrift FT2 verwendet, hinter PAGE bleibt sie wirkungslos,

L R C Diese Angaben richten die Textzeilen innerhalb ihres Rechtecks aus:
L = linksbündig, R = rechtsbündig, C = zentriert.

Gi Diese Angabe unterlegt das Ausgaberechteck mit dem Grauton G1 ... G16 aus PAGEP, hinter dem Statement PAGE bleibt sie wirkungslos.

COi Diese Angaben unterlegt das Ausgaberechteck mit der Farbe CO1 ... CO16 aus PAGEP, hinter dem Statement PAGE bleibt sie wirkungslos.

START = | y
RBy
REy
RMy
RSy
RTy | **,** | x
RBx
REx
RMx
RSx
RTx
C

Diese Angabe verändert die Lage der Texte auf der Seite. Sie erlaubt, mehrere Tabellen aus XTAB-Statements oder Texte aus TEXT-Statements mit der Angabe CONT auf einer Seite zu platzieren.

y und *x* legen die obere linke Position des Textanfangs fest.
Die Angaben RM ... RT positionieren den Text relativ zu den Druckausgaben, die zuvor durch die Statements XTAB oder TEXT auf das gleiche Blatt ausgegeben wurden.

C Für den Abstand x kann auch der Buchstabe C angegeben werden, zum Beispiel in der Form
```
START=0,C
```
Dadurch wird der Text horizontal auf der Seite zentriert.

x Position des linken Textanfangs.
Bei Druckausgabe mit dem Statement PAGE ist hier die Position als Anzahl 0 ... 32 767 von Druckzeichen anzugeben. Bei Ausgabe mit dem Statement PAGEP ist dies die Position in 0 ... 500 Millimetern oder in einer der bei PAGEP beschriebenen Maßeinheiten. Mit negativen Angaben kann auch in den linken Rand hinein gedruckt werden.

Ohne eine Angabe RM ... RT ist x der Abstand des Textanfangs vom linken Rand des Blattes. So beginnt der Text mit
```
START=0, 10
```
im Abstand 10 von linken Rand. Die Breite des Randes wird mit den Statements PAGE und PAGEP festgelegt.

Steht vor x eine Angabe RM ... RT, so bezieht sich der Abstand nicht auf den linken Rand, sondern auf den durch RM ... festgelegten Referenzpunkt. Zum Beispiel beginnt der Text mit
```
START=0, RE10
```
im Abstand 10 rechts von der vorigen XTAB- oder TEXT-Ausgabe auf dem Blatt.

y Position des oberen Textanfangs.
Bei Druckausgabe mit dem Statement PAGE ist hier die Position als Anzahl 0 ... 32 767 von Druckzeilen anzugeben. Bei Ausgabe mit dem Statement PAGEP ist dies die Position in 0 ... 500 Millimetern oder in einer der bei PAGEP beschriebenen Maßeinheiten. Mit negativen Angaben kann auch in den oberen Rand hinein gedruckt werden.

Ohne eine Angabe RM ... RT ist y der Abstand des Textanfangs vom oberen Rand des Blattes. So beginnt der Text mit
```
START=10, 0
```
im Abstand 10 vom oberen Rand. Die Höhe dieses Randes lässt sich mit den Statements PAGE und PAGEP festlegen.

Steht vor y eine Angabe RM ... RT, so bezieht sich der Abstand nicht auf den oberen Rand, sondern auf den durch RM ... festgelegten Referenzpunkt. Zum Beispiel beginnt der Text
```
START=RE10, 0
```
im Abstand 10 unterhalb der vorigen XTAB- oder TEXT-Ausgabe auf dem Blatt.

RB Durch diese Angabe wird der Anfang der Zählerwerte der vorigen XTAB-Ausgabe zum Referenzpunkt. In der x-Richtung ist dies das rechte Ende der Spalte mit den Zeilentexten, in der y-Richtung das untere Ende der Spaltenüberschriften.

RE Durch diese Angabe wird das Ende der vorigen XTAB- oder TEXT-Ausgabe zum Referenzpunkt. In der x-Richtung ist RE das rechte Ende dieser Ausgabe, in der y-Richtung das untere Ende einschließlich eventueller BOTTOM-Texte, jedoch ohne Fußnoten aus FOOTNOTE.

RM Durch diese Angabe wird die maximale, aus XTAB- oder TEXT-Statements stammende Ausgabe zum Referenzpunkt. In der x-Richtung ist das der am weitesten rechts auf dem Blatt gedruckte Punkt, in der y-Richtung der am weitesten unten gedruckte Punkt. Dazu zählen eventuelle BOTTOM-Texte, jedoch keine Fußnoten aus FOOTNOTE.

RS Durch diese Angabe wird der Anfang der vorigen XTAB- oder TEXT-Ausgabe zum Referenzpunkt.
In der x-Richtung ist RS der linke Anfang dieser Ausgabe, in der y-Richtung der obere Anfang, einschließlich eventueller TITLE- oder FILTER-Angaben.

RT Durch diese Angabe wird der Tabellenanfang der der vorigen XTAB-Ausgabe zum Referenzpunkt. In der x-Richtung ist dies der linke Anfang der vorigen Tabelle, in der y-Richtung der obere Anfang der Spaltenüberschriften, hinter eventuellen TITLE- oder FILTER-Angaben.

Wurden vorher auf einer Seite keine Tabellen oder Texte gedruckt, so führen die relativen Positionen RM ... RT zum gleichen Ergebnis wie die absoluten Werte x oder y.

Fehlt die Angabe START, so wird mit START = RM0, 0 gearbeitet.

Beispiel: Text rechts von einer Tabelle

```
XTAB   ROW=TOTAL;C20
       COL=TOTAL;C10
       CONT

TEXT   'Diese Tabelle enthält nur ungewichtete Fälle'
       WIDTH=50
       STYLE=FT5
       START=0,RE5
```

Diese Statements erzeugen eine Kreuztabelle und stellen mit dem TEXT-Statement einen zusätzlichen Text rechts daneben. Das Ergebnis könnte folgendermaßen aussehen:

	Gesamt	Beilage bemerkt	
		ja	nein
Gesamt	909	465	444
Interesse an Weiterbelieferung			
Ich wäre stark daran interessiert	241	125	116
Ich wäre etwas daran interessiert	226	108	118
Es wäre mir gleichgültig	220	119	101
Ich fände es störend	222	113	109

Diese Tabelle enthält nur ungewichtete Fälle

Die Angabe CONT im Statement XTAB verhindert den Blattvorschub, sodass der zusätzliche Text auf der gleichen Seite erscheint.

Im Statement TEXT wählt die Angabe STYLE=FT5 eine größere Schrift aus.

WIDTH=50 gibt eine Breite von 50mm für den Text vor.

START=R0,RE5 plaziert den Text: 5mm rechts der Tabelle, oben auf der Seite.

Beispiel: Text mit Tonfläche über einer Tabelle

```
TEXT 'Wie hoch ist das Nettoeinkommen aller ständig '-
     'in Ihrem Haushalt lebenden Personen zusammengenommen?'
       STYLE=G3
       WIDTH=118
       MARGINS=T:3,B:3,L:3,R:3
       CONT

XTAB   COL=TOTAL;C24
       ROW=TOTAL;C30.1;...4
       STYLE=G3
       LINES=L14
       START=RE2,0
```

Diese Statements erzeugen eine Kreuztabelle und stellen einen zusätzlichen Text darüber. Das Ergebnis könnte folgendermaßen aussehen:

Wie hoch ist das Nettoeinkommen aller ständig in Ihrem Haushalt lebenden Personen zusammengenommen?

	Gesamt	Geschlecht	
		männlich	weiblich
Gesamt	909	459	450
Haushaltsnettoeinkommen			
bis 999 Euro	90	51	39
1000 - 1999 Euro	92	56	36
2000 - 2999 Euro	101	47	54
3000 - 3999 Euro	106	56	50

Das Statement TEXT erzeugt den Text oberhalb der Tabelle in einem Kasten von 118mm Breite.

STYLE=G3 hinterlegt den Text und die Tabelle mit einem Grauton.

MARGINS sorgt für einen Rand von 3mm um den Text und START=0,0.8 rückt den Text um 0.8 mm nach rechts.

Die Angabe CONT im Statement TEXT verhindert den Blattvorschub, sodass die folgende Tabelle auf der gleichen Seite erscheint.

Im Statement XTAB unterlegt STYLE=G3 die Tabelle mit einem Grauton und LINES=L14 erzeugt die weißen Linien darin.

START=RE2,0 rückt die Tabelle 2mm unter die TEXT-Ausgabe.

–TST

TST-Statements dienen dazu, Fehler in den einzelnen statistischen Fällen zu erkennen und dazugehörige Texte und Variablenwerte in das Zählprotokoll auszugeben. Sie sind Folgestatements zu ADD-PROC, MOD-PROC, UPD-PROC und OUTPUT-EXT.

Im TST-Statement lassen sich beliebig komplizierte Prüfbedingungen formulieren. Ist diese Prüfbedingung für einen statistischen Fall nicht erfüllt, so wird eine Fehlermeldung ausgedruckt. Neben der Fall-Identifikation des statistischen Falles werden auch die Werte der an der Prüfbedingung beteiligten Variablen aufgeführt.
Die Druckaufbereitung ist dabei fest vorgegeben.

Das Statement PRT erzeugt ähnliche Meldungen wie TST, besitzt aber keine eigene Prüfbedingung.
Das Statement REPORT gibt ebenfalls Werte der statistischen Fälle aus. Im Gegensatz zu PRT und TST erfolgt die Ausgabe in eigene Dateien und in frei wählbarer Druckaufbereitung.
Die Anzahl der PRT- und TST-Meldungen ist Teil der Fehlerstatistik, die am Ende der Datendurchläufe ausgedruckt werden kann. Siehe dazu im Statement OPTIONS die Angaben ESUM und ESUMS.

Folgende Angaben sind hinter TST möglich:

---

logischer Ausdruck

Dieser ist immer erforderlich und muss vor allen anderen Angaben im Statement stehen. Zu statistischen Fällen, die die Bedingung des logischen Ausdrucks nicht erfüllen, erscheint eine Fehlermeldung im Zählprotokoll. Siehe Abschnitt *Logische Ausdrücke*.

'text' Dies ist eine beliebige Textzeile, die als Fehlermeldung ausgegeben wird, wenn der logische Ausdruck *nicht* erfüllt ist. Fehlt in einem TST-Statement eine solche Textzeile, so wird dafür der logische Ausdruck selbst gedruckt.

Mit der Funktion UNDO aus dem Abschnitt *Textzeilen* lässt sich dieser Text auch aus Variablen oder Textelementen übernehmen. So wird für das Statement

```
-TST  C10.Z1  UNDO(C10.1)
```

Der Merkmalstext von C10.1 als Fehlertext verwendet.

Die folgenden Angaben hinter dem logischen Ausdruck geben im Fehlerfall Variablenwerte im Zählprotokoll aus. Fehlen solche Angaben, so werden stattdessen die Werte der Variablen aus dem logischen Ausdruck ausgegeben.

Cn
Cm . . . n
Cn[i]
Cn[i] . . . j

zur Auswahl von Bedingungsvariablen

Nn
Nm . . . n
Nn[i]
Nn[i] . . . j

zur Auswahl von numerischen Variablen

Tn
Tm . . . n
Tn[i]
Tn[i] . . . j

zur Auswahl von Textvariablen

Beispiele

```
MOD-PROC
-TST (C10.1 | ...9) & .Z1
```

Dieses TST-Statement prüft für jeden statistischen Fall die Werte der Bedingungsvariablen C10:
Es muß wenigstens eines der Merkmale 1 ... 9 dieser Variablen erfüllt sein.
Die Angabe Z1 verlangt zusätzlich, dass genau ein Merkmale erfüllt ist.

Bei der Auswertung der statistischen Fälle könnten daraus folgende Fehlermeldungen gedruckt werden:

```
                                              STMT
FALL-ID: 1672
   C10 = 7 9      (C10.1|..9) & .Z1          25

FALL-ID: 1675
   C10 =          (C10.1|..9) & .Z1          25
```

Unter der Identifikation des jeweiligen Falles werden die Werte der betroffenen Variablen ausgewiesen, der nicht erfüllte logische Ausdruck sowie die Statement-Nummer des TST-Statements aus dem Protokoll.

```
MOD-PROC
-TST (C10.1 | ...9) & .Z1 'Variable C10 falsch' N7 C20
```

In diesem Beispiel ist ein spezieller Fehlertext vorgegeben zusammen mit den Variablenwerten von N7 und C20. Die Werte von C10 erscheinen dabei nicht. Die Fehlerliste könnte dann folgendermaßen aussehen:

```
                                              STMT
FALL-ID: 1672
   N7 = 10        Variable C10 falsch         25
   C20 = 3 8

FALL-ID: 1675
   N7 = Z0        Variable C10 falsch         25
   C20 =
```

UPD-PROC

Die Folgestatements von UPD-PROC verarbeiten die statistischen Fälle, zu deren Fall-Identifikation sowohl Daten aus INPUT-EXT als auch aus INPUT-INT oder INPUT-SEC vorliegen.

Für die Bearbeitung statistischer Fälle, zu denen nur Daten aus INPUT-EXT und keine aus INPUT-INT oder INPUT-SEC vorliegen, ist das Statement ADD-PROC zuständig. Es kann daher nie vorkommen, dass für einen statistischen Fall gleichzeitig ADD-PROC und UPD-PROC wirksam werden.

In einem Verarbeitungslauf darf das Statement UPD-PROC nur einmal vorkommen.
Es besitzt selbst keine weiteren Angaben, dafür aber beliebig viele Folgestatements.
Diese müssen direkt hinter UPD-PROC liegen und mit einem Strich - am Anfang der Zeile beginnen.
Die Folgestatements sind in der Reihenfolge einzugeben, in der sie später ausgeführt werden sollen:
Zur Verarbeitung werden zunächst die Daten aus INPUT-INT eingelesen, die dort in Form von Variablen gespeichert sind. Danach können die Variablen durch die Folgestatements neue Werte erhalten.
Der Wert der Variablen aus INPUT-INT kann also mit den Daten aus INPUT-EXT verändert werden.

–Cn = Merkmalswerte
Die Bedingungsvariable Cn erhält Merkmalswerte zugewiesen.

–Nn = numerischer Wert
Die numerische Variable Nn erhält einen Wert zugewiesen.

–Tn = Textwert
Die Textvariable Tn erhält einen Wert zugewiesen.

Genauere Angaben zu diesen Folgestatements befinden sich im Abschnitt Wertezuweisungen .

–DO **–EDO**
Beginn und Ende einer Schleife: Die zwischen DO und EDO liegenden Statements werden mehrfach ausgeführt. Siehe DO-Statement.

–IF logischer Ausdruck **–EIF**
Die Verarbeitung der Statements zwischen IF und EIF hängt ab vom Wert des logischen Ausdrucks. Siehe IF-Statement.

–LANG Sprachschlüssel zur Auswahl von Variablentexten. Siehe LANG-Statement.

–PRT <Variable> <Text>
Dieses Statement druckt Variablenwerte und feste Texte aus. Siehe PRT-Statement.

–REPORT Ausgabe in Textdatei. Siehe REPORT-Statement.

–CLEAN logischer Ausdruck <**HEAD=**'text'>
Dieses Statement dient zur Bereinigung fehlerhafter Variablenwerte durch hinter CLEAN liegende Wertezuweisungen. Diese Wertezuweisungen werden nur ausgeführt, wenn der logische Ausdruck erfüllt ist. Im Zählprotokoll gibt eine CLEAN-Statistik eine Übersicht über die ausgeführten Korrekturen.

–TST logischer Ausdruck <Variable> <Text>
Dieses Statement prüft für die einzelnen statistischen Fälle, ob der logische Ausdruck erfüllt ist. Wenn nicht, wird der Text zusammen mit den Werten der angegebenen Variablen als Fehlermeldung ausgedruckt. Siehe TST-Statement.

–END < **CASE** | **ALL** >

Dieses Statement beendet die Verarbeitung des laufenden Falles in UPD-PROC.
Die hinter END stehenden Folgestatements werden nicht mehr ausgeführt.
Ohne weitere Angaben wird die Verarbeitung des laufenden Falles hinter den Folgestatements von UPD-PROC fortgesetzt.
CASE beendet darüber hinaus die Verarbeitung des laufenden Falles durch alle noch folgenden Statements.
ALL bricht die gesamte Verarbeitung vorzeitig ab.

Darüber hinaus können aber auch die Daten in den gerade eingelesenen Datensätzen durch Folgestatements von UPD-PROC verändert oder überschrieben werden. Siehe Abschnitt Externe Datenfelder und Wertezuweisungen . Die entsprechenden Statements haben folgende Form:

–A ... **=** Textwert
Es wird Text in den gerade eingelesenen Datensatz gestellt.

–AK ... **=** Textwert
Es wird Text in den gerade eingelesenen Datensatz einer CSV-Datei gestellt, mit Schlüsselwort und Gleichheitszeichen davor.

–AS ... **=** Textwert
Es wird Text in den gerade eingelesenen Datensatz einer CSV-Datei gestellt.

–B ... **=** Merkmalswerte
Es werden Merkmalswerte als einzelne Bits in den gerade eingelesenen Datensatz gestellt.

–D ... **=** numerischer Wert
Es wird ein numerischer Wert als Dezimalzahl in den gerade eingelesenen Datensatz gestellt.

–DK ... **=** numerischer Wert
Es wird ein numerischer Wert als Dezimalzahl in den gerade eingelesenen Datensatz einer CSV-Datei gestellt, mit Schlüsselwort und Gleichheitszeichen davor.

–DS ... **=** numerischer Wert
Es wird ein numerischer Wert als Dezimalzahl in den gerade eingelesenen Datensatz einer CSV-Datei gestellt.

–F ... **=** numerischer Wert
Es wird eine Dualzahl mit Vorzeichen (Integer) in den gerade eingelesenen Datensatz gestellt.

–G ... **=** numerischer Wert
Es wird eine Dualzahl ohne Vorzeichen (unsigned Integer) in den gerade eingelesenen Datensatz gestellt.

–I ... **=** Merkmalswerte
Es werden Merkmalswerte als positive Dualzahlen in den gerade eingelesenen Datensatz gestellt.

–K... **=** Merkmalswerte
Es werden Merkmalswerte im Column-Binary-Format in den gerade eingelesenen Datensatz gestellt.

–L ... **=** Merkmalswerte
Es werden Merkmalswerte 1, ... 9, 0, -, & in den gerade eingelesenen Datensatz gestellt.

–M ... **=** Merkmalswerte
Es werden Merkmalswerte in Form von Dezimalzahlen in den gerade eingelesenen Datensatz gestellt.

–MK ... **=** Merkmalswerte
Es werden Merkmalswerte in Form von Dezimalzahlen in den gerade eingelesenen Datensatz einer CSV-Datei gestellt , mit Schlüsselwort und Gleichheitszeichen davor.

–MS ... **=** Merkmalswerte
Es werden Merkmalswerte in Form von Dezimalzahlen in den gerade eingelesenen Datensatz einer CSV-Datei gestellt.

–P ... **=** numerischer Wert
Es wird ein numerischer Wert als gepackte Dezimalzahl in den gerade eingelesenen Datensatz gestellt.

–U ... **=** Merkmalswerte
Es werden Merkmalswerte als Zeichen 0 und 1 in den gerade eingelesenen Datensatz gestellt.

–UK ... **=** Merkmalswerte
Es werden Merkmalswerte als Zeichen 0 und 1 in den gerade eingelesenen Datensatz einer CSV-Datei gestellt, mit Schlüsselwort und Gleichheitszeichen davor.

–US ... **=** Merkmalswerte

Es werden Merkmalswerte als Zeichen 0 und 1 in den gerade eingelesenen Datensatz einer CSV-Datei gestellt

Weitere Angaben über das Zusammenspiel von UPD-PROC mit anderen Statements befinden sich im Abschnitt *Verarbeitung einzelner statistischer Fälle*.

Beispiel

```
INPUT-EXT    FNAME=EINGABE.EXT    ID=D(1..4)
INPUT-INT    FNAME=EINGABE.INT
...
UPD-PROC
-N7=D1(5..8)
```

Das Statement INPUT-EXT fordert eine externe Eingabedatei an und das Statement INPUT-INT eine interne Eingabedatei. Um die Datensätze aus den beiden Dateien verarbeiten zu können, müssen alle Eingabesätze eine Fall-Identifikation besitzen.

Liegt ein Fall aus INPUT-EXT zur Verarbeitung vor und ein dazu passender Fall aus INPUT-INT mit gleicher Fall-Identifikation, so wird er durch UPD-PROC und seine Folgestatements verarbeitet.

Wird dagegen kein passender Fall auf INPUT-INT gefunden, so regelt das Statement ADD-PROC die Verarbeitung.

Mit dem Folgestatement von UPD-PROC wird aus den Bytes 5 bis 8 des Datensatzes aus INPUT-EXT eine maximal vierstellig Zahl gelesen und der Variablen N7 zugewiesen.
Da INPUT-EXT keine Angabe zum Dateiformat enthält, müssen die Werte im D-Feld auf PCs und UNIX-Rechnern im ASCII-Format vorliegen und auf Großrechnern im EBCDIC-Format. Das gleiche gilt für die Ziffern der Fall-Identifikation ID.

WEIGHT

Das Statement WEIGHT ermittelt anhand vorgegebener Soll-Werte Repräsentanzgewichte für die einzelnen statistischen Fälle und stellt sie in eine numerische Variable.
Dazu sind Wichtungsbedingungen aus Bedingungsvariablen mit den zugehörigen Soll-Werten anzugeben. Zum Beispiel:

```
                    10 12 17 12
```

für eine Variable C1 mit zwei Ausprägungen und eine Variable C2 mit vier Ausprägungen.

Eine solche Wichtungsbedingung kann aus maximal zehn Bedingungsvariablen bestehen. Bis zu 30 solcher Wichtungsbedingungen sind in einem WEIGHT-Statement möglich. Das Produkt aus der Anzahl der Sollwerte aller Wichtungsbedingungen eines WEIGHT-Statements darf allerdings nicht über 2 000 000 liegen.

Liegt nur eine Wichtungsbedingung vor (Zellengewichtung), so gibt es stets eine einzige Lösung der Wichtungsaufgabe.

Bei mehreren Wichtungsbedingungen (Randgewichtung) existieren meistens sehr viele Lösungen. Das Programm wählt mit mathematischen Optimierungsverfahren solche Lösungen aus, die die erhobenen Daten möglichst wenig verfälschen, also sehr große und sehr kleine Gewichte vermeiden. WEIGHT liefert dabei exakte Lösungen der Optimierungsaufgaben, ohne Näherungsverfahren zu verwenden, die oft wegen zu weniger Iterationsschritte keine optimalen Ergebnisse erzeugen. Wenn eine optimale Lösung existiert, so wird diese auch gefunden. Mit einer innovativen Zielfunktion für diese Optimierung (FUNCTION=1) vermeidet WEIGHT sehr kleine Gewichte.

Reihenfolge der Verarbeitung

Die Gewichte werden direkt nach der Verarbeitung von ADD-PROC, MOD-PROC, UPD-PROC, FUSION und SELECT ermittelt. Das Statement SELECT kann dafür benutzt werden, die Gewichtung auf eine Teilstichprobe zu beschränken.

Die durch WEIGHT ermittelten Gewichte stehen zur Verarbeitung durch XTAB, CODEBOOK, HOLECOUNT, OUTPUT-EXT, OUTPUT-INT bereits im gleichen Verarbeitungslauf zur Verfügung.

In einem Verarbeitungslauf sind beliebig viele WEIGHT Statements zulässig. Sie werden in der Reihenfolge der Eingabe abgearbeitet. Dabei kann jedes WEIGHT Statement mit der Angabe OLD auf die Fallgewichte zurückgreifen, die in davor liegenden WEIGHT Statements ermittelt wurden.

Wenn die Soll-Werte nicht erreicht werden ...

Folgende Gründe können zu Abweichungen von den gewünschten Soll-Werte führen:

- Es wurden zu enge Mindest- oder Höchstwerte MIN oder MAX für die Gewichte vorgegeben.
- Die Zielvariable aus NEW oder NEWFACT. besitzt zu wenige Dezimal- oder Nachkommastellen.
- Die Soll-Werte weichen zu stark von den Ist-Werten der Stichprobe ab.

Sind solche Abweichungen unbedingt zu vermeiden, so sollten weniger wichtige Randbedingungen in ein erstes WEIGHT-Statement verlagert werden. Das dort erstellte Gewicht ist in einem zweiten WEIGHT-Statement mit OLD als Vorgewicht anzufordern. Dieses zweite WEIGHT-Statement sollte die exakt einzuhaltenden Soll-Vorgaben enthalten.

Auf diese Weise entstehen in der Regel noch brauchbare Ergebnisse, auch wenn das zweite WEIGHT-Statement die Gewichte des ersten etwas verfälscht.

Folgende Angaben sind im Statement WEIGHT möglich:

Ca **C**a***C**b***** ...	**=** w w w ...

Hier werden Wichtungsbedingungen als Bedingungs-Variable mit Soll-Werten *w* zu den einzelnen Merkmalen vorgegeben.

Die einfache Wichtungsbedingung

```
C100=10 22 25 33 11
```

zum Beispiel ordnet den 5 Merkmalen der Variablen C100 Sollwerte zu.
Das Merkmal C100.1 erhält den Sollwert 10 und das Merkmal C100.5 den Sollwert 11.

Die kombinierte Wichtungsbedingung

```
C1*C2=21.5   8.1   7.8   9.2
      10.5  12.5  17.8  12.6
```

geht von einer Variablen C1 mit 2 Merkmalen und einer Variablen C2 mit 4 Merkmalen aus.
Die Kombination der beiden Merkmale C1.1 und C2.1 erhält den Soll-Wert 21.5,
die Kombination C1.1 und C2.2 den Wert 8.1 bis hin zu den Merkmalen C1.1 und C2.4 mit dem Sollwert 9.2. Danach erhält die Kombination C1.2 und C2.1 den Soll-Wert 10.5 usw.

Eine kombinierte Wichtungsbedingung darf aus maximal 10 Bedingungs-Variablen bestehen.
Bis zu 100 Wichtungsbedingungen sind in einem WEIGHT-Statement möglich.

Enthält ein WEIGHT-Statement nur eine Wichtungsbedingung, so liegt eine Zellengewichtung vor. Bei zwei oder mehr Wichtungsangaben ist es eine Randgewichtung, zu deren Lösung das Programm die mit der Angabe FUNCTION ausgewählte Optimierungs-Strategie verwendet.

Die in den Wichtungsbedingungen verwendeten C-Variablen dürfen weder Mehrfachnennungen besitzen noch Fälle ohne Angabe. CNTA protokolliert solche Fehler und beendet die Verarbeitung vorzeitig.

w Diese Soll-Werte müssen größer oder gleich Null sein. Sie dürfen aus maximal 9 Ziffern mit bis zu 8 Nachkommastellen bestehen. Als Dezimalzeichen ist der Punkt zu verwenden. Die Soll-Werte sind im Statement wahlweise durch Kommas oder Leerstellen voneinander zu trennen.

Die Soll-Werte werden als Verhältniszahlen verstanden. Es sind nach Belieben Prozentwerte, Absolutwerte oder andere Verhältniszahlen möglich. Die Wichtung wird stets so durchgeführt, dass die gewichteten Zählergebnisse im gleichen Verhältnis zueinander stehen, wie die entsprechenden Soll-Werte. Die Gesamtzahl der gewichteten Fälle stimmt in der Regel mit der Anzahl ungewichteter Fälle überein. Durch die Angabe TOTAL kann jedoch hiervon abgewichen werden.

MAX = *m*

Diese Angabe begrenzt die Fallgewichte nach oben. Der Wert *m* muss größer als MIN sein und darf bis zu 9 Ziffern besitzen.

Zum Beispiel verhindert MAX=5.25 Fallgewichte über 5.25.

Neben MAX bestimmen die Zielvariablen NEW= ... und NEWFACT= ... das maximale Fallgewicht: Ist die Zielvariable unter DEFS zum Beispiel durch –N1:3,2 definiert, so wird der größtmögliche Wert 9.99 dieser Variablen zum maximalen Fallgewicht.
Fehlt die Angabe MAX, so bestimmt die Zielvariable alleine das maximale Fallgewicht.

Enthält das Statement WEIGHT die Angabe OLD oder TOTAL, so bezieht sich das Gewicht MAX auf das neue Gewicht einschließlich des alten Gewichts OLD und der Hochrechnung auf TOTAL.

Zu kleine maximale Werte können Effektivität und Informationsgehalt der Wichtungsergebnisse verschlechtern.

MIN = *m*

Diese Angabe begrenzt die Fallgewichte nach unten. Der Wert *m* muß größer als 0 und kleiner als MAX sein und darf bis zu 9 Ziffern besitzen.

Zum Beispiel verhindert MIN=0.12 Fallgewichte unter 0.12.

Neben MIN bestimmen die Zielvariablen NEW= ... und NEWFACT= ... das minimale Fallgewicht: Ist die Zielvariable unter DEFS durch –N1:3,2 definiert, so wird der kleinstmögliche positive Wert 0.01 dieser Variablen zum minimalen Fallgewicht. Fehlt die Angabe MIN, so bestimmt die Zielvariable alleine das minimale Fallgewicht.

Enthält das Statement WEIGHT die Angabe OLD oder TOTAL, so bezieht sich das Gewicht MIN auf das neue Gewicht einschließlich des alten Gewichts OLD und der Hochrechnung auf TOTAL.

Zu große minimale Werte können Effektivität und Informationsgehalt der Wichtungsergebnisse verschlechtern.

TOTAL = *x*

Hier kann die gewünschte Gesamtzahl *x* der gewichteten Fälle nach der Durchführung von WEIGHT festgelegt werden. Für *x* ist ein Zahlenwert aus maximal 18 Ziffern auch mit Nachkommastellen zulässig. Als Dezimalzeichen ist der Punkt zu verwenden.

Fehlt diese Angabe, so stimmt die gewichtete Fallzahl nach WEIGHT mit der Anzahl eingelesener Fälle überein. Ist die Angabe OLD vorhanden, so stimmt die gewichtete Fallzahl mit der Anzahl der eingelesenen Fälle überein, die mit diesem alten Gewicht ermittelt wurde.

TOTAL sorgt für einen festen Faktor, mit dem die durch WEIGHT ermittelten Gesamtgewichte zum Abschluss multipliziert werden. Die neuen Fallgewichte werden *nach* Multiplikation mit diesem Faktor auf die durch MIN und MAX vorgegebenen Werte beschränkt.

NEW = Nn

Diese Angabe legt die numerische Variable *Nn* fest, die die neuen Fallgewichte aufnehmen soll. Es ist darauf zu achten, dass diese Variable genügend Vor- und Nachkommastellen für die Fallgewichte besitzt. Siehe dazu auch die Angaben MIN und MAX.

Ist im Statement WEIGHT auch ein altes Fallgewicht OLD angegeben, so ist dieses alte Gewicht bereits im neuen Fallgewicht der Variablen aus NEW enthalten. Siehe dazu auch die Angabe NEWFACT.

Mindestens eine der Angaben NEW oder NEWFACT muss im Statement WEIGHT vorhanden sein. Beide sind auch gleichzeitig möglich.

NEWFACT = Nn

Enthält das Statement WEIGHT ein altes Fallgewicht OLD, so erhält die Variable *Nn* aus NEW die endgültigen Wichtungsfaktoren einschließlich des alten Fallgewichts. Es ist darauf zu achten, dass diese Variable genügend Vor- und Nachkommastellen für die Fallgewichte besitzt. Siehe dazu auch die Angaben MIN und MAX.

Mit NEWFACT werden dagegen nur die Faktoren in die Variable *Nn* ausgegeben, mit denen die Vorgewichte OLD zu multiplizieren sind, um das um das endgültige Fallgewicht zu erreichen.

Mindestens eine der Angaben NEW oder NEWFACT muss im Statement WEIGHT vorhanden sein. Beide sind auch gleichzeitig möglich.

Ohne ein altes Fallgewicht OLD liefern NEW und NEWFACT die gleichen Gewichte.

OLD = Na

Hier kann ein bereits bestehendes Fallgewicht in Form einer numerischen Variablen *Na* angegeben werden. Die Ist-Werte werden beim Einlesen der Daten mit den Werten dieser Variablen gewichtet. Ein altes Fallgewicht kann bereits bei der Datenerfassung ermittelt worden sein, zum Beispiel als Haushaltsgewicht. Es kann seine Werte aber auch durch die Angabe NEW=Na in einem vorausgegangenen WEIGHT Statement erhalten haben.

Das Optimierungsverfahren von CNTA versucht die Wichtungsfaktoren, mit denen Na zu multiplizieren ist, möglichst nahe bei 1 zu halten. Ohne die Angabe OLD werden die resultierenden Gesamtgewichte selbst nahe bei 1 gehalten.

Ein altes Gewicht OLD darf auch den Wert 0 oder Z0 (keine Angabe) besitzen. Solche statistischen Fälle werden aus dem Wichtungsverfahren ausgeschlossen und erhalten als neues Fallgewicht den Wert 0. Negative Werte im alten Fallgewicht *Na* gelten dagegen als Fehler; sie werden protokolliert und führen zum vorzeitigen Ende der Verarbeitung.

FUNCTION = | **E** | **S** | **M** | **I** < , n > |

Diese Angabe bestimmt das Verfahren zur Ermittlung der Randgewichte. Die folgenden Funktionen arbeiten mit mathematischen Optimierungsverfahren, die ein optimales Ergebnis im Sinne der jeweiligen Zielfunktion liefert:

FUNCTION=E arbeitet mit der einfachen symmetrischen Zielfunktion. Sie liefert die besten Effektivitätswerte, nimmt dafür jedoch sehr kleine Gewichte in der Nähe von Null in Kauf sowie größere Informationsverluste im Sinne der Shannon-Entropie.

FUNCTION=S liefert Wichtungsergebnisse mit dem kleinstmöglichen Informationsverlust im Sinne der Shannon-Entropie. Dieses Verfahren vermeidet kleine Gewichte, zeigt aber etwas geringere Effektivitätswerte im Vergleich zu FUNCTION=E.

FUNCTION=M vermeidet vorrangig kleine Gewichte in der Nähe von Null bei leichten Effektivitätsverlusten im Vergleich zur FUNCTION=E und Informationsverlusten im Vergleich zu FUNCTION=S.

Bei sehr umfangreichen Wichtungsaufgaben mit vielen Randbedingungen führen diese Optimierungsverfahren manchmal langen Laufzeiten oder Speichermangel . Dann empfiehlt sich:

FUNCTION=I verzichtet auf die mathematischen Optimierungen bei geringerem Laufzeit- und Speicherbedarf. Sie liefert bei nicht vollständig lösbaren Wichtungsaufgaben oft geringere Soll-Ist-Abweichungen als die optimierenden Funktionen, achtet jedoch nicht auf Effektivität und Informationsgehalt.
Das Verfahren arbeitet iterativ bis eine Lösung des Wichtungsaufgabe erreicht ist.
Die Angabe *n* hinter I legt die maximale Anzahl von Iterationen fest, bei denen die Suche nach einer Lösung abgebrochen wird. Ohne diese Angabe *n* wird mit maximal 50 000 Iterationen gearbeitet.

Fehlt die Angabe FUNCTION, so wird zunächst FUNCTION=E verwendet. Entstehen dabei Soll-Ist-Abweichungen, so wird versucht, diese mit FUNCTION= I zu reduzieren.

Einzelheiten zu diesen Funktionen werden weiter hinten beschrieben.

SELECT = *logischer Ausdruck*

Nur Fälle, die den logischen Ausdruck erfüllen, nehmen an der Wichtung teil. In allen anderen Fällen wird der Wert der Gewichtsvariablen aus NEW oder NEWFACT nicht verändert.
Zum Beispiel vergibt

```
SELECT=C10.1
```

nur denjenigen Fällen Gewichte, die die Bedingung C10.1 erfüllen.

Beschreibung der Wichtungsverfahren

FUNCTION = E

Es seien

- z Anzahl Ist-Wert-Zellen zu allen Kombinationen der Randbedingungen mit Werten ungleich 0
- a_i die durch den Parameter OLD gewichtete Anzahl der statistischen Fälle einer Ist-Wert-Zelle
- x_i die gesuchten Wichtungsfaktoren, mit denen $a_i\ x_i$ die vorgegebenen Sollwerte erreicht
- m das durch MIN bestimmte Minimalgewicht
- M das Maximalgewicht aus MAX.
- s ist die Summe der mit OLD gewichteten Fälle.

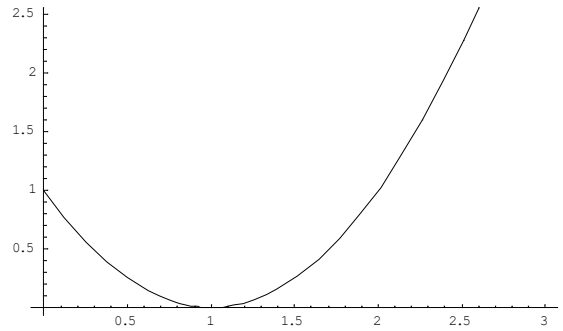
Dann liefert dieses Verfahren eine Lösung, die die folgenden Bedingungen erfüllt:

$$a_i\ x_i \geq m$$

$$a_i\ x_i \leq M$$

$$f(x_1 \ldots x_z) = \sum_{i=1\ldots z} a_i(x_i - 1)^2 = min$$

Die Zielfunktion $f(x_1 \ldots x_z)$ zeigt im Prinzip folgenden Verlauf. Sie gibt an, wie stark ein Gewicht der x-Achse bei der Optimierung vermieden werden soll.



Dieses Verfahren ist weit verbreitet, da die zugehörige Optimierungsaufgabe mit einfachen mathematischen Mitteln zu lösen ist. Es liefert stets die besten Effektivitätswerte, da diese genau für dieses Verfahren definiert sind.
Im Vergleich zu FUNCTION=I und FUNCTION=S werden jedoch sehr kleine Gewichte in der Nähe von Null akzeptiert sowie größere Informationsverluste im Sinne der Shannon-Entropie.

FUNCTION = S

Es seien

- z Anzahl Ist-Wert-Zellen zu allen Kombinationen der Randbedingungen mit Werten ungleich 0
- a_i die durch den Parameter OLD gewichtete Anzahl der statistischen Fälle einer Ist-Wert-Zelle
- x_i die gesuchten Wichtungsfaktoren, mit denen $a_i\ x_i$ die vorgegebenen Sollwerte erreicht
- m das durch MIN bestimmte Minimalgewicht
- M das Maximalgewicht aus MAX.
- s ist die Summe der mit OLD gewichteten Fälle.

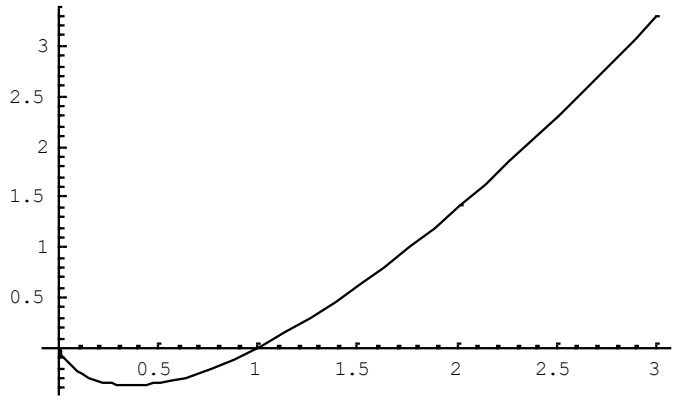
Dann liefert dieses Verfahren eine Lösung, die die folgenden Bedingungen erfüllt:

$$a_i\ x_i \geq m$$

$$a_i\ x_i \leq M$$

$$f(x_1 \ldots x_z) = \sum_{i=1\ldots z} \frac{a_i x_i}{s} \ln(x_i) = min$$

Die Zielfunktion $f(x_1 \ldots x_z)$ zeigt im Prinzip folgenden Verlauf. Sie gibt an, wie stark ein Gewicht der x-Achse bei der Optimierung vermieden werden soll.



Dieses Verfahren arbeitet nach dem Prinzip des minimalen Informationsverlusts im Sinne der Shannon-Entropie.
Siehe GABLER, HOFFMEYER-ZLOTNIK, KREBS: Gewichtung in der Umfragepraxis: Kriterien der Gewichtung einer nationalen Bevölkerungsstichprobe. *Springer Fachmedien Wiesbaden* und JOACHIM MERZ: Die konsistente Hochrechnung von Mikrodaten nach dem Prinzip des minimalen Informationsverlusts. *Allgemeines Statistisches Archiv 67: 342-366.*

Dies Verfahren liefert die geringsten Informationsverluste im Sinne der Shannon-Entropie.
Es vermeidet im Vergleich zu FUNCTION=E sehr kleine Gewichte nahe 0, liefert jedoch eine geringere Effektivität.

FUNCTION = M

Es seien

- z Anzahl Ist-Wert-Zellen zu allen Kombinationen der Randbedingungen mit Werten ungleich 0
- a_i die durch den Parameter OLD gewichtete Anzahl der statistischen Fälle einer Ist-Wert-Zelle
- x_i die gesuchten Wichtungsfaktoren, mit denen $a_i\ x_i$ die vorgegebenen Sollwerte erreicht
- m das durch MIN bestimmte Minimalgewicht
- M das Maximalgewicht aus MAX.
- s ist die Summe der mit OLD gewichteten Fälle.

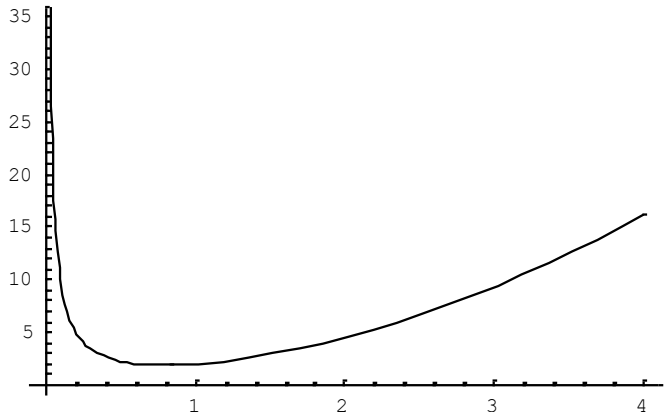
Dann liefert dieses Verfahren eine Lösung, die die folgenden Bedingungen erfüllt:

$$a_i\ x_i \geq m$$

$$a_i\ x_i \leq M$$

$$f(x_1 \ldots x_z) = \sum_{i=1\ldots z} a_i\ (x_i{}^2 + \frac{1}{x_i}) = min$$

Die Zielfunktion $f(x_1 \ldots x_z)$ zeigt im Prinzip folgenden Verlauf. Sie gibt an, wie stark ein Gewicht der x-Achse bei der Optimierung vermieden werden soll.



Dieses CNT-spezifische Verfahren vermeidet kleine Gewichte nahe bei 0 im Vergleich zu FUNCTION=E und reduziert die Effektivität etwas weniger als FUNCTION=S. Es stellt einen Kompromiss zwischen den beiden vorigen dar.

FUNCTION = I

Diese Funktion arbeitet nach dem heuristischen Prinzip der iterativen Zellengewichtung.
Sie benötigt bei größeren Wichtungsaufgaben weniger Laufzeit und Hauptspeicher.
Bei nicht vollständig lösbaren Wichtungsaufgaben liefert sie oft geringere Soll-Ist-Abweichungen als die optimierenden Funktionen, achtet jedoch nicht auf Effektivität und Informationsgehalt.

Bewertung der Wichtungsergebnisse

Zur Messung der Effektivität der Lösungen für die Gewichte ohne OLD:

$$\frac{(\sum_{i=1\ldots z} a_i x_i)^2}{s \sum_{i=1\ldots z} a_i {x_i}^2} 100$$

Zur Messung der Effektivität der Lösungen für die Gewichte mit OLD:

$$\frac{(\sum_{i=1\ldots z} a_i x_i)^2}{n \sum_{i=1\ldots z} (a_i x_i)^2} 100$$

Informationsverlust im Sinne der Shannon-Entropie für die Gewichte ohne OLD:

$$V = \sum_{i=1\ldots z} \frac{a_i x_i}{s} \ln (x_i)$$

Informationsverlust im Sinne der Shannon-Entropie für die Gewichte mit OLD:

$$V = \sum_{i=1\ldots z} \frac{a_i x_i}{s} \ln (o_i x_i \frac{n}{s})$$

Informationsgehalt im Zählprotokoll :

$$\begin{cases} (1 - V) \cdot 100 & f\ddot{u}r\ V \leq 1 \\ 0 & f\ddot{u}r\ V > 1 \end{cases}$$

Dabei ist

- z Anzahl Ist-Wert-Zellen zu allen Kombinationen der Randbedingungen mit Werten ungleich 0
- n Anzahl der Fälle ungewichtet
- a_i die mit OLD und TOTAL gewichtete Anzahl Fälle der Ist-Wert-Zelle i
- o_i das Gewicht OLD der Ist-Wert-Zelle i
- x_i die gesuchten Wichtungsfaktoren, mit denen $a_i\ x_i$ die vorgegebenen Sollwerte erreicht
- s ist die Summe der mit OLD und TOTAL gewichteten Fälle.

Siehe GABLER, HOFFMEYER-ZLOTNIK, KREBS: Gewichtung in der Umfragepraxis: Kriterien der Gewichtung einer nationalen Bevölkerungsstichprobe. *Springer Fachmedien Wiesbaden* und JOACHIM MERZ: Die konsistente Hochrechnung von Mikrodaten nach dem Prinzip des minimalen Informationsverlusts. *Allgemeines Statistisches Archiv 67: 342-366.*

Vergleich von Wichtungsergebnissen

Die folgenden Auszüge aus Zählprotokollen zeigen die typische Wirkung verschiedener Wichtungsfunktionen FUNCTION bei gleicher Aufgabenstellung.

```
FUNCTION=E

Effektivität                                                  52.7
Informationsgehalt nach Shannon                               81.3
Standardabweichung                                           0.948

Prozent der Fälle
    |
    |          .
    |          X
 20 +          X
    |          X
    |         :X.
    |         XXX
    |         XXX
 10 +         XXX
    |         XXX.                                        . < 1%
    |       .:XXXX.                                       : < 2%
    |   :   XXXXXXX:   :                                  X = 2%
    |  :X:::XXXXXXXX:::X...:  . .        . .
  0 --+----+----+----+----+--+----+----+----+----+----+->  Gewicht
      0  0.5    1  1.5    2  5   10   15   20   25   30
```

```
FUNCTION=S

Effektivität                                                  52.6
Informationsgehalt nach Shannon                               81.7
Standardabweichung                                           0.950

Prozent der Fälle
    |
    |          .
    |          X
 20 +         .X
    |         XX
    |         XX
    |         XX:
    |         XXX
 10 +         XXX
    |         XXX                                         . < 1%
    |    :  ::XXX:.                                       : < 2%
    |    X  XXXXXXX:.  .                                  X = 2%
    |   :X::XXXXXXXXX..X:..:... .        . .
  0 --+----+----+----+----+--+----+----+----+----+----+->  Gewicht
      0  0.5    1  1.5    2  5   10   15   20   25   30
```

```
FUNCTION=M

Effektivität                                                  52.5
Informationsgehalt nach Shannon                               81.6
Standardabweichung                                           0.952

Prozent der Fälle
    |
    |
    |          :
 20 +         :X
    |         XX
    |         XX
    |         XX.
    |         XXX
 10 +        .XXX
    |    :   XXXX                                         . < 1%
    |    X   XXXX:.                                       : < 2%
    |    X. :XXXXXX:    .                                 X = 2%
    |    XX XXXXXXX::..X..:. . .        . .
  0 --+----+----+----+----+--+----+----+----+----+----+->  Gewicht
      0  0.5    1  1.5    2  5   10   15   20   25   30
```

Beispiel Zellengewichtung

```
WEIGHT  C108*C1011=14 16 20 10 20 20
                  11 20 25 30 8  6
        NEW=N1
```

Das Statement WEIGHT enthält nur eine Wichtungsbedingung C108*C1011, verlangt also eine Zellengewichtung, bei der jede Merkmalskombination einen Sollwert verlangt. In diesem Beispiel wird unterstellt, dass die Variable C108 über zwei Merkmale verfügt und die Variable C1011 über sechs Merkmale. Daraus ergeben sich 12 Wichtungszellen, deren Soll-Werte hinter dem Gleichheitszeichen anzugeben sind.
Entscheidend bei diesen Sollwerten ist ihr Verhältnis zueinander. Dies dürfen Absolut- aber auch Prozentwerte sein, die sich nicht unbedingt auf 100 aufsummieren müssen.

Der erste Sollwert 14 bezieht sich auf Merkmalskombination C108.1/C1011.1, der folgende 16 auf die Kombination C108.1/C1011.2 bis hin zu 6 für C108.2/C1011.6.

Die Angabe NEW=N1 stellt die fertigen Gewichte der einzelnen statistischen Fälle in die Variable N1.

Das WEIGHT Statement erzeugt ein Protokoll, das folgendermaßen aussehen könnte:

```
------------------------------------------------------------------
                  WEIGHT Statement 13
------------------------------------------------------------------
C108*C1011:   Summe der Sollwerte im WEIGHT-Statement        200
Anzahl Eingabefälle                                       123210
Ausgabevariable NEW                                           N1
Kleinstes mögliches Gewicht gemäß NEW=N1                  0.0001
Größtes mögliches Gewicht gemäß NEW=N1                 99999.9999

Effektivität                                               82.7%
Informationsverlust im Sinne der Shannon-Entropie           7.6%
Standardabweichung der Ausgabegewichte NEW                 0.458

Prozent der Fälle
 30 +
    |           :
    |           X
    |           X
    |          . X
 20 +          X X :
    |          X X X
    |          X X X
    |          X X X
    |     :    X X X
 10 +     X   :X X X                                    . = ...0.9%
    |     X:  XX X X                                    : = 1%...1.9%
    |     XX  XX X X                                    X = 2%
    |     XX  XX X X       :
    |     XX  XX X X       X
  0 --+----+----+----+----+--+----+----+----+----+----+----+----+----+----+
      0  0.5    1  1.5    2  5   10   15   20   25   30   35   40   45   50   Gewicht

                                                    Ausgabegewichte
    Ist-Werte     Soll-Werte     Ergebnisse   Differenz   Minimum Maximum Mittel

  7692   6.2%   8625   7.0%   8625   7.0%    -      -     1.12    1.12    1.12   {Männer}{14 - 19 Jahre}
 11641   9.4%   9857   8.0%   9857   8.0%    -      -     0.85    0.85    0.85           {20 - 29 Jahre}
 14147  11.5%  12321  10.0%  12321  10.0%    -      -     0.87    0.87    0.87           {30 - 39 Jahre}
 14689  11.9%   6161   5.0%   6161   5.0%    -      -     0.42    0.42    0.42           {40 - 49 Jahre}
  9177   7.4%  12321  10.0%  12321  10.0%    -      -     1.34    1.34    1.34           {50 - 59 Jahre}
  4177   3.4%  12321  10.0%  12321  10.0%    -      -     2.95    2.95    2.95           {60 - 64 Jahre}
  7215   5.9%   6777   5.5%   6777   5.5%    -      -     0.94    0.94    0.94   {Frauen}{14 - 19 Jahre}
 11403   9.3%  12321  10.0%  12321  10.0%    -      -     1.08    1.08    1.08           {20 - 29 Jahre}
 14610  11.9%  15401  12.5%  15401  12.5%    -      -     1.05    1.05    1.05           {30 - 39 Jahre}
 14777  12.0%  18482  15.0%  18482  15.0%    -      -     1.25    1.25    1.25           {40 - 49 Jahre}
  9396   7.6%   4928   4.0%   4928   4.0%    -      -     0.52    0.52    0.52           {50 - 59 Jahre}
  4286   3.5%   3696   3.0%   3696   3.0%    -      -     0.86    0.86    0.86           {60 - 64 Jahre}

123210 100.0% 123210 100.0% 123210 100.0%    -      -     0.42    2.95    1.00   TOTAL
------------------------------------------------------------------------------------------------------------------
```

Die erste Protokollzeile zeigt zu den Wichtungsbedingung C108*C1011 die Summe 200 der Soll-Werte.
Danach erscheint die Gesamtzahl 123210 der gelesenen Fälle.

Es folgen drei Zeilen zur Ausgabevariablen NEW mit dem durch die Variable N1 bestimmten Wertebereich der Ausgabegewichte zwischen 0.0001 und 99999.9999.

Die anschließenden Werte geben Hinweise auf die Qualität der erzeugten Gewichte. Sie sind im obigen Abschnitt *Bewertung der Wichtungsergebnisse* beschrieben.
Die *Effektivität* lässt auf die Anzahl statistisch wirksamer Fälle nach der Wichtung schließen. Sie basiert auf der Veränderung der Varianzen durch die Gewichte.
Der *Informationsverlust im Sinne der Shannon-Entropie* liefert einen alternativer Hinweis auf die Veränderung der Daten durch die Wichtung. Gleiches gilt für die *Standardabweichung der Ausgabegewichte*.

Das Histogramm gibt eine Übersicht über die Verteilung der erzeugten Gewichte.

Die dahinter stehende Tabelle zeigt zu den Merkmalskombinationen von C108 und C1011.2 neben den Soll- und Istwerten die durch die Gewichte tatsächlich erreichten Ergebnisse. Die Spalte *Differenzen* zwischen den Ist- und Ergebniswerten ist hier leer. Sie enthält nur bei erfolglosen Randgewichtungen Werte.

Die letzten Spalten unter *Ausgabegewichte* liefern eine Übersicht über die zu den einzelnen Zellen erzeugten Gewichte, so wie sie in die Zielvariable NEW gestellt werden.

Beispiel: Randgewichtung

```
WEIGHT  C108*C1011=14 16 20 10 20 20
                  11 20 25 30  8  6
        C106=79 21
        NEW=N1
        MIN=0.3
```

Das Statement WEIGHT enthält die beiden Randbedingungen C108*C1011 und C106, verlangt also eine Randgewichtung, bei der die einzelnen Randbedingungen voneinander unabhängige Sollwerte erhalten.
In diesem Beispiel wird unterstellt, dass die Variable C108 über zwei Merkmale verfügt, die Variable C1011 über sechs und C106 über zwei Merkmale. Die erste Bedingung verlangt daher 12 Sollwerte und die zweite zwei. Entscheidend bei diesen Sollwerten ist ihr Verhältnis zueinander. Dies dürfen Absolut- aber auch Prozentwerte sein, die sich nicht unbedingt auf 100 aufsummieren müssen.
Der erste Sollwert 14 bezieht sich auf Merkmalskombination C108.1/C1011.1, der folgende 16 auf die Kombination C108.1/C1011.2 bis hin zu 6 für C108.2/C1011.6.
Die Angabe NEW=N1 stellt die fertigen Gewichte der einzelnen statistischen Fälle in die Variable N1.
MIN=0.3 verhindert Gewichte kleiner als 0.3.

Das WEIGHT Statement erzeugt ein Protokoll, das folgendermaßen aussehen könnte:

```
------------------------------------------------------------
                    WEIGHT Statement 17
------------------------------------------------------------
C108*C1011:   Summe der Sollwerte im WEIGHT-Statement         200
C106:         Summe der Sollwerte im WEIGHT-Statement         100
Anzahl Eingabefälle                                        123210
Ausgabevariable NEW                                           N1
Kleinstes mögliches Gewicht gemäß MIN                     0.3000
Kleinstes mögliches Gewicht gemäß NEW=N1                  0.0001
Größtes mögliches Gewicht gemäß NEW=N1                 99999.9999

FUNCTION=2
Effektivität                                               82.7%
Informationsverlust im Sinne der Shannon-Entropie           7.6%
Standardabweichung der Ausgabegewichte NEW                0.458

Prozent der Fälle
    |
    |
    |          :
 20 +          X
    |          X .
    |          X X
    |          X X
    |          X.X:
 10 +     :.   XXXX.                                    . = ...0.9%
    |     XX  :XXXXX                                   : = 1%...1.9%
    |     XX  XXXXXX                                   X = 2%
    |     XX  XXXXXX       :
    |     XX: XXXXXX:      X
  0 --+----+----+----+---+--+----+----+----+----+----+----+----+----+
      0  0.5    1  1.5    2  5   10   15   20   25   30   35   40   45   50   Gewicht

                                                  Ausgabegewichte
    Ist-Werte    Soll-Werte    Ergebnisse   Differenz   Minimum Maximum Mittel

  7692   6.2%   8625   7.0%   8625   7.0%    -      -     1.11    1.16    1.12   {Männer}{14 - 19 Jahre}
 11641   9.4%   9857   8.0%   9857   8.0%    -      -     0.84    0.88    0.85           {20 - 29 Jahre}
 14147  11.5%  12321  10.0%  12321  10.0%    -      -     0.86    0.91    0.87           {30 - 39 Jahre}
 14689  11.9%   6161   5.0%   6161   5.0%    -      -     0.41    0.46    0.42           {40 - 49 Jahre}
  9177   7.4%  12321  10.0%  12321  10.0%    -      -     1.33    1.38    1.34           {50 - 59 Jahre}
  4177   3.4%  12321  10.0%  12321  10.0%    -      -     2.94    2.99    2.95           {60 - 64 Jahre}
  7215   5.9%   6777   5.5%   6777   5.5%    -      -     0.93    0.98    0.94   {Frauen}{14 - 19 Jahre}
 11403   9.3%  12321  10.0%  12321  10.0%    -      -     1.07    1.12    1.08           {20 - 29 Jahre}
 14610  11.9%  15401  12.5%  15401  12.5%    -      -     1.05    1.09    1.05           {30 - 39 Jahre}
 14777  12.0%  18482  15.0%  18482  15.0%    -      -     1.24    1.29    1.25           {40 - 49 Jahre}
  9396   7.6%   4928   4.0%   4928   4.0%    -      -     0.51    0.56    0.52           {50 - 59 Jahre}
  4286   3.5%   3696   3.0%   3696   3.0%    -      -     0.85    0.90    0.86           {60 - 64 Jahre}

 98347  79.8%  97336  79.0%  97336  79.0%    -      -     0.41    2.94    0.99   {Deutschland West}
 24863  20.2%  25874  21.0%  25874  21.0%    -      -     0.46    2.99    1.04   {Deutschland Ost}

123210 100.0% 123210 100.0% 123210 100.0%    -      -     0.41    2.99    1.00   TOTAL
------------------------------------------------------------------------------------------------------
```

In den beiden ersten Protokollzeilen werden zu den Randbedingungen C108*C1011 sowie C106 die Summen der 200 und 100 der Soll-Werte ausgewiesen. Dahinter erscheint die Gesamtzahl der eingelesenen Fälle.

Es folgen Zeilen zur Ausgabevariablen NEW mit dem durch die Variable N1 bestimmten Wertebereich der Ausgabegewichte zwischen 0.0001 und 99999.9999. Die Angabe MIN=0.3 im WEIGHT-Statement begrenzt das kleinstmögliche Gewicht zusätzlich auf 0.3.

Die Angabe FUNCTION=2 zeigt das verwendete Optimierungsverfahren an.

Die anschließenden Werte geben Hinweise auf die Qualität der erzeugten Gewichte. Sie sind im obigen Abschnitt Bewertung der Wichtungsergebnisse genauer beschrieben.
Die *Effektivität* lässt auf die Anzahl statistisch wirksamer Fälle nach der Wichtung schließen. Sie basiert auf der Veränderung der Varianzen durch die Gewichte.
Der *Informationsverlust im Sinne der Shannon-Entropie* liefert einen alternativer Hinweis auf die Veränderung der Daten durch die Wichtung. Gleiches gilt für die *Standardabweichung der Ausgabegewichte*.

Das Histogramm gibt eine Übersicht über die Verteilung der erzeugten Gewichte.

Die Tabelle dahinter zeigt zu allen Merkmalskombinationen der Wichtungsbedingungen neben den Soll- und Istwerten die durch die Gewichte tatsächlich erreichten Ergebnisse. Die Spalte *Differenzen* zwischen den Soll- und Ergebniswerten ist hier leer. Sie enthält nur bei erfolglosen Randgewichtungen Werte.

Die letzten Spalten *Ausgabegewichte* liefern eine Übersicht über die zu den einzelnen Zellen erzeugten Gewichte, so wie sie in die Zielvariable NEW gestellt werden.

XADD XADDB

Die Statements XADD und XADDB erstellen Kreuztabellen, genau wie XTAB, drucken sie aber nicht.
Die Zählergebnisse werden stattdessen auf die Werte des vorausgehenden XTAB addiert, bevor die entsprechende Tabelle gedruckt wird. Zu einem XTAB Statement können beliebig viele XADD und XADDB Statements angegeben werden; sie müssen aber alle direkt hinter dem zugehörigen XTAB liegen.

Die Anzahl von Zeilen oder Spalten in XADD und XADDB darf nicht größer sein als im dazugehörigen XTAB, wohl aber kleiner. Die Addition beginnt links oben: Die erste Zeile und Spalte von XADD oder XADDB wird auf die erste Zeile und Spalte von XTAB addiert usw. Es werden immer ganze Zeilen und Spalten aus XADD und XADDB auf die entsprechenden Zeilen und Spalten von XTAB addiert.

Mit der Angabe ASTART kann diese Addition in XTAB aber auch nach rechts oder unten verschoben werden.

XTAB kann mehrere FILTER-Angaben enthalten. Jede dieser Angaben erzeugt eine eigene Tabelle.
Ebenso dürfen XADD und XADDB mehrere FILTER-Angaben enthalten. Aus jeder davon entsteht wieder eine Tabelle, die entsprechend aufaddiert wird. Dabei dürfen XADD und XADDB nicht mehr Tabellen erzeugen, als in XTAB vorliegen, wohl aber weniger.

Bei XADD erfolgt die Addition der Zählergebnisse, nachdem alle Rechenoperationen aus numerischen Ausdrücken ausgeführt sind, jedoch vor der Druckaufbereitung mit Prozentuierung, Indexbildung usw.
Siehe Abschnitt Numerische Ausdrücke bei XTAB unter COL/ROW/FILTER. Enthält XADD zum Beispiel die Angabe

```
XADD    ROW=MEAN(N1); N1*N2
```

so wird zunächst der Mittelwert von N1 errechnet, die Summen von N1 und N2 multipliziert und danach erst die Ergebnisse dieser Rechenschritte auf die Tabelle aus XTAB addiert.

Im Unterschied dazu werden bei XADDB zunächst die Teilergebnisse der Zählungen von XADDB auf XTAB addiert und erst danach die Rechenoperationen aus numerischen Ausdrücken ausgeführt.
Dies setzt voraus, dass XTAB und XADDB in den zu addierenden Zeilen und Spalten gleichartige numerische Ausdrücke besitzen. Wenn zum Beispiel XTAB die Angabe

```
XTAB      ROW=MEAN(N1); N1*N2
```

enthält, so ist dahinter die folgende Angabe zulässig:

```
XADDB   ROW=MEAN(N17); TOTAL*C1.1
```

Falsch wäre jedoch die Angabe:

```
XADDB   ROW=TOTAL; TOTAL-C1.1
```

weil in der ersten Zeile der Ausdruck MEAN(N1) in XTAB einen anderen Rechenvorgang verlangt als der einfache Ausdruck TOTAL in XADDB. Darüber hinaus verlangt XTAB in der zweiten Zeile eine Multiplikation, XADDB dagegen eine Subtraktion.
CNTA erkennt solche Fehler bei der Statementprüfung und gibt dazu entsprechende Fehlermeldungen aus.

Folgende Angaben sind in XADD und XADDB möglich:

| **COL** | **=** z **;** ... | Mit ROW und COL werden die Zeilen- und Spalten-Bedingungen für die
| **COLS** | | Kreuztabellen festgelegt und mit FILTER die Filter-Bedingungen.
Die Auswertungsregeln sind die gleichen wie bei XTAB.

| **ROW** | **=** z **;** ... | ROWS, COLS und FILTERS gelten für das laufende Statement und für alle
| **ROWS** | | folgenden XADD, XADDB und auch XTAB Statements.
Die Angaben ROW, COL und FILTER gelten nur für das laufende XADD oder

| **FILTER** | **=** z **;** ... | XADDB und haben Vorrang vor ROWS, COLS und FILTERS aus früheren
| **FILTERS** | | Statements. Ebenso gelten Angaben aus XTAB zu FILTERS, ROWS und
COLS auch für die folgenden XADD- und XADDB-Statements.

In XADD und XADDB müssen ROW und COL stets wirksam sein. Die Angabe FILTER kann fehlen; dann wird FILTER=TOTAL angenommen.
In den Zählausdrücken z dürfen hinter einem Doppelpunkt auch Texte und Druckregeln angegeben werden. Zum Drucken werden jedoch nur die Texte und Druckregeln des zugehörigen XTAB herangezogen.

| **ASTART** | = r, s | Diese Angabe legt fest, in welcher Zeile und Spalte des XTAB mit der Addition
| **ASTARTS** | | der ersten Zeile und Spalte des XADD oder XADDB begonnen werden soll.
Fehlt diese Angabe, so werden die Werte der ersten Zeile und Spalte von XADD oder XADDB auf die Werte der ersten Zeile und Spalte von XTAB addiert.
ASTART und ASTARTS erlauben es, von dieser Regel abzuweichen. ASTART gilt nur für das laufende Statement, ASTARTS dagegen auch für alle folgenden XADD oder XADDB Statements.

r Zeile 1...32767 von XTAB, auf die die Werte der ersten Zeile von XADD oder XADDB zu addieren sind. Die zweite Zeile von XADD oder XADDB wird dann auf die Zeile r+1 von XTAB addiert usw.

s Spalte 1...32767 von XTAB, auf die die Werte der ersten Spalte von XADD oder XADDB zu addieren sind. Die zweite Spalte von XADD oder XADDB wird dann auf die Spalte s+1 von XTAB addiert usw.

| **FADD** | Die Angabe FADD bewirkt, dass auch die Filterwerte von XADD auf die Filterwerte
| **FADDS** | von XTAB oberhalb der Tabelle addiert werden.
| **NOFADD** | Fehlt diese Angabe, so werden nur die Tabellenzähler, nicht jedoch die
| **NOFADDS** | Filterwerte addiert.
FADD gilt nur für das laufende XADD oder XADDB Statement, FADDS dagegen auch für alle folgenden XADD oder XADDB.
NOFADD macht die Angabe FADDS aus früheren Statements für das laufende Statement rückgängig.
NOFADDS macht die Angabe FADDS aus früheren Statements für das laufende und alle folgenden Statements rückgängig.

| **GROUP** | **=** | **ID** | Durch diese Angabe wird eine besondere Form der Gruppenverarbeitung
| **GROUPS** | | **ID1** | ermöglicht.
| **ID2** | Es gelten die gleichen Regeln, wie bei XTAB.
| **ID3** |
| **ID4** | GROUP gilt nur für das laufende Statement, GROUPS dagegen auch für alle folgenden XTAB, XADD und XADDB.

Umgekehrt ist die Angabe GROUPS aus einem XTAB Statement auch für alle folgenden XADD und XADDB wirksam.

Beispiel: Ganze Tabellen addieren

```
XTAB    FILTER=C1.1    ROW=TOTAL;C10      COL=TOTAL;C100
        PRINT=ABS
        PRINT1=PROW1,%
XADD    FILTER=C1.2    ROW=TOTAL;C11      COL=TOTAL;C101
```

In diesen Statements erzeugt zunächst XTAB eine Tabelle aus den Variablen C10 und C100, gefiltert mit C1.1 und XADD eine zweite Tabelle aus C11 und C101, gefiltert mit C1.2.

Anschließend addiert das Statement XADD seine Tabelle auf die des davorliegenden Statements XTAB. Schließlich wird diese aufaddierte Tabelle nach den Regeln von XTAB ausgegeben. Das Ergebnis könnte folgendermaßen aussehen:

	TOTAL	Anzeige der Marke gesehen	
		ja	nein
TOTAL	1204	1041	163
Bewertung der Marken A und B			
1	226	192	34
2	341	288	53
3	316	262	54
4	219	203	16
5	102	96	6

Beispiel: Unechte Kreuztabelle aus Einzelzeilen

```
XTAB   COL=TOTAL;C1
       ROW=TOTAL;0:'Bewertung Marke A',ABS,1;
                 0:'Bewertung Marke B',ABS,1
XADD   COL=TOTAL;C1   ROW=MEAN(N1)   ASTART=2,1
XADD   COL=TOTAL;C2   ROW=MEAN(N2)   ASTART=3,1
```

Diese Statements könnten die folgende unechte Kreuztabelle erzeugen:

	TOTAL	Anzeige der Marke gesehen	
		ja	nein
TOTAL	1204	1041	163
Bewertung Marke A	4,6	4,6	4,4
Bewertung Marke B	2,3	2,3	2,5

Im Statement XTAB liefert *COL=TOTAL;C1* die Spaltenbedingungen für die erste Tabellenzeile und die Texte für die Spaltenüberschriften.

ROW=TOTAL erzeugt die erste Zeile mit Anzahl aller Fälle von C1.

In den folgenden Zeilen sorgt 0: für zwei Zeilen, die beide keine Zählergebnisse sondern nur die Werte 0 enthalten. Hinter 0 stehen die Zeilentexte und die Angaben ABS,1 zur Aufbereitung der Ergebnisse als Absolutwerte mit einer Nachkommastelle.

Das erste XADD-Statement erzeugt mit *MEAN(N1)* nur eine Zeile mit den Mittelwerrten von N2 und addiert diese wegen der Angabe *ASTART=2,1* in die zweite Zeile der Tabelle von XTAB. Die Angabe *COL=TOTAL;C1* sorgt für die Spaltenbedingungen.

Das zweite XADD-Statement stellt entsprechend die Mittelwerte von N2 in die dritte Zeile. Die Spaltenbedingungen *COL=TOTAL;C2* unterscheiden sich vom vorigen XADD. Daher lassen sich diese beiden Zeilen nicht in einer *echten* Kreuztabelle erzeugen.

XTAB

Das Statements XTAB erzeugt Kreuztabellen. Es gibt eine Fülle von Kombinations- und Gestaltungsmöglichkeiten, die in den folgenden Abschnitten ausführlich beschrieben werden. Hier zunächst eine kurze Übersicht:

COL
ROW
FILTER
Diese drei Angaben sind formal gleichartig aufgebaut. Sie bilden das Kernstück des XTAB Statements. Durch COL werden die Zählanweisungen für die Tabellenspalten und die Kopftexte eingegeben.
Mit ROW werden analog die Zählanweisungen für die Tabellenzeilen festgelegt sowie die Zeilentexte der Tabellen.
FILTER schließlich legt die Filterbedingungen für die Tabelle, eventuelle globale Wichtungsfaktoren und entsprechende Filter- oder Wichtungstexte oberhalb der Tabelle fest.

BOTTOM Mit BOTTOM lassen sich beliebige Textzeilen eingeben, die unterhalb der Tabellen als freier Text gedruckt werden.

CONT Die Angabe von CONT sorgt dafür, dass nach dem Ausdruck einer Tabelle kein Blattvorschub durchgeführt wird. Die nächste Tabelle kann dann auf das gleiche Blatt gedruckt werden.

FONT Mit FONT lassen sich auf einem Seitendrucker die verschiedenen Bestandteile einer Tabelle in unterschiedlichen Schriftarten drucken (siehe Statement PAGEP).

GROUP Mit GROUP wird erreicht, dass in der Tabelle neben der normalen Kreuzauswertung zusätzlich mit einer besonderen Form der Gruppenverarbeitung gezählt werden kann:
Alle statistischen Fälle, die in der Eingabe direkt hintereinander liegen und in bestimmten Merkmalen übereinstimmen, werden als Gruppe aufgefasst. In den Zählbedingungen von ROW, COL und FILTER kann dann festgelegt werden, ob jeder Fall einzeln auszuwerten ist, oder ob nur die Gruppe in den entsprechenden Zählern verrechnet werden soll.

LINE Mit LINE lassen sich auf einem Seitendrucker verschiedene Linien in einer Tabelle ausgeben. Siehe Statement PAGEP.

PRINT PRINT steuert die Aufbereitung der Zählergebnisse.
Dabei können die Werte jeder Tabellenzelle gleichzeitig auf vier verschiedene Weisen ausgegeben werden. Neben Absolutwerten mit verschiedenen Nachkommastellen existiert eine Vielzahl von Möglichkeiten der Prozentuierung. Indexwerte können ermittelt und Rangziffern ausgedruckt werden. Schließlich können durch t-Tests Werte verglichen und bei signifikanter Abweichung markiert werden.

RCOL Mit XTAB können Tabellen erzeugt werden, die wesentlich größer sind als ein Druckblatt. CNTA zerlegt diese Tabellen dann automatisch, so dass sie auf mehreren Seiten hintereinander ausgegeben werden. Durch RCOL wird festgelegt, wie viele Anfangsspalten - zum Beispiel als Total-Spalten - in jeder Fortsetzungstabelle zu wiederholen sind.

RROW Diese Angabe legt analog zu RCOL fest, wie viele Anfangszeilen in automatisch erstellten Fortsetzungstabellen - zum Beispiel als Basiszeilen - zu wiederholen sind.

SAME Mit SAME werden die Zählergebnisse des vorigen XTAB-Statements ohne erneute Auswertung der einzelnen statistischen Fälle in das laufende XTAB übernommen.

SORT Durch SORT lassen sich die Zeilen und Spalten einer Tabelle ganz oder teilweise nach auf- oder absteigenden Werten sortieren.

START Mit START lässt sich die Positionierung der Tabelle auf der Ausgabeseite verändern. Zusammen mit der Angabe CONT lassen sich damit mehrere Tabellen beliebig auf einer Seite anordnen.

STUBHEAD Diese Angabe druckt beliebige Texte im Tabellenkopf oberhalb der Zeilentexte.

SUPPRESS Mit Hilfe von SUPPRESS lassen sich viele Tabellenteile unterdrücken: Zählerzeilen ohne Zählergebnisse, Filterangaben über den Tabellen, Leerzeilen zwischen den Zählergebnissen usw.

TAB Durch TAB werden die Breiten der Tabellenspalten festgelegt sowie die Trennzeichen und Linienelemente zwischen den Spalten.

STYLE Mit STYLE werden die Regeln zur Aufbereitung der Tabellentexte festgelegt: Die Texte lassen sich durch automatischen Zeilenumlauf in die vorgegebenen Spaltenbreiten einpassen. Kopftexte werden gedreht sowie rechtsbündig, linksbündig oder zentriert ausgegeben. Beim automatischen Umlauf der Zeilentexte können Fortsetzungszeilen nach rechts eingezogen werden. Mehrzeilig eingegebene Texte lassen sich zu einzeiligen Texten umformen. Fortsetzungszeilen von Zeilentexten können nach rechts eingerückt werden.

TITLE Mit TITLE lassen sich beliebige Textzeilen eingeben, die oberhalb der Tabellen als freier Text gedruckt werden.

VTAB Diese Angabe sorgt dafür, dass Tabellen mit Randlinien auf einer vorgegebenen Höhe enden, auch wenn sie weniger Zeilen mit Zählergebnissen besitzen.

Automatische Zerlegung von Tabellen

Mit Hilfe von XTAB können fast beliebig große Tabellen erstellt werden: Bis zu 32 767 Zeilen und Spalten sind möglich. Ihre Anzahl wird allerdings durch den verfügbaren Hauptspeicher begrenzt.
Tabellen, die zu groß für eine Druckseite sind, zerlegt CNTA automatisch in Teiltabellen, die auf ein Blatt passen. Dabei kann das Format der Druckseiten mit dem PAGE- oder PAGEP-Statement den jeweiligen Wünschen angepasst werden.

Beim Zerlegen solcher Tabellen versucht CNTA, zusammengehörige Texte möglichst nicht zu trennen. Im XTAB-Statement können auch Solltrennstellen vorgegeben werden. Dies geschieht durch die Angabe :SP hinter dem gewünschten Zählausdruck. Eine solche Trennstelle wird aber nur dann benutzt, wenn dort auch eine Trennung erforderlich ist. Daher können Solltrennstellen auch im Übermaß angegeben werden.

Bei Tabellen, die gleichzeitig zu hoch und zu breit für die Druckseiten sind, werden zunächst so viele Tabellenzeilen ausgewählt, wie auf einer Seite Platz finden. Diese werden in der Breite aufgetrennt und gesondert gedruckt. Danach wird das Verfahren mit den nächsten Zeilen fortgesetzt.

Ist eine Tabelle mit nur wenigen Zeilen zu breit für eine Seite, so versucht CNTA zunächst, die Fortsetzungstabelle unter die erste Teiltabelle auf die gleiche Seite zu stellen. Dies Verfahren wird auch mehrfach wiederholt, bis die Seite voll ist. Erst danach wird die nächste Seite eingestellt.

Durch die Angaben RROWS und RCOLS im XTAB-Statement kann festgelegt werden, ob und wie viele Zeilen oder Spalten aus dem ersten Tabellenteil in den Fortsetzungstabellen wiederholt werden sollen. Insbesondere kann dadurch festgelegt werden, ob die Zeilen- oder Spaltentexte in den Fortsetzungstabellen zu wiederholen sind.

XTAB: COL / ROW / FILTER

COL **COLS**	**=** *s* **;** . . .	**ROW** **ROWS**	**=** *z* **;** . . .	**FILTER** **FILTERS**	**=** *f* **;** . . .

COL, ROW und FILTER legen die Zählbedingungen für die Kreuztabellen fest.
Die Angaben *s, z* und *f* liefern, durch Semikolon voneinander getrennt, die Spalten-, Zeilen- und Filterbedingungen der Tabelle. Zum Beispiel:

```
XTAB  FILTER=C1.1  ROW=TOTAL;C10;C20  COL=TOTAL;C100
```

Für jeden Kreuzungspunkt einer Zeile mit einer Spalte besitzen die Tabellen einen Zähler von doppelter Gleitkomma-Genauigkeit.
Bei der Auswertung eines jeden statistischen Falles werden zunächst die Werte der Filteroperanden f, der Zeilenoperanden *z* und der Spaltenoperanden *s* ermittelt. Sodann wird für jeden Kreuzungspunkt der Tabelle der Filterwert mit dem Zeilenwert und dem Spaltenwert multipliziert. Das Ergebnis wird dann in der Tabelle zum Zähler des Kreuzungspunkts addiert.

Ist dabei einer der Operanden kein Zahlenwert, sondern eine Bedingung oder ein Merkmal, so wird für erfüllte Bedingungen der Wert 1 verwendet, sonst der Wert 0.

ROW, COL und FILTER gelten für das laufende XTAB-Statement, ROWS, COLS und FILTERS dagegen auch für die folgenden XTAB-Statements ohne dass dort neue Angaben erforderlich sind.

Die Angaben ROW und COL müssen vorhanden sein um eine Kreuztabelle zu erstellen. FILTER kann fehlen, dann wird FILTER=TOTAL angenommen.

Fehlen die Angaben ROW, COL und FILTER gleichzeitig, so wird keine Tabelle erzeugt. Solche XTAB Statements legen allgemeine Angaben wie TABS, PRINTS usw. für die folgenden Tabellen fest.

Für die Zählbedingungen s, z und f gibt es die folgenden Gestaltungsmöglichkeiten:

XTAB, ROW / COL / FILTER: elementare Zähloperanden

Hinter ROW, COL und FILTER können zunächst die folgenden elementaren Zähloperanden verwendet werden, zum Beispiel in der Form ROW = C10.1; C12.7; TOTAL.
Dabei erzeugt jeder Zähloperand hinter ROW eine Zeile, hinter COL eine Spalte und hinter FILTER eine eigene Tabelle.

Cn **.** a Dieser Operand besitzt den Wert 1, wenn das Merkmal a = 1 ... 9 999 der Bedingungsvariablen Cn erfüllt ist. Andernfalls hat Cn.a den Wert 0.
Zum Beispiel liefert C10.7 den Wert 1, wenn das Merkmal 7 der Variablen C10 erfüllt ist, sonst den Wert 0. Auch negative Merkmalsnummern sind für a möglich: C10.–1 ist das letzte Merkmal der Variablen, C10.–2 das vorletzte usw.

Cn Diese Angabe erzeugt zu jedem Merkmal a der Variablen Cn einen Operanden Cn.a.
Besitzt die Variable C1 vier Merkmale, so ist die Angabe C1 gleichwertig mit

```
C1.1;C1.2;C1.3;C1.4
```

Cn **. S** Dieser Operand liefert die Anzahl aller erfüllten Merkmale der Bedingungsvariablen Cn.
So steht C10.S für die Summe aller erfüllten Merkmale der Variablen C10.
Diese Summe lässt sich auch auf ausgewählte Merkmale einer Variablen beschränken. So zählt

```
C10.S(8,13..15)
```

die Merkmale 8, 13, 14 und 15 der Variablen C10, die den Wert 1 besitzen. Die übrigen Merkmale von C10 werden nicht berücksichtigt. Auch negative Merkmalsnummern sind hier möglich:
–1 ist das letzte Merkmal einer Variablen, –2 das vorletzte usw.

Cn **. F** Diese Angabe erzeugt zu jeder möglichen Anzahl von Angaben pro Fall einen Operanden.
Besitzt die Variable C10 zum Beispiel drei Merkmale, so erzeugt C10.F je einen Operanden für *0 Angaben von 3*, für *1 Angabe von 3*, für *2 Angaben von 3* und für *3 Angaben von 3*.
Für jeden statistischen Fall erhält nur jener Operand de
n Wert 1, dessen Anzahlbedingung erfüllt ist. In den Tabellen erscheinen nur die Zeilen oder Spalten, zu denen auch Fälle vorliegen.
Die festen Texte *Angabe von* und *Angaben von* in den Tabellen lassen sich mit den Folgestatements FP und FS von DEFS durch Texte eigener Wahl ersetzen.
Diese Angabe lässt sich auch auf ausgewählte Merkmale einer Variablen beschränken. So erzeugt

```
C11.F(8,13..15)
```

vier Operanden für 0 bis 4 Angaben in den Merkmale 8, 13, 14 und 15 der Variablen C11
Auch negative Merkmalsnummern sind möglich: –1 ist das letzte Merkmal einer Variablen, –2 das vorletzte usw.

Cn **. Z**k Dieser Operand ermittelt die Anzahl von Angaben der Variablen Cn für jeden statistischen Fall.
Er erhält den Wert 1, wenn die Anzahlbedingung Zk erfüllt ist, andernfalls den Wert 0.
Folgende Bedingungen sind möglich:

Z0 = kein Merkmal ist erfüllt, alle haben den Wert 0,
Z1 = genau eines der Merkmale besitzt den Wert 1,
Z2 = genau zwei der Merkmale besitzen den Wert 1,
Z3 = drei oder mehr Merkmale besitzen den Wert 1.

Es sind auch kombinierte Anzahlbedingungen möglich:

Z01 = keines oder genau ein Merkmal sind erfüllt,
Z123 = eines oder mehr Merkmale sind erfüllt, usw.

So verlangt C10.Z12, dass genau eines oder zwei der Merkmale der Variablen C10 erfüllt sind.
Anzahlbedingungen lassen sich auch auf ausgewählte Merkmale einer Variablen beschränken.
So ist die Bedingung

```
C10.Z1(8,13..15)
```

erfüllt, wenn genau eines der Merkmale 8, 13, 14 oder 15 der Variablen C10 den Wert 1 besitzt.
Auch negative Merkmalsnummern sind möglich: –1 ist das letzte Merkmal einer Variablen, –2 das vorletzte usw.

TOTAL Dieser Operand besitzt den Wert 1 = wahr für jeden statistischen Fall.
Mit seiner Hilfe lassen sich Zwischen- und Endsummen bilden.

Nn Numerische Variable, die hier auszuwerten ist.

ADD Dieser Operand besitzt den Wert 1 = wahr für neu hinzukommende statistische Fälle:
Das sind Fälle, zu denen nur Daten aus INPUT-EXT vorliegen, nicht jedoch aus INPUT-INT oder INPUT-SEC. In allen anderen Fällen besitzt ADD den Wert 0.

UPD Dieses Merkmal besitzt den Wert 1 = wahr für statistische Fälle, zu denen externe Daten aus INPUT-EXT und gleichzeitig interne Daten aus INPUT-INT oder INPUT-SEC vorliegen. In allen anderen Fällen besitzt UPD den Wert 0.

OLD Dieser Operand besitzt den Wert 1 = wahr für alte statistische Fälle, zu denen nur Daten aus INPUT-INT oder INPUT-SEC vorliegen, nicht jedoch aus INPUT-EXT. In allen anderen Fällen besitzt OLD den Wert 0.

UPDSEC Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle aus INPUT-EXT oder INPUT-INT, zu denen ein Fall mit passender Identifikation aus einer INPUT-SEC-Datei eingelesen wurde.
In allen anderen Fällen besitzt UPDSEC den Wert 0 = *falsch*.

ADDSEC Dieser Operand besitzt den Wert 1 = *wahr* für statistische Fälle aus INPUT-SEC, zu denen kein Fall aus INPUT-EXT oder INPUT-INT mit passender Identifikation vorhanden ist. Die Bedingung ADDSEC ist auch für alle statistischen Fälle aus INPUT-SEC mit der Angabe ADD oder ADDP erfüllt. In allen anderen Fällen besitzt ADDSEC den Wert 0 = *falsch*.

CIDi **. B**
CIDi **. E** Diese Operanden prüfen den Gruppenwechsel für Fall-Identifikationen:
Mehrere direkt hintereinander liegende statistische Fälle werden als Gruppe aufgefasst, wenn sie in ihren Fall-Identifikationen übereinstimmen.
Um eine solche Gruppenverarbeitung zu ermöglichen, müssen die Eingabedaten eine entsprechende Fall-Identifikation besitzen. Bei externen Dateien muss in INPUT-EXT eine solche Angabe ID ... ID4 gemacht werden. Bei internen Dateien muss diese Identifikation bei der Erstellung der Dateien vorgelegen haben.

Durch die Angabe CID, CID1 ... CID4 wird festgelegt, mit welchen Fall-Identifikationen die Gruppenbildung durchzuführen ist. Mit CID werden diejenigen Fälle zu einer Gruppe zusammengefasst, die direkt hintereinander liegen und die gleiche Fall-Identifikation ID besitzen. Bei CID1 bilden die Fälle eine Gruppe, die sowohl in ID als auch in ID1 übereinstimmen. CID4 schließlich verlangt, dass die statistischen Fälle in allen Fall-Identifikationen ID ... ID4 gleiche Werte besitzen.

Durch die zusätzliche Angabe .B und .E wird festgelegt, ob die entsprechende Bedingung zu Beginn oder am Ende einer Gruppe erfüllt sein soll:
CID.B ist nur für den ersten statistischen Fall jeder Gruppe erfüllt und für alle anderen Fälle nicht, CID.E gilt entsprechend nur für den letzten Fall jeder Gruppe.

IDi Numerische Fall-Identifikationen ID ... ID4.
Diese Operanden sind hier nur zulässig, wenn die entsprechende Fall-Identifikation auch numerische Werte besitzt, nicht jedoch bei Textwerten.

k Numerische Konstante. Sie kann aus maximal 9 Ziffern bestehen.
Als Dezimalzeichen ist der Punkt zu verwenden, das Komma wird nicht akzeptiert. Die Schreibweisen .9 und 9. sind unzulässig, dafür ist 0.9 und 9.0 anzugeben.

Die folgenden Funktionen erlauben als Argument eine einzelne C- oder N-Variable. Zum Beispiel liefert

```
ROW=MEAN(C10)
```

den Mittelwert aus den erfüllten Merkmalen der Variablen C10.

Aus den C-Variablen dürfen auch einzelne Merkmale ausgewählt werden. So liefert

```
ROW=MEAN(C10(8,13..15))
```

den Mittelwert aus den Merkmalen 8, 13, 14 und 15 der Variablen C10. Auch negative Merkmalsnummern sind dabei möglich: -1 ist das letzte Merkmal einer Variablen, -2 das vorletzte usw. So schließt

```
ROW=MEAN(C10(1..-2))
```

das letzte Merkmal der Variablen C10 aus dem Mittelwert aus.

Bedingungsvariable Cn werden nach folgenden Regeln ausgewertet:
Statistische Fälle, für die keines der Merkmale von Cn erfüllt ist, nehmen nicht an der Auswertung teil.
Sind für einen statistischen Fall mehrere Merkmale der Variablen Cn gleichzeitig erfüllt, so wird jedes dieser Merkmale wie ein eigener Fall behandelt.

Die Werte Z0 (keine Angabe) von numerischen Variablen Nn nehmen nicht an der Auswertung teil.

Mit Folgestatements von DEFS können diesen Funktionen Texte zugewiesen werden. So versehen die Statements

```
DEFS
-MEAN 'Mittelwert'
```

die Funktion MEAN in den Kreuztabellen mit dem Text *Mittelwert.*

MAX(Cn**)** Größtes erfülltes Merkmal der Bedingungsvariablen Cn.
MAX(Nn**)** Größter Wert der numerischen Variablen Nn.

MEAN(Cn**)** Mittelwert m aus den erfüllten Merkmalen der Bedingungsvariablen Cn.
MEAN(Nn**)** Mittelwert m aus den Werten der numerischen Variablen Nn, die von Z0 verschieden sind.

$$m = \frac{\sum x}{n}$$

x = erfüllte Merkmalsnummern der Variablen Cn für alle statistischen Fälle.
= Werte der Variablen Nn, die von Z0 (keine Angabe) verschieden sind.
n = Anzahl der erfüllten Merkmale der Variablen Cn für alle statistischen Fälle.
= Anzahl der der Variablenwerte von Nn, die von Z0 (keine Angabe) verschiedenen sind.

MED(Cn**)** Median M aus den erfüllten Merkmalen der Bedingungsvariablen Cn.
MED(Nn**)** Median M aus den Werten der numerischen Variablen Nn, die von Z0 verschieden sind.

$$M = m + \frac{\frac{n}{2} - \sum_{i=1}^{m-1} n_i}{n_m}$$

m = größtes Merkmal von Cn, das kleiner oder gleich dem Median ist.
= größter Wert von Nn, der kleiner oder gleich dem Median ist.
n = Anzahl der erfüllten Merkmale der Variablen Cn für alle statistischen Fälle.
= Anzahl der Variablenwerte von Nn, die von Z0 (keine Angabe) verschieden sind.
n_i = Anzahl der erfüllten Merkmale Cn.i aller statistischen Fälle.
= Anzahl der Variablenwerte von Nn in der Klasse i, die von Z0 verschieden sind.

MIN(Cn**)** Kleinstes erfülltes Merkmal der Bedingungsvariablen Cn.
MIN(Nn**)** Kleinster Wert der numerischen Variablen Nn, der von Z0 (keine Angabe) verschieden ist.

MODL(Cn**)** Kleinster Modalwert (häufigster Wert) der erfüllten Merkmal der Bedingungsvariablen Cn.
MODL(Nn**)** Kleinster Modalwert der numerischen Variablen Nn, der von Z0 (keine Angabe) verschieden ist.

Besitzt die Variable mehrere Modalwerte, so weist MODL den kleinsten davon aus.
Die Angabe PRINT ... =MOD ... versieht mehrdeutige Modalwerte mit einer Markierung und gibt unterhalb der Tabelle eine Fußnote dazu aus.

MODU(Cn**)** Größter Modalwert (häufigster Wert) der erfüllten Merkmal der Bedingungsvariablen Cn.
MODU(Nn**)** Größter Modalwert der numerischen Variablen Nn, der von Z0 (keine Angabe) verschieden ist.

Besitzt die Variable mehrere Modalwerte, so weist MODU den größten davon aus.
Die Angabe PRINT ... =MOD ... versieht mehrdeutige Modalwerte mit einer Markierung und gibt unterhalb der Tabelle eine Fußnote dazu aus.

PERCTL(Cn **,** p**)** p-Perzentil aus den erfüllten Merkmalen der Bedingungsvariablen Cn.
PERCTL(Nn **,** p**)** p-Perzentil aus den Werten der numerischen Variablen Nn, die von Z0 verschieden sind.

$$perctl(p) = \begin{cases} W_k & für\ d = 0 \\ W_{k+1} & für\ d \geq 1 \\ (1-d)\cdot W_k + d \cdot W_{k+1} & für\ d \in \left]0,1\right[\ und\ H_{k+1} \geq 1 \\ (1-\frac{d}{H_{k+1}})\cdot W_k + \frac{d}{H_{k+1}} \cdot W_{k+1} & für\ d \in \left]0,1\right[\ und\ H_{k+1} < 1 \end{cases}$$

m	Anzahl der Variablenwerte ungleich Z0
$W_1 < W_2 < ... W_m$	die unterschiedlichen Variablenwerte in aufsteigender Folge
H_i	Anzahl Fälle mit dem Wert W_i
$H = \sum_{i=1}^{m} H_i$	Anzahl Fälle mit Nennungen in der Variablen.
p	Prozentsatz 1…99 des Perzentils
$H_p = \frac{p \cdot (H+1)}{100}$	Anzahl Fälle zum Prozentsatz
$k \in \{1..m\}$	der Index für den gilt: $\sum_{i=1}^{k} H_i \leq H_p$ und $\sum_{i=1}^{k+1} H_i > H_p$
$d = H_p - \sum_{i=1}^{k} H_i$	Anzahl Fälle, die in die Klasse zum Wert W_{k+1} fallen.

STD(Cn**)** Standardabweichung *s* aus den erfüllten Merkmalen der Bedingungsvariablen Cn.
STD(Nn**)** Standardabweichung *s* aus den Werten der numerischen Variable Nn, die von Z0 verschieden sind.

$$s = \sqrt{\sum \frac{(x-m)^2}{n-1}}$$

m	Mittelwert der Variablen, wie bei MEAN(Cn) und MEAN(Nn) beschrieben.
x	erfüllte Merkmalsnummern der Variablen Cn für alle statistischen Fälle. Werte der Variablen Nn, die von Z0 (keine Angabe) verschieden sind.
n	Anzahl der erfüllten Merkmale der Variablen Cn für alle statistischen Fälle. Anzahl der Variablenwerte von Nn, die von Z0 (keine Angabe) verschieden sind.

STE(C*n*) Standardfehler *e* aus den erfüllten Merkmalen der Bedingungsvariablen Cn.
STE(N*n*) Standardfehler *e* aus den Werten der numerischen Variablen Nn, die von Z0 verschieden sind.

$$e = \frac{s}{\sqrt{n}}$$

s Standardabweichung, wie bei STD(Cn) und STD(Nn) beschrieben.
n Anzahl der erfüllten Merkmale der Variablen Cn für alle statistischen Fälle.
Anzahl der Variablenwerte von Nn, die von Z0 (keine Angabe) verschieden sind.

KOMP Komplement für C-Variable.
Dieser Operand liefert die Zahl der Fälle, die in den davorliegenden Zeilen oder Spalten nicht enthalten sind. Sein Wert hängt von den davorstehenden Operanden der Form Cn.m, Cn.Zi, Cn.S sowie davorstehenden logischen Ausdrücken ab.

Zu Beginn von ROW, COL und FILTER besitzt KOMP für jeden statistischen Fall den Wert 1 oder *wahr*. KOMP erhält den Wert 0 oder *falsch*, wenn die davorstehenden Operanden eine der folgenden Bedingungen erfüllen:

- eine einzeln stehende Bedingung der Form Cn.m oder Cn.Zi besitzt den Wert 1 oder *wahr*,
- ein einzeln stehender Operand Cn.S besitzt einen von 0 verschiedenen Wert,
- ein logischer Ausdruck besitzt den Wert 1 oder *wahr*,
- in einem numerischen Ausdruck ist eine der Bedingungen Cn.m, Cn.Zi oder KOMP erfüllt,
- in einem numerischen Ausdruck ist der Operand Cn.S von 0 verschieden,
- ein weiteres, davorstehendes KOMP besitzt den Wert 1 oder *wahr*.

Zum Beispiel:

```
ROW=TOTAL;C1.1;C1.2;C1.Z0;KOMP
```

Hier liefert KOMP die Anzahl der Fälle, für die die beiden Merkmale C1.1 und C1.2 nicht gesetzt sind und für die auch die die Bedingung C1.Z0 nicht erfüllt ist. Operanden wie TOTAL, N1 oder MEAN(N1) haben keinen Einfluss auf KOMP.

In dem Beispiel

```
ROW=TOTAL;C1.1&C1.2;KOMP
```

liefert KOMP die Zahl der Fälle, für die der logische Ausdruck C1.1 & C1.2 nicht erfüllt ist, für die also C1.1 oder C1.2 den Wert 0 besitzt.

Für

```
ROW=TOTAL;C1.1+C1.2;KOMP
```

liefert KOMP die Zahl der Fälle, für die im numerischen Ausdruck C1.1 + C1.2 keines der Merkmale C1.1 und C1.2 erfüllt ist.

Innerhalb einer Staffel {…} ist KOMP nur abhängig von den Zeilen oder Spalten der gleichen Staffel. Zum Beispiel:

```
ROW=TOTAL;C10.1;{C1.1;C1.2;KOMP}{C2.1;C2.2;KOMP}
```

Hier steht KOMP in der ersten Staffel für die Fälle, für die C1.1 und C1.2 nicht erfüllt sind und KOMP in der zweiten Staffel für die Fälle, für die C2.1 und C2.2 nicht erfüllt sind.

Hinter einer Staffelung {…} ist KOMP nur abhängig von den Zeilen oder Spalten, die zwischen dem Ende der Staffel und KOMP liegen. Zum Beispiel:

```
ROW=TOTAL;{C10.1}{C1.1;C1.2};C2.1;C2.2;KOMP
```

Hier liefert KOMP die Anzahl der Fälle, für die C2.1 und C2.2 nicht erfüllt sind; unabhängig von den davorstehenden Angaben.

Die elementaren Operanden lassen sich auf vielfältige Weise zu komplexen Zählausdrücken kombinieren. Dies wird im Abschnitt *COL/ROW/FILTER: Ausdrücke* beschrieben.

Beispiel: Zähloperanden ohne Text

```
DEFS
-C1:2
-C100:3
-N1:9
-N100:9,5

XTAB   COL=TOTAL;N1;  C100
       ROW=TOTAL;C1.1;.Z1;.S;N100;MEAN(N100)
```

Diese Statements könnten die folgende Tabelle erstellen:

TOTAL 4594

	TOTAL	N1	C100.1	C100.2	C100.3
TOTAL	4594	4597	3	238	1101
C1.1	992	985	1	76	329
C1.Z1	4594	4597	3	238	1101
C1.S	4594	4597	3	238	1101
N100	13641	13629	3	412	2395
MEAN(N100)	3	3	1	2	2

Zur Auswertung eines statistischen Falles werden für jede Tabellenzelle die Variablenwerte aus dem Filter, der Zeile und der Spalte miteinander multipliziert und zu dem bisherigen Wert der Zelle hinzuaddiert.

Da die Zähloperanden nicht vertextet wurden, werden die Operandennamen als Text in den Tabellen verwendet.

Abkürzungen

Wird in direkt hintereinander liegenden Zähloperanden die gleiche Bedingungsvariable Cn angesprochen, so muß Cn nur im ersten Zähloperanden ausgeschrieben werden:
Statt: C80.1 ; C80.2 ; C80.Z0 ; C80.S
genügt: C80.1 ; .2 ; .Z0 ; .S

Erscheinen aufeinander folgende Merkmale einer Bedingungsvariablen Cn in Zähloperanden direkt hintereinander, so genügt es, das erste und das letzte Merkmal anzugeben:
Statt: C1.1 ; .2 ; .3 ; .4 ; .5
genügt: C1.1 ; ... 5

Auch negative Merkmalsnummern sind möglich:
Statt: C1.1 ; .2 ; .3 ; .4 ; .5
genügt: C1.1 ; ... –2
sofern die Variable C1 gerade sechs Merkmale besitzt.

Sollen alle Merkmale einer Bedingungsvariablen ausgewertet werden, so genügt es, die Variable ohne Merkmalsangaben zu schreiben:
Statt: C100.1 ; ... 8
genügt: C100
sofern die Variable C100 gerade acht Merkmale besitzt.

Werden mehrere direkt hintereinander liegende Bedingungsvariable angesprochen, so genügt es, die erste und die letzte Variable anzugeben:
Statt: C10 ; C11 ; C12 ; C13 ; C15
genügt: C10 ; ... 15
Hierbei wird unterstellt, dass die Variable C14 nicht definiert ist.

Werden mehrere direkt hintereinander liegende numerische Variable angesprochen, so genügt es, die erste und die letzte Variable anzugeben:
Statt: N10 ; N11 ; N12 ; N13 ; N15
genügt: N10 ; ... 15
Hierbei wird unterstellt, dass die Variable N14 nicht definiert ist.

Wurde eine Variable mit Index definiert und werden mehrere direkt hintereinander liegende Indexwerte dieser Variablen angesprochen, so genügt es, die Variable nur beim ersten Mal vollständig mit Index anzugeben:
Statt: N2[10] ; N2[11] ; N2[12] ; N2[13]
genügt: N2[10] ; ... [13]
oder auch: N2[10] ; ... 13

Statt: N10[1] ; N10[3] ; N10[5] ; N10[7]
genügt: N10[1] ; [3] ; [5] ; [7]

XTAB, ROW / COL / FILTER: Texte zu den Zähloperanden

Die Zählergebnisse werden standardmäßig mit Texten versehen.
Oberhalb der Tabelle können zu den FILTER-Angaben Texte ausgegeben werden. Im Tabellenkopf werden zu den COL-Operanden Texte gedruckt, und am Anfang jeder Zeile stehen die aus den ROW-Operanden entnommenen Texte.

Für die verschiedenen Zähloperanden in ROW, COL und FILTER gelten folgende Regeln:

Cn **.** a — Wurde der Bedingungsvariablen *Cn* und dem Merkmal *a* unter DEFS ein Variablentext und ein Merkmalstext zugewiesen, so werden diese Texte in der Tabelle verwendet. Dabei wird der Merkmalstext unterhalb des Variablentextes ausgegeben.
Das Gleiche gilt, wenn die Variable bereits auf INPUT-INT definiert ist und von dort die Texte mitbringt.
Besitzt *Cn* einen Variablentext, das Merkmal *a* aber keinen Merkmalstext, so wird der Variablentext ausgegeben und darunter die Merkmalsnummer *a* gedruckt.
Ist ein Merkmalstext zu *a* vorhanden aber kein Variablentext für *Cn*, so wird nur der Merkmalstext gedruckt.
Fehlen schließlich Variablen- und Merkmalstext, so wird die Angabe *Cn.a* direkt gedruckt.

Cn **. Z**i — Wenn vorhanden, wird zunächst der Variablentext von *Cn* gedruckt und darunter die Anzahlbedingung *Zi* selbst.
Wurde *Zi* unter DEFS ein Text zugewiesen, so erscheint stattdessen dieser Text.
Besitzt *Cn* keinen Text, wohl aber *Zi*, so wird nur der Text von *Zi* gedruckt.
Besitzt weder *Cn* noch *Zi* einen Text, so wird *Cn.Zi* direkt ausgegeben.

Cn **. S** — Wenn vorhanden, wird zunächst der Variablentext von *Cn* gedruckt und darunter das Zeichen *S*.
Wurde der Anzahlbedingung *S* unter DEFS ein Text zugewiesen, so erscheint dieser Text anstelle des Zeichens S. Besitzt Cn keinen Text, wohl aber *S*, so wird nur der Text von *S* gedruckt.
Besitzt weder *Cn* noch *S* einen Text, so wird *Cn.S* selbst ausgegeben.

Nn — Besitzt *Nn* einen Variablentext, so wird dieser gedruckt, andernfalls die Variablenbezeichnung *Nn* selbst.

TOTAL
IDi
CIDi
ADD
UPD
OLD — Für diese Operanden werden die unter DEFS eingegebenen Texte verwendet.
Sind solche Texte nicht vorhanden, so werden die Operanden selbst als Text ausgegeben.

Ausdrücke — Für Ausdrücke – zum Beispiel N10*N20 – wird der Ausdruck selbst als Text in die Tabellen gestellt.

Funktionen — Funktionen mit nur einer Variablen als Argument – zum Beispiel MEAN(N10) – erhalten als Text den der Variablen. Besitzt die Variable keinen Text, so wird stattdessen die Funktion selbst verwendet.
Für Funktionen mit mehreren Argumenten – zum Beispiel RCH(N10:2 N20:3) – wird ebenfalls die Funktion als Text ausgegeben.

Wird einem Text in DEFS eine Fußnote zugewiesen, so erscheint diese Fußnote am Ende jeder Tabelle, die den Text enthält.

Beispiel: Zähloperanden mit Text

```
DEFS
-C1:2              1='Männer'  2='Frauen'
-C101:3    'Alter' 1='14 - 18 Jahre' 2='19 - 40 Jahre' 3='über 40 Jahre'
-N10:9,5   'Gewicht'
-N100:9,5  'jährliche Gesamtausgaben'
-TOTAL     'Gesamt'
-MEAN      'Mittelwert'
-S         'Summe'
-Z1        'genau eine Angabe'

XTAB   COL=TOTAL;N10;C100
       ROW=TOTAL;C1.1;.Z1;.S;N100;MEAN(N100)
```

Diese Statements könnten folgende Tabelle erzeugen:

Gesamt 1204

	Gesamt	Gewicht	Alter		
			14 - 18 Jahre	19 - 40 Jahre	über 40 Jahre
Gesamt	1204	1248	226	341	316
Männer	1041	1092	192	288	262
Frauen	163	156	34	53	54
genau eine Angabe	1204	1248	226	341	316
Summe	1204	1248	226	341	316
jährliche Gesamtausgaben	62416	88105	11989	18416	16306
Mittelwert	66	71	63	66	67

In den Folgestatements von DEFS haben hier alle Zähloperanden Texte erhalten.
Diese Texte werden in der Tabelle in die verfügbaren Spaltenbreiten eingepasst.

XTAB, ROW / COL / FILTER: erweiterte Druckaufbereitung

z **:** < | 'text' | > < **:ABS** < ,d > > < **:FT**i > < **:L**i > < **:R**g > < **:SP** > < **:F**u > < | **,G**i | >
| **C**n | | **,CO**i |
| **C**n**.**a |
| **N**n |
| **T**n |
| **NT** |

Hinter jedem Zähloperanden z sind nach einem Doppelpunkt die obigen Angaben möglich.
Mit ihnen lassen sich Texte zu den Zähloperanden abweichend von den Standardregeln ausgeben.
Weiter können die Zählergebnisse anders aufbereitet, sowie Linien und Tonflächen erzeugt werden.
Die Anzahl und Reihenfolge der Angaben hinter dem Doppelpunkt ist beliebig.

z Dies ist ein Zählausdruck hinter ROW, COL oder FILTER, wie zum Beispiel TOTAL, C1, N98.
Es ist auch möglich, den Zählausdruck z wegzulassen und nur den Doppelpunkt anzugeben. Die obigen Angaben dahinter erzeugen dafür eine Zeile oder Spalte ohne Zählergebnisse aber mit den besonderen Druckaufbereitungen.

'text' Hier kann ein ein- oder mehrzeiliger Text angegeben werden, wie im Abschnitt *Textzeilen* beschrieben, der den Text des Operanden z ersetzt. Es kann auch ein Textelement in der Form (n) verwendet werden.

Cn Der Variablentext der Bedingungsvariablen Cn ersetzt den Text des Operanden z.

Cn . a Der Variablentext der Bedingungsvariablen Cn zusammen mit dem Merkmalstext des Merkmals a ersetzt den Text des Operanden z.

Nn Der Variablentext der numerischen Variablen Nn ersetzt den Text des Operanden z.

Tn Der Variablentext der Textvariablen Tn ersetzt den Text des Operanden z.

NT Zu dem Operanden z wird *kein* Text ausgegeben.

ABS In der entsprechenden Zeile oder Spalte werden abweichend von den Angaben aus PRINT...PRINT3 nur absolute Werte gedruckt. Die Angabe ABS macht für die entsprechende Zeile oder Spalte alle Angaben aus PRINT...PRINT3 unwirksam.
Die Zählergebnisse zu ABS werden in jedem Fall rechtsbündig in die Spalte gestellt.
So sorgen die Angaben

```
PRINT=PROW1,2  ROW=TOTAL:ABS;C1   COL=TOTAL:ABS;C12
```

in einem XTAB-Statement zunächst dafür, dass alle Zählergebnisse als Prozentwerte mit zwei Nachkommastellen ausgegeben werden, mit der ersten Tabellenzeile als Basis.
Die Angaben ABS hinter ROW und COL bewirken, dass die Werte der ersten Zeile und Spalte abweichend davon als Absolutwerte ohne Nachkommastelle erscheinen.
ABS kann auch in oberen Staffelstufen angegeben werden.
Es ist dann für alle dazugehörigen unteren Staffelstufen wirksam.

d Anzahl 1 ... 15 der Nachkommastellen, mit denen die durch ABS angeforderten Absolutwerte auszugeben sind. Fehlt diese Angabe d, so erscheinen die Absolutwerte ohne Nachkommastellen.
Wird für einen Zähler unter ROW und COL gleichzeitig ABS angegeben, mit einer unterschiedlichen Anzahl von Nachkommastellen, so wird die höchste der beiden Angaben wirksam.
Wird ABS gleichzeitig in einer oberen und unteren Staffelstufe mit unterschiedlicher Anzahl von Nachkommastellen angegeben, so hat die Angabe aus der unteren Staffelstufe Vorrang vor der oberen.

FTi Diese Angabe sorgt dafür, dass die Texte und Zählerwerte einer Zeile, einer Spalte oder eines Filters in einer anderen Schriftart gedruckt werden, als durch die Angabe STYLE in XTAB festgelegt wurde.
Es sind die Angaben FT1 bis FT16 möglich, so wie sie im Statement PAGEP definiert sind.
Bei Druckausgabe mit PAGE auf Zeilendrucker bleibt FTi wirkungslos.
Werden gleichzeitig in ROW und COL mit FTi verschiedene Schriften angefordert, so hat die Angabe aus ROW Vorrang vor der Angabe aus COL.

Die Angabe STYLE im Statement XTAB bestimmt ebenfalls Schriftarten für die Zeilen und Spalten. Diese gelten aber nur für die Kopf- und Zeilentexte, nicht für Zähler. Die Angaben aus ROW= und COL=... haben Vorrang vor denen aus STYLE. Schriftangaben FTi hinter PRINT . . . PRINT3 wiederum haben Vorrang vor den Angaben aus ROW= oder COL=.

Eine neue Schriftart FTi kann auch in Staffelungen angegeben werden. Dabei beeinflusst FTi auch alle dahinter liegenden Staffelstufen sowie die Zählerwerte in den zugehörigen Zeilen oder Spalten.

Gi
COi Diese Angabe unterlegt Zeilen oder Spalten mit einem Grauton G1 ... G16 oder einer Farbe CO1 ... CO16 aus PAGEP. Dabei werden neben den Zählergebnissen auch die zugehörigen Zeilen- und Spaltentexte mit Tonflächen versehen. Bei C-Variablen mit Variablentext wird jedoch nur der Merkmalstext eingefärbt. Die Angabe INSERT versieht dagegen die Zeilen und Spalten vollständig mit Tonflächen. Hinter FILTER bleiben Farb- und Grautonangaben wirkungslos, ebenso mit dem Statement PAGE.

Zum Beispiel unterlegt das Statement

```
XTAB  ROW=TOTAL:G5;C17.1:G3;..9;.10..20
```

Zeilentext und Zählerwerte der ersten Tabellenzeile mit dem Grauton G5. Die Zeilen zu C17.1 bis C17.9 werden mit dem Grauton G3 versehen und die Zeilen zu C17.10 bis C17.20 erscheinen ohne Grauton.

Grautöne und Farben sind auch in Staffelungen möglich. Eine solche Angabe wirkt auch auf alle dahinter liegenden Staffelstufen sowie die Zählerwerte der dazugehörigen Zeilen oder Spalten. Angaben in unteren, rechten Staffeln haben Vorrang vor denen in oberen, linken Staffeln.
Zum Beispiel unterlegt das Statement:

```
XTAB   COL={:'Geschlecht',G3}{C1.1;.2:G0;.3:G1}
```

im Tabellenkopf den Text *Geschlecht* mit dem Grauwert G3. Der Text und die Zählergebnisse zur Spalte C1.1 werden mit dem gleichen Grauwert versehen. Die Angabe G0 sorgt dafür, dass der Text zu C1.2 und die Zählergebnisse der zweiten Spalte ohne Graufläche ausgegeben werden. Die Angabe G1 versieht den Text von C1.3 und die Zählergebnisse der letzten Spalte mit dem Grauton G1.

Neben der Angabe INSERT erzeugt auch STYLE im Statement XTAB Tonflächen für die Zeilen und Spalten, STYLE jedoch nur für die Kopf- und Zeilentexte, nicht für Zähler. INSERT versieht die Zeilen und Spalten vollständig mit Tonflächen einschließlich aller Textstufen.
Schließlich lassen sich mit der Angabe PRINT im Statement XTAB einzelne Zählergebnisse mit Tonflächen unterlegen, dies auch abhängig von Bedingungen.
Die Angaben aus ROW= und COL= haben Vorrang vor denen aus INSERT und diese Vorrang vor STYLE.

Mit G0 oder CO0 lässt sich die Ausgabe eines zuvor festgelegten Grautons verhindern.

Li Linie L1 bis L16, die *vor* der laufenden Zeile oder Spalte gedruckt werden soll.
Dabei sind Li die Linien, so wie sie im Statement PAGEP festgelegt wurden.
Mit L0 kann die Ausgabe einer Linie verhindert werden.
Hinter FILTER bleibt diese Angabe wirkungslos, ebenso wenn mit PAGE auf Zeilendrucker ausgegeben wird.
Linien Li können auch in oberen Staffelstufen angegeben werden. Dann wird die Linie vor der ersten Zeile oder Spalte für diesen Operanden ausgegeben.
Werden gleichzeitig in einer oberen und unteren Staffelstufe Linien angegeben, so hat die Linie der oberen Staffelstufe Vorrang vor der unteren.

Rg Mit Hilfe dieser Angabe können die Kopftexte einer Tabelle gedreht werden.
Rg ist nur hinter COL möglich und nur dann wirksam, wenn das Statement PAGEP aktiv ist.
KYOCERA-Drucker müssen zusätzlich die Kommandos PRESCRIBE II beherrschen.
Es können folgende Angaben gemacht werden:
R90: Dreht die Texte um 90 Grad gegen den Uhrzeigersinn, Leserichtung von unten nach oben.
R-90: Dreht die Texte um 90 Grad im Uhrzeigersinn, Leserichtung von oben nach unten.
R0: Sorgt für normale, waagerechte Ausgabe der Texte.

SP Hiermit werden Solltrennstellen für die automatische Tabellenzerlegung vorgegeben.
CNTA zerlegt Tabellen, die zu groß für eine Druckseite sind, in mehrere Teiltabellen.
Dabei wird versucht, zusammengehörige Texte möglichst nicht voneinander zu trennen, und bevorzugt *hinter* Zeilen und Spalten mit der Angabe SP zu zerlegen.
Ist keine Tabellenauftrennung erforderlich, so bleibt SP wirkungslos.
Diese Angaben können daher bedenkenlos auch im Übermaß angegeben werden.

Fu Fußnotennummer 1 ... 99999. Das unter der Nummer (u) in DEFS definierte Textelement wird als Fußnote am Seitenende unter den Tabellen dieses XTAB Statements gedruckt.
Wurde zum Beispiel unter DEFS das folgende Textelement definiert:

```
-(27) 'Diese Angaben beruhen auf Schätzwerten'
```

so sorgt die Angabe

```
ROWS=...;N139:F27;...
```

dafür, dass dieser Text als Fußnote ausgegeben wird.
Die Fußnote erscheint jedoch nicht, wenn die entsprechende Zeile oder Spalte aufgrund einer SUPPRESS-Angabe nicht ausgegeben wird.
Wird eine zu große Tabelle von CNTA auf mehrere Seiten verteilt, so wird die Fußnote nur auf den Seiten gedruckt, auf denen die entsprechende Zeile oder Spalte auch wirklich erscheint. Die Angabe F0 macht bereits bestehende Fußnotenangaben, die bei der Definition von Variablen vergeben wurden, für die laufende Zeile oder Spalte unwirksam.
Eine hier angegebene Fußnote Fu hat Vorrang vor einer unter DEFS zugeordneten und ersetzt diese in der laufenden Zeile oder Spalte.

Beispiel: Aufbereitung der Zählerwerte

```
DEFS
-C1:2              1='Männer'  2='Frauen'
-C100:3   'Alter'  1='14 - 18 Jahre' 2='19 - 40 Jahre' 3='über 40 Jahre'
-N100:9,5 'jährliche Gesamtausgaben'
-(38)     '*) Fälle ohne Angabe sind nicht im Mittelwert enthalten.'

XTAB   COL=TOTAL;C100
       ROW=TOTAL;C1;MEAN(N100):ABS,2,'Mittel *)',F38
       PRINT=ABS
       PRINT1=PROW1,1,%
       STYLE=G2
       LINE=L11
```

Diese Statements könnten folgende Tabelle erzeugen:

Gesamt 4594

	Gesamt		Alter					
			14 - 18 Jahre		19 - 40 Jahre		über 40 Jahre	
Gesamt	4594	100,0%	3	100,0%	238	100,0%	1101	100,0%
Männer	992	21,6%	1	33,3%	76	31,9%	329	29,9%
Frauen	3602	78,4%	2	66,7%	162	68,1%	772	70,1%
Mittel *)	2,99		1,13		1,78		2,19	

*) Fälle ohne Angabe sind nicht im Mittelwert enthalten.

Die Angabe *PRINT=ABS* gibt die Zählergebnisse als Absolutwerte aus. *PRINT1=PROW1,1,%* stellt die Ergebnisse zusätzlich rechts daneben als Prozentwerte, mit einer Nachkommastelle und dem Zeichen % dahinter.

Für die Zeile mit dem Mittelwert ist diese Aufbereitung aber nicht sinnvoll. Durch die Angabe *ABS, 2, 'Mittel *)', F38* hinter dem Doppelpunkt wird eine abweichende Regel angegeben. Diese hat Vorrang vor PRINT und PRINT1.

ABS, 2 erzeugt einen Absolutwert mit zwei Nachkommastellen,

*'Mittel *)'* liefert einen neuen Zeilentext und. *F38* sorgt für eine Fußnote untrhalb der Tabelle mit dem Text des Textelements (38)

Die Angabe *STYLE=G2* unterlegt die Tabelle mit einer grauen Tonfläche und *LINES=L11* sorgt für die weißen Linien zwischen den Zellen.

Beispiel: Tonflächen, Linien, Schriftarten

```
XTAB  LINE=L7,ROW:L0
      COL=TOTAL:CO12;C1:L2
      ROW=TOTAL:'Basis-Fälle = 100%',FT3;
          C100.1:L2,G2; .2:G2; .3:G2;
          C2030.1:L2;.2;..6;
          C1.1:'Männer',L2;
            .2:'~FT3~Frauen'
      PRINT=R(1)ABS
            R(2..6 9..-1)PROW1,1,%
            R(7)PROW1,1,%,FT3
            R(8)PROW1,1,%,CO16
```

Dieses Statement zeigt verschiedene Möglichkeiten, Tabellenteile mit Tonflächen und unterschiedlichen Schriften zu versehen:

Gesamt 4594

	Gesamt	Männer	Frauen
Basis-Fälle = 100%	**4594**	**992**	**3602**
Alter			
14 - 18 Jahre	0,1%	0,1%	0,1%
19 - 40 Jahre	5,2%	7,7%	4,5%
über 40 Jahre	24,0%	33,2%	21,4%
Schulabschluss			
Volksschule, Hauptschule	2,4%	1,5%	2,6%
Weiterführende Schule ohne Abitur	13,5%	15,6%	12,9%
Abitur	**12,6%**	**15,0%**	**11,9%**
Studium ohne Abschluss	5,2%	3,5%	5,7%
Studium mit Abschluss	65,3%	63,2%	65,8%
kein Abschluss	1,0%	1,1%	1,0%
Männer	21,6%	100,0%	-
Frauen	78,4%	-	100,0%

Die Angabe *LINES=L7,ROW:L0* wählt die kräftigere Linie L7 für die Tabelle aus, wobei *LROW:L0* die Linien zwischen den Zeilen unterdrückt.
Hinter ROW sorgen *C100.1:L2...* und *C2030.1:L2* für die zwei dünnen Linien L2 zwischen den Zeilen.

In COL erzeugt die Angabe *C1:L2* die beiden dünnen Linien L2 zwischen den Spalten.

Die graue Tonfläche *G2* der Alters-Zeilen entsteht durch *C100.1:L2,G2; .2:G2; .3:G2* hinter ROW.
In COL versieht die Angabe *TOTAL:CO12* die erste Zählerspalte mit der Farbe CO12.
Die Angabe *R(8)PROW1,1,%,CO16* hinter PRINT unterlegt die Zähler der achten Zeile mit der blauen Farbe CO16.

Die Angabe *TOTAL:'Basis-Fälle = 100%',FT3* wählt die fette Schrift FT3 für die erste Zeile aus.
Hinter PRINT erzeugt *R(7)PROW1,1,%,FT3* die fette Schrift FT3 für die Zähler der Zeile 7.
.2:'~FT3~Frauen' gibt den Text der letzten Zeile ebenfalls in der fetten Schrift FT3 aus.

Beispiel: Rotierte Kopftexte

```
XTAB  COL=TOTAL;C2030
      ROW=TOTAL:'Basis-Fälle = 100%':ABS;C100
      PRINT=PROW1,1,%
      STYLE=COL(-1):R90
```

Dieses Statement kann folgende Tabelle erzeugen:

Gesamt 4594

	Gesamt	Schulabschluss					
		Volksschule, Hauptschule	Weiterführende Schule ohne Abitur	Abitur	Studium ohne Abschluss	Studium mit Abschluss	kein Abschluss
Basis-Fälle = 100%	4594	110	621	579	240	2998	46
Alter							
14 - 18 Jahre	0,1%	0,9%	0,2%	0,2%	-	-	-
19 - 40 Jahre	5,2%	5,5%	8,9%	9,5%	5,8%	3,4%	15,2%
über 40 Jahre	24,0%	11,8%	18,8%	26,3%	17,9%	25,5%	23,9%

Die Angabe *COL(-1):R90* dreht die Kopftexte der unteren Stufe um 90 Grand gegen den Uhrzeigersinn.

XTAB: ROW / COL / FILTER: leere Zähloperanden

Hinter ROW oder COL kann der Zähloperand z zwischen zwei Semikolons auch fehlen.
Dann wird an dieser Stelle eine Zeile oder Spalte ohne Zählergebnisse gedruckt.
Diese Einrichtung dient dazu, zusätzliche Leerzeilen, Überschriftstexte oder Linien in die Zeilen oder Spalten der Tabellen einzufügen.

So wird zum Beispiel durch

```
ROW = TOTAL; ;C10.1 ...
```

zwischen die Zeile TOTAL und die Zeile zu C10.1 eine zusätzliche Leerzeile ohne Zählergebnisse und Texte eingefügt.

Zu einer solchen Leerzeile oder Leerspalte sind hinter einem Doppelpunkt auch alle Angaben des vorausgehenden Abschnitts *Erweiterte Druckaufbereitung* möglich, mit Ausnahme von ABS.
Zum Beispiel wird durch

```
ROW=TOTAL; :'Werbeaussagen'; C10 ...
```

hinter TOTAL eine Textzeile ohne Zählergebnisse aber mit dem Zeilentext *Werbeaussagen* eingefügt. Solche Texte zu leeren Zeilen dürfen sogar in den Zählerbereich der Tabelle hinein reichen.

Ein fehlender Zähloperand mit Textangaben kann auch in oberen (linken) Staffelstufen von ROW, COL oder FILTER verwendet werden. Er sorgt dort für zusätzliche Textzeilen, ohne die Zählergebnisse zu beeinflussen.

Beispiel: Leere Zeilen und Spalten

```
XTAB  COL=TOTAL;
          :'leere Spalte';;
          C10
      ROW=TOTAL;
          :'leere Zeile';
          C100;;
          C2030
      INSERT=R(1...-1)L2
```

Dieses Statement zeigt die Möglichkeit, Zeilen und Spalten ohne Zählergebnisse in die Tabellen einzufügen:

Gesamt 4594

	Gesamt	leere Spalte		Männer	Frauen
Gesamt	4594			992	3602
leere Zeile					
Alter					
14 - 18 Jahre	3			1	2
19 - 40 Jahre	238			76	162
über 40 Jahre	1101			329	772
Schulabschluss					
Volksschule, Hauptschule	110			15	95
Weiterführende Schule ohne Abitur	621			155	466
Abitur	579			149	430
Studium ohne Abschluss	240			35	205
Studium mit Abschluss	2998			627	2371
kein Abschluss	46			11	35

Hinter COL erzeugt *; : 'leere Spalte'* eine Spalte ohne Zählergebnisse mit einem Kopftext. Die Angabe ; ; direkt dahinter erstellt eine völlig leere Spalte.

Hinter ROW sorgt *; : 'leere Zeile'* für eine Zeile mit Zeilentext und ohne Zählergebnisse. Die Angabe *; ;* dahinter fügt eine völlig leere Zeile ein.

Durch *INSERT=R(1...-1)L2* entstehen die waagerechten Linien zwischen den Tabellenzeilen.

XTAB, ROW / COL / FILTER: Staffelung zum Wichten und Filtern

Eine Staffel besteht aus Zähloperanden *z*, die in geschweifte Klammern eingeschlossen sind:

```
{z; z; …}
```

Zwei direkt hintereinander stehende Staffeln

```
{a; b; … }{x; y; … }
```

bewirken, dass ihre Zählausdrücke paarweise miteinander kombiniert werden. Dies in der Reihenfolge:

```
ax; ay; … ; bx; by; …
```

Zur Auswertung eines solchen Paares *ax* wird für jeden statistischen Fall der Wert von *a* mit dem von x multipliziert und das Ergebnis in die entsprechenden Tabellenzähler addiert.

Es können bis zu zehn solcher Staffeln in der Form

```
{a; a; …}{b; b; …} … {x; x; …}
```

hintereinander stehen.

Mit Hilfe dieser Staffelungen lassen sich ganze Tabellen oder Teile davon mit festen Gewichten, Projektionsfaktoren, Fallgewichten oder zusätzlichen Filterbedingungen versehen.

Bei Staffelungen in FILTER wird für jeden kombinierten Operanden *ax* eine eigene Tabelle erstellt. Zum Beispiel liefert die Angabe

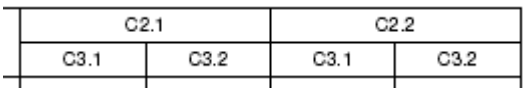
```
FILTER={N100}{TOTAL;C10.1;C10.2}
```

drei Tabellen.
Die erste davon wird mit N100 und TOTAL gefiltert: TOTAL sorgt dafür, dass alle statistischen Fälle auszuwerten sind, und N100 bewirkt, dass alle Werte der Tabelle pro Fall mit dem Wert von N100 multipliziert werden. N100 dient hier also als Fallgewicht.
Die zweite Tabelle wird mit N100 und C10.1 gefiltert: Es werden nur solche statistischen Fälle ausgewertet, die die Bedingung C10.1 erfüllen. Sämtliche Werte der Tabelle werden wieder pro Fall mit dem Wert von N100 multipliziert.
Die dritte Tabelle wird analog erstellt. Hier wird lediglich statt C10.1 die Bedingung C10.2 verwendet.

Bei COL wird für jeden kombinierten Operanden *ax* eine eigene Spalte eingerichtet.
Zum Beispiel führt die Angabe

```
COL={C2.1;C2.2}{C3.1;C3.2}
```

zu folgendem Tabellenkopf:



In der ersten Spalte werden die Fälle gezählt, für die sowohl die Bedingung C2.1 als auch die Bedingung C3.1 erfüllt ist. Für die übrigen Spalten gelten die entsprechenden Kombinationen.

Bei ROW wird für jeden kombinierten Operanden *ax* eine eigene Zeile eingerichtet. Die Staffelung führt dabei zum Einrücken der Texte. Zum Beispiel könnte die Angabe
ROW={C2.1;C2.2}{C3.1;C3.2} zu folgenden Zeilentexten führen:

C2.1	
C3.1	1158
C3.2	1139
C2.2	
C3.1	1178
C3.2	1119

Der Wert 1158. in der ersten Tabellenzeile entsteht durch Multiplikation des Wertes 0 oder 1 der Bedingung C2.1 mit dem Wert 0 oder 1 der Bedingung C10.1wird. Das Ergebnis wird in die entsprechende Tabellenzelle addiert.

Staffeln in ROW oder COL können auch numerische oder logische Ausdrücke enthalten. Jedoch dürfen zwei verschiedenen Stufen einer Staffelung nicht gleichzeitig Ausdrücke besitzen. In FILTER sind numerische Ausdrücke nicht möglich.

Beispiel

```
XTAB  ROW=TOTAL;{TOTAL:'ungewichtet';N1:'gewichtet'}{C10.1;C10.2}
      COL=TOTAL;{C1030}{C100}
```

Dieses Statement zeigt Staffelungen in den Zeilen und im Kopf:

Gesamt 4594

	Gesamt	F3: Tätigkeit als leitender Angestellter								
		unteres Management: schwierige Aufgaben, nach Vorgabe selbständige Ausführung			mittleres Management: selbständiger Leistung, begrenzte Personal und Budgetverantwortung			oberes Management: umfassende Führungsaufgaben und Entscheidungsbefugnisse		
		Alter			Alter			Alter		
		14 - 18 Jahre	19 - 40 Jahre	über 40 Jahre	14 - 18 Jahre	19 - 40 Jahre	über 40 Jahre	14 - 18 Jahre	19 - 40 Jahre	über 40 Jahre
Gesamt	4594	-	5	26	-	6	86	-	5	16
ungewichtet										
Männer	992	-	1	4	-	-	19	-	-	-
Frauen	3602	-	4	22	-	6	67	-	5	16
gewichtet										
Männer	40	-	0	0	-	-	1	-	-	-
Frauen	108	-	0	1	-	0	2	-	0	0

XTAB, ROW / COL / FILTER: logische und numerische Ausdrücke

Als Zähloperanden für die Tabellenzeilen ROW und Tabellenspalten COL sind auch logische und numerische Ausdrücke möglich. Dabei ist streng zwischen logischen und numerischen Ausdrücken zu unterscheiden: Logische Ausdrücke enthalten die Operatoren | & ^ < > =. In numerischen Ausdrücken dürfen diese nicht vorkommen.

Als Zähloperanden für die Tabellenzeilen ROW und Tabellenspalten COL sind auch logische und numerische Ausdrücke möglich. Dabei ist streng zwischen diesen beiden zu unterscheiden: Logische Ausdrücke enthalten die Operatoren | & ^ < > =. In numerischen Ausdrücken dürfen diese nicht vorkommen.

Numerische Ausdrücke

In XTAB bestehen diese aus den Operanden des Abschnitts *Elementare Zähloperanden* und den folgenden Operatoren:

		Beispiele	
+ –	Vorzeichen;	–25 oder –N132	
**	Potenzierung;	N10 ** 5.8	
*	Multiplikation;	N10 * N20	
/	Division;	N10 / 10	Division durch 0 führt zum Ergebnis 0.
+	Addition;	N10 + N20	
–	Subtraktion;	N10 – 15.3	

Bei der Auswertung von Ausdrücken wird grundsätzlich mit doppelt genauen Gleitkommazahlen gearbeitet. Das erlaubt eine Genauigkeit von 16 Dezimalstellen.

Zur Auswertung eines Ausdrucks wird für jeden Operanden getrennt die Summe der Werte für alle statistischen Fälle ermittelt. Erst danach werden diese Summen miteinander verrechnet.

Für den Ausdruck

```
COL=N10*C20.1
```

wird zunächst die Summe der Werte der Variablen N10 für alle statistischen Fälle gebildet und getrennt davon die Summe der erfüllten Merkmale C20.1. Erst zur Ausgabe der Tabelle werden diese beiden Summen miteinander multipliziert. In dem Beispiel

```
COL=N10;C20.1;N10*C20.1
```

läuft das darauf hinaus, dass die in der Tabelle ausgewiesenen Werte der beiden ersten Spalten miteinander multipliziert in der dritten Spalte erscheinen.

Es besteht ein wesentlicher Unterschied zwischen den numerischen Ausdrücken in XTAB und denen der Folgestatements von MOD-PROC, ADD-PROC usw.:
Für den numerischen Ausdruck

```
-N100=N10*C20.1
```

hinter MOD-PROC werden die Werte der Variablen N10 und des Merkmals C20.1 für jeden statistischen Fall miteinander multipliziert und das Ergebnis der Variablen N100 zugewiesen.

Die Ausdrücke in XTAB dürfen auch die folgenden Funktionen enthalten. Dazu wird zunächst für jedes Argument der Funktion die Summe der Werte aller statistischen Fälle gebildet. Diese Summe wird dann als Argument in die Funktion eingesetzt:

ABS Absolutbetrag.
Zum Beispiel liefert ABS(N10) den absoluten Betrag der Summe der Werte von N10.

CL Ceiling = kleinste ganze Zahl, die größer oder gleich dem Argument ist.
Zum Beispiel liefern CL(1.1) und CL(2) beide den Wert 2.

COS Cosinus.
Die Argumente müssen im Bogenmaß angegeben werden. Zum Beispiel liefert COS(3.1416) den Wert -1. Zur Umrechnung: Bogenmaß = Grad * 2 ⊏ / 360.

EXP Exponentialfunktion.
Zum Beispiel liefert EXP(N10) die gleichen Werte wie (2.7182...)**N10.

FL Floor = größte ganze Zahl, die kleiner oder gleich dem Argument ist.
Zum Beispiel liefern FL(1.8) und FL(1) beide den Wert 1.

LN Logarithmus zur Basis e.
Zum Beispiel liefert LN(1) den Wert 0. Zu nicht positiven Argumenten ergibt LN den Wert 0.

SIN Sinus.
Die Argumente müssen im Bogenmaß angegeben werden. Zum Beispiel liefert SIN(3.1416) den Wert 0. Zur Umrechnung: Bogenmaß = Grad * 2 ⊏ / 360.

SQRT Quadratwurzel.
Zum Beispiel liefert SQRT(N10) die Quadratwurzel der Werte der Variablen N10.
Zu negativen Argumenten ergibt SQRT den Wert 0.

Diese Funktionen lassen als Argumente auch zusammengesetzte Ausdrücke zu. Zum Beispiel:
ABS((N10 + N20)**2)

Jeder numerische Ausdruck in XTAB kann bis zu 255 Operatoren und Operanden besitzen und verträgt praktisch unbegrenzt viele Klammern (und).

Die Operatoren eines Ausdrucks werden nach dieser Prioritätenfolge abgearbeitet:

hohe Priorität	( )	Klammern um Teilausdrücke und Funktionsargumente
	+ –	Vorzeichen Plus und Minus
	**	Potenzierung
	* /	Multiplikation und Division
niedrige Priorität	+ –	Addition und Subtraktion

Stehen mehrere Operatoren gleicher Priorität ohne Klammern direkt hintereinander, wird von links nach rechts ausgewertet.

Der Wert des Ausdrucks

```
N10+N20/10+ABS(N30-10)*ID
```

wird daher auf folgende Weise errechnet:

Zunächst werden für alle statistischen Fälle die Summen zu den einzelnen Operanden gebildet: Die Summen aller Werte zu N10, die Summen aller Werte zu N20, die Summen aller Werte zu N30 und die Summen aller ID-Werte. Erst am Ende der Auswertung aller statistischen Fälle wird der obige Ausdruck mit den aufgelaufenen Summen berechnet.
Dazu wird zunächst der Quotient N20/10 ermittelt und zwischengespeichert. Danach wird, der Klammern wegen, die Differenz N30-10 gebildet und der absolute Betrag daraus ermittelt. Dieser wird mit der Summe der Werte aus ID multipliziert und das Ergebnis zwischengespeichert. Schließlich werden alle zwischengespeicherten Summanden auf die Summe der N10-Werte addiert.

Logische Ausdrücke

In XTAB sind auch alle logischen Ausdrücke möglich, wie im Abschnitt *Logische Ausdrücke* vorne beschrieben. Zusätzlich ist in XTAB noch der Operand KOMP möglich.

Zum Beispiel:

```
ROW=(C17.1|.4)&C20.Z1
```

Logische Ausdrücke in XTAB werden jedoch – anders als die numerischen – fallweise ausgewertet. So werden in dem Beispiel

```
ROW=((N1*N2)>17)&C20.1
```

für jeden statistischen Fall die Werte von N1 und N2 miteinander multipliziert. Ist das Ergebnis größer 17 und gleichzeitig die Bedingung C20.1 erfüllt, so wird der vorliegende Fall in der entsprechenden Tabellenzeile gezählt.

Der Ausdruck

```
ROW=N1*N2
```

enthält dagegen keinen der Operatoren & | ¬ < > = . Es liegt also ein numerischer Ausdruck vor, für den zunächst die Summen von N1 und N2 für alle statistischen Fälle getrennt gebildet werden. Erst bei der Tabellenausgabe werden diese Summen miteinander multipliziert.

Beispiel: Ausdrücke

```
XTAB ROW=TOTAL;
          2+(N1+N100)/3:'numerischer Ausdruck';
          2+(N1+N100)/3;
          C3000.1|.4|.5:'logischer Ausdruck';
          C3000.1|.4|.5
     COL=TOTAL;C10.1;.2
```

Dieses Statement kann folgende Tabelle erzeugen:

Gesamt 4594

	Gesamt	Männer	Frauen
Gesamt	4594	992	3602
numerischer Ausdruck	4598	767	3833
2+(N1+N100)/3	4598	767	3833
logischer Ausdruck	1476	200	1276
C3000.1|.4|.5	1476	200	1276

Der erste numerische Ausdruck *2+(N1+N100) / 3* ist mit dem Text *numerischer Ausdruck* versehen, der als Zeilentext in der Tabelle erscheint.

Die zweite Zeile enthält den gleichen numerischen Ausdruck, jedoch ohne Text. Dafür wird der Ausdruck selbst als Zeilentext verwendet.

Ganz analog wird mit den logischen Ausdrücken in den beiden letzten Zeilen verfahren.

Abkürzungen

Wird in einem numerischen Ausdruck hintereinander mehrmals die gleiche Bedingungsvariable Cn angesprochen, so muß Cn nur im ersten Operanden ausgeschrieben werden:

Statt: C80.1 + C80.2 * C80.Z3 + C80.S
Genügt: C80.1 + .2 * .Z3 + .S

Eine solche Abkürzung kann auch Variable aus davor stehenden Zählausdrücken verwenden, die durch ein Semikolon abgetrennt sind:

Statt: C80.1 + C80.2 ; C80.3 + C80.4 ; C80.5 + C80.6
Genügt: C80.1 + .2 ; .3 + .4 ; .5 + .6

Stehen in einem Ausdruck mehrere Merkmale der gleichen Bedingungsvariablen direkt hintereinander, lückenlos aufsteigend und mit dem gleichen Operator verknüpft, so genügt es, das erste und letzte Merkmal aufzuführen:

Statt: C80.1 + C80.2 + C80.3 + C80.4 + C80.5
Genügt: C80.1 + ..5

Sind in einem Ausdruck mehrere direkt hintereinander liegende numerische Variable mit dem gleichen Operator verknüpft, so muß nur die erste und die letzte Variable aufgeführt werden:

Statt: N12 + N13 + N16 + N17
Genügt: N12 + .. 17

In diesem Beispiel wird unterstellt, dass die Variablen N14 und N15 nicht definiert sind.

Wird in einem Ausdruck mehrmals hintereinander die gleiche Variable mit Index angesprochen, kann die Variablenbezeichnung weggelassen werden:

Statt: N10[1] + N10[2] + N10[3] + N10[4] + N10[5]
genügt: N10[1] + .. [5]
oder auch: N10[1] + ...5

Statt: N10[1] + N10[3] + N10[5] + N10[7]
genügt: N10[1] + [3] + [5] + [7]

Wird mehrmals hintereinander die gleiche Funktion mit direkt aufeinander folgenden Variablen benutzt, so muß die Funktion nur einmal geschrieben werden:
Statt: MEAN(N1) ; MEAN(N2) ; MEAN(N3) ; MEAN(N4)
genügt: MEAN(N1) ; .. 4

Entsprechend kann abgekürzt werden, wenn Variablen mit Index direkt hintereinander in einer Funktion verwendet werden:
Statt: MEAN(N2[10]) ; MEAN(N2[11]) ; MEAN(N2[12]) ; MEAN(N2[13])
Genügt: MEAN(N2[10]) ; .. [13]
oder auch: MEAN(N2[10]) ; .. 13

XTAB, ROW / COL / FILTER: numerische Konstanten

Numerische Konstanten bestehen aus maximal 18 Ziffern und dürfen auch Nachkommastellen besitzen.
Als Dezimalzeichen ist der Punkt zu verwenden, das Komma wird nicht akzeptiert.

Solche Konstanten werden zunächst in ROW oder COL als Teil numerischer Ausdrücke verwendet.
Ein Beispiel ist die Konstante 1 in

```
ROW= ... ; N3/(TOTAL-1) ; ...
```

Hier wird zunächst die Summe der Werte N3 aus allen statistischen Fällen gebildet sowie von der Anzahl aller Fälle der konstante Wert 1 abgezogen. Der Quotient dieser beiden Zwischenwerte wird schließlich in die Tabelle ausgegeben. (Siehe im Abschnitt *Ausdrücke* unter COL/ROW/FILTER).

Numerische Konstanten können aber auch als eigenständige Zähloperanden in ROW oder COL erscheinen. Eine solche Konstante wird anstelle eines Zählergebnisses in alle Zellen der entsprechenden Zeile oder Spalte gestellt. So führt zum Beispiel die Angabe

```
ROW= ... ; 10.5 ; ...
```

dazu, dass in einer Zeile der Tabelle durchgängig der Wert 10.5 erscheint.

In den oberen (linken) Stufen einer Staffelung werden einzelne Konstanten, die nicht Teil eines numerischen Ausdrucks sind, als feste Wichtungs- oder Projektionsfaktoren verarbeitet. Die Zählbeiträge der einzelnen statistischen Fälle werden vor der Addition in die Tabelle mit diesem konstanten Wert multipliziert. (Siehe im vorigen Abschnitt *Staffelung zum Wichten und Filtern*).

Beispiel

```
XTAB  ROW=TOTAL;
          17;
          17:'A';
          N1/0.95;
          {1.25}{N1}
      COL=TOTAL;
          25;
          25:'B'
```

Dieses Statement zeigt die Wirkung von numerischen Konstanten an verschiedenen Stellen einer Tabelle.

	Gesamt		B
Gesamt	4594	25	25
	17		
A	17		
N1/0.95	156		
Gewicht	185	25	25

Werden Konstanten als eigener Zääöhloperand verwendet, wie hier die 17 in ROW und die 25 in COL, so erscheinen diese Werte direkt in der Tabelle.

Treffen in einer Tabelle Zeilen mit Konstanten oder numerischen Ausdrücken auf Spalten, die ebenfalls Konstante oder numerische Ausdrücke besitzen, so werden keine Werte ausgewiesen.

XTAB, ROW / COL / FILTER: Klassenbildung

Die Angaben PRINT ... PRINT3 für die Aufbereitung der Zählergebnisse, SORT zum Sortieren der Zeilen und Spalten sowie ROUND und PROSUM zum Runden der Zählergebnisse können auf Teile einer Tabelle begrenzt werden.

Dazu sind die Nummern der betroffenen Zeilen und Spalten anzugeben. Zum Beispiel prozentuiert

```
PRINT=R(2 7..10)PROW1
```

die Werte der Zeilen 2 und 7 bis 10 auf die Werte der ersten Zeile.

Die gesondert zu behandelnden Zeilen oder Spalten lassen sich in ROW oder COL aber auch zu Klassen zusammenfassen. Dazu sind die gewünschten Zählausdrücke in geschweifte Klammern { ... } einzuschließen, mit einem Buchstaben A ... Z davor. Zum Beispiel:

```
XTAB  ROW=TOTAL;B{C1.1;..5}
      PRINT=R(B)PROW1
```

Hier bilden die Zähloperanden *C1.1 ; ..5* die Klasse B. Hinter PRINT bezieht sich die Angabe R(B) auf alle Zählausdrücke der Klasse B in ROW. Die Werte zu TOTAL werden durch diese PRINT-Anweisung nicht aufbereitet.

Jeder Klassenbuchstabe A ... Z kann gleichzeitig in ROW und COL erscheinen und für unterschiedliche Zwecke verwendet werden. Klein- und Großbuchstaben beschreiben die gleiche Klasse.

Nicht direkt hintereinander stehende Zähloperanden lassen sich trotzdem zu einer Klasse zusammenfassen:

```
ROW=TOTAL;A{C1};.S;A{C2};.S
PRINT=R(A)PROW1
```

Hier bilden die Merkmale C1.1 ... und C2.1 ... zusammen die Klasse A, auf die sich PRINT mit R(A) bezieht.

Klassen lassen sich schachteln:

```
COL=TOTAL;A{B{C47};C{C95}}
```

Hier bilden die Merkmale der Variablen C47 die Klasse B, die der Variablen C95 die Klasse C und beide zusammen die Klasse A.

Innerhalb einer Klassenklammer { . . . } sind Staffelungen oder die t-Tests MTT und MTTU möglich:

```
ROW=A{{C1}{TOTAL;C101};MTT{MEAN(N17);MEAN(N21)}}
```

Umgekehrt lassen sich Klassenklammern auch innerhalb von t-Tests MTT und MTTU angeben:

```
COL=MTT{TOTAL;A{C25}}
```

Schließlich sind Klassenklammerungen auch in Staffeln möglich, dort aber nur auf der unteren Stufe:

```
COL={TOTAL;N10;1.375}{TOTAL;A{C25}}
```

Beispiel: Klassenbildung

```
XTAB  COL=TOTAL;C10
      ROW=B{TOTAL};
          P{C100};
          M{MEAN(N100)}
      PRINT=R(B)ABS R(P)PROW1,% R(M)ABS,2
```

Dieses Statement kann folgende Tabelle erzeugen:

	Gesamt	Männer	Frauen
Gesamt	4594	992	3602
Alter			
14 - 18 Jahre	0%	0%	0%
19 - 40 Jahre	5%	8%	4%
über 40 Jahre	24%	33%	21%
jährliche Gesamtausgaben			
Mittelwert	2,99	2,31	3,18

XTAB, ROW / COL / FILTER: Nettoreichweite zur Mediaplanung

In ROW, COL und FILTER lassen sich die Nettoreichweiten von Medienstreuplänen ermitteln.
Dabei sind auch Wirkungskurven für die wirksamen Reichweiten möglich.

Zur exakten Auswertung der Streupläne für die Kontaktklassen ist ein hoher Rechenaufwand erforderlich (Faltung von binomialverteilten Wahrscheinlichkeiten). CNTA vermeidet diesen Aufwand mit Hilfe eines Approximationsverfahrens, das bei guten Laufzeiten sehr gute Näherungswerte liefert.

Zur Ermittlung der Nettoreichweiten stehen folgende Funktionen zur Verfügung, die außer in XTAB und XADD auch in den Wertezuweisungen hinter ADD-PROC, MOD-PROC und UPD-PROC zulässig sind:

RCH (Nn_1 **:** m_1 **N**n_2 **:** m_2 ... **<** ; k_1 **:** w_1 k_2 **:** w_2 ... **>)**

Diese Funktion ermittelt die Nettoreichweite *r* eines Medienstreuplans.

Hierbei müssen die Variablen *Nn* Kontaktwahrscheinlichkeiten zwischen 0.0 und 1.0 für die Medien des Streuplans enthalten. Der Wert m hinter jeder Variablen und dem Doppelpunkt gibt die Belegungshäufigkeit oder Frequenz 1, 2 ... des entsprechenden Mediums an. Es ist eine praktisch unbegrenzte Anzahl von Argumentpaaren erlaubt, numerische Ausdrücke sind als Argumente aber nicht möglich. Die Berechnung erfolgt auf folgende Weise:

$$r = \sum \left(1 - (1 - p_1)^{m_1} (1 - p_2)^{m_2} \ldots\right)$$

p_i = Kontaktwahrscheinlichkeit aus den Variablen Nn_i.

m_i = Belegungshäufigkeit des Mediums der Variablen Nn_i.

Der Wert Z0 (keine Angabe) in den beteiligten N-Variablen wird wie 0 behandelt, und Wahrscheinlichkeitswerte über 1 wie 1.

Zusätzlich kann zu einer solchen Reichweitenfunktion eine Wirkungskurve zur Ermittlung der wirksamen Reichweite des Streuplans angegeben werden. Dazu sind folgende Angaben möglich:

k_i = obere Grenze 1 ... 65 635 der jeweiligen Kontaktklasse. Sie muss größer als die Grenze der vorausgehenden Klasse sein.

w_i = Kontaktgewicht 0.0 ... 1.0 der entsprechenden Kontaktklasse. Jedes Gewicht darf bis zu 8 Nachkommastellen besitzen.

Siehe hierzu auch den folgenden Abschnitt *Kontaktklassen zur Mediaplanung*.
Nettoreichweiten von Streuplänen können auch pro Fall hinter ADD-PROC, UPD-PROC und MOD-PROC ermittelt werden. Siehe dazu *Numerische Ausdrücke*.

RCHM (Nn$_1$ Nn$_2$ ...)

Diese Funktion ermittelt die Mehrfachleser der angegebenen Titel, also diejenigen, die alle Titel gleichzeitig lesen. Die Berechnung erfolgt durch:

$$r = p_1 \cdot p_2 \ldots$$

p_i = Kontaktwahrscheinlichkeit aus den Variablen Nn$_i$.

RCHX (Nn$_1$ Nn$_2$ ...)

Diese Funktion ermittelt die Exklusivleser des ersten Titels, also die Leser des ersten Titels, die die übrigen Titel nicht lesen. Die Berechnung erfolgt zum Beispiel bei drei Titeln durch:

$$r(p_1\ p_2\ p_3) = p_1(1 - p_2 \cdot p_3 + p_2 + p_3)$$

p_i = Kontaktwahrscheinlichkeit aus den Variablen Nn$_i$.

Beispiel

```
XTAB  PRINT=ABS       PRINT1=PROW1,%
      ROWS=TOTAL;;
           RCH(N2:6 N3:2 N4:4):'Plan 1';
           RCH(N2:5 N3:4 N4:5):'Plan 2'
     COLS=TOTAL;C10
```

Dieses Statement kann folgende Tabelle erzeugen:

	Gesamt		Männer		Frauen	
Gesamt	4594	100%	992	100%	3602	100%
Plan 1	1669	36%	401	40%	1267	35%
Plan 2	2076	45%	488	49%	1588	44%

In ROWS ermittelt die Funktion RCH die Nettoreichweiten der beiden folgenden Streupläne:

Medium	**Plan 1**	**Plan 2**
N2	6 x	5 x
N3	2 x	4 x
N4	4 x	5 x

PRINT und PRINT1 weisen die Ergebnisse als Absolut- und Prozentwerte aus.

XTAB: Kontaktklassen zur Mediaplanung

Hier besteht die Möglichkeit, Medienstreupläne und Kontaktklassen vorzugeben und zu jeder Klasse die Reichweiten, die kumulierten Reichweiten, die Kontakte und die kumulierten Kontakte auszuweisen. Dies kann in ROW, COL oder FILTER geschehen. Zu jeder vorgegebenen Kontaktklasse wird entsprechend eine Zeile, Spalte oder Tabelle erstellt.

Die dazu erforderlichen Angaben EC, ECC, ECR, ECRC ähneln der Reichweitenfunktion RCH; siehe im Abschnitt *Nettoreichweite zur Mediaplanung*. Sie werden von CNTA nicht als numerischer Ausdruck verstanden. Daher können sie auch in FILTER verwendet werden, um zu jeder Kontaktklasse eine eigene Tabelle zu erstellen. Weiter können zum Beispiel Kontaktklassen in ROWS angegeben werden und gleichzeitig Ausdrücke wie MEAN oder andere Rechenoperationen in COLS vorkommen.

Zur exakten Auswertung der Streupläne für die Kontaktklassen ist ein hoher Rechenaufwand erforderlich (Faltung von binomialverteilten Wahrscheinlichkeiten). CNTA vermeidet diesen Aufwand mit Hilfe eines Approximationsverfahrens, das bei guten Laufzeiten sehr gute Näherungswerte liefert.

EC(Nn_1 **:** m_1 **N**n_2 **:** m_2 ... **;** k_1 k_2 ... **)**

Dieser Operand wertet den vor dem Semikolon stehenden Medienstreuplan nach den hinter dem Semikolon stehenden Kontaktklassen aus (Exposure Classes).
Pro Kontaktklasse liefert er für jeden ausgewerteten statistischen Fall die Anzahl der Kontakte in dieser Klasse.
Dazu müssen die Werte der Variablen *Nn* Wahrscheinlichkeiten zwischen 0.0 und 1.0 für die Medienkontakte sein. Der Wert *m* hinter jeder Variablen und dem Doppelpunkt ist die Belegungshäufigkeit oder Frequenz 1, 2 ... für die davor stehende Variable im Streuplan. Es ist eine praktisch unbegrenzte Anzahl von Argumentpaaren erlaubt, Ausdrücke sind als Argumente aber nicht möglich.

Besitzt eine der N-Variablen des Streuplans den Wert Z0 (keine Angabe), so wird dieser wie 0 verarbeitet. Wahrscheinlichkeitswerte über 1 werden wie 1 behandelt.

Mit Hilfe der Werte $k = 1 \ldots 65\,635$ hinter dem Semikolon werden die oberen Grenzen der einzelnen Kontaktklassen festgelegt. So sorgt die Angabe:

```
EC( ... ; 2 3 5 10 )
```

für die Auswertung der folgenden Kontaktklassen:

```
1...2
3
4...5
6...10
11...
```

Zu jeder dieser Kontaktklassen wird eine Zeile, Spalte oder Tabelle erstellt, abhängig davon, ob EC in ROW, COL oder FILTER angegeben wurde.
Zu jeder Klasse verwendet CNTA dabei automatisch die hier gezeigten Von-Bis-Angaben als Zeilen-, Spalten- oder Filtertext.

ECC (Nn_1 : m_1 Nn_2 : m_2 ... ; k_1 k_2 ...)

Diese Funktion arbeitet analog zu EC. Anstelle der Kontakte pro Kontaktklasse werden jedoch die kumulierten Kontakte der Kontaktklassen ausgewiesen. Die erste Klasse enthält daher stets die Kontaktsumme des Streuplans.

ECR (Nn_1 : m_1 Nn_2 : m_2 ... ; k_1 k_2 ...)

Diese Funktion arbeitet analog zu EC. Anstelle der Kontakte pro Kontaktklasse werden jedoch die Nettoreichweiten des Streuplans pro Kontaktklasse ausgewiesen.

ECRC (Nn_1 : m_1 Nn_2 : m_2 ... ; k_1 k_2 ...)

Diese Funktion arbeitet analog zu EC. Anstelle der Kontakte pro Kontaktklasse werden jedoch die kumulierten Nettoreichweiten pro Kontaktklasse ausgewiesen. Die erste Klasse enthält daher stets die Nettoreichweite des Streuplans.

Beispiel

```
XTAB   PRINT=ABS   PRINT1=PROW1,%
       ROWS=TOTAL;;
            ECR(N2:6 N3:12 N4:4 ; 2 4 ):'Reichweiten';;
            EC(N2:5 N3:14 N4:5 ; 2 4 ):'Kontakte'
       COLS=TOTAL;C10
```

Dieses Statement kann folgende Tabelle erzeugen:

	Gesamt		Männer		Frauen	
Gesamt	4594	100%	992	100%	3602	100%
Reichweiten						
1...2	2763	60%	591	60%	2172	60%
3...4	709	15%	189	19%	521	14%
5...	7	0%	7	1%	-	-
Kontakte						
1...2	4095	89%	891	90%	3204	89%
3...4	2148	47%	597	60%	1551	43%
5...	-	-	-	-	-	-

Hinter ROWS ermittelt ECR die Reichweiten in den vorgegebenen Kontaktklassen für den Streuplan:

Medium	**Frequenz**
N2	6
N3	12
N4	4

Hinter dem Semikolon werden die Kontaktklassen 1 ... 2 und 3 ... 4 durch ihre Obergrenzen festgelegt.
Die Angabe EC in ROWS ermittelt die Kontakte zu gleichem Streuplan und Kontaktklassen.
PRINT und PRINT1 weisen die Ergebnisse als Absolut- und Prozentwerte aus.

XTAB: BOTTOM

```
BOTTOM  | = | 't'    <,  | L | > |
BOTTOMS |   |            | R |   |
                          | C |
                 | NO |
```

BOTTOM druckt einen Text unter der Tabelle, vor eventuellen Fußnoten.

BOTTOMS Diese Angabe gilt für das laufende und alle folgenden XTAB-Statements, bis sie durch neue BOTTOMS ersetzt wird.

BOTTOM Diese Angabe gilt nur für das laufende XTAB-Statement.
Eventuelle Angaben BOTTOMS aus früheren XTAB-Statements werden für das laufende XTAB unwirksam.

't' Der ein- oder mehrzeilige Text, der unter die Tabellen gedruckt werden soll; wie im Abschnitt Textzeilen beschrieben.

L R C Mit den Angaben L, R oder C wird der Text direkt unterhalb der Tabelle ausgegeben:
L = linksbündig, R = rechtsbündig, C = zentriert.
Zum Beispiel stellt
BOTTOM='Frage 24',C
den Text mittig unter die Tabelle.

Fehlen diese Angaben, so wird der Text linksbündig direkt unterhalb der Tabellen ausgegeben.

NO setzt die Angabe BOTTOMS aus früheren XTAB-Statements außer Kraft.

Beispiel

```
XTAB  COL=TOTAL;C10
      ROW=TOTAL;C2030
      BOTTOM='Diese Tabelle enthält nur Teilnehmer über 25 Jahre'
```

Diese Statements erzeugen folgende Tabelle:

	Gesamt	Männer	Frauen
Gesamt	4594	992	3602
Schulabschluss			
Volksschule, Hauptschule	110	15	95
Weiterführende Schule ohne Abitur	621	155	466
Abitur	579	149	430

Diese Tabelle enthält nur Teilnehmer über 25 Jahre

XTAB: CONT

Diese Angabe verhindert den Blattvorschub am Ende der Tabelle. Die nächste Tabelle oder TEXT-Ausgabe wird auf die gleiche Seite gestellt wie die laufende Tabelle.

Mit der zusätzlichen Angabe START lassen sich die Tabellen an jede gewünschte Position der Seite rücken. So lassen sich mehrere Tabellen auf einer Seite sowohl nebeneinander als auch übereinander in beliebigen Abständen ausgeben.

Die Fußnotentexte mehrerer Tabellen werden gesammelt am Ende jeder Seite ausgegeben.

Beispiel

```
XTAB   ROW=TOTAL;C549
       COL=C548
       SUPPRESS=FILTER
       CONT

XTAB   ROW=TOTAL;C549
       COL=C547
```

Diese Statements können folgende Tabellen erzeugen:

	Geschlecht	
	Männlich	Weiblich
TOTAL	1041	163
Schulabschluß		
Volksschule / Hauptschule	218	22
Mittel-/höhere Schule ohne Abitur	449	72
Abitur / Universität	374	69

	Alter				
	bis 29 Jahre	30 bis 39 Jahre	40 bis 49 Jahre	50 bis 59 Jahre	60 Jahre und älter
TOTAL	226	341	316	219	102
Schulabschluß					
Volksschule / Hauptschule	48	53	61	53	25
Mittel-/höhere Schule ohne Abitur	82	157	144	97	41
Abitur / Universität	96	131	111	69	36

Die Angabe CONT im ersten XTAB-Statement verhindert nach Ausgabe der ersten Tabelle den Vorschub auf eine neue Seite. Die Tabelle des zweiten XTAB-Statements erscheint auf der gleichen Seite.

Da das zweite XTAB-Statement keine START-Angabe besitzt, wird die zweite Tabelle direkt unter die erste gestellt.

XTAB: GROUP

GROUP	**=**	**ID**
GROUPS		**ID1**
		ID2
		ID3
		ID4

Durch GROUP wird eine besondere Form der Gruppenverarbeitung ermöglicht:
Alle statistischen Fälle, die in der Eingabe direkt hintereinander liegen und die in der hier angegebenen Fall-Identifikation übereinstimmen, werden als eine Gruppe aufgefasst.

Mit GROUP=ID1 wird zum Beispiel festgelegt, dass diejenigen statistischen Fälle eine Gruppe bilden, die in den Fall-Identifikationen ID und ID1 übereinstimmen.

Durch GROUP wird in ausgewählte Tabellenzellen nur einmal pro Gruppe ein Wert addiert; dann nämlich, wenn zum ersten Mal innerhalb einer Gruppe die FILTER-, ROW- und COL-Bedingungen für diese Zelle erfüllt sind. Für alle noch folgenden Fälle der gleichen Gruppe wird diese Tabellenzelle dann nicht mehr bearbeitet.

GROUP gilt für das laufende XTAB-Statement, GROUPS für das laufende und alle folgenden XADD und XTAB-Statements.

Ohne GROUP werden die Werte jedes statistischen Falles in den Tabellen gezählt, wenn nur die entsprechenden FILTER-, ROW- und COL-Bedingungen erfüllt sind.

GROUP ist Voraussetzung zur Gruppenverarbeitung. Zusätzlich sind in ROW, COL oder FILTER die für die Gruppenverarbeitung vorgesehenen Zähloperanden mit G- zu kennzeichnen. Zum Beispiel:

```
ROW=TOTAL;G-TOTAL;G-N10;G-MEAN(N100)
```

Hier wird die erste Tabellenzeile mit dem Operanden TOTAL konventionell ausgewertet.
Die restlichen Tabellenzeilen werden wegen der Angabe G- in Gruppen verarbeitet.

In Filter bewirkt ein G-Operand, dass die gesamte Tabelle der Gruppenverarbeitung unterworfen wird. In oberen Staffelstufen führt ein G-Ausdruck dazu, dass die dazugehörigen Zeilen oder Spalten mit der Gruppenregel verarbeitet werden. So ist zum Beispiel

```
ROW={TOTAL;G-TOTAL}{C10.1;C10.2}
```

gleichwertig mit:

```
ROW=C10.1;C10.2;G-C10.1;G-C10.2
```

Auch in numerischen Ausdrücken ist Gruppenverarbeitung möglich. So wird durch

```
G-MEAN(N100)
```

der Mittelwert der Variablen N100 ermittelt, auch dies nur für den jeweils ersten auszuwertenden Fall pro Gruppe.

Durch

```
N10/G-TOTAL
```

in ROW oder COL werden zunächst die Werte von N10 addiert und anschließend dividiert durch die Anzahl der verarbeiteten Gruppen.

Beispiel

```
INPUT-EXT  FNAME=DATEN.EXT        ID=D(1,3)      ID1=D(5,2)
INPUT-INT  FNAME=DATEN.INT        IDSHORT

DEFS
-C10:2    1='Kaffee gekauft'  2='keinen Kaffee'/'gekauft'
-N10:6,2  'Ausgaben für Kaffee in DM'

UPD-PROC
-N10=D1(8,4)2
-C10.1=N10>=0

XTAB  GROUP=ID
      COL=TOTAL;C1
      ROW=TOTAL;G-C10.1;N10/G-C10.1:ABS,2,'DM für Kaffee'/'im Monat'
      SUPPRESS=FILTER
```

Diese Statements könnten folgende Tabelle erzeugen:

	TOTAL	Männer	Frauen
TOTAL	291	139	152
Kaffee gekauft	87	34	53
DM für Kaffee im Monat	47.94	54.04	44.03

In diesem Beispiel werden Daten aus dem internen Datenbestand INPUT-INT und dem externen Bestand INPUT-EXT zusammen verarbeitet.

Der interne Bestand enthält persönliche Daten zu den Befragten. Davon soll hier nur die Variable C1 mit dem Geschlecht interessieren und eine dreistellige Personennummer als Fall-Identifikation ID.

Der externe Datenbestand enthält zu jeder dieser Personen Daten aus dem Haushaltsbuch eines Monats: Für jeden Einkaufstag ist dort pro Person ein Datensatz gespeichert. Ein solcher Datensatz enthält die dreistellige Personennummer als Fall-Identifikation ID sowie das Tagesdatum ID1 des Einkaufstages.

Da in INPUT-EXT eine zweistufige Fall-Identifikation vorliegt, in INPUT-INT aber nur eine einstufige, ist im Statement INPUT-INT die Angabe IDSHORT erforderlich. Diese Angabe sorgt für folgende Verarbeitung:
Zu jeder Personennummer wird zunächst der Datensatz aus INPUT-INT gelesen. Dazu werden aus INPUT-EXT die Datensätze mit der gleichen Personennummer hinzugelesen.

Jeder Datensatz aus INPUT-INT bildet mit einem dazugehörigen Datensatz aus INPUT-EXT einen statischen Fall. Die Variablenwerte aus INPUT-INT, hier etwa das Geschlecht 1, bleiben für alle statistischen Fälle mit gleicher Personennummer aus INPUT-EXT unverändert erhalten.

Die Daten aus INPUT-EXT werden dann durch die Statements von UPD-PROC in Variable gestellt: N10 erhält den Betrag in DM, der an dem jeweiligen Tag für Kaffee ausgegeben wurde. In C10.1 wird gespeichert, ob an diesem Tag auch Kaffee gekauft wurde.

In dem XTAB-Statement ermöglicht die Angabe GROUP=ID, dass in der Tabelle nicht nur Daten pro Fall (oder Tag) gezählt werden können, sondern auch Daten pro Person.

Die Zeile TOTAL in ROW enthält keine weitere Angabe. Daher werden dort wie gewohnt alle statistischen Fälle gezählt, hier also die Anzahl der Einkaufsvorgänge.
In der zweiten Zeile sorgt die Angabe G- vor C10.1 dafür, dass das Merkmal C10.1 nur einmal pro Personennummer ID gezählt werden soll, nämlich beim ersten Erscheinen für eine Personennummer. Hier wird also die Anzahl von Personen gezählt, die in dem Monat wenigstens einmal Kaffee gekauft haben.

In der letzten Zeile werden die DM-Beträge N10 für Kaffeekäufe für alle statistischen Fälle aufaddiert. Würde dort N10/C10.1 stehen, so würde die Gesamtsumme für Kaffeekauf durch die Anzahl der Kaffeekäufe dividiert.
Das Ergebnis wäre der durchschnittliche Betrag pro Kaffeekauf. Hier wird aber durch G-C10.1 dividiert. Dies ist aber gerade die Anzahl von Personen, die wenigstens einmal im Monat Kaffee gekauft haben. Das Ergebnis ist daher der durchschnittliche Betrag, der pro Kaffeekäufer im Monat insgesamt ausgegeben wurde.

Abkürzungen

Werden in hintereinander liegenden Zähloperanden Abkürzungen verwendet und ist der Ausgangsoperand mit der Angabe G- versehen, so gilt G- auch für die folgenden Operanden.

Statt: G-C80.1 ; G-C80.2 ; G-C80.Z0
genügt: G-C80.1 ; .2 ; .Z0

Statt: (G-C80.1+G-C80.2)/G-C80.3
genügt: (G-C80.1+.2)/.3

Statt: G-C1.1 ; G-C1.2 ; G-C1.3 ; G-C1.4
genügt: G-C1.1 ; ..4

Statt: G-N10 ; G-N11 ; G-N12 ; G-N10+G-N11+G-N12
genügt: G-N10 ; ..12 ; G-N10+ ..12

XTAB: INSERT

INSERT **INSERTS**	=	**R(** z z ...**)** <n> **C(** s s ...**)** <n> **NO**	**L**i **G**i **CO**i **SPACE**	**, R(** z z ...**)** ...

Mit INSERT lassen sich zusätzliche Leerzeilen und waagerechte Linien in die Tabellenzeilen einfügen oder Tabellenzeilen und -spalten mit Farbe oder Grautönen unterlegen. So erzeugt die Angabe

```
XTAB INSERT=R(6 11 21)L3
```

zum Beispiel vor den Zeilen 6, 11 und 21 der Tabelle eine waagerechte Linie L3.

Auch lassen sich einzelne Tabellenspalten durchgehend mit Tonflächen unterlegen. So unterlegt die Angabe

```
XTAB INSERT=C(6 11 21)G6
```

die Spalten 6, 11 und 21 mit dem Grauton G6 und zwar durchgehend durch alle Kopftexte und Zählerzeilen.

Die hinter R(...) und C(...) angegebenen Zeilen- und Spaltennummern verstehen sich nach eventueller Sortierung durch SORT oder Entfernung durch SUPPRESS. Somit erlaubt INSERT, in regelmäßigen Abständen Lesehilfen in die Tabellen einzufügen, auch wenn die Zeilen erst beim Drucken auf nicht vorhersehbare Weise angeordnet werden.

INSERT gilt für das laufende XTAB-Statement, INSERTS für das laufende und alle folgenden XTAB-Statements.

---

s Dies sind die Nummern 1 . . . 32 767 der Tabellenspalten, die mit einer Tonfläche zu unterlegen sind. Als Spalte wird dabei eine vollständige Angabe zwischen zwei Semikolons ; aus COL verstanden, auch wenn diese in der Tabelle aus zwei Teilspalten besteht.

Mehrere direkt hintereinander liegende Spaltennummern lassen sich als Von-Bis-Werte in der Form a .. . b angeben. Die Angabe

```
INSERT=C(2..5 8..12)G3
```

unterlegt zum Beispiel die Spalten 2 bis 5 und 8 bis 12 mit einer Tonfläche.

Mehrere direkt hintereinander liegende Spaltennummern können auch mit einer Anzahlangabe (a) in Klammern festgelegt werden. So werden mit

```
INSERT=R(2(4) 8(5))G3
```

gleichfalls die Spalten 2 bis 5 und 8 bis 12 mit einer Tonfläche versehen.
Die Anzahlangabe (a) darf zwischen 1 und 255 liegen.

Es sind auch negative Spaltennummern möglich. Dabei ist –1 die letzte Spalte der Tabelle, –2 die vorletzte usw. So versieht

```
INSERT=C(2..-2)G3
```

alle Tabellenspalten mit einer Tonfläche, mit Ausnahme der ersten und der letzten Spalte.

Sind Einfügungen in festen Spaltensprüngen gewünscht, so kann hinter einer Von-Bis-Angabe a . . b eine Schrittweite (s) in Klammern angegeben werden. Mit

```
INSERT=C(1..-1(2))CO11
```

wird zum Beispiel jede zweite Spalte mit der Farbe CO11 unterlegt.
Die Schrittweite (s) darf zwischen 1 und 255 liegen.

z Dies sind die Nummern 1 . . . 32 767 der Tabellenzeilen, die mit zusätzlichen Linien, Tonflächen oder Leerzeilen zu versehen sind. Linien und Leerzeilen werden immer *vor* die hier angegebene Zeile gestellt. Bei Tonflächen wird die Zeile selbst unterlegt.
Als Zeile wird dabei eine vollständige Angabe zwischen zwei Semikolons ; aus ROW verstanden, auch wenn diese in der Tabelle aus mehreren Text- oder Zählerzeilen besteht.

Mehrere direkt hintereinander liegende Zeilennummern lassen sich als Von-Bis-Werte in der Form a . . . b angeben. Die Angabe

```
INSERT=R(2..10 20..25)G3
```

unterlegt zum Beispiel die Zeilen 2 bis 10 und 20 bis 25 mit einer Tonfläche.

Mehrere direkt hintereinander liegende Zeilennummern können auch mit einer Anzahlangabe (a) in Klammern festgelegt werden. So werden mit

```
INSERT=R(2(9) 20(6))G3
```

gleichfalls die Zeilen 2 bis 10 und 20 bis 25 mit einer Tonfläche versehen.
Die Anzahlangabe (a) darf zwischen 1 und 255 liegen.

Es sind auch negative Zeilennummern möglich. Dabei ist –1 die letzte Zeile der Tabelle, –2 die vorletzte usw. So versieht zum Beispiel

```
INSERT=R(2..-2)G3
```

alle Tabellenzeilen mit einer Tonfläche, mit Ausnahme der ersten und der letzten Zeile.

Sind Einfügungen in festen Zeilensprüngen gewünscht, so kann hinter einer Von-Bis-Angabe a . . . b eine Schrittweite (s) in Klammern angegeben werden. Mit

```
INSERT=R(6..-1(5)) SPACE
```

wird zum Beispiel vor den Zeilen 6, 11, 16 usw. eine Leerzeile eingefügt.
Die Schrittweite (s) darf zwischen 1 und 255 liegen.

n Mit der Angabe n = 1 ... 255 gilt die Einfügung zusätzlich für jeweils n hintereinander liegende Zeilen oder Spalten, beginnend bei jeder Spalte s oder Zeile z.

Die Angabe

```
XTAB INSERT=R(1..-1(6))2,G3
```

unterlegt zunächst jede sechste Zeile mit dem Grauton G3. Wegen der Angabe 2 wird jedoch nicht nur eine Zeile unterlegt, sondern jeweils zwei hintereinander liegende, also die Zeilen 1, 2, 7, 8 usw.

Gi
COi Diese Eintragung unterlegt die durch R(...) oder C(...) angegebenen Zeilen oder Spalten mit dem Grauton G1 bis G16 oder der Farbe CO1 bis CO16 aus dem Statement PAGEP.
Mit dem Statement PAGE ist diese Angabe wirkungslos.
Die Angaben STYLE und PRINT erzeugen ebenfalls Tonflächen. Gleiches gilt für die erweiterte Druckaufbereitung hinter ROW und COL. Dort wird neben den Tabellenzählern jedoch nur die unterste Textstufe eingefärbt, während INSERT alle Textstufen der Zeilen und Spalten mit Tonflächen versieht, die aus C-Variablen mit Variablentexten oder Staffelungen entstehen.

Li Diese Angabe stellt vor jede zu R(...) angegebenen Zeile *z* eine Linie L1 ... L16 so wie sie im Statement PAGEP festgelegt wurde. Hinter dem Statement PAGE ist diese Angabe wirkungslos.
Linien sind für C(...) nicht möglich.

SPACE Diese Eintragung fügt vor jede der Tabellenzeilen R(...) eine Leerzeile ein.
SPACE ist hinter C(...) nicht möglich.

NO setzt die Angabe INSERTS aus früheren XTAB-Statements außer Kraft.

Beispiel: Linien und Tonflächen einfügen

```
XTAB  ROWS=TOTAL:'Basis-Fälle = 100',FT3;
          N1021;..1037
      COLS=TOTAL:'Gesamt';C2:L1
      STUBHEAD='LpA-Reichweite %'
      PRINT=R(1)ABS R(2..-1)PROW1,1,%
      SORT=ROWS1(2..-1)
      INSERT=R(2..-1(8))4,G3  R(2..-1(4))L1
```

Dieses Statement könnte folgende Tabelle erzeugen:

LpA-Reichweite %	Gesamt	West	Ost
Basis-Fälle = 100	**200**	**57**	**143**
ADAC Motorwelt	14,2%	12,3%	15,0%
Das Beste	9,0%	5,4%	10,4%
Burda Moden	8,4%	3,8%	10,2%
Capital	2,1%	1,5%	2,4%
Carina	2,7%	4,8%	1,8%
Cinema	0,3%	0,1%	0,4%
Cosmopolitan	2,5%	3,1%	2,2%
Coupe	1,4%	0,7%	1,6%
Deine Gesundheit	5,0%	-	6,9%
Eltern	5,6%	11,0%	3,5%
essen & trinken	1,6%	2,2%	1,4%
Extra Rätsel	4,2%	0,1%	5,8%
Flora	1,3%	0,1%	1,8%
Frau im Leben	0,3%	1,0%	-
GEO	1,1%	2,2%	0,7%
Glücks Rätsel	2,8%	0,1%	3,8%
Goldene Gesundheit	0,9%	1,5%	0,6%

Wegen SORT=ROWS1(2..-1) werden die Zeilen der Tabelle ab der zweiten Zeile nach den Werten der ersten Spalte sortiert.

Hinter INSERT unterlegt die Angabe R(2..-1(8))4,G3 zunächst einige Tabellenzeilen mit dem Grauton G3 und zwar ab der zweiten Zeile in Schritten von acht Zeilen jeweils vier hintereinander liegende Zeilen.

Anschließend fügt die Angabe R(2..-1(4))L1 noch Linien L1 in die Tabelle ein. Dies geschieht wieder ab der zweiten Zeile in Schritten von vier Zeilen.
Dadurch werden die Tonflächen jeweils oben und unten durch waagerechte Linien begrenzt.

XTAB: LINE

LINE | **LINES** **=** <**L**i> <**,VER:L**i> <**,HOR:Li>** <**,COL:L**i> <**,ROW:L**i>

Mit LINE lassen sich Linien für verschiedene Teile der Tabellen auswählen. Zum Beispiel sorgt die Angabe

```
LINE=L5,ROW=L0
```

dafür, dass in der Tabelle überall die Linie L5 verwendet wird. Nur zwischen den Zählerzeilen werden wegen ROW=L0 keine Linien ausgegeben.

Die Angabe LINE ist nur zusammen mit dem Statement PAGEP wirksam. Bei der Arbeit mit PAGE lassen sich Linien nur aus einzelnen Zeichen zusammensetzen. Das geschieht mit Hilfe der Angabe LCHARS im Statement PAGE sowie TAB im Statement XTAB.

Fehlt LINE oder LINES, so wird L3 für VER, HOR und COL verwendet, sowie L0 für ROW.

Linienangaben hinter ROW, COL oder INSERT haben Vorrang von LINE.

LINES Die Angaben gelten für das laufende XTAB und alle folgenden XTAB-Statements, bis sie durch neue Angaben LINE oder LINES ersetzt werden.

LINE Diese Angabe gilt nur für das laufende XTAB-Statement.
Eventuelle Angaben LINES aus früheren XTAB-Statements werden für das laufende XTAB unwirksam, dahinter aber wieder aktiv.

VER Hiermit werden alle senkrechten Linien einer Tabelle festgelegt. Das sind die rechte und die linke Randlinie der Tabelle, aber auch die Linien zwischen den Zählerspalten.
Die Linie vor jeder einzelnen Zählerspalte kann durch Angaben hinter COL noch verändert werden.
Siehe Abschnitt *COL/ROW/FILTER: Erweiterte Druckaufbereitung*.

HOR Damit lassen sich die waagerechten Randlinien der Tabelle verändern. Das sind die obere und die untere Randlinie der Tabelle sowie die Linie zwischen den Kopftexten und der ersten Zählerzeile.
Weitere horizontale Linien zwischen den Zählerzeilen lassen sich durch Angaben hinter ROW erzeugen. Siehe Abschnitt *COL/ROW/FILTER: Erweiterte Druckaufbereitung*.

COL Dies sind die waagerechten Linien innerhalb der Kopftexte, die durch Staffelungen entstehen.
Hiermit können zum Beispiel zwischen den Kopftexten dünnere Linien gedruckt werden als darüber und darunter.

ROW Dies sind horizontale Linien zwischen den Zählerzeilen. Weitere horizontale Linien zwischen den Zählerzeilen lassen sich durch Angaben hinter ROW erzeugen und durch INSERT.
Siehe Abschnitt *XTAB, COL/ROW/FILTER: Erweiterte Druckaufbereitung* und Abschnitt *XTAB, INSERT* .

Li Diese Angabe wählt eine der Linien L1 ... L16 aus, so wie sie im Statement PAGEP definiert sind.
Durch L0 wird die Ausgabe einer Linie an der entsprechenden Stelle unterbunden.

Beispiel

```
XTAB  ROW=TOTAL:'Basis';;;C10;;C11
      COL=TOTAL:'Gesamt';C20;C30

XTAB  ROW=TOTAL:'Basis';:L3;C10;:L3;C11
      COL=TOTAL:'Gesamt';C20.1;.2:L0;C30.1;.2:L0;..3
      LINE=HOR:L7,VER:L7
```

Diese Statements könnten folgende Tabellen erzeugen:

		Geschlecht		Alter		
	Gesamt	Männer	Frauen	...30	31...49	50...
Basis	309	152	157	96	89	124
C10.1	100	47	53	34	34	32
C10.2	104	51	53	36	24	44
C10.3	105	54	51	26	31	48
C11.1	111	50	61	40	25	46
C11.2	108	58	50	30	35	43
C11.3	90	44	46	26	29	35

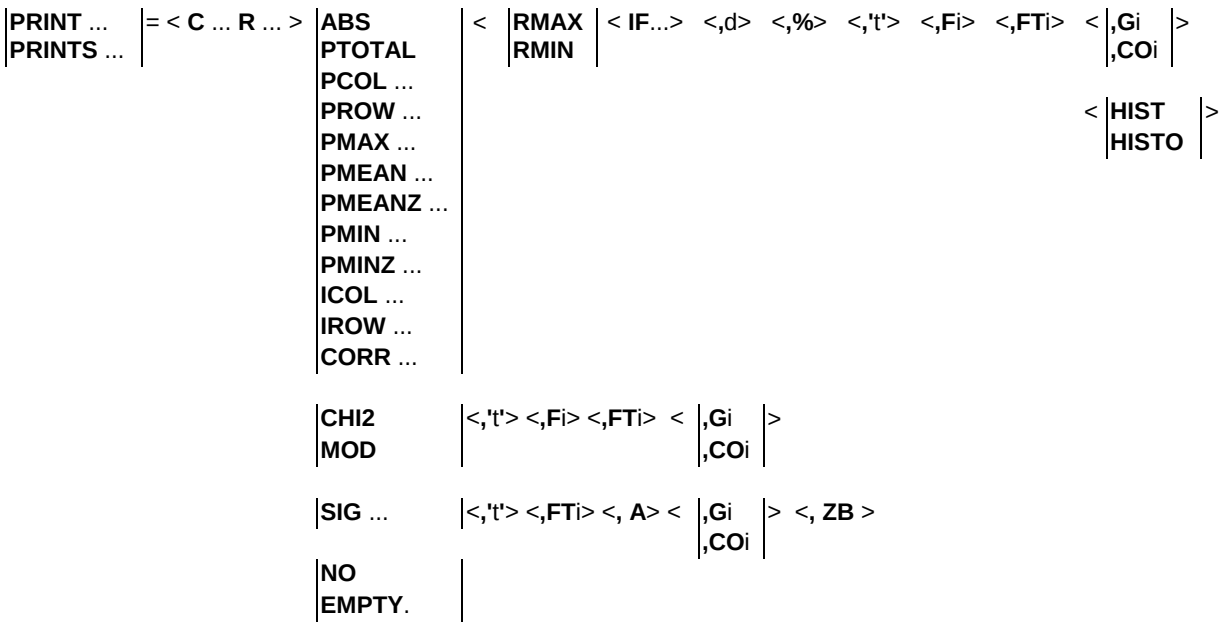
		Geschlecht		Alter		
	Gesamt	Männer	Frauen	...30	31...49	50...
Basis	309	152	157	96	89	124
C10.1	100	47	53	34	34	32
C10.2	104	51	53	36	24	44
C10.3	105	54	51	26	31	48
C11.1	111	50	61	40	25	46
C11.2	108	58	50	30	35	43
C11.3	90	44	46	26	29	35

Zum Vergleich wird hier dieselbe Tabelle mit unterschiedlich breiten Linien gedruckt.

Das erste XTAB enthält keine LINE-Angabe. Die Tabelle wierd daher mit der Linie L3 erstellt. Im zweiten XTAB-Statement erzeugt die Angabe *LINE = HOR:L7* die stärkeren waagerechten Linien *L7* über und unter den Kopftexten sowie am Ende der Tabelle. *VER:L7* erzeugt alle senkrechten Linien und *COL:L3* die waagerechten Linien innerhalb des Kopfbereichs.

In ROW sorgen die Angaben *;:L3* hinter *Basis* und zwischen *C10* und *C11* für eine Leerzeile mit einer schmalen waagerechten Linie. In COL unterbindet *:L0* die Ausgabe einer Linie vor den entsprechenden Spalten.

XTAB: PRINT



PRINT... Diese Angaben steuern die Aufbereitung der Zählergebnisse in den Tabellen.
Für jeden Kreuzungspunkt einer Zeile aus ROW mit einer Spalte aus COL erzeugt XTAB einen Zählerwert. Dieser Wert kann innerhalb seiner Tabellenzelle gleichzeitig auf bis zu vier verschiedene Arten aufbereitet werden. Die Angabe PRINT ... PRINT3 wählt die Position in der Tabellenzelle, in der der Wert auszugeben ist:

PRINT	PRINT1
PRINT2	PRINT3

Die Angaben hinter PRINT ... PRINT3 legen die Art Aufbereitung fest. So zeigen die beiden Angaben

```
PRINT=ABS    PRINT1=PROW1
```

den Zählerwert jeder Tabellenzelle auf zwei verschiedene Weisen in einer Zeile: Links als Absolutwert ohne Nachkommastellen und rechts als Prozentwert, mit den Werten der ersten Tabellenzeile als Basis.

Besitzt ein XTAB-Statement keine der Angaben PRINT ... PRINT3, so wird mit der Voreinstellung:

```
PRINT=ABS
```

gearbeitet und für PRINT1 ... PRINT3 keine Werte ausgegeben.

PRINTS Die Angaben PRINT ... PRINT3 gelten nur für das laufende XTAB-Statement.
PRINTS ... PRINTS3 haben die gleiche Wirkung wie PRINT ... PRINT3, gelten jedoch für das laufende und alle folgenden XTAB-Statements, bis sie durch neue PRINTS-Angaben ersetzt werden.

Zeilen- und Spaltenangaben

Die PRINT-Angaben verlangen an verschiedenen Stellen Zeilen- oder Spaltennummern.
Dabei erzeugt jeder durch ein Semikolon abgetrennte Zähloperand aus ROW eine Zeile und jeder Operand aus COL eine Spalte. So entstehen aus

```
XTAB ROW=TOTAL;C10.1
```

zwei Zeilen, egal wie viele Text- oder Zählerzeilen dazu ausgegeben werden.

Leere Zeilen oder Spalten sind mitzuzählen. Zum Beispiel erzeugt

```
XTAB ROW=TOTAL;;C10.1
```

drei Zeilen.

Die Zeilen und Spalten können als einzelne Nummern 1 ... 32 767 wie

```
12
```

oder Von-Bis-Werte wie

```
17..25  oder  17...25
```

eingegeben werden. Dies sind stets die Nummern vor eventueller Sortierung durch die Angabe SORT und vor der Unterdrückung von Zeilen oder Spalten durch SUPPRESS.

Die Nummern lassen sich auch direkt hintereinander stellen. So erzeugt

```
PRINT=C(1 5..7 10..12)ABS C(13..15)PROW1
```

Absolutwerte in den Spalten 1, 3, 5 ... 7 und 10 ... 12. Die Spalten 13 ... 15 erhalten Prozentwerte mit Zeile 1 als Basis.

Auch negative Zeilen- und Spaltenangaben sind möglich. -1 ist die letzte Zeile oder Spalte einer Tabelle, -2 die vorletzte usw. So sorgt

```
PRINT1=C(2..7)ABS C(10..-1)PROW1
```

für Absolutwerte in den Spalten 2 ... 7. Ab Spalte 10 bis hin zur letzten Spalte werden Prozentwerte mit Zeile 1 als Basis gedruckt. Die übrigen Spalten bleiben in der Position PRINT1 leer.

Sind in ROW oder COL Zeilen oder Spalten mit Klassenbuchstaben A ... Z{ } gekennzeichnet (siehe Abschnitt *XTAB, Klassenbildung von Zähloperanden*), so lässt sich anstelle der Zeilen- oder Spaltennummern auch der Klassenbuchstabe A ... Z verwenden:

```
XTAB ROW=TOTAL; A{C1.1;...5};B{TOTAL;C2}
     PRINT=R(1)ABS  R(A)PROW1  R(B)PROW1,1
```

Die Angabe PRINT sorgt für Absolutwerte in der Zeile 1. Die Zeilen der Klasse A werden auf Zeile 1 prozentuiert und ohne Nachkommastellen ausgegeben. Die Zeilen der Klasse B werden ebenfalls auf Zeile 1 prozentuiert aber mit einer Nachkommastelle aufbereitet.

Klassenangaben und Von-Bis-Werte können auch kombiniert werden. Zum Beispiel

```
PRINT=C(1..7 A B -1)ABS
```

C... **R**... Mit diesen Angaben werden die dahinter stehenden Aufbereitungsregeln auf bestimmte Zeilen oder Spalten beschränkt. Für die Angabe von Zeilen- und Spaltennummern siehe oben bei *Zeilen- und Spaltenangaben*. Ohne solche Angaben C... und R... gelten die Regeln für die ganze Tabelle.

C(s...**)**
Diese Angabe beschränkt die dahinter stehenden Druckregeln auf die Tabellenspalten s ... .

```
PRINT=C(2..7)ABS C(10..12 A)PROW1
```

erzeugt Absolutwerte in den Spalten 2 ... 7 und Prozentwerte mit Zeile 1 als Basis in den Spalten 10 ... 12 sowie den Spalten mit dem Klassenbuchstaben A. In den übrigen Spalten bleibt die Position PRINT leer.

R(z...**)**
Diese Angabe beschränkt die folgenden Druckregeln auf die Tabellenzeilen z ... .

```
PRINT=R(1)ABS R(2..-1)PROW1
```

erzeugt Absolutwerte in der ersten Zeile und Prozentwerte in den übrigen Zeilen mit Zeile 1 als Basis.

C(s...**) R(**z...**)**
Die C- und R-Angaben lassen sich kombinieren, um die Aufbereitungsregeln auf die Spalten s ... und Zeilen z ... zu beschränken. So erzeugt die Angabe

```
PRINT=C(2..7)PROW1
      C(10..12) R(1)ABS
                R(2..5)PCOL1
```

in den Spalten 2 bis 7 Prozentwerte mit Zeile 1 als Basis.
In den Spalten 10 bis 12 sorgt R(1) ABS für Absolutwerte in der ersten Zeile und R(2..5) PCOL1 druckt in den Spalten 10 bis 12 die Werte der Zeilen 2 bis 5 als Prozentwerte mit der ersten Spalte als Basis.
R(1) ABS und R(2..5) PCOL1 haben keine Auswirkung auf die Spalten 2 bis 7.

ABS Die Zählergebnisse werden als Absolutwerte, nicht prozentuiert ausgegeben. Zum Beispiel liefert

```
PRINT=R(1..2)ABS
```

Absolutwerte in den beiden ersten Zeilen der Tabelle.

PCOL ... Die Zählergebnisse werden auf die Werte einer Spalte prozentuiert.
Für die Angabe der Spaltennummer siehe oben bei *Zeilen- und Spaltenangaben*.
Folgende Angaben sind möglich:

PCOL j
Die Zähler werden innerhalb jeder Zeile auf den Wert der Spalte *j* prozentuiert.

```
PRINT=PCOL1
```

prozentuiert die Werte jeder Zeile auf den Wert der ersten Spalte in der Zeile.

Mit *j* ist die Spaltennummer vor eventueller Sortierung durch SORT und Spaltenunterdrückung durch SUPPRESS gemeint. Damit ist es möglich, auf Spalten zu prozentuieren, die in der Tabelle gar nicht erscheinen. Es lassen sich auch die Spaltennummern nach SUPPRESS und SORT angeben. Dazu ist der Buchstabe S vor die Spaltennummer zu stellen. Zum Beispiel PCOLS1 oder PCOLS-1.

Schließlich kann die Prozentuierungsbasis auch über Klassenbuchstaben A{ } ... Z{ } festgelegt werden. Mit dem Statement

```
XTAB  COL=A{C1.Z123};B{C1};A{C2.Z123};B{C2}
      PRINT=C(B)PCOLA
```

werden die Werte aus den Spalten B{C1} auf die erste davor liegende Spalte A{C1.Z123} prozentuiert und die Werte der Spalten B{C2} auf die erste davor liegende Spalte A{C2.Z123}.
Durch PROWA wird ab der aktuellen Zeile nach links in der Tabelle gesucht und die erste Spalte mit der Klasse A{ } als Prozentuierungsbasis verwendet. Mit der Angabe PCOL-A wird ab der aktuellen Spalte nach rechts gesucht und die erste Spalte mit der Klasse A{ } als Basis benutzt.

PCOL j **ROW** m
Alle Zähler werden auf den Wert in Spalte *j* und Zeile *m* prozentuiert.

```
PRINT=PCOL-1ROW-1
```

prozentuiert alle Zähler der Tabelle auf den Wert der letzten Spalte und letzten Zeile.

PMAX ... Die Zählergebnisse werden auf den größten Absolutwert prozentuiert.
Für die Angabe von Zeilen- und Spaltennummern siehe oben bei *Zeilen- und Spaltenangaben*.
Folgende Angaben sind möglich:

PMAX
Prozentuierungsbasis ist der größte Absolutwert der Tabelle. Zum Beispiel:
```
PRINT=PMAX
```

PMAX(C(s...**))**
Die Zähler werden innerhalb jeder Zeile auf den größten Absolutwert der Spalten s ... prozentuiert.
```
PRINT=PMAX(C(1..-1))
```
prozentuiert die Zähler jeder Zeile auf den größten Wert aus allen Spalten dieser Zeile.

PMAX(R(z...**))**
Die Zähler werden innerhalb jeder Spalte auf den größten Absolutwert der Zeilen z ... prozentuiert.
```
PRINT=PMAX(R(1..-1))
```
prozentuiert die Zähler jeder Spalte auf den größten Wert aus allen Zeilen dieser Spalte.

PMAX(C(s...**) R(**z...**))**
Die Zähler werden auf den größten Absolutwert der Spalten s ... und Zeilen z ... prozentuiert.
```
PRINT=R(2..-1)C(2..-1)PMAX(R(2..-1)C(2..-1))
```
liefert Prozentwerte für alle Zähler der Tabelle ab Zeile 2 und Spalte 2. Basis der Prozentuierung ist der größte Absolutwert, wieder ab Zeile 2 und Spalte 2 der Tabelle.

PMEAN ... Die Zählergebnisse werden auf den Mittelwert von Absolutwerten prozentuiert.
PMEANZ ... PMEAN schließt die Werte 0 bei der Bildung des Mittelwertes aus, PMEANZ schließt sie ein.
Für die Angabe von Zeilen- und Spaltennummern siehe oben bei *Zeilen- und Spaltenangaben*.
Folgende Angaben sind möglich:

PMEAN oder **PMEANZ**
Prozentuierungsbasis ist der Mittelwert aus allen Werten der Tabelle. Zum Beispiel:
```
PRINT=PMEAN
```

PMEAN(C(s...**))** oder **PMEANZ(C(**s...**))**
Der Mittelwert wird innerhalb jeder Zeile aus den Zählern der Spalten s ... gebildet.
```
PRINT=PMEAN(C(1..-1))
```
prozentuiert die Zähler jeder Zeile auf den Mittelwert aus allen Spalten dieser Zeile.

PMEAN(R(z...**))** oder **PMEANZ(R(**z...**))**
Der Mittelwert wird innerhalb jeder Spalte aus den Zählern der Zeilen z ... gebildet.
```
PRINT=PMEAN(R(1..-1))
```
prozentuiert die Zähler jeder Spalte auf den Mittelwert aus allen Zeilen dieser Spalte.

PMEAN(R(z...**) C(**s...**))** oder **PMEAN(R(**z...**) C(**s...**))**
Der Mittelwert wird aus allen Werten der Zeilen z ... und Spalten s ... gebildet.
```
PRINT=R(2..-1)C(2..-1)PMEAN(R(2..-1)C(2..-1))
```
liefert Prozentwerte für alle Zähler der Tabelle ab Zeile 2 und Spalte 2. Basis der Prozentuierung ist der Mittelwert aus allen Tabellenwerten ab Zeile 2 und Spalte 2.

PMIN ...
PMINZ ...

Die Zählergebnisse werden auf den kleinsten Absolutwert prozentuiert.
PMIN schließt die Werte 0 bei der Bildung des kleinsten Wertes aus, PMINZ schließt sie ein.
Für die Angabe von Zeilen- und Spaltennummern siehe oben bei *Zeilen- und Spaltenangaben*.
Folgende Angaben sind möglich:

PMIN
Prozentuierungsbasis ist der kleinste Absolutwert der Tabelle.

```
PRINT=PMIN
```

prozentuiert auf den kleinsten von 0 verschiedenen Wert der ganzen Tabelle.

PMIN(R(z...**))**
Die Zähler werden innerhalb jeder Spalte auf den kleinsten Absolutwert der Zeilen z ... prozentuiert.

```
PRINT=PMIN(R(1..-1))
```

prozentuiert die Zähler jeder Spalte auf den kleinsten Wert aus allen Zeilen dieser Spalte.

PMIN(C(s...**))**
Die Zähler werden innerhalb jeder Zeile auf den kleinsten Absolutwert der Spalten s ... prozentuiert.

```
PRINT=PMIN(C(1..-1))
```

prozentuiert die Zähler jeder Zeile auf den kleinsten Wert aus allen Spalten dieser Zeile.

PMIN(R(z...**) C(**s...**))**
Die Zähler werden auf den kleinsten Absolutwert der Zeilen z ... und Spalten s ... prozentuiert.

```
PRINT=PMIN(R(2..-1)C(2..-1))
```

liefert Prozentwerte für alle Zähler der Tabelle. Basis der Prozentuierung ist der kleinste Absolutwert ab Zeile 2 und Spalte 2 der Tabelle.

PROW ...

Die Zählergebnisse werden auf die Werte einer Zeile prozentuiert. Für die Angabe der Zeilennummer siehe oben bei *Zeilen- und Spaltenangaben*. Folgende Angaben sind möglich:

PROW m
Die Zähler werden innerhalb jeder Spalte auf den Wert der Zeile *m* prozentuiert.

```
PRINT=PROW1
```

prozentuiert die Werte jeder Spalte auf den Wert der ersten Zeile in dieser Spalte.

Mit *m* ist die Zeilennummer vor eventueller Sortierung durch SORT oder Zeilenunterdrückung durch SUPPRESS gemeint. Damit ist es möglich, auf Zeilen zu prozentuieren, die in der Tabelle gar nicht erscheinen. Es lassen sich auch die Zeilennummern nach SUPPRESS und SORT angeben. Dazu ist der Buchstabe S vor die Zeilennummer zu stellen. Zum Beispiel PROWS1 oder PROWS-1.

Schließlich kann die Prozentuierungsbasis auch über Klassenbuchstaben A{ } ... Z{ } festgelegt werden. Mit dem Statement

```
XTAB  ROW=A{C1.Z123};B{C1};A{C2.Z123};B{C2}
      PRINT=PROWA
```

werden alle Werte aus den Zeilen B{C1} auf die erste davor liegende Zeile A{C1.Z123} prozentuiert und die Werte der Zeilen B{C2} auf die erste davor liegende Zeile A{C2.Z123}. Durch PROWA wird ab der aktuellen Zeile nach oben in der Tabelle gesucht und die erste Zeile mit der Klasse A{ } als Prozentuierungsbasis verwendet. Mit der Angabe PROW-A wird ab der aktuellen Zeile nach unten gesucht und die erste Zeile mit der Klasse A{ } als Basis benutzt.

PROW m **COL** j
Alle Zähler werden auf den Wert in Zeile *m* und Spalte *j* prozentuiert.

```
PRINT=PROW-1COL-1
```

prozentuiert alle Zähler der Tabelle auf den Wert der letzten Zeile und letzten Spalte.

PTOTAL Die Zählergebnisse werden auf den Filterwert der Tabelle prozentuiert. Zum Beispiel

```
PRINT=PTOTAL
```

Ist im XTAB-Statement kein FILTER angegeben, so wird auf die Anzahl der verarbeiteten Fälle prozentuiert.

ICOL... Die Zählergebnisse werden als Indexwerte von Prozentwerten ausgegeben.
IROW... Für die Angabe der Zeilen- und Spaltennummer siehe oben bei *Zeilen- und Spaltenangaben*.
Es gelten alle schon bei PROW und PCOL beschriebenen Möglichkeiten.
Folgende Angaben sind zulässig:

ICOL j **ROW** m
Prozentuierungsbasis ist die Spalte *j*, Indexbasis die Zeile *m*.
Im folgenden Beispiel soll der Indexwert *i* aus dem Zählerwert *a* für

```
PRINT=ICOL1ROW1
```

ermittelt werden:

	Spalte 1	laufende Spalte
Zeile 1	s	r q = r / s * 100
laufende Zeile	b	a p = a / b * 100 i = p / q * 100

Zunächst wird zu dem laufenden Zählerwert *a* der Prozentwert *p* zur Basis *b* errechnet, genau wie bei PCOL1.
Danach wird der Zählerwert *r* aus der laufenden Spalte und der Zeile 1 entnommen. Aus ihm wird mit dem Basiswert *s* aus Zeile 1 und Spalte 1 der Prozentwert *q* ermittelt, wieder wie bei PCOL1.
Schließlich wird aus *p* und *q* der Indexwert $i = p / q * 100$ ermittelt.

IROW m **COL** j
Diese Angabe liefert das gleiche Ergebnis wie ICOL j ROW m.

ICOL j **MAX**... **ICOL** j **MIN**... **ICOL** j **MEAN**...
Zunächst werden die Zählergebnisse auf die Werte der Spalte *j* prozentuiert.
Danach wird der Wert der Funktion MAX ... MEAN ... oder MIN ... aus entsprechenden Prozentwerten ermittelt, so wie oben bei *PMAX* ... *PMIN* ... *PMEAN* ... beschrieben.

Schließlich wird der Prozentwert der laufenden Tabellenzelle noch einmal auf diesen Funktionswert prozentuiert. Zum Beispiel erzeugt

```
PRINT=ICOL1MAX(R1..-1))
```

Indexwerte aus den Prozentwerten mit Spalte 1 als Basis. Der größte Prozentwert jeder Spalte dient als Indexbasis und erhält den Indexwert 100.

IROW m **MAX**... **IROW** m **MIN**... **IROW** m **MEAN**...
Zunächst werden die Zählergebnisse auf die Werte der Zeile *m* prozentuiert.
Danach wird der Wert der Funktion MAX ... MEAN ... oder MIN ... aus entsprechenden Prozentwerten ermittelt, so wie oben bei *PMAX* ... *PMIN* ... *PMEAN* ... beschrieben.

Schließlich wird der Prozentwert der laufenden Tabellenzelle noch einmal auf diesen Funktionswert prozentuiert. Zum Beispiel erzeugt

```
PRINT=IROW1MIN(R1..-1))
```

Indexwerte aus den Prozentwerten mit Zeile 1 als Basis. Der kleinste Prozentwert jeder Spalte dient als Indexbasis und erhält den Indexwert 100.

RMAX | **RMIN** < **C** | **R** > < **Z** >

Durch diese Angaben werden Rangzahlen 1, 2, ... nach der Größe der Zählergebnisse ausgegeben. Vor Vergabe der Rangzahlen werden die Zählergebnisse nach der davor stehenden Aufbereitungsregel verrechnet. Zum Beispiel erzeugt

```
PRINT=PROW1 RMAX
```

Rangziffern aus allen Prozentwerten PROW1 der Tabelle.

RMAX

Diese Angabe gibt dem größten Wert die Rangzahl 1. So wird durch

```
PRINT=ABS RMAX
```

für den größten Absolutwert der Tabelle die Rangzahl 1 ausgegeben.

RMIN

Bei dieser Angabe erhält der kleinste Wert die Rangzahl 1. So wird mit

```
PRINT=PROW1 RMIN
```

für den kleinsten Prozentwert der Tabelle mit Zeile 1 als Basis die Rangzahl 1 ausgegeben.

C

Diese Angabe vergibt die Rangzahlen getrennt für jede Spalte. Zum Beispiel:

```
PRINT=PROW1 RMAXC
```

Sind weder C noch R angegeben, werden die Rangzahlen über alle Zeilen und Spalten hinweg vergeben. Zum Beispiel:

```
PRINT=ABS RMAX
```

Hier werden für alle Tabellenzellen Rangzahlen ausgegeben. Jede Rangzahl erscheint dabei nur einmal in der Tabelle.

R

Diese Angabe vergibt die Rangzahlen getrennt für jede Zeile. Zum Beispiel:

```
PRINT=PROW1 RMAXR
```

Sind weder C noch R angegeben, werden die Rangzahlen über alle Zeilen und Spalten hinweg vergeben. Zum Beispiel:

```
PRINT=ABS RMAX
```

Hier werden für alle Tabellenzellen Rangzahlen ausgegeben. Jede Rangzahl erscheint dabei nur einmal in der Tabelle.

Z

Mit dieser Angabe nehmen auch die Zählergebnisse 0 an der Rangbildung teil.
Ohne sie werden für die Werte 0 keine Rangzahlen ausgegeben. Zum Beispiel:

```
PRINT=ABS RMAXCZ
```

MOD <,'t'> <,**F**u> <,**FT**i> < | ,**G**i | ,**CO**i | >

Diese Angabe dient dazu, mehrdeutige Modalwerte zu kennzeichnen. Sie wirkt nur auf Zähler, die mit den Modalwert-Funktionen MODL oder MODU erzeugt wurden.

Zum Beispiel

```
XTAB  ROW=TOTAL;MODL(C10)
      PRINT=ABS
      PRINT1=MOD
```

Dieses Statement stellt zunächst mit PRINT=ABS den Modalwert von C10 in die Tabelle. Besitzt die Variable C10 mehrere Modalwerte, so gibt PRINT1=MOD das Zeichen x in die Tabelle.
Unterhalb der Tabelle erscheint dazu die Fußnote:

x) Mehrere Modalwerte

Besitzt C10 nur einen Modalwert, so wird weder das Zeichen x noch die Fußnote ausgegeben.

Die Angabe 't' hinter MOD ersetzt das Zeichen x durch ein oder mehrere andere Zeichen.

Fu hinter MOD ersetzt den Fußnotentext *x) Mehrere Modalwerte* durch das Textelement (u).

Mit FTi lässt sich die Schriftart wählen, mit der die Markierungszeichen auszugeben sind und durch Gi lassen sich diese mit einer Tonfläche unterlegen.

CHI2 <,'t'> <,**F**u> <,**FT**i> < | ,**G**i | ,**CO**i | >

Diese Angabe bewirkt einen Chi^2-Test, der die Zeilen- und Spaltenbedingungen einer Tabelle oder Teiltabelle miteinander vergleicht.
Die beteiligten Zähler werden in der Tabelle mit frei wählbaren Zeichen markiert.
Der resultierende Chi^2-Wert und dazugehörige Signifikanzwert werden in einer Fußnote unterhalb der Tabelle ausgegeben.

Der Signifikanzwert gibt den kleinsten Fehler an, bei dem die Hypothese *die Zeilen- und Spaltenbedingungen sind statistisch unabhängig* abzulehnen ist: Kleine Werte weisen auf signifikant abhängige Bedingungen hin, große lassen auf unabhängige Bedingungen schließen.

An dem Test nehmen die Zeilen und Spalten teil, die vor der Angabe CHI2 mit R(...) und C(...) ausgewählt sind. Fehlen solche Zeilen- und Spalten-Angaben, so nehmen alle Zeilen oder Spalten teil

Die am Test beteiligten Tabellenzellen werden mit den Zeichen aus der Druckmaske 't' gekennzeichnet. Fehlt eine solche Druckmaske, so wird dafür das Zeichen x verwendet.

Der ermittelte Chi-Quadrat-Wert und der dazugehörige Signifikanzwert werden hinter der Fußnote Fu in der Form

Chi-Quadrat = x.xx Signifikanz = y.yy%

ausgegeben. Dieser Fußnotentext stammt aus dem Textelement (u). Ist keine Fußnote Fu angegeben, so erscheint dafür der Text:

x) Chi-Quadrat-Test dieser Tabellenwerte: Chi-Quadrat = x.xx Signifikanz = y.yy%

Der Abschnitt *CHI2: Zweidimensionaler Chi-Quadrat-Test* enthält weitere Angaben zu diesem Thema.

SIG < **(**g...**)** > **,**s1 < **,**s2> < **,**s3> < **,E**s1 > < **,E**s2> < **,E**s3>

< | **,H** | > < **,A** > < **,M** > <,'t'> <**,FT**i> < **,ZB** > < | **,G**i | >
| **,L** | | **,CO**i |
| **,F** |
| **,S** |

Diese Angabe sorgt für die Markierung von Zählergebnissen, die sich im Sinne eines t-Tests signifikant voneinander unterscheiden.
Dazu sind zunächst in ROW oder COL durch die Funktionen MTT... das Verfahren und die miteinander zu vergleichenden Werte festzulegen.
Genaueres enthält der Abschnitt *t-Test zur Markierung signifikanter Unterschiede* in XTAB.

Die Angabe

```
XTAB ROW=TOTAL;MTT{MEAN(C17)}
```

stellt zum Beispiel die Mittelwerte der Variablen C17 in die letzte Zeile der Tabelle. Die Angabe MTT vergleicht die Werte dieser Zeile und untersucht sie durch t-Tests auf signifikante Unterschiede. Die zusätzliche Angabe

```
PRINT=SIG,2
```

legt das Signifikanz-Niveau für diese Tests auf 2% fest.

Sind nicht alle Werte einer Zeile oder Spalte miteinander zu vergleichen, so lässt sich der Wirkungsbereich von SIG durch davor stehende Angaben R... oder C... einschränken. Zum Beispiel bewirkt

```
PRINT=C(1..4)SIG,1 C(5..6)SIG,2
```

einen Signifikanz-Vergleich auf dem 1%-Niveau für die Spalten 1 bis 4 und einen Vergleich auf dem 2%-Niveau für die Spalten 5 bis 6.

s1 s2 s3 Signifikanz-Niveaus für den t-Test als Prozentwerte 0.01 ... 50.00 im Sinne des zweiseitigen Tests. Das Programm markiert Wertepaare in der Tabelle, die beim Vergleich dieses Niveau unterschreiten, mit den Buchstaben a ... z.
Je kleiner diese Schwelle gewählt wird desto sicherer unterscheiden sich die markierten Werte.
Sind mehrere Signifikanz-Niveaus angegeben, so wird die schwächste Bedingung (mit dem größten Wert) durch Kleinbuchstaben gekennzeichnet, die zweite durch Großbuchstaben und die dritte durch Großbuchstaben mit Ausrufungszeichen dahinter. Die Angabe TMARK in XTAB ersetzt die Buchstaben a ... z durch andere Zeichen.

Es1 **E**s2 **E**s3
Schwellenwerte 0.001 ... 99.999 der Effektstärken nach J. Cohen. Zum Beispiel

```
PRINT=SIG,5,E0.2,E0.5,E0.8
```

Diese Angaben sind nur bei t-Tests mit MTT und MTTU möglich. Sie setzen ein einziges Signifikanzniveau *s1* voraus. Danach werden nur solche Wertepaare mit den Buchstaben a ... z markiert, die sich im Sinne des t-Tests signifikant unterscheiden und zusätzlich die vorgegebene Effektstärken *Esi* überschreiten.
Sind mehrere Schwellenwerte angegeben, so wird die schwächste Bedingung (mit dem kleinsten Wert) durch Kleinbuchstaben gekennzeichnet, die nächste durch Großbuchstaben und die dritte durch Großbuchstaben mit Ausrufungszeichen dahinter.
Die Angabe TMARK in XTAB ersetzt die Buchstaben a ... z durch andere Zeichen.

A Diese Angabe sortiert die Markierungszeichen a ... z für signifikante Unterschiede alphabetisch. Enthält das Statement XTAB die Angabe TMARK, so wird nach der Reihenfolge der Zeichen hinter TMARK sortiert. Ohne die Angabe A werden die Markierungszeichen a ... z nach Signifikanzwerten sortiert.

M Die Angabe M stellt bei signifikanten Unterschieden ein Minuszeichen - vor das Markierungszeichen a ... z des kleineren Wertes.

H L F S Ohne diese Angaben werden bei signifikanten Unterschieden beide Werte mit einem Buchstaben markiert.

H markiert nur den größeren der beiden signifikant unterschiedlichen Werte.

L markiert nur den kleineren der beiden signifikant unterschiedlichen Werte.

F markiert nur den ersten der beiden signifikant unterschiedlichen Werte. Zum Beispiel markiert

```
PRINT=C(2..8)SIG(1),5,F
```

nur die Werte der Spalten 2 ... 8.

S markiert nur den zweiten der beiden signifikant unterschiedlichen Werte. Zum Beispiel markiert

```
PRINT=C(2..8)SIG(1),5,S
```

nur die Werte der Spalte 1.

ZB Liefert der t-Test keine signifikanten Unterschiede, so wird ein waagerechter Strich – ausgegeben. Die Angabe ZB gibt stattdessen das Leerzeichen aus. Zum Beispiel

```
PRINT=C(2..5)SIG(1),5,ZB
```

(g ...**)** Die Angabe von C ... oder R ... vor SIG begrenzt den Vergleich auf ausgewählte Zeilen oder Spalten. Durch weitere Zeilen oder Spalten (g ..) direkt hinter SIG werden die Zeilen oder Spalten vor SIG nicht untereinander, sondern nur mit denen hinter SIG verglichen.

Für (g ...) gelten die bei *Zeilen- und Spaltenangaben* beschriebenen Regeln. So vergleicht zum Beispiel

```
PRINT=C(1..5)SIG(6..10),2
```

die Werte der Spalten 1 bis 5 mit denen der Spalten 6 ... 10 auf dem 2%-Niveau.

Sind in ROW oder COL Zeilen oder Spalten mit Klassenangaben A ... Z{ ..} gekennzeichnet, so lassen sich anstelle der Zeilen- oder Spaltennummern auch die Klassenbuchstaben A ... Z verwenden:

```
COL=TOTAL;A{C7};B{C10}
PRINT=C(A)SIG(B),1
```

In COL sind die Merkmale der Variablen C7 mit A markiert und die der Variablen

C10 mit B. D
ie Angabe PRINT vergleicht nun die Merkmale der Klasse A mit denen der Klasse B durch t-Tests.

Gi Grauton G1 bis G16 oder Farbe CO1 bis CO16 aus dem Statement PAGEP, mit dem die.
COi Zählerwerte unterlegt werden sollen. Das geschieht aber nur, wenn zu der Tabellenzelle signifikante Unterschiede gefunden wurden. Hinter dem Statement PAGE ist diese Angabe wirkungslos.
Ohne zusätzliche Angaben unterlegen Gi und COi die nur die Ausgabebereiche des jeweiligen PRINT ... PRINT3. G5(L1) unterlegt dagegen die ganze erste Zeile der Tabellenzelle, also den Bereich von PRINT und PRINT1, G5(L2) die zweite Zeile, also den Bereich von PRINT2 und PRINT3 und G5(L12) den ganzen Ausgabebereich von PRINT, PRINT1, PRINT2 und PRINT3.

CORR Diese Angabe erstellt Korrelations-Koeffizienten aus den Werten der Zeilen- und Spaltenvariablen. Dazu sind in ROW und COL N-Variable, C-Variable mit Merkmalen und Mittelwerte MEAN von C- und N-Variablen möglich.

Für Zeilen oder Spalten mit Staffelungen wird der Korrelationskoeffizient zwischen den Variablen der untersten Staffelstufe gebildet. Zum Beispiel

```
XTAB  ROW={C5.1}{N1}  COL=C10.1
```

Hier werden für CORR die Werte der Variablen N1 und C10.1 miteinander verglichen. C5.1 ist lediglich eine Filterbedingung. Dagegen werden in

```
XTAB  ROW={N1}{C5.1}  COL=C10.1
```

die Werte der Variablen C5.1 mit C10.1 verglichen und N1 dient als Gewicht bei der Ermittlung der Fallzahl.

Diese Koeffizienten k (Pearson-Korrelation oder Produkt-Moment-Korrelation) werden auf folgende Weise ermittelt:

$$k = \frac{n\sum x_1 x_2 - \sum x_1 \sum x_2}{\sqrt{n\sum {x_1}^2 - \left(\sum x_1\right)^2} \cdot \sqrt{n\sum {x_2}^2 - \left(\sum x_2\right)^2}}$$

x_i = Werte der beteiligten Variablen.

n = Anzahl beteiligter Fälle.

NO In der entsprechenden Tabellenposition wird kein Zählergebnis ausgegeben.
Diese Angabe dient vor allem dazu, PRINTS-Angaben aus vorausgehenden XTAB-Statements rückgängig zu machen. Zum Beispiel:

```
PRINT=NO
```

EMPTY Wie bei PRINT=NO wird in der entsprechenden Tabellenposition kein Zählergebnis ausgegeben. Abweichend von PRINT=NO wird jedoch in den CSV- und XLSX-Dateien eine leere Spalte an der entsprechenden PRINT-Position ausgegeben. Zum Beispiel:

```
PRINT=EMPTY
```

IF | **IFA** < v1 > g v2 < ,d > < ,**%** > < ,'t' > < ,**F**i > < ,**FT**i > < **,G**i | **,CO**i >

Diese Angaben machen die Aufbereitung der Zählergebnisse vom Vergleich verschiedener Tabellenwerte abhängig. Dabei ist *v1* der erste Vergleichswert, *g* eine Vergleichsbedingung und *v2* der zweite Vergleichswert. Die dahinter stehenden Aufbereitungsregeln

,d ,% ,'t' ,FTi ,Gi ,Fi

werden nur wirksam, wenn die durch *v1, g* und *v2* vorgegebene Bedingung erfüllt ist.

In der Angabe

```
PRINT=ABS,1 IF =MAX,FT3
```

sorgt ABS,1 dafür, dass die Zählergebnisse als Absolutwerte mit einer Nachkommastelle erscheinen. Vor =MAX ... fehlt der erste Vergleichswert. Daher werden alle Tabellenwerte mit dem maximalen Wert der Tabelle verglichen und bei Übereinstimmung in der Schrift FT3 ausgegeben.

Es lassen sich beliebig viele IF-Bedingungen direkt hintereinander angeben.
Jede Druckaufbereitungsregel bleibt solange wirksam, bis sie durch eine dahinter stehende, erfüllte IF-Bedingung ersetzt wird. Durch

```
PRINT=ABS,2 IF>50,FT5,1
            IF>100,'<&>'
            IF>200,FT9
```

werden Absolutwerte mit zwei Nachkommastellen ausgegeben, Absolutwerte über 50 in der Schrift FT5, jedoch nur mit einer Nachkommastelle. Liegt ein Wert über 100, so wird er ebenfalls in der Schrift FT5 mit einer Nachkommastelle gedruckt, zusätzlich aber noch in spitze Klammern gesetzt. Werte über 200 erscheinen in der Schrift FT9, in spitzen Klammern und mit einer Nachkommastelle.

IF Diese Angabe vergleicht die Tabellenwerte in der aufbereiteten Form: Ist der aktuelle Wert prozentuiert, so werden Prozentwerte verglichen, ist er absolut, so werden Absolutwerte verglichen.

Die Angabe

```
PRINT=PCOL1 IF >50,CO3
```

prozentuiert alle Tabellenwerte wegen PCOL1 auf die Werte der ersten Spalte.
Da zwischen IF und >ROW1 kein Prüfwert *v1* angegeben ist, dient der Prozentwert der laufenden Tabelle zum Vergleich:
Ist dieser Prozentwert größer als 50, so wird er mit der Farbe CO3 unterlegt.

IFA Diese Angabe vergleicht stets die absoluten, nicht verrechneten Tabellenwerte, unabhängig davon, wie der laufende Zählerwert aufbereitet wird.

Die Angabe

```
PRINT=PROW1 IFA ROW1<80,FT3
```

prozentuiert alle Tabellenwerte wegen PROW1 auf die Werte der ersten Zeile.
Wegen IFA werden die Absolutwerte mit dem darüber stehenden Absolutwert der ersten Zeile verglichen und nicht die durch PROW1 erzeugten Prozentwerte:
Ist der Absolutwert der ersten Zeile kleiner als 80, so werden alle Zähler dieser Spalte in der Schrift FT3 ausgegeben.

---

v1 Erster Vergleichswert der Tabelle als eine der folgenden Angaben:
ROWi, COLi, TOTAL, MAX... MIN... oder MEAN...
Ohne diese Angabe *v1* wird der Wert der laufenden Tabellenzelle als Vergleichswert verwendet.

Die Angabe
```
PRINT=PROW1 IFA ROW1<10,' '
```
erzeugt zunächst Prozentwerte mit den Werten der ersten Zeile als Basis..
IFA sorgt für den Vergleich der Absolutwerte. ROW1<10 prüft statt des Wertes der laufenden Zelle den Absolutwert der ersten Zeile in der laufenden Spalte.
Wegen ,' ' werden Leerzeichen anstelle der Prozentwerte ausgegeben, sobald der Wert in Zeile 1 der laufenden Spalte kleiner als 10 ist.

g Hier ist einer der folgenden Vergleichsoperatoren anzugeben:

<	kleiner	>=	größer oder gleich
>	größer	^=	ungleich (kann auch ¬= geschrieben werden)
=	gleich	^<	nicht kleiner (kann auch ¬< geschrieben werden)
<=	kleiner oder gleich	^>	nicht größer (kann auch ¬> geschrieben werden)
IN	mit Vergleichsintervall		

v2 Zweiter Vergleichswert der Tabelle.

Hinter dem Vergleichsoperator IN ist hier eine numerische Von-Bis-Angabe der Form (v .. b) erforderlich. So sorgt zum Beispiel
```
PRINT=ABS IF IN(0..40),F1 IF IN(41..80),F2
```
dafür, dass zu Absolutwerten zwischen 0 und 40 wegen F1 das Textelement (1) und zu Werten zwischen 41 und 80 wegen F2 das Textelement (2) als Fußnote ausgegeben wird.

Hinter den Vergleichsoperatoren < > = ... sind folgende Angaben möglich:
Eine konstante Zahl, auch mit Nachkommastellen und Vorzeichen.
Eine der unten beschriebenen Funktionen
ROWi, COLi, TOTAL, MAX... MIN... oder MEAN...

So sorgt die Angabe
```
PRINT=PROW1 IF <1.5,1
```
wegen PROW1 für Prozentwerte ohne Nachkommastellen mit der ersten Zeile als Basis.
Die Angabe IF < 1.5 vergleicht die Prozentwerte mit dem Wert 1.5. Ist diese Bedingung erfüllt, so werden die Werte wegen der Angabe ,1 mit einer Nachkommastelle ausgegeben.

Die Angabe
```
PRINT=ABS IF >=MAX-1.5,G20
```
ermittelt wegen MAX den größten Absolutwert der Tabelle und zieht davon die Konstante 1,5 ab. Alle Werte der Tabelle, die größer als dieser Vergleichswert sind, werden mit dem Grauton G20 unterlegt. Damit lassen sich die größten Werte einer Tabelle hervorheben, die nur geringfügig vom Maximalwert abweichen.

ROW i Der Vergleichswert für einen Zähler wird aus der Zeile *i* der laufenden Spalte entnommen.
Mit *i* ist die Zeilennummer vor eventueller Sortierung durch SORT oder Zeilenunterdrückung durch SUPPRESS gemeint. In der Angabe
```
PRINT=ABS IF =ROW1,CO3
```
vergleicht IF jeden Tabellenwert mit dem Wert der ersten Zeile seiner Spalte. Bei Gleichheit wird der Tabellenwert mit der Farbe CO3 unterlegt.

COL*i* Der Vergleichswert für einen Zähler wird aus der Spalte *i* der laufenden Zeile entnommen. Mit *i* ist die Spaltennummer vor eventueller Sortierung durch SORT oder Zeilenunterdrückung durch SUPPRESS gemeint. In der Angabe

```
PRINT=ABS IF =COL1,CO3
```

vergleicht IF jeden Tabellenwert mit dem Wert der ersten Spalte seiner Zeile. Bei Gleichheit wird der Tabellenwert mit der Farbe CO3 unterlegt.

TOTAL Der Filterwert der Tabelle als Vergleichswert verwendet.
Hinter TOTAL ist noch eine numerische Konstante mit Vorzeichen Plus + oder Minus – möglich. Die Angabe

```
PRINT=ABS IF >TOTAL-10,FT3
```

gibt alle Werte, die höchstens um 10 kleiner sind als der Filterwert, in der Schrift FT3 aus.

MIN MAX MEAN MINZ MAXZ MEANZ

Diese Funktionen ermitteln das Maximum, Minimum oder den Mittelwert aus allen Zählerwerten der Tabelle. Bei MIN, MAX und MEAN wird dazu der Wert 0 ausgeschlossen, bei MINZ, MAXZ und MEANZ wird der Wert 0 einbezogen.
Hinter diesen Funktionen ist noch eine numerische Konstante mit Vorzeichen Plus + oder Minus – möglich. Die Angabe

```
PRINT=PROW1 IF <MEAN-10,CO4
            IF >MEAN+10,CO12
```

prozentuiert wegen PROW1 alle Werte auf die erste Zeile. Das erste IF unterlegt diejenigen Werte, die um mehr als 10 Prozentpunkte kleiner sind als der mittlere Prozentwert, mit der Farbe CO4. Das zweite IF unterlegt alle Werte, die um mehr als 10 Prozentpunkte größer sind als der mittlere Prozentwert, mit der Farbe CO12.

MAX(R(z...**)) MIN(R(**z...**)) MEAN(R(**z...**))**

Hier gelten zunächst die gleichen Regeln wie bei MIN, MAX und MEAN.
Die Angabe R(z...) hinter den Funktionen ermittelt jedoch Maximum, Minimum und Mittelwert nicht aus der ganzen Tabelle sondern nur aus den Zeilen (z...) der jeweiligen Spalte. Für diese Zeilen gelten die bei *Zeilen- und Spaltenangaben* beschriebenen Regeln. Zum Beispiel

```
PRINT=ABS IF =MAX(R(1..-1)),CO12
```

MAX(C(s...**)) MIN(C(**s...**)) MEAN(C(**s...**))**

Hier gelten zunächst die gleichen Regeln wie bei MIN, MAX und MEAN.
Die Angabe C(s...) hinter den Funktionen ermittelt jedoch Maximum, Minimum und Mittelwert nicht aus der ganzen Tabelle sondern nur aus den Spalten (s...) der jeweiligen Zeile. Für diese Spalten gelten die bei *Zeilen- und Spaltenangaben* beschriebenen Regeln. Zum Beispiel:

```
PRINT=ABS IF =MAX(C(1..-1)),CO12
```

MAX(R(z...**) C(**s...**)) MIN(R(**z...**) C(**s...**)) MEAN(R(**z...**) C(**s...**))**

Hinter den Funktionen können gleichzeitig Zeilen R und Spalten C angegeben werden. Dann werden die Funktionswerte nur aus den Zeilen (z...) und Spalten (s...) gebildet, unabhängig von der Zeile oder Spalte des auszugebenden Zählers.
Für die Zeilen (z...) und Spalten (s...) gelten die bei *Zeilen- und Spaltenangaben* beschriebenen Regeln. Zum Beispiel:

```
PRINT=PROW1 IF <MEAN(R(2..-1)C(1..-1)),FT3
```

d **,%** **,'t'** **,F**i **,FT**i | **,G**i | **,CO**i | | **,HIST** | **,HISTO**

d Anzahl 0 ... 18 Nachkommastellen, die in den Zählergebnisse zu drucken sind. Die auszugebenden Werte werden dazu passend gerundet. Ohne diese Angabe erscheinen keine Nachkommastellen. Das Dezimalzeichen wird durch DCHAR im Statement PAGE oder PAGEP festgelegt.

't' Dies sind 1 ... 16 in Hochkommas eingeschlossene Textzeichen, die vor oder hinter den Zählergebnissen ausgedruckt werden. Dabei markiert das Zeichen & die Stelle, an der die Zähler erscheinen sollen. Bei der Druckausgabe wird es durch den Zählerwert ersetzt. So erzeugt

```
PRINT=ABS,'(& EUR)' ...
```

zum Beispiel die Druckausgabe (105 EUR)

Die Angabe 't' kann auch mehrere Zeichen & enthalten. Jedes Zeichen & wird dann durch ein einzelnes Zeichen des Zählergebnisses ersetzt, von rechts beginnend.
Besitzt das Zählergebnis mehr Zeichen als & in 't' enthalten sind, so nimmt das äußerste linke & die noch verbleibenden Zeichen auf.. Besitzt das Zählergebnis dagegen weniger Zeichen, so verschwinden für die überzähligen & in der Ausgabe. Mit

```
PRINT=ABS,'(&.&&& EUR)' ...
```

wird das Zählergebnis 123 als (123 EUR) ausgegeben, der Wert 9324 als (9.234 EUR) und der Wert 120456 schließlich als (120.456 EUR).

Ohne das Zeichen & wird nur der Text 't' ohne Zählerwerte ausgegeben. Damit lassen sich feste Werte in die Tabellenzellen stellen, unabhängig von den Zählergebnissen. Zum Beispiel können sehr kleine Zählergebnisse durch Text oder einzelne Zeichen ersetzt werden. Weiteres dazu oben bei IF.

Mit der Funktion UNDO aus dem Abschnitt *Textzeilen* lässt sich der Text zu den Zählerwerten auch aus Variablen oder Textelementen übernehmen. Enthält das Textelement (10) zum Beispiel den Text *keine Angabe*, so wird dieser mit

```
PRINT= ... ABS,UNDO(10)
```

anstelle eines Zählerwertes ausgegeben.

% Durch diese Angabe wird hinter den Zählerwerten das Zeichen % gedruckt. '&%' führt zum gleichen Resultat.

Fi Die Angabe Fi stellt für i = 1 ... 9999 den Text des Textelements (i) als Fußnote unter die Tabelle, sobald ein Tabellenwert nach dieser Druckregel ausgegeben wird. Die Statements

```
DEFS
-(9999) '*) Fallzahl unter 30'
...
XTAB  PRINT=PROW1 IFA <30,'*',F9999
```

bewirken, dass die Prozentwerte PROW1 der Tabelle durch Sterne * ersetzt werden, wenn die dazugehörigen Absolutwerte kleiner als 30 sind. Als Fußnote erscheint dann der Text

```
*) Fallzahl unter 30
```

FTi Diese Angabe legt die Schriftart fest, in der die zugehörigen Zählerwerte zu drucken sind.
Es sind die Angaben FT1 bis FT9 möglich, so wie sie im Statement PAGEP festgelegt wurden.
Bei Druckausgabe auf Zeilendrucker bleibt diese Angabe wirkungslos.
Fehlt eine solche Angabe FTi hinter PRINT ... PRINT3, so werden die Zählerwerte in der durch FONT=NUM:FTi im Statement XTAB festgelegten Schrift gedruckt.
Die hinter PRINT ... PRINT3 ausgewählte Schriftart FTi hat auch Vorrang vor den Schriften FTi hinter ROW oder COL in XTAB.

Gi Grauton G1 bis G16 oder Farbe CO1 bis CO16 aus dem Statement PAGEP, mit dem die.
COi Zählerwerte unterlegt werden sollen. Nach PAGE bleibt diese Angabe wirkungslos.

Die Angaben G0 oder CO0 machen frühere Unterlegungen rückgängig, zum Beispiel aus einer davor liegenden IF-Angabe. So unterlegt zum Beispiel

```
PRINT=ABS, CO8   IF<=100, CO4   IF<=10, CO0
```

alle Zählerwerte zunächst mit der Farbe CO8. Werte kleinergleich 100 werden wegen IF>=100,CO4 durch die Farbe CO4 hervorgehoben und Werte kleinergleich 10 wegen IF<=10,CO0 nicht farbig unterlegt.

Der Wert 0 wird normalerweise als Strich -- ausgegeben und durch eine Angabe der Form IF<10,COi nicht mit einer Farbe unterlegt; es sei denn der Strich wird über die Angabe NCHAR im Statement PAGE oder PAGEP durch ein anderes Zeichen ersetzt oder dem Wert 0 wird in der Form IF=0,COi einzeln eine Farbe zugewiesen.

COi(Ln) Gi(Ln)
Ohne zusätzliche Angaben unterlegen Gi und COi die nur die Ausgabebereiche des jeweiligen PRINT ... PRINT3. G5(L1) unterlegt dagegen die ganze erste Zeile der Tabellenzelle, also den Bereich von PRINT und PRINT1, G5(L2) die zweite Zeile, also den Bereich von PRINT2 und PRINT3 und G5(L12) den ganzen Ausgabebereich von PRINT, PRINT1, PRINT2 und PRINT3.

COi (v ... b) Gi (v ... b)
Die Zahlenwerte v und b machen die Farbintensität der Tonfläche abhängig von den Zählerwerten der Tabelle. Die Angabe

```
PRINT2=ABS,CO3(1...100)
```

gibt Werte kleiner oder gleich 1 ohne Tonfläche aus, Werte größer oder gleich 100 werden mit der Farbe CO3 unterlegt. Die Werte zwischen 1 und 100 werden mit Farbwerten zunehmender Intensität unterlegt.
Die Angabe

```
PRINT2=ABS,CO3(100...1)
```

verfährt umgekehrt: Werte größer oder gleich 100 erhalten keine Tonfläche, Werte kleiner oder gleich 1 die volle Farbe CO3 und die Werte dazwischen werden mit entsprechend abgestufter Intensität unterlegt.

Die Angaben Li und (v ... b) lassen sich auch kombinieren. Zum Beispiel:

```
PRINT=ABS,CO3(L1 100...1)
```

HIST Diese Angaben unterlegen die Zählergebnisse mit einem grünen Histogrammbalken, dessen Länge
HISTO proportional vom jeweiligen Zählerwert abhängt.
Mit einer zusätzlichen Angabe COi oder Gi lassen sich beliebig andere Farben auswählen. Zahlenwerte größer oder gleich 100 führen zu einem Balken maximaler Länge, Werte kleiner oder gleich 0 erhalten keinen Balken. Mit den Angaben COi(v ... b) und Gi(v ... b) lassen sich beliebige andere Von-Bis-Werte vorgeben. Siehe Beispiel *Unterlegung mit Tonflächen*.

Beispiel: Mehrere Aufbereitungen pro Tabellenzelle

```
XTAB   COL=TOTAL;C1;N1
       ROW=TOTAL;C2
       PRINT=ABS
       PRINT2=PCOL1,1,%
       PRINT3=PROW1,%
```

Dieses Statement könnte folgende Tabelle erzeugen:

	Gesamt		Männer		Frauen	
Gesamt	4594		992		3602	
	100,0%	100%	21,6%	100%	78,4%	100%
Alter						
14 - 18 Jahre	3		1		2	
	100,0%	0%	33,3%	0%	66,7%	0%
19 - 40 Jahre	238		76		162	
	100,0%	5%	31,9%	8%	68,1%	4%
über 40 Jahre	1101		329		772	
	100,0%	24%	29,9%	33%	70,1%	21%

Die Angabe PRINT=ABS gibt die Zählergebnisse in der ersten Zeile jeder Tabellenzelle als Absolutwert aus. Damit ist der unverrechnete Zählerwert gemeint; er kann auch ein negatives Vorzeichen besitzen.

Da keine Angabe PRINT1 vorliegt, werden die Werte für PRINT rechtsbündig in die Spalten gestellt.

Die Angaben PRINT2 und PRINT3 sorgen für die zusätzliche Aufbereitung der Zählergebnisse in der zweiten Zeile jeder Tabellenzelle.

PRINT2=PCOL1,1,% erstellt Prozentwerte mit einer Nachkommastelle und dem Zeichen % dahinter. Basis für die Prozentuierung sind die Werte der ersten Spalte der jeweiligen Zeile.

PRINT3=PROW1,% liefert Prozentwerte rechts von den Werten PRINT2, ohne Nachkommastellen und mit dem Zeichen % dahinter. Basis für die Prozentuierung sind die Werte der ersten Zeile der jeweiligen Spalten.

Beispiel: Unterschiedliche Aufbereitung der Zähler

```
XTAB  COL=TOTAL;C1
      ROW=TOTAL;C2;N1:'Betrag'
      PRINT=R(1..-2)ABS
            R(-1)ABS,'& €'
      PRINT1=R(2..-2)PROW1,1,%
            R(-1)PCOL1,%
```

Dieses Statement kann folgende Tabelle erzeugen:

	Gesamt	Männer	Frauen
Gesamt	4594	992	3602
C2.1	2294 49,9%	508 51,2%	1786 49,6%
C2.2	2300 50,1%	484 48,8%	1816 50,4%
Betrag	148 € 100%	40 € 27%	108 € 73%

Die Angaben PRINT und PRINT1 geben die Zählergebnisse in jeder Tabellenzelle auf zwei verschiedene Arten nebeneinander aus.

Mit R(1..-2)ABS bei PRINT werden ab Zeile 1 bis zur vorletzten Zeile die Ergebnisse als Absolutwerte ausgewiesen.

R(-1)ABS,'& €' gibt die Werte der letzten Zeile ebenfalls als Absolutwert aus, jedoch mit dem Zeichen € dahinter.

PRINT1 enthält keine Angabe zur ersten Zeile. Daher werden dort auch keine Werte für PRINT1 ausgegeben. Wegen R(2..-2)PROW1,1,% werden die Werte von Zeile 2 bis zur vorletzten Zeile prozentuiert, mit der ersten Zeile als Basis, einer Nachkommastelle und dem Zeichen % dahinter.

Durch R(-1)PCOL1,% wird schließlich erreicht, dass die Werte der letzten Zeile ebenfalls zu prozentuieren sind, mit der ersten Spalte als Basis, ohne Nachkommastellen, mit dem Zeichen % dahinter.

Beispiel: Aufbereitung nach Klassenbuchstaben

```
XTAB  ROW=TOTAL;A{C2.1;.2}
      COL=TOTAL;B{C1.1;.2;N10}
      PRINT=R(1)ABS
            R(A)C(1)ABS
                C(B)PROW1,%
```

Dieses Statement könnte folgende Tabelle erzeugen:

	Gesamt	Männer	Frauen
Gesamt	4594	992	3602
C2.1	2294	51%	50%
C2.2	2300	49%	50%

In ROW werden die Merkmale C2.1 ; .2 durch die Angabe A{...} zu einer Klasse A zusammengefasst, auf die sich die folgende PRINT-Anweisung sehr einfach beziehen kann.

Hinter COL bilden die Zähloperanden C1.1 ; .2 ; N10 die Klasse B. Siehe bei XTAB im Abschnitt *Klassenbildung von Zähloperanden* unter COL/ROW/FILTER.

Die Angabe PRINT=R(1)ABS sorgt zunächst für Absolutwerte in Zeile 1.

Die Anweisung R(A) dahinter bewirkt, dass alle noch folgenden Angaben in PRINT nur für die Zeilen der Klasse A gelten.

Hinter R(A) sorgen die Angaben C(...) dafür, dass in den Zeilen der Klasse A die Spalten unterschiedlich aufbereitet werden: C(1)ABS liefert Absolutwerte in Spalte 1 und

C(B)PROW1,% ergibt Prozentwerte in den Spalten der Klasse B , mit Zeile 1 als Basis und Prozentzeichen % dahinter.

Beispiel: Prozent- und Indexwerte

```
XTAB  ROW=TOTAL;C2.1;.2
      COL=TOTAL;C1
      PRINT=PROW1,%
      PRINT2=IROW1COL1,1
```

Dieses Statement kann folgende Tabelle liefern:

	Gesamt	Männer	Frauen
Gesamt	100% 100,0	100% 100,0	100% 100,0
C2.1	50% 100,0	51% 102,6	50% 99,3
C2.2	50% 100,0	49% 97,5	50% 100,7

Die Angabe PRINT=PROW1,% sorgt für die Ausgabe der Zählergebnisse als Prozentwerte mit der Zeile 1 als Prozentuierungsbasis und dem Zeichen % dahinter.

PRINT2 gibt direkt darunter die Werte noch einmal aus. IROW1 sorgt zunächst für die gleiche Prozentuierung wie bei PRINT. Die Eintragung COL1 dahinter bewirkt aber, dass diese Prozentwerte noch einmal auf die Prozentwerte der ersten Spalte prozentuiert werden (Indexwerte). Durch ,1 dahinter wird schließlich eine Nachkommastelle angefordert.

Die Angabe ABS hinter ROW=TOTAL sorgt dafür, dass die Werte der TOTAL-Zeile von diesen Aufbereitungen ausgenommen sind und nur als Absolutwerte gedruckt werden.

Beispiel: Verschiedene Schriften

```
XTAB  COLS=TOTAL:'Gesamt';C11.2;..3
      ROWS=TOTAL:'Basis',FT4;C16;MEAN(N1232):'  Mittelwert'
      PRINT=R(2..-2)PCOL1,1,%
      PRINT1=R(2..-2)PROW1,2,FT3
      PRINT3=R(2..-2)ABS,FT4
             R(-1)ABS,2,'& €',FT3
```

Dieses Statement kann folgende Tabelle erzeugen:

	Gesamt	Alter	
		Von 21 bis 25 Jahre	Von 26 bis 30 Jahre
Basis			
Ortsgrösse Wohnort			
Bis 50.000 Einwohner	100,0% **33,96** 1560	6,1% **39,92** 95	21,4% **30,34** 334
Mehr als 50.000 bis 100.000 Einwohner	100,0% **8,40** 386	5,4% **8,82** 21	21,2% **7,45** 82
Mehr als 100.000 bis 250.000 Einwohner	100,0% **11,82** 543	4,6% **10,50** 25	27,8% **13,71** 151
Mehr als 250.000 bis 500.000 Einwohner	100,0% **9,27** 426	5,6% **10,08** 24	24,6% **9,54** 105
Mehr als 500.000 bis 1 Million Einwohner	100,0% **13,43** 617	4,7% **12,18** 29	27,4% **15,35** 169
Mehr als 1 Million Einwohner	100,0% **20,70** 951	4,2% **16,81** 40	25,4% **21,98** 242
Mittelwert	**59,79 €**	**35,53 €**	**43,90 €**

Da das Statement XTAB keine Angabe FONT enthält, werden alle Tabellenteile, zu denen keine Schriftangaben vorliegen, mit der Schrift FT2 gedruckt. Das gilt auch für die durch die Angabe PRINT aufbereiteten Zählergebnisse.

Zur ersten Tabellenzeile wird hinter ROWS die Schrift FT4 verlangt. Das führt dazu, dass sowohl der Zeilentext 'Basis' als auch alle Zählergebnisse der Zeile in einer größeren Schrift erscheinen.

Hinter PRINT1 wird die fette Schrift FT3 angefordert. Die Schrift der Zeilentexte wird dadurch nicht verändert, nur die Zählergebnisse werden fett gedruckt.

Hinter PRINT3 wird zunächst von der zweiten bis zur vorletzten Zeile die größere Schrift FT4 verlangt und für die letzte Zeile die fette Schrift FT3. Diese Angaben haben ebenfalls keinen Einfluss auf die Zeilentexte.

Beispiel: Zähler mit Bedingungen aufbereiten

```
XTAB  COL={MEAN(N200):'Durchschnitt 1...6 der Zufriedenheit'}
          {TOTAL:'Insgesamt';C1}
      ROW=C200
      PRINT=ABS,1 IF >=MAX(R(1..-1))-0.5,FT5
                  IF <=MIN(R(1..-1))+0.5,G4
                  IF >= MAX-0.5,' *  &'
      BOTTOM='Die besten Werte jeder Spalte sind fett gedruckt, '-
             'die schlechtesten grau unterlegt.'/
             'Die besten Werte der gesamten Tabelle sind'-
             ' zusätzlich mit einem Stern gekennzeichnet.'/
             'Zu den besten und schlechtesten Werten zählen auch'-
             ' Ergebnisse die um maximal 0.5 Punkte'/
             'davon abweichen.'
```

Dieses Statement kann folgende Tabelle erzeugen:

	Durchschnitt 1...6 der Zufriedenheit		
	Insgesamt	Männer	Frauen
Lieferzeit	* **4,7**	3,1	* **5,0**
Qualität	3,9	3,1	4,0
Service	3,8	3,2	3,9
Garantie	**4,2**	**4,1**	4,2
Preis	3,2	2,8	3,4

Die besten Werte jeder Spalte sind fett gedruckt, die schlechtesten grau unterlegt.
Die besten Werte der gesamten Tabelle sind zusätzlich mit einem Stern gekennzeichnet.
Zu den besten und schlechtesten Werten zählen auch Ergebnisse die um maximal 0.5 Punkte davon abweichen.

Die Angabe PRINT = ABS,1 im Statement XTAB sorgt zunächst dafür, dass alle Zählergebnisse als Absolutwerte mit einer Nachkommastelle erscheinen.

Aufgrund der ersten Bedingung IF >= MAX(R(1 . . -1)) - 0.5, FT5 wird in jeder Spalte der maximale Wert über alle Zeilen ermittelt. Jeder Zählerwert dieser Spalte, der sich um höchstens 0,5 Punkte vom Maximalwert unterscheidet, wird in der fetten Schrift FT5 gedruckt.

Die zweite Bedingung IF <= MIN(R(1 . . -1)) + 0.5, G4 bewirkt entsprechend, dass die kleinsten Werte der Spalte mit dem Grauton G4 unterlegt werden.

Mit der letzten Bedingung IF >= MAX - 0.5, ' * &' wird der größte Wert aus allen Zeilen und Spalten der Tabelle ermittelt. Zählergebnisse, die sich um nicht mehr als 0,5 Punkte von diesem größten Wert unterscheiden, werden zusätzlich mit einem Stern versehen.

Beispiel: Rangzahlen

```
XTAB  COL=TOTAL;C1
      ROW=TOTAL;C20
      PRINT=R(2..-1)PROW1,1,%
      PRINT1=R(1)ABS
             R(2..-1)PROW1 RMAXC,'(&)'
```

Dieses Statement kann folgende Tabelle erzeugen:

	Gesamt		Männer		Frauen	
Gesamt	4594		992		3602	
Alter						
Von 18 bis 20 Jahre	0,1%	(8)	0,1%	(7)	0,1%	(8)
Von 21 bis 25 Jahre	5,2%	(5)	7,7%	(5)	4,5%	(6)
Von 26 bis 30 Jahre	24,0%	(2)	33,2%	(1)	21,4%	(3)
Von 31 bis 35 Jahre	26,2%	(1)	25,0%	(2)	26,5%	(1)
Von 36 bis 40 Jahre	19,6%	(4)	15,3%	(3)	20,8%	(4)
Von 41 bis 50 Jahre	20,2%	(3)	14,8%	(4)	21,6%	(2)
Von 51 bis 60 Jahre	4,4%	(6)	3,5%	(6)	4,6%	(5)
Über 60 Jahre	0,4%	(7)	0,1%	(7)	0,5%	(7)

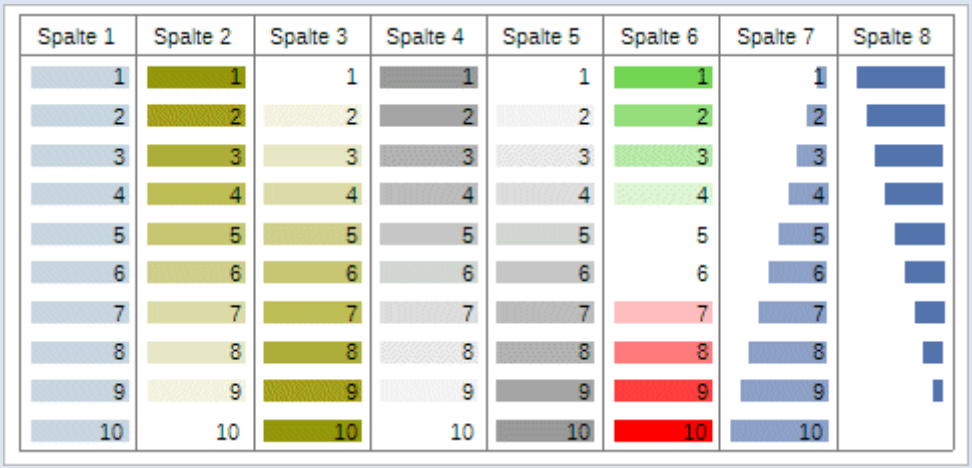
Die Angabe PRINT sorgt ab der zweiten Zeile für Prozentwerte mit der ersten Zeile als Basis, einer Nachkommastelle und dem Prozentzeichen dahinter.

PRINT1 stellt Absolutwerte in die erste Zeile und gibt ab der zweiten Zeile wegen RMAXC Rangzahlen aus, de getrennt pro Spalte vergeben werden.

PROW1 RMAXC ermittelt die Rangzahlen anhand der Prozentwerte mit Zeile 1 als Basis. Die Angabe ' (&)' stellt diese Rangzahlen in Klammern.

Beispiel: Unterlegung mit Tonflächen

```
XTAB . . .
      PRINT=C(1)ABS,CO16
            C(2)ABS,CO5(10...1)
            C(3)ABS,CO5(1...10)
            C(4)ABS,G8(10...1)
            C(5)ABS,G8(1...10)
            C(6)ABS,CO2(6...10) IF<=5,CO10(5...0)
            C(7)ABS,HIST,CO15(0...10)
            C(8)ABS,HISTO,CO14(10...1)
```



Spalte 1	Spalte 2	Spalte 3	Spalte 4	Spalte 5	Spalte 6	Spalte 7	Spalte 8
1	1	1	1	1	1	1	
2	2	2	2	2	2	2	
3	3	3	3	3	3	3	
4	4	4	4	4	4	4	
5	5	5	5	5	5	5	
6	6	6	6	6	6	6	
7	7	7	7	7	7	7	
8	8	8	8	8	8	8	
9	9	9	9	9	9	9	
10	10	10	10	10	10	10	

`C(1)ABS CO16`
unterlegt die Zählergebnisse der Spalte 1 mit der Farbe CO16 unabhängig von ihren Werten.

`C(2)ABS,CO5(10...1)`
versieht die Werte der Spalte 2 mit der Farbe CO5, wobei die Intensität der Farbe von den Zählergebnissen abhängt: Die Angabe (10...1) gibt dem Wert 1 die volle Farbe CO5, während größere Werte immer schwächer eingefärbt werden.

`C(3)ABS,CO5(1...10)`
Diese Angabe liefert in Spalte 3 den entgegen gesetzten Verlauf der Farben: Der Wert 10 erhält die volle Farbe CO5, während kleinere Werte zunehmend schwächer eingefärbt werden.

`C(4)ABS,G8(10...1)`
`C(5)ABS,G8(1...10)`
Diese Angaben erzeugen in Spalte 4 und 5 ähnliche Ergebnisse, jedoch mit dem Grauton G8 anstelle einer Farbe.

`C(6)ABS,CO2(6...10) IF<=5,CO10(5...0)`
In Spalte 6 werden Werte unter 5 grün und Werte über 5 rot unterlegt. Die Intensität der Farben steigt mit dem Abstand der Werte von 5.

`C(7)ABS,HIST,CO15(0...10)`
`C(8)ABS,HISTO,CO14(10...1)`
In den Spalten 7 und 8 machen die Angaben HIST und HISTO die Länge der Farbbalken von den Zählergebnissen abhängig. HISTO unterdrückt zusätzlich die Ausgabe der Zählergebnisse.

Beispiel: Histogramme

```
XTAB COL=TOTAL;C1
     ROW=TOTAL;C1090
     PRINT=R(1)ABS
           R(2...-1)ABS,HIST,CO11(50...2000)
```

Dieses Statement kann folgende Tabelle erzeugen:

	Gesamt	Männer	Frauen
Gesamt	4594	992	3602
F9: Betriebsgröße (Mitarbeiter)			
Kleinstunternehmen: weniger als 10	328	87	241
Kleinunternehmen: von 10 bis 99	1104	258	846
Mittlere Unternehmen: von 100 bis 499	957	212	745
Großunternehmen: 500 und mehr Beschäftigte	1971	396	1575

Hinter PRINT stellt R(2 ... -1)ABS, HIST die Zählergebnisse ab Zeile 2 als Absolutwerte mit Histogrammbalken in die Tabelle.

CO11 bestimmt die Farbe der Balken und (50...2000) den Wertebereich, den diese abdecken sollen.

XTAB: RCOL

RCOL | **=** n
RCOLS |

Diese Angabe wird wirksam, wenn Tabellen zu breit sind, um auf einer Seite Platz zu finden.
Solche Tabellen werden automatisch an geeigneten Spaltengrenzen aufgetrennt und in mehreren Teilen hintereinander gedruckt.
RCOL legt fest, wie viele Spalten - zum Beispiel als Total-Spalten - am Anfang jeder Fortsetzungstabelle zu wiederholen sind.

Beim Auftrennen in Teiltabellen versucht CNTA zunächst, die Tabellenteile hintereinander auf eine Seite zu stellen. Erst wenn der Platz dort nicht ausreicht, wird die nächste Teiltabelle auf eine neue Seite ausgegeben.

Ohne die Angabe RCOL wird mit RCOL=2 gearbeitet.

RCOLS Diese Angabe gilt für das laufende und alle folgenden XTAB-Statements, bis sie durch neue RCOLS ersetzt wird.

RCOL Diese Angabe gilt nur für das laufende XTAB-Statement.
Eventuelle Angaben RCOLS aus früheren XTAB-Statements werden für das laufende XTAB unwirksam.

n Hier ist die Anzahl 0 ... 9 von Spalten anzugeben, die vom Tabellenanfang in jeder Fortsetzungstabelle zu wiederholen sind.
Durch SUPPRESS=FCOL unterdrückte Spalte werden dabei nicht mitgezählt.

RCOL=1 wiederholt nur die Textspalte, RCOL=2 die Textspalte mit der ersten Zählerspalte usw.
RCOL=0 wiederholt keine Spalten, auch nicht die Textspalte.

Beispiel

```
XTAB  COL=TOTAL;C2.1;..6
      ROW=TOTAL;C1
      RCOL=1
```

Diese Statements könnten folgende Tabellen liefern:

	TOTAL	C2.1	C2.2	C2.3
TOTAL	24	1	1	2
C1.1	11	-	-	2
C1.2	13	1	1	-

	C2.4	C2.5	C2.6
TOTAL	6	1	4
C1.1	2	1	1
C1.2	4	-	3

Hierbei wird unterstellt, dass nicht alle Spalten der Tabelle auf eine Seite passen. CNTA trennt dann die Spalten an einer geeigneten Stelle auf und zerlegt die eine Tabelle in mehrere. Diese Teiltabellen werden übereinander ausgegeben, wenn möglich sogar auf der gleichen Seite.

Die Angabe RCOL=1 sorgt dafür, dass in den Fortsetzungstabellen die erste Spalte des Tabellenanfangs wiederholt wird. In diesem Beispiel ist dies gerade die Spalte mit den Zeilentexten.

XTAB: RROW

RROW | **=** n
RROWS |

Diese Angabe wird wirksam, wenn Tabellen zu viele Zeilen besitzen, um auf einer Seite Platz zu finden. Solche Tabellen werden automatisch an geeigneten Zeilengrenzen aufgetrennt und in mehreren Teilen hintereinander ausgegeben
.
RROW legt fest, wie viele Zeilen - zum Beispiel als Basis-Zeilen - am Anfang jeder Fortsetzungstabelle zu wiederholen sind. Jede dieser aufgetrennten Tabellen beginnt auf einer neuen Seite.

Fehlt die Angabe RROW, so wird mit RROW=2 gearbeitet:
Die Spaltenüberschriften und die erste Zählerzeile werden beim Aufbrechen der Tabelle wiederholt.

RROWS Diese Angabe gilt für das laufende und alle folgenden XTAB-Statements, bis sie durch neue RROWS ersetzt wird.

RROW Diese Angabe gilt nur für das laufende XTAB-Statement.
Eventuelle Angaben RROWS aus früheren XTAB-Statements werden für das laufende XTAB unwirksam.

n Hier ist die Anzahl 0 ... 9 von Zeilen anzugeben, die vom Tabellenanfang in jeder Fortsetzungstabelle zu wiederholen sind. Eine durch SUPPRESS=FROW unterdrückte Zeile wird dabei nicht mitgezählt.

RROW=1 wiederholt nur die Spaltenüberschriften, RCOL=2 die Spaltenüberschriften mit der erste Zählerzeile usw. RCOL=0 wiederholt keine Zeilen, auch nicht die Spaltenüberschriften.

Beispiel

```
XTAB  ROW=TOTAL;;C2.1;..10
      COL=TOTAL;C1
      SUPPRESS=FILTER,SPACES
      RROW=3
```

Dieses Statement könnte folgende Tabellen erzeugen:

	TOTAL	C1.1	C1.2	C1.3
TOTAL	24	7	4	13
C2.1	-	-	-	-
C2.2	1	-	-	1
C2.3	1	-	-	1
C2.4	1	-	1	-
C2.5	3	-	1	2

	TOTAL	C1.1	C1.2	C1.3
TOTAL	24	7	4	13
C2.6	4	1	1	2
C2.7	1	1	-	-
C2.8	3	1	-	2
C2.9	2	2	-	-
C2.10	3	1	-	2

In diesem Beispiel wird unterstellt, dass die Tabelle dieses XTAB-Statements in der Höhe nicht auf eine Seite passt. CNTA trennt dann die Tabelle bei geeigneten Zeilen auf und druckt sie als unabhängige Teiltabellen auf mehreren Seiten.

Die Angabe RROW=3 sorgt dafür, dass vor jeder Fortsetzungstabelle sowohl die Spaltenüberschriften als auch die ersten beiden Tabellenzeilen wiederholt werden.
In diesem Beispiel ist das zusätzlich zu den Spaltenüberschriften die Zeile TOTAL und die darauf folgende Leerzeile.

XTAB: SAME

Diese Angabe übernimmt die Werte der vorigen Tabelle ohne neue Zählung.. Das spart Laufzeit bei unterschiedlicher Aufbereitung gleicher Zählergebnisse.

Durch SAME werden die Angaben ROW, COL, FILTER und GROUP des vorausgegangenen XTAB-Statements in das laufende Statement übernommen. Die Auswertung der einzelnen statistischen Fälle entfällt, dafür werden die fertigen Zählergebnisse des vorigen XTAB in das laufende XTAB übernommen.
Dies kann zu einer beträchtlichen Verkürzung der Auswertungszeiten führen.

Die Angabe SAME schließt die Verwendung von ROW, COL, FILTER oder GROUP im laufenden Statement aus. Alle anderen Angaben eines XTAB-Statements sind jedoch zulässig und dürfen sehr wohl vom vorigen XTAB abweichen.

SAME ist immer dann zu empfehlen, wenn in mehreren direkt hintereinander liegenden XTAB-Statements die gleichen Zählergebnisse auf unterschiedliche Weise aufbereitet werden sollen.

SAME lässt sich auch dann verwenden, wenn zwischen dem laufenden XTAB und dem davor liegenden XTAB-Statement ein oder mehrere XADD- oder XADDB-Statements liegen. SAME sorgt dann für die Übernahme der Zählerwerte einschließlich der aus XADD oder XADDB stammenden Angaben.

Beispiel

```
XTAB  COLS=TOTAL;C16     ROWS=TOTAL;;N1;...4
      PRINT=ABS,1
XTAB  SAME  PRINT=PROW1,1,%
XTAB  SAME  PRINT=PCOL1,1,%
```

Diese Statements könnten die folgenden Tabellen erzeugen:

		Alter in Jahren			
	TOTAL	...18	19...35	36...65	66...
TOTAL	1000.0	259.0	239.0	262.0	240.0
BUNTE	927.8	240.0	219.0	230.8	238.0
NEUE REVUE	230.5	60.2	53.9	62.6	53.9
QUICK	1359.3	344.7	315.3	374.7	324.6
STERN	349.6	90.9	86.5	89.8	82.5

		Alter in Jahren			
	TOTAL	...18	19...35	36...65	66 ...
TOTAL	100.0%	100.0%	100.0%	100.0%	100.0%
BUNTE	92.8%	92.7%	91.6%	88.1%	99.2%
NEUE REVUE	23.1%	23.2%	22.5%	23.9%	22.5%
QUICK	135.9%	133.1%	131.9%	143.0%	135.3%
STERN	35.0%	35.1%	36.2%	34.3%	34.4%

		Alter in Jahren			
	TOTAL	...18	19...35	36...65	66...
TOTAL	100.0%	25.9%	23.9%	26.2%	24.0%
BUNTE	100.0%	25.9%	23.6%	24.9%	25.7%
NEUE REVUE	100.0%	26.1%	23.4%	27.1%	23.4%
QUICK	100.0%	25.4%	23.2%	27.6%	23.9%
STERN	100.0%	26.0%	24.7%	25.7%	23.6%

Das erste XTAB-Statement liefert dabei auf gewohnte Weise die erste Tabelle.
Die Angabe PRINT=ABS,1 sorgt für die Aufbereitung der Zählergebnisse als Absolutwerte mit einer Nachkommastelle.

Im zweiten XTAB-Statement übernimmt SAME die Zählergebnisse aus dem ersten XTAB, ohne die statistischen Fälle neu auszuwerten.
PRINT=PROW1,1,% bewirkt hier aber die Ausgabe von Prozentwerten, mit der ersten Zeile als Basis.

Entsprechend arbeitet das dritte XTAB.
PRINT=PCOL1,1,% liefert ebenfalls Prozentwerte, hier jedoch mit der ersten Spalte als Basis.

XTAB: SORT

SORT =	**ROWS** i	< v >	< **,**	**ROWS** i	... >
SORTS	**AROWS** i	**(** x **)** ...		**AROWS** i	
	AROWZ i	**(** x **)** v ...		**AROWSZ** i	
	COLS i	**[** x **]** ...		**COLS** i	
	ACOLS i	**[** x **]** v ...		**ACOLS** i	
	ACOLSZ i			**ACOLSZ** i	
	ROWSA			**ROWSA**	
	COLSA			**COLSA**	
	NO				

Diese Angabe sortiert die Zeilen oder Spalten einer Tabelle. Dies geschieht nach Ermittlung der Zählergebnisse, nach Verrechnung der Statements XADD und XADDB und nachdem die Folgestatements zu XTAB abgearbeitet sind, jedoch vor der Aufbereitung der Zählergebnisse durch PRINT ... PRINT3.

Als Zeile oder Spalte gilt jeder Zähloperand aus ROW oder COL, der durch ein Semikolon abgetrennt ist.
Eine Tabellenzeile kann aus mehreren Druckzeilen bestehen, zum Beispiel durch gleichzeitige Angabe von PRINT und PRINT2 oder durch mehrzeilige Texte.

Leere Zeilen oder Spalten der Form ... ; ; ... oder ... ; :'text' ; ... sind bei der Zeilen- oder Spaltennummerierung für SORT mitzuzählen. Gleiches gilt für Zeilen und Spalten, die durch SUPPRESS unterdrückt wurden.

Die Sortierung lässt sich durch Angaben der Form (x) oder [x] auf ausgewählte Zeilen oder Spalten beschränken: (x) sortiert die in den Klammern angegebenen Zeilen oder Spalten untereinander.
[x] dagegen vertauscht Gruppen von Zeilen oder Spalten, abhängig von den Werten ihrer ersten Zeile oder Spalte, ohne innerhalb der Gruppen zu sortieren.

Die Sortierregeln hinter SORT werden nacheinander von links nach rechts abgearbeitet.

Die Zeile oder Spalte, nach deren Werten sortiert wird, kann auch durch ein SUPPRESS Statement unterdrückt werden.

SORTS Diese Angabe gilt für das laufende und alle folgenden XTAB-Statements, bis sie durch neue SORTS ersetzt wird.

SORT Diese Angabe gilt nur für das laufende XTAB-Statement. Eventuelle Angaben SORTS aus früheren XTAB-Statements werden für das laufende XTAB unwirksam.

ROWS i Die Zeilen der Tabelle werden nach den Werten der Zählerspalte i = 1 ... 32 767 sortiert.
Dies erfolgt in absteigender Folge: Große Werte stehen vorne, kleine weiter hinten und negative ganz hinten. Die Angabe i kann auch fehlen; dann wird nach den Werten der Spalte 1 sortiert. Für i können auch negative Werte i = -1 ... -32 767 verwendet werden. Bei -1 wird nach den Werten der letzten Spalte der Tabelle sortiert, bei -2 nach der vorletzten Spalte usw.

AROWS i Wie ROWSi, jedoch wird in aufsteigender Folge sortiert: Kleine Werte stehen vorne, größere hinten und negative ganz vorn in der Tabelle.
Die Werte 0 werden als *keine Angabe* verstanden und an das Ende der sortierten Zeilen gestellt.

AROWSZ i Wie AROWSi, jedoch werden die Werte 0 als normales Zählergebnis interpretiert und entsprechend einsortiert.

ROWSA Die Zeilen der Tabelle werden nach den Zeilentexten alphabetisch aufsteigend sortiert.

COLS i Die Spalten der Tabelle werden nach den Werten der Zählerzeile i = 1 ... 32 767 sortiert.
Dies erfolgt in absteigender Folge: Große Werte stehen vorn, kleine weiter hinten und negative ganz hinten. Die Angabe i kann auch fehlen; dann wird nach den Werten der Zeile 1 sortiert.
Für i können auch negative Werte i = -1 ... -32 767 verwendet werden. Bei -1 wird nach den Werten der letzten Zeile der Tabelle sortiert, bei -2 nach der vorletzten Zeile usw.

ACOLS i Wie COLSi, jedoch wird in aufsteigender Folge sortiert:
Kleine Werte stehen vorn, größere hinten und negative ganz vorn in der Tabelle.
Die Werte 0 werden als *keine Angabe* verstanden und an das Ende der sortierten Spalten gestellt.

ACOLSZ i Wie ACOLSi, jedoch werden die Werte 0 als normales Zählergebnis interpretiert und entsprechend einsortiert.

COLSA Die Spalten der Tabelle werden nach den Spaltentexten alphabetisch aufsteigend sortiert.

NO Hiermit wird die Angabe SORTS= ... aus früheren XTAB Statements unwirksam gemacht.

(x) Diese Angabe sortiert nur ausgewählte Zeilen oder Spalten der Tabelle.
Dazu sind die zu sortierenden Zeilen oder Spalten in der Form (v . . . b) mit Von- und Bis-Nummern 1 ... 32 767 anzugeben. Auch negative Werte sind zulässig: -1 ist die letzte Zeile oder Spalte der Tabelle, -2 die vorletzte usw.

Die Angabe

```
SORT=ROWS1(2..5)(9..-1)
```

sortiert die Zeilen 2 ... 5 untereinander nach den Werten der Spalte 1 und anschließend ab Zeile 9 die restlichen Zeilen.

Sind in ROW oder COL Zeilen oder Spalten mit Klassenangaben A ... Z{ } versehen, so sind anstelle der Zeilen- oder Spaltennummern auch die Klassenbuchstabe A ... Z möglich:

```
ROW=TOTAL;A{C1.1;...5};B{C2.1;...3}
SORT=ROWS1(A)(B)
```

Hier sind in ROW die Operanden C1.1;...5 mit der Klassenangabe A gekennzeichnet und die Operanden C2.1 ... 3 mit der Klassenangabe B. SORT sortiert nun die Zeilen der Klasse A untereinander und anschließend die Zeilen der Klasse B.

Die Angaben

```
ROW=TOTAL;A{C1.1;...5};A{C2.1;...3}
SORT=ROWS1(A)
```

liefern das gleiche Ergebnis: Die Zeilen der Operanden C1.1;...5 werden untereinander sortiert und unabhängig davon die Zeilen der Operanden C2.1 ... 3.

[x] Diese Angaben vertauschen Gruppen von Zeilen oder Spalten ohne die Zeilen oder Spalten innerhalb der Gruppen zu sortieren. Die Gruppen werden anhand ihrer ersten Zeilen oder Spalten sortiert. Zur Auswahl der Gruppen x sind keine Zeilen- oder Spaltennummern möglich sondern nur die Klassenbuchstaben A ... Z.

Die Angaben

```
ROW=TOTAL;B {C1.1;...5};B{C2.1;...3}
SORT=ROWS1 [B]
```

vertauschen die beiden Klassen B abhängig von den Werten ihrer ersten Zeilen. Die Zeilen innerhalb der Klassen B bleiben unsortiert.

Werden eckige und runde Klammern [x] und (x) gleichzeitig verwendet, so sind die eckigen Klammern vor den runden anzugeben. Zum Beispiel:

```
SORT=ROWS1 [B] (B)
```

Hier tauscht [B] zunächst die Klassen B gegeneinander aus, abhängig von den Werten ihrer ersten Zeilen. Danach sortiert (B) die Zeilen innerhalb der einzelnen Klassen B.

Mehrstufige Klassenangaben begrenzen die Vertauschungen:

```
ROW=A{C10.1;B{C1.1;D{C1.2;...4}};
            B{C2.1;D(C2.2;...5}}};
    A{C10.1;B{C3.1;D{C3.2;...6}}};
SORT=ROWS1 [A] [B] (D)
```

Zunächst vertauscht [A] die beiden Klassen A miteinander, da diese nicht in übergeordneten Klassen liegen.
Danach vertauscht [B] die beiden Klassen B{C1.1 ... } und B{C2.1 ... } gegeneinander , da sie in der gleichen übergeordneten Klasse A liegen. Die Klasse B{C3.1 ... } wird nicht verschoben, da sie in einer getrennten Klasse A liegt.
Abschließend sortiert (D) die Zeilen jeweils innerhalb der drei Klassen D{ }.

v Ohne Verrechnungsregeln wird nach den absoluten, nicht prozentuierten Werten der jeweiligen Zeile oder Spalte sortiert.
Mit diesen Angaben kann auch nach Prozent- oder Indexwerten sortiert werden:

ABS	Es wird nach den Absolutwerten sortiert.
PCOL m	Es wird nach Prozentwerten sortiert, mit den Werten der Spalte *m* als Prozentuierungsbasis.
PCOL m **ROW** j	Es wird nach Prozentwerten sortiert, mit dem Wert in Spalte *m* und Zeile *j* als Prozentuierungsbasis.
PROW m	Es wird nach Prozentwerten sortiert, mit den Werten der Zeile *m* als Prozentuierungsbasis.
PROW m **COL** j	Es wird nach Prozentwerten sortiert, mit dem Wert in Zeile *m* und Spalte *j* als Prozentuierungsbasis.
PTOTAL	Es wird nach Prozentwerten sortiert, mit dem FILTER-Wert als Prozentuierungsbasis.
ICOL m **ROW** j	Es wird nach Indexwerten sortiert. Prozentuierungsbasis sind dabei die Werte der Spalte *m* und Indexbasis die Werte der Zeile *j*.
IROW m **COL** j	Es wird nach Indexwerten sortiert. Prozentuierungsbasis sind dabei die Werte der Zeile *m* und Indexbasis die Werte der Spalte *j*.

Im Beispiel

```
SORT=ROWS1 PCOL2
PRINT=IROW1COL2
```

werden die Tabellenwerte als Index ausgegeben. Die Zeilen werden jedoch nach den Prozentwerten der Spalte 1 sortiert, mit den Werten der Spalte 2 als Prozentuierungsbasis.

```
SORT=ROWS1 (2..5)PROW1 (7..10)PCOL1
```

Hier werden zunächst die Zeilen 2 ... 5 untereinander sortiert, mit den Werten aus Spalte 1, prozentuiert auf die die erste Zeile. Danach werden die Zeilen 7 ... 10 untereinander sortiert, mit den Indexwerten auf Zeile 5 und Spalte 6 als Sortierwerte.

```
SORT=ROWS1 [A]ABS (A)PROW1
```

sortiert wegen ROWS1 die Zeilen nach den Werten der ersten Spalte. Zunächst vertauscht [A]ABS die Klassen A untereinander, abhängig von den Absolutwerten ihrer ersten Zeilen.
Danach sortiert (A)PROW1 die Zeilen innerhalb der Klassen A nach Prozentwerten.

```
SORT=ROWS1 (A)PROW1 , COLS1 [A]ABS
```

sortiert zunächst wegen ROWS1 ... die Zeilen der Klassen A nach Prozentwerten der Spalte 1 und vertauscht danach wegen COLS1 ... die Spalten der Klassen A untereinander, abhängig von den Absolutwerten der Zeile 1.

Beispiel: Zeilen sortieren

```
XTAB  ROW=TOTAL;;S{C1};;S{C2}   COL=TOTAL;C3
      SORT=ROWS1(S)
```

Dieses Statement könnte folgende Tabelle erzeugen:

	TOTAL	C3.1	C3.2	C3.3
TOTAL	120	40	42	39
C1.3	31	12	10	9
C1.4	30	10	11	10
C1.1	30	10	11	10
C1.2	29	8	11	10
C2.1	42	15	14	13
C2.3	40	13	14	13
C2.2	39	12	14	13

SORT=ROWS1 sortiert die Tabellenzeilen nach den Werten der ersten Spalte in absteigender Reihenfolge.

Die Angabe (S) dahinter beschränkt die Sortierung auf die Zeilen der beiden mit S markierten Zeilengruppen S{C1} und S{C2}. Jede Gruppe wird dabei für sich sortiert.

Beispiel: Gruppen von Zeilen sortieren

```
XTAB ROW=TOTAL;;G{C1.S;Z{C1}};;G{C2.S;Z{C2}}
     COL=TOTAL;C3
     SORT=ROWS1 [G] (Z)
```

Dieses Statement könnte folgende Tabelle erzeugen:

	TOTAL	C3.1	C3.2	C3.3
TOTAL	120	40	42	39
C2.S	96	32	34	31
C2.1	34	12	11	10
C2.3	32	10	11	10
C2.2	31	10	11	10
C1.S	72	24	25	23
C1.3	19	7	6	6
C1.4	18	6	6	6
C1.1	18	6	6	6
C1.2	17	5	6	6

SORT=ROWS1 sortiert die Tabellenzeilen nach den Werten der ersten Spalte in absteigender Reihenfolge.

Die Angabe [G] dahinter vertauscht die beiden mit G{...} markierten Gruppen von Zeilen gegeneinander, abhängig von den Werten der jeweils ersten Zeilen C1.S und C2.S; dies ohne die Zeilen innerhalb der Gruppen zu sortieren.

Mit (Z) werden anschließend die durch Z{...} markierten Zeilen untereinander sortiert, jede der beiden Gruppen Z für sich.

XTAB: START

START = | y | **RB**y | **RE**y | **RM**y | **RS**y | **RT**y | **,** | x | **RB**x | **RE**x | **RM**x | **RS**x | **RT**x | **C**

Diese Angabe verändert die Lage der Tabellen auf der Seite. Sie dient insbesondere dazu, mehrere Tabellen aus XTAB-Statements oder Texte aus TEXT-Statements mit der Angabe CONT auf einer Seite zu platzieren.

Durch y und x wird die obere linke Position des Tabellenanfangs festgelegt.
Dabei gelten die Texte aus TITLE, FILTER und BOTTOM als Teil der Tabelle. Sie werden zusammen mit der Tabelle verschoben, nicht jedoch die Fußnoten aus FOOTNOTE.

Mit den Angaben RM ... RT kann eine Tabelle relativ zu den Druckausgaben positioniert werden, die zuvor durch die Statements XTAB oder TEXT auf das gleiche Blatt ausgegeben wurden.

Wurden vorher auf einer Seite keine Tabellen oder Texte gedruckt, so führen die relativen Positionen RM ... RT zum gleichen Ergebnis wie die absoluten Werte x oder y.

Fehlt die Angabe START, so wird mit START = RM0, 0 gearbeitet.

STARTS Diese Angabe gilt für das laufende und alle folgenden XTAB-Statements, bis sie durch ein neues STARTS ersetzt wird.

START Diese Angabe gilt nur für das laufende XTAB-Statement.
Sie hat Vorrang vor einem STARTS aus einem früheren XTAB-Statement.

C Für den Abstand x kann auch der Buchstabe C angegeben werden, zum Beispiel in der Form

```
START=0, C
```

Dadurch wird die Tabelle horizontal auf der Seite zentriert, einschließlich der TITLE-, FILTER- und BOTTOM-Texte.

x Position des linken Tabellenanfangs.
Bei Druckausgabe mit dem Statement PAGE ist hier die Position als Anzahl 0 ... 32 767 von Druckzeichen anzugeben. Bei Ausgabe mit dem Statement PAGEP ist dies die Position in 0 ... 500 Millimetern oder in einer der bei PAGEP beschriebenen Maßeinheiten. Mit negativen Angaben kann auch in den linken Rand hinein gedruckt werden.

Ohne eine Angabe RM ... RT ist *x* der Abstand des Tabellenanfangs vom linken Rand des Blattes. So beginnt die Tabelle mit

```
START=0, 10
```

im Abstand 10 von linken Rand. Die Breite dieses Randes lässt sich mit den Statements PAGE und PAGEP festlegen.

Steht vor *x* eine Angabe RM ... RT, so bezieht sich der Abstand nicht auf den linken Rand, sondern auf den durch RM ... festgelegten Referenzpunkt. Zum Beispiel beginnt die Tabelle mit

```
START=0, RE10
```

im Abstand 10 rechts von der vorigen XTAB- oder TEXT-Ausgabe auf dem Blatt.

y Position des oberen Tabellenanfangs.
Bei Druckausgabe mit dem Statement PAGE ist hier die Position als Anzahl 0 ... 32 767 von Druckzeilen anzugeben. Bei Ausgabe mit dem Statement PAGEP ist dies die Position in 0 ... 500 Millimetern oder in einer der bei PAGEP beschriebenen Maßeinheiten. Mit negativen Angaben kann auch in den oberen Rand hinein gedruckt werden.
Dabei gelten die Texte aus TITLE und FILTER als Teil der Tabelle.

Ohne eine Angabe RM ... RT ist *y* der Abstand des Tabellenanfangs vom oberen Rand des Blattes. So beginnt die Tabelle mit

```
START=10, 0
```

im Abstand 10 vom oberen Rand. Die Höhe dieses Randes lässt sich mit den Statements PAGE und PAGEP festlegen.

Steht vor *y* eine Angabe RM ... RT, so bezieht sich der Abstand nicht auf den oberen Rand, sondern auf den durch RM ... festgelegten Referenzpunkt. Zum Beispiel beginnt die Tabelle mit

```
START=RE10, 0
```

im Abstand 10 unterhalb der vorigen XTAB- oder TEXT-Ausgabe auf dem Blatt.

RB Durch diese Angabe wird der Anfang der Zählerwerte der vorigen XTAB-Ausgabe zum Referenzpunkt. In der x-Richtung ist dies das rechte Ende der Spalte mit den Zeilentexten, in der y-Richtung das untere Ende der Spaltenüberschriften.

RE Durch diese Angabe wird das Ende der vorigen XTAB- oder TEXT-Ausgabe zum Referenzpunkt. In der x-Richtung ist RE das rechte Ende dieser Ausgabe, in der y-Richtung das untere Ende einschließlich eventueller BOTTOM-Texte, jedoch ohne Fußnoten aus FOOTNOTE.

RM Durch diese Angabe wird die maximale, aus XTAB- oder TEXT-Statements stammende Ausgabe zum Referenzpunkt. In der x-Richtung ist das der am weitesten rechts auf dem Blatt gedruckte Punkt, in der y-Richtung der am weitesten unten gedruckte Punkt. Dazu zählen eventuelle BOTTOM-Texte, jedoch keine Fußnoten aus FOOTNOTE.

RS Durch diese Angabe wird der Anfang der vorigen XTAB- oder TEXT-Ausgabe zum Referenzpunkt. In der x-Richtung ist RS der linke Anfang dieser Ausgabe, in der y-Richtung der obere Anfang, vor eventuellen TITLE- oder FILTER-Angaben.

RT Durch diese Angabe wird der Tabellenanfang der der vorigen XTAB-Ausgabe zum Referenzpunkt. In der x-Richtung ist dies der linke Anfang der vorigen Tabelle, in der y-Richtung der obere Anfang der Spaltenüberschriften, hinter eventuellen TITLE- oder FILTER-Angaben.

Beispiel: Mehrere Tabellen auf einem Blatt

```
XTAB  TITLE='Tabelle 1'
      ROW=TOTAL;C3
      COL=C4
      CONT

XTAB  TITLE='Tabelle 2'
      ROW=TOTAL;C2
      COL=C4
      START=0,RM3
      CONT

XTAB  TITLE='Tabelle 3'
      ROW=TOTAL;C1
      COL=C4
      START=RM2,0
```

Diese Statements erzeugen folgende Tabelle:

Tabelle 1

	C4.1	C4.2	C4.3
TOTAL	399	405	400
C3.1	136	146	121
C3.2	151	131	122
C3.3	112	128	157

Tabelle 2

	C4.1	C4.2	C4.3
TOTAL	399	405	400
C2.1	130	132	135
C2.2	134	140	141
C2.3	135	133	124

Tabelle 3

	C4.1	C4.2	C4.3
TOTAL	399	405	400
C1.1	106	96	109
C1.2	95	110	92
C1.3	101	106	93
C1.4	97	93	106

Da das erste XTAB keine Angabe START enthält, steht die dazugehörige Tabelle oben links auf der Seite. Die Angabe CONT bewirkt, dass die nächste Tabelle auf der gleichen Seite ausgegeben wird.

Im zweiten XTAB-Statement sorgt die Angabe START=0,RE3 dafür, dass die dazugehörige Tabelle 0 mm unter dem oberen Blattrand und 3 mm neben der vorigen Tabelle beginnt. CONT bewirkt wiederum, dass auch die folgende Tabelle auf die gleiche Seite gestellt wird.

Die Angabe START=RM2,0 im letzten XTAB-Statement stellt die Tabelle 2 mm unter die bisherigen Tabellen, 0 mm vom linken Rand des Blattes entfernt. Da kein CONT vorhanden ist, ist dies die letzte Tabelle auf dem Blatt.

XTAB: STUBHEAD

```
STUBHEAD  | = | 't'  <,  | L | >       |
STUBHEADS |   |          | R |         |
                         | C |         |
          | NO                         |
```

Diese Angabe stellt einen Text in den Tabellenkopf, über den Zeilentexten in der Höhe der Spaltenüberschriften. Durch die Angabe STYLE=STUBHEAD ... lässt sich dieser Text mit einer Tonfläche unterlegen.

STUBHEADS

Diese Angabe gilt für das laufende und alle folgenden XTAB-Statements, bis sie durch ein neues STUBHEADS ersetzt wird.

STUBHEAD

Diese Angabe gilt nur für das laufende XTAB-Statement, und sie hat Vorrang vor einem eventuell gültigen STUBHEADS aus einem früheren XTAB-Statement.

' t' Der ein- oder mehrzeilige Text, der als Titel gedruckt werden soll; wie im Abschnitt *Textzeilen* beschrieben.

L R C Diese Angaben richten den Text über den Zeilentexten aus:
L = linksbündig, R = rechtsbündig, C = zentriert.
Ohne sie wird der Text linksbündig ausgegeben.

Zum Beispiel stellt
STUBHEAD=' Fr age 24' , C
den Text mittig über die Zeilentexte.

NO setzt die Angabe STUBHEADS aus früheren XTAB-Statements außer Kraft.

Beispiel

```
XTAB  ROW=TOTAL;C3
      COL=TOTAL;C2
      STUBHEAD='Mitglied einer Kirche'
```

Dieses Statement könnte folgende Tabelle erzeugen:

Mitglied einer Kirche		Geschlecht	
	Gesamt	Männer	Frauen
Gesamt	8636	4122	4514
bin Mitglied einer Kirche	5829	2614	3215
bin aus der Kirche ausgetreten	1431	789	642
war nie Kirchenmitglied	1354	704	650
keine konkrete Angabe	22	15	7

XTAB: STYLE

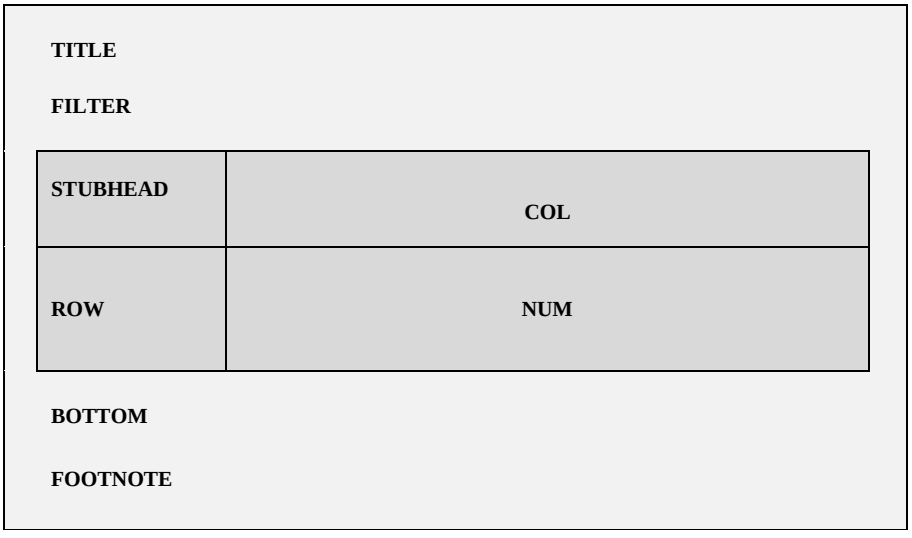
Mit STYLE werden die Regeln zur Aufbereitung der Tabellen verändert:

- Den einzelnen Tabellenteilen lassen sich verschiedene Schriften zuordnen.
- Die Tabellenteile STUBHEAD, ROW und COL lassen sich mit Grautönen oder Farbe unterlegen.
  Die Texte können auf unterschiedliche Weise in die vorhandenen Spaltenbreiten eingepasst werden.
- Kopftexte lassen sich um 90 Grad im und gegen den Uhrzeigersinn drehen.
- Texte zu BOTTOM, COL, ROW, STUBHEAD und TITLE lassen sich rechtsbündig, linksbündig oder zentriert ausgeben.
- Texte zu STUBHEAD und COL lassen sich oben unten oder in der Mitte der frei verfügbaren Höhe positionieren.
- Beim automatischen Zeilenumlauf in ROW lassen sich die Fortsetzungszeilen nach rechts einrücken.
- Mehrzeilig eingegebene Texte können zu einer Zeile zusammengefügt werden.

Dazu sind folgende Angaben möglich:

STYLE **STYLES**	**=** < allgemeine Aufbereitungsregeln > <	**,BOTTOM** **,COL<(s)>** **,FILTER** **,FOOTNOTE** **,NUM** **,ROW<(s)>** **,STUBHEAD** **,TITLE**	spezielle Aufbereitungsregeln >

Die Angabe STYLES gilt für das laufende und alle folgenden XTAB-Statements, bis sie durch ein neues STYLES ersetzt wird. STYLE gilt nur für das laufende XTAB-Statement und hat Vorrang vor STYLES aus früheren XTAB-Statements. Folgende Tabellenteile lassen sich mit STYLE verändern:



BOTTOM Aufbereitungsregeln für die BOTTOM-Texte.

STUBHEAD Aufbereitungsregeln für die STUBHEAD-Texte.

TITLE Aufbereitungsregeln für die TITLE-Texte.

FILTER Aufbereitungsregeln für die FILTER-Texte.

FOOTNOTE Aufbereitungsregeln für die Fußnoten-Texte.

NUM Aufbereitungsregeln für die Zählergebnisse in den Tabellenzellen.

COL Aufbereitungsregeln für die Spaltenüberschriften. Zum Beispiel unterlegt

```
STYLE=COL:G6
```

alle Spaltenüberschriften mit dem Grauton G6. Diese Aufbereitungsregeln beschränken sich auf die Kopftexte, im Gegensatz zu den Font- und Farbangaben hinter COL= ... , die bis hinunter in den Zählerbereich wirken.

Eine Angabe (*s*) hinter COL begrenzt die Aufbereitungsregeln auf einzelne Textstufen in den Tabellenköpfen. Dabei ist *s* eine Stufennummer 1 ... 20 mit 1 für die oberste Textstufe, 2 für die zweithöchste usw. Weiter sind die Angaben –1 ... –20 möglich, mit –1 für die letzte vorhandene Stufe einer Spalte, –2 die vorletzte usw. Es sind auch Angaben in der Form COL(1...3) oder COL(2... –1) für mehrere Stufen möglich.

Im Statement

```
STYLE=COL(-1):R90,COL(1):R0
```

dreht COL(–1):R90 die Spaltenüberschriften der letzten Stufe um 90 Grad.
COL(1):R0 dreht die Texte der ersten Stufe um 0 Grad, gibt sie also normal aus. Da diese Angabe hinter COL(–1):R90 steht, macht sie die Drehung für einstufige Texte rückgängig.

Treffen für eine Textstufe mehrere Angaben zu, so wird die am weitesten rechts stehende wirksam. So verwendet das Statement

```
STYLE=COL(1):FT5,COL(-1):FT2
```

für eine einstufige Spaltenüberschrift den Font FT2.

ROW Aufbereitungsregeln für die Zeilentexte. So gibt

```
STYLE=ROW:FT3
```

alle Zeilentexte in der Schrift FT3 aus. Font-Angaben beschränken sich auf die Zeilentexte, die Hintergrundfarben wirken bis in den Zählerbereich hinein.

Eine Angabe (*s*) hinter ROW begrenzt die Aufbereitungsregeln auf einzelne Textstufen in den Zeilentexten. Dabei ist *s* eine Stufennummer 1 ... 20 mit 1 für die oberste Textstufe, 2 für die zweithöchste usw. Weiter sind die Angaben –1 ... –20 möglich, mit –1 für die letzte vorhandene Stufe einer Zeile, –2 die vorletzte usw.

So sorgt die Angabe

```
STYLE=ROW(1):FT5,ROW(2..-1):FT1
```

für die folgenden Zeilentexte:

Basis
Geschlecht
männlich
weiblich

Dabei legt ROW(1):FT3 die Schrift FT3 für die oberste Textstufe fest. ROW(2..-1):FT1 weist allen Texten von der zweiten bis zur letzten Stufe die Schrift FT1 zu.

Treffen für eine Textstufe mehrere Angaben zu, so wird die am weitesten rechts stehende wirksam. Zum Beispiel erzeugt

```
STYLE=ROW(1):FT5,ROW(-1):FT2
```

die Textzeilen

Basis
Geschlecht
männlich

Die direkt hinter STYLE= angegebenen allgemeinen Aufbereitungsregeln gelten für alle Tabellenteile, die speziellen Aufbereitungsregeln nur für die davor stehende Komponente. Folgende Regeln sind verfügbar:

< | **:KEEP** | **:FIT** | **:FITM** | **:TRUNC** | > **<:UNDO>** **<:FT**i> < | **:L** | **:R** | **:C** | > < | **:T** | **:B** | **:M** | > < | **:G**i | **:CO**i | > **<:R**g> **<:I**e>

KEEP Mit dieser Angabe werden die Textzeilen genau so ausgegeben, wie sie in den Statements eingegeben wurden. Zu lange Zeilen werden bei PAGE abgeschnitten und am Zeilenende mit den Zeichen *) markiert oder bei PAGEP in voller Länge über andere Tabellenteile hinweg gedruckt. Dazu wird eine Fehlermeldung 6xx ausgegeben.

FIT Bei dieser Angabe trennt CNTA zu lange Zeilen in mehrere Zeilen auf.
Die Auftrennung erfolgt an Wortgrenzen. Eine maschinelle Silbentrennung findet nur statt, wenn ein einzelnes Wort länger als eine Zeile ist.
FIT ist die Voreinstellung, wenn STYLE oder STYLES fehlen

Falls zum Beispiel der Text

```
'Schleswig-Holstein'/'Niedersachsen'/'Hamburg'
```

nicht in eine Spalte passt, könnte mit STYLE=FIT daraus werden:

```
Schleswig-
Holstein
Niedersach-
sen
Hamburg
```

Da Programme gelegentlich fehlerhaft trennen, lassen sich mit dem Zeichen _ Trennstellen in den Worten vorgeben. Lieferte zum Beispiel das Programm die Trennung

```
Nied-
ersachsen
```

so wird durch Eingabe von

```
'Nieder_sachsen'
```

die Trennung

```
Nieder-
sachsen
```

erreicht. Falls die Spalte breit genug ist, wird das Wort nicht getrennt und ohne _ ausgegeben.
Das Zeichen _ vor einem Wort verhindert die Trennung des Wortes vollständig.

Zum Beispiel erreicht man mit

```
'Niedersachsen _Hamburg'
```

dass das Wort

```
Hamburg
```

stets ohne Trennung ausgegeben wird.

FITM Diese Angabe arbeitet wie FIT. Zusätzlich sind Silbentrennungen auch dann erlaubt, wenn mehrere Worte in einer Zeile stehen. Damit lässt sich besonders in den Tabellenköpfen Platz sparen.

TRUNC Diese Angabe ist gleichwertig mit KEEP, jedoch werden zu lange Texte an den Zeilenenden abgeschnitten. Dazu erscheint keine Fehlermeldung 6xx.

UNDO Diese Angabe formt die Texte vor der Ausgabe zu einer einzigen Zeile um.
Leerzeilen und Zeilen, die nur aus Sonderzeichen bestehen, werden entfernt. Doppelte Leerzeichen werden beseitigt und Silbentrennungen rückgängig gemacht. Zum Beispiel entsteht aus den Eingabezeilen

```
/'Schleswig-'/'Holstein'//'  Nieder-'/'sachsen  '/
'Hamburg  Bremen'/'------------------'
```

die Textzeile

```
'Schleswig-Holstein Niedersachsen Hamburg Bremen'.
```

Ohne die Angabe UNDO bleiben Leerzeilen, Leerzeichen und Zeilenübergänge erhalten.

L R C Diese Angaben richten die Texte für BOTTOM, COL, ROW, STUBHEAD und TITLE horizontal aus:
L = linksbündig, R = rechtsbündig, C = zentriert.
Zum Beispiel stellt

```
STYLE=COL:C,COL(-1)R
```

die Texte aus COL mittig in die Spalten, die letzten Texte jeder Spalte werden jedoch nach rechts ausgerichtet.

T B M Diese Angaben richten die Texte für COL und STUBHEAD vertikal aus:
T = oben, B = unten, M = mittig.
Zum Beispiel stellt

```
STYLE=COL:M,COL(-1):B
```

die Texte aus COL mittig in die frei verfügbare Höhe der Köpfe, die letzten Texte jeder Spalte werden jedoch nach unten ausgerichtet.

I*e* Mit dieser Angabe hinter ROW werden die durch FIT oder FITM erzeugten Fortsetzungszeilen eingerückt: Die erste Zeile des Textes wird normal ausgegeben, alle Fortsetzungszeilen beginnen dagegen um die Distanz *e* nach rechts verschoben.
Ist das Statement PAGE wirksam, so ist für *e* die Anzahl Zeichen anzugeben, um die die Fortsetzungszeilen einzurücken sind. Bei PAGEP ist der Einzug *e* in Millimetern oder in einer der anderen Maßeinheiten anzugeben, die beim Statement PAGEP beschrieben sind.
Zum Beispiel könnte

```
STYLE=ROW:FIT:I2
```

aus der Variablen

```
-C1:2 1='selbständig, freiberuflich' 2='leitender Angestellter'
```

die folgende Zeilentexte erzeugen:

```
selbständig,
  freiberuflich
leitender
  Angestellter
```

FT*i* Diese Angabe legt die Schriftart FT1 ... FT16 aus PAGEP fest, in der die verschiedenen Tabellenteile zu drucken sind.

Hinter dem Statement PAGE ist diese Angabe wirkungslos.

Die hier angegebenen Schriften werden immer dann verwendet, wenn keine abweichenden Angaben in den Textzeilen und aus der erweiterten Druckaufbereitung hinter ROW, COL oder FILTER vorliegen.

Fehlen die Schriftangaben FTi aus STYLE, so wird mit der Voreinstellung FT4 für TITLE und FT2 für alle anderen Tabellenteile gearbeitet.

Gi
COi Diese Angabe unterlegt die Tabellenteile COL, ROW, NUM oder STUBHEAD mit dem Grauton G1 bis G16 oder der Farbe CO1 bis COi aus PAGEP.
Hinter dem Statement PAGE ist diese Angabe wirkungslos.

Mit den Angaben INSERT und PRINT sowie durch die erweiterte Druckaufbereitung hinter ROW und COL lassen sich ebenfalls Grautöne und Farben ausgeben.

Rg Mit Hilfe dieser Angabe können die Kopftexte einer Tabelle gedreht werden.
Sie ist nur bei COL und STUBHEAD möglich und nur dann wirksam, wenn das Statement PAGEP aktiv ist. Es können folgende Angaben gemacht werden:

R90: Dreht die Texte um 90 Grad gegen den Uhrzeigersinn, Leserichtung von unten nach oben.
R-90: Dreht die Texte um 90 Grad im Uhrzeigersinn, Leserichtung von oben nach unten.
R0: Sorgt für normale, waagerechte Ausgabe der Texte.

Durch

```
STYLE=COL(-1):R90,COL(1):R0
```

werden die Kopftexte der letzten Stufe wegen COL(-1) : R90 um 90 Grad gedreht, die Texte der höheren Textstufen werden unverändert waagerecht ausgegeben. Besitzt eine Spalte nur eine Textstufe, so wird deren Text wegen der letzten Angabe COL(1) : R0 nicht gedreht.

Die Angaben FIT und FITM hinter STYLE verlieren ihre Wirkung bei R90 und R-90. Wegen der Rotation sind hier keine Spaltenbreiten wirksam, in die der Text eingepasst werden könnte.

Beispiel: Tabelle mit Tonflächen

```
XTAB COLS=TOTAL:'Total';C20
     ROWS=TOTAL:'Basis';C1180
     LINES=L11
     STYLE=G2,COL:CO16,COL(1):FT3,
              ROW:CO12,ROW(1):FT3,
              STUBHEAD:G4
```

Dieses Statement kann folgende Tabelle erzeugen:

	Total	Alter							
		Von 18 bis 20 Jahre	Von 21 bis 25 Jahre	Von 26 bis 30 Jahre	Von 31 bis 35 Jahre	Von 36 bis 40 Jahre	Von 41 bis 50 Jahre	Von 51 bis 60 Jahre	Über 60 Jahre
Basis	4594	3	238	1101	1203	902	926	200	18
F18: Erwartete Gehaltsentwicklung									
Steigen	1627	2	94	441	444	297	290	54	5
Stagnieren	2839	1	135	644	736	578	593	136	13
Sinken	128	-	9	16	23	27	43	10	-

Die Angabe STYLE=G2 unterlegt zunächst die ganze Tabelle mit dem Grauton G2.

COL:CO16 unterlegt alle Kopftexte mit der Farbe CO16 und COL(1):FT3 gibt die Kopftexte der obersten Stufe in fetter Schrift aus.

ROW:CO12 unterlegt alle Zeilentexte mit der Farbe CO12 und ROW(1):FT3 gibt die Zeilentexte der obersten Stufe in Fetter Schrift aus.

STUBHEAD:G4 färbt den Kopf über den Zeilentexten in dem Grauton G3.

LINES=L11 erzeugt die weißen Linien innerhalb und um die Tabelle.

Beispiel: Gedrehte Kopftexte

```
XTAB ROW=TOTAL;C2030
     COL=TOTAL;C1060.1;...12
     STYLE=COL(-1):R90:B
     TAB=40,10,8
```

Dieses Statement kann folgende Tabelle erzeugen:

	TOTAL	F6: Wirtschaftszweig											
		Architektur-, Ingenieurbüros	Automobilindustrie	Banken und Finanzen	Bergbau	Bildung und Erziehung	Chemische Industrie	Consulting	EDV	Einzelhandel	Energie / Versorger	Facility Management	Gastronomie
TOTAL	4594	111	258	327	5	88	105	188	325	94	103	24	44
Schulabschluss													
Volksschule, Hauptschule	110	1	7	1	-	1	4	1	8	4	4	-	4
Weiterführende Schule ohne Abitur	621	7	29	49	-	1	22	14	63	21	8	5	12
Abitur	579	4	15	56	2	3	11	15	51	15	7	2	12
Studium ohne Abschluss	240	2	4	10	-	4	4	12	31	3	1	4	2
Studium mit Abschluss	2998	97	195	210	3	79	64	144	170	50	83	13	14
kein Abschluss	46	-	8	1	-	-	-	2	2	1	-	-	-

Hinter STYLE dreht COL(-1):R90 die Texte der untersten Textstufe im Tabellenkopf um 90 Grad. Die Angabe B dahinter richtet diese Texte nach unten aus.

Die Angabe TAB=40,10,8 sorgt für die schmalen Spalten der Tabelle.

Beispiel: Verschiedene Schriften

```
PAGEP FONT15='Lucida Handwriting',N,12p,G9
      FONT7=H,I,9p,CO14
XTAB STYLE=TITLE:FT15,
           FILTER:FT5,
           STUBHEAD:FT4,
           BOTTOM:FT10,
           COL:FT9,COL(1):FT3,
           ROW:FT7,ROW(1):FT3,
           NUM:FT7
     COLS=total:'Gesamt';C2010
     ROWS=total:'Basis-Fälle = 100%';c2020
     TITLE='Alter des ~U~Haushaltsvorstands'
     STUBHEAD='~B~Struktur~E~ in Prozent'
     BOTTOM='Es wurde nur die ~I~zweite Welle~E~ ausgewertet'
     PRINT=R(1)ABS R(2...4,6..-1)PROW1,1,% R(5)PROW1,1,%,FT8
```

Diese Statements können folgende Tabelle erzeugen:

Alter des Haushaltsvorstands

TOTAL **4594**

Struktur in Prozent	Gesamt	S1: Geschlecht	
		Weiblich	Männlich
Basis-Fälle = 100%	4594	992	3602
S2: Alter			
Von 18 bis 20 Jahre	0,1%	0,1%	0,1%
Von 21 bis 25 Jahre	5,2%	7,7%	4,5%
Von 26 bis 30 Jahre	24,0%	33,2%	21,4%
Von 31 bis 35 Jahre	**26,2%**	**25,0%**	**26,5%**
Von 36 bis 40 Jahre	19,6%	15,3%	20,8%
Von 41 bis 50 Jahre	20,2%	14,8%	21,6%
Von 51 bis 60 Jahre	4,4%	3,5%	4,6%
Über 60 Jahre	0,4%	0,1%	0,5%

Es wurde nur die *zweite Welle* ausgewertet

Zunächst werden im Statement PAGEP zwei Schriften umdefiniert:
Für FT15 wird die Schrift *Lucida Handwriting* ausgewählt. N sorgt für normalen Schriftstil, 12p für die Schriftgröße 12 Punkt und G9 für einen Grauton als Schriftfarbe.
Der Schrift FT7 wird mit H die Schrift Helvetica zugewiesen. Durch I wird sie kursiv, wegen 9p in der Schriftgröße 9 Punkt und CO14 macht Blau zur Schriftfarbe.

Im Statement XTAB werden den einzelnen Tabellenteilen mit STYLE verschiedene Schriften zu gewiesen. Insbesondere erhält TITLE die modifizierte Schrift FT15 sowie ROW und NUM die blaue Schrift FT7. COL(1):Ft3 und ROW(1):FT3 ordnen den obersten Textstufen der Kopf- und Zeilentexte die Schrift FT3 zu.

Hinter PRINT sorgt R(5) PROW1,1,%,FT8 in den Zählergebnissen der Tabellenzeile 5 für die abweichende Schrift FT8.

Innerhalb der Texte werden einzelne Worte mit ~U~ unterstrichen, mit ~I~ kursiv und mit ~B~ in fetter Schrift ausgegeben. (Siehe Abschnitt Textzeilen).

XTAB: SUPPRESS

SUPPRES **SUPPRESS**	**=**	Tabellenelemente **NO**

Die Ausgabe der unten aufgeführten Tabellenelemente lässt sich durch SUPPRESS unterdrücken.
Ohne sie werden alle Tabellenelemente ausgegeben.

Zum Beispiel verhindert

```
SUPPRESS=FILTER,SPACES
```

die Ausgabe der Filtertexte und Filterwerte und beseitigt die Leerzeilen zwischen den einzelnen Zählerzeilen. Die Angabe

```
SUPPRESS=EMPTY
```

in einem späteren XTAB-Statement unterdrückt nur leere Tabellenzeilen. Die Filterangaben und Leerzeilen werden wieder ausgegeben.

SUPPRESS Diese Angabe gilt für das laufende und alle folgenden XTAB-Statements, bis sie durch ein neues SUPPRESS ersetzt wird.

SUPPRES Diese Angabe gilt nur für das laufende XTAB-Statement, und sie hat Vorrang vor einer Angabe SUPPRESS aus einem voranstehenden XTAB-Statement.

Folgende Tabellenelemente lassen sich unterdrücken werden:

COLTEXT die Spaltenüberschriften und der Text STUBHEAD werden nicht gedruckt.

EMPTY die Zeilen, deren Werte alle Null sind, werden nicht gedruckt.

FOOTNOTES
die Fußnotentexte werden nicht gedruckt. Es handelt sich hier um Fußnoten, die durch die Angabe Fi hinter ROW, COL, FILTER und PRINT erzeugt oder einzelnen Variablen bei der Definition hinter DEFS fest zugeordnet wurden.

FILTER der Filterwert und die Filtertexte werden nicht gedruckt.

FCOL die erste Zählerspalte der Tabelle wird nicht gedruckt. Dies ist, anders als bei SUPPRESS=COLS(...), die erste Spalte *vor* eventueller Sortierung der Tabelle durch SORT.

FROW die erste Zählerzeile der Tabelle wird nicht gedruckt. Dies ist, anders als bei SUPPRESS=ROWS(...), die erste Zeile *vor* eventueller Sortierung der Tabelle durch SORT.

FVAL der Filterwert wird nicht gedruckt, wohl aber die Filtertexte.

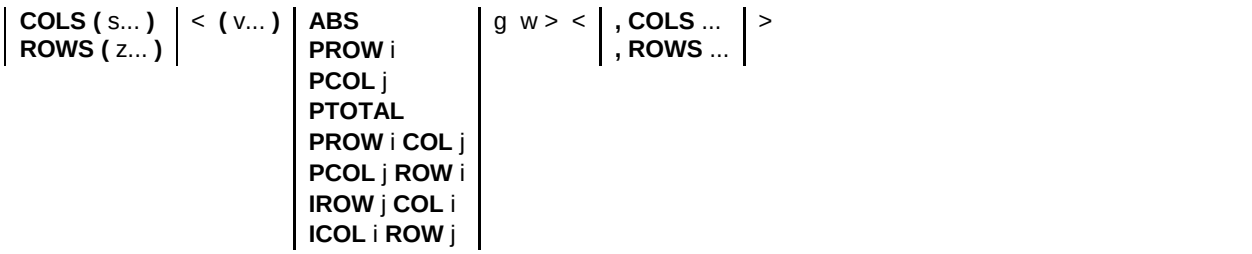
LCOL die letzte Zählerspalte der Tabelle wird nicht gedruckt. Dies ist, anders als bei SUPPRESS=COLS(...), die letzte Spalte *vor* eventueller Sortierung der Tabelle durch SORT.

LROW die letzte Zählerzeile der Tabelle wird nicht gedruckt. Dies ist, anders als bei SUPPRESS=ROWS(...), die letzte Zeile *vor* eventueller Sortierung der Tabelle durch SORT.

NO setzt die Angabe SUPPRESS aus früheren XTAB-Statements außer Kraft.
Das bedeutet, dass alle unterdrückbaren Komponenten wieder ausgegeben werden.

STUBS die Zeilentexte und der Text STUBHEAD werden nicht gedruckt.

SPACES die Leerzeilen zwischen den Zählerzeilen werden nicht gedruckt.



Die hinter COLS in Klammern aufgeführten Tabellenspalten (s...) werden nicht gedruckt. Entsprechend werden mit ROWS die Tabellenzeilen (z ...) nicht gedruckt.
Die Werte dieser unterdrückten Zeilen und Spalten lassen sich trotzdem zum Sortieren und als Prozentuierungsbasis verwenden. Zum Beispiel verhindert

```
SUPPRESS=ROWS(1,5..6)
```

die Ausgabe der Zeilen 1, 5 und 6.

Das Unterdrücken von Zeilen und Spalten lässt sich auch von Tabellenwerten abhängig machen. Die Angabe

```
SUPPRESS=COLS(1),COLS(5..6)(1)ABS<20
```

zum Beispiel unterdrückt stets die Spalte 1 und zusätzlich die Spalten 5 bis 6, wenn deren absolute Zählerwerte in Zeile 1 kleiner als 20 sind.

(s... **)** Es sind die Nummern 1 ... 32 767 der zu entfernenden Spalten anzugeben, durch Kommas voneinander getrennt. Von- und Bis-Angaben sind in der Form 1 . . 5 möglich, Aufzählungen als 7, 9, 12 und Klassenangaben A ... Z, sofern in COL einzelne Spalten mit A{ } ... Z{ } gekennzeichnet wurden.
Auch negative Werte sind zulässig: -1 ist die letzte Spalte der Tabelle, -2 die vorletzte usw. So entfernt das Statement

```
SUPPRESS=COLS(2..5,9..-1)
```

die Spalten 2 ... 5 und ab Spalte 9 den Rest der Tabelle. Durch die Statements

```
COL=TOTAL;A{C1.1;...5};B{TOTAL;C2}
SUPPRESS=COLS(1,B)
```

werden die erste Spalte und alle Spalten der Klasse B unterdrückt.

Die Spaltennummern (s...) werden *nach* eventueller Sortierung der Tabelle verstanden.
Zum Beispiel werden mit den Angaben

```
SORT=COLS1
SUPPRESS=COLS(4..-1)
```

alle Spalten der Tabelle zunächst nach den Werten der Zeile 1 sortiert. Danach werden ab der vierten Spalte alle Spalten unterdrückt, sodass nur die drei Spalten mit den größten Werten erscheinen.

(z... **)** Es sind die Nummern 1 ... 32 767 der zu entfernenden Zeilen anzugeben, durch Kommas voneinander getrennt. Von- und Bis-Angaben sind in der Form 1 . . 5 möglich, Aufzählungen als 7, 9, 12 und Klassenangaben A ... Z, sofern in ROW einzelne Zeilen mit A{ } ... Z{ } gekennzeichnet wurden.
Auch negative Werte sind zulässig: - 1 ist die letzte Zeile der Tabelle, - 2 die vorletzte usw.
So entfernt das Statement

```
SUPPRESS=ROWS(2..5,9..-1)
```

die Zeilen 2 ... 5 und ab Zeile 9 alle restlichen Zeilen der Tabelle.

Durch die Statements

```
ROW=TOTAL;A{C1.1;...5};B{TOTAL;C2}
SUPPRESS=ROWS(1,B)
```

werden die erste Zeile und alle Zeilen der Klasse B unterdrückt.

Die Zeilennummern (z...) werden *nach* eventueller Sortierung der Tabelle verstanden.
Zum Beispiel werden mit den Angaben

```
SORT=ROWS1
SUPPRESS=ROWS(4..-1)
```

alle Zeilen der Tabelle zunächst nach den Werten der Spalte 1 sortiert. Danach werden ab der vierten Zeile alle Zeilen unterdrückt, sodass nur die drei Zeilen mit den größten Werten erscheinen.

(v... **)** Mit dieser Angabe werden Zeilen oder Spalten abhängig von bestimmten Tabellenwerten unterdrückt.
So verhindert zum Beispiel

```
SUPPRESS=ROWS(2..-1)(1)ABS<20
```

ab der zweiten Zeile die Ausgabe aller Zeilen, deren absolute Zählerwerte in Spalte 1 kleiner als 20 sind.
Von- und Bis-Angaben sind für (v...) in der Form (1 . . 5) möglich, Aufzählungen als (7, 9, 12) und Klassenangaben A ... Z aus ROW oder COL. Die Zeilen- und Spaltennummern (v...) sind die Nummern *vor* eventueller Sortierung der Tabellen durch SORT.
Sind für (v...) mehrere Zeilen oder Spalten angegeben, so wird die Ausgabe nur unterdrückt, wenn die dahinter stehende Bedingung für alle Zeilen oder Spalten aus (v...) erfüllt ist.

ABS Mit dieser Angabe werden die Absolutwerte der Tabelle mit dem festen Zahlenwert w verglichen.

PROW i Die Tabellenwerte werden nach Prozentuierung auf die Zeile i mit dem festen Zahlenwert w verglichen.

PCOL j Die Tabellenwerte werden nach Prozentuierung auf die Spalte j mit dem festen Zahlenwert w verglichen.

PTOTAL Die Tabellenwerte werden nach Prozentuierung auf den Filterwert mit dem festen Zahlenwert w verglichen.

PROWi**COL**j Die Tabellenwerte werden als Prozentwerten aufbereitet und danach mit dem festen Zahlenwert w
PCOLj**ROW**i verglichen. Dabei ist i die Basiszeile und j die Basisspalte.

IROWi**COL**j Die Tabellenwerte werden als Indexwerte von Prozentwerten aufbereitet und danach mit dem
ICOLj**ROW**i festen Zahlenwert w verglichen. Dabei ist i die Basiszeile und j die Basisspalte.
Siehe dazu auch im Abschnitt *XTAB PRINT* die Angaben IROW... und ICOL...

g An der Stelle von g ist einer der folgenden Vergleichsoperatoren anzugeben:

<	kleiner
>	größer
=	gleich
<=	kleiner oder gleich
>=	größer oder gleich
¬=	ungleich (stattdessen kann auch ^= geschrieben werden)
¬<	nicht kleiner (stattdessen kann auch ^< geschrieben werden)
¬>	nicht größer (stattdessen kann auch ^> geschrieben werden)

w Der feste Vergleichswert w ist eine Zahl aus maximal neun Ziffern und bis zu acht Nachkommastellen. Er darf auch ein Minuszeichen als Vorzeichen besitzen.
Zum Beispiel bewirkt die Angabe

```
SUPPRESS=COLS(2..-1)(15..17)PROW1<2.5
```

dass ab der zweiten Spalte alle Spalten der Tabelle unterdrückt werden, die in den Zeilen 15 bis 17 nur Prozentwerte kleiner als 2,5 enthalten. Als Prozentuierungsbasis wird dafür die erste Tabellenzeile verwendet. Der Vergleich mit dem Wert 2,5 findet vor eventueller Rundung der Prozentwerte für die Druckausgabe statt.
Durch die Angabe

```
SUPPRESS=COLS(1..-1)(1)ABS=0
```

werden alle Spalten der Tabelle unterdrückt, die in der ersten Zeile Nullen enthalten.

TABLES g w

Mit TABLES wird die Ausgabe ganzer Tabellen verhindert, wenn ihr Filterwert die dahinter stehende Vergleichsbedingung erfüllt.

g An der Stelle von g ist einer der folgenden Vergleichsoperatoren anzugeben:

<	kleiner
>	größer
=	gleich
<=	kleiner oder gleich
>=	größer oder gleich
¬=	ungleich (stattdessen kann auch ^= geschrieben werden)
¬<	nicht kleiner (stattdessen kann auch ^< geschrieben werden)
¬>	nicht größer (stattdessen kann auch ^> geschrieben werden)

w Der feste Vergleichswert w ist eine Zahl aus maximal neun Ziffern und bis zu acht Nachkommastellen. Er darf auch ein Minuszeichen als Vorzeichen besitzen.
Zum Beispiel bewirkt die Angabe

```
SUPPRESS=TABLES < 50
```

dass diejenigen Tabellen nicht ausgegeben werden, deren Filterwerte kleiner als 50 sind.

VARLABELF
VARLABELC
VARLABELR
VARLABELS

Diese Angaben verhindern die Ausgabe der Variablentexte von C-Variablen in den Tabellen:

VARLABELF unterdrückt die Variablentexte im Filter
VARLABELC unterdrückt die Variablentexte in den Tabellenköpfen
VARLABELR unterdrückt die Variablentexte in den Zeilen
VARLABELS unterdrückt die Variablentexte überall

Beispiel

```
XTAB  COL=TOTAL;C1
      ROW=TOTAL;C2.1;..3
```

Dieses Statement ohne die Angabe SUPPRESS könnte folgende Tabelle erzeugen:

TOTAL 24

	TOTAL	C1.1	C1.2	C1.3
TOTAL	24	7	4	13
C2.1	-	-	-	-
C2.2	9	3	2	4
C2.3	10	4	2	4

```
XTAB   COL=TOTAL;C1
       ROW=TOTAL;C2.1;..3
       SUPPRESS = FILTER, SPACES, EMPTY, STUBS
```

Dieses Statement ist, mit Ausnahme der Angabe SUPPRESS, identisch mit dem vorigen XTAB. Die entsprechende Tabelle sieht dann folgendermaßen aus:

TOTAL	C1.1	C1.2	C1.3
24	7	4	13
9	3	2	4
10	4	2	4

Durch SUPPRESS=FILTER wird die Zeile mit den Filter-Angaben über der Tabelle entfernt.

SUPPRESS=SPACES beseitigt die Leerzeilen zwischen den Zählerzeilen.

SUPPRESS=EMPTY verhindert die Ausgabe der Zeile zu C2.1, da diese nur die Zählerwerte 0 enthält.

SUPPRESS=STUBS schließlich unterdrückt die Ausgabe der Zeilentexte.

XTAB: TAB

TAB	=	b, b ...
TABS		't' b , 't' b ...

Diese Angabe legt die Breite der Tabellenspalten fest. Zum Beispiel sorgt

```
TAB=30,20,15
```

dafür, dass die erste Tabellenspalte zur Aufnahme der Zeilentexte 30mm breit ist, die zweite Spalte 20mm und alle folgenden 15 mm. Diese Maßangaben gelten, wenn das Statement PAGEP aktiv ist. Hinter dem Statement PAGE sind dies 30, 25 und 15 Zeichen.

Jede durch ein Komma abgetrennte Angabe gilt für eine Tabellenspalte. Bis zu 128 Spalten können hier beschrieben werden. Die jeweils letzte Angabe gilt für alle noch folgenden Spalten. Es müssen daher nur so viele Werte angegeben werden, wie unterschiedliche Spalten vorliegen.

Die TAB-Angaben sind den entsprechenden Spaltenangaben aus COL fest zugeordnet.
Werden die Tabellenspalten mit Hilfe von SORT=COLSi umsortiert, so werden die TAB-Angaben entsprechend mitsortiert.

Fehlen die Angaben TAB oder TABS, so wird bei Ausgabe nach PAGE mit den Werten TABS = 19, 14 gearbeitet und bei Ausgabe nach PAGEP mit TABS = 65mm, 18mm.

TABS Diese Angabe liefert Werte für das laufende und alle folgenden XTAB-Statements, bis sie durch neue TABS ersetzt wird.

TAB Diese Angabe gilt nur für das laufende XTAB-Statement.
Eventuelle Angaben TABS aus früheren XTAB-Statements werden für das laufende XTAB unwirksam.

b Breite der Tabellenspalte.
Ist das Statement PAGE wirksam, so ist die Breite als Anzahl Zeichen 3 ... 128 anzugeben.
Bei Ausgabe mit PAGEP ist die Breite in 5 ... 250 Millimeter anzugeben. Auch die anderen bei PAGEP beschriebenen Maßeinheiten sind möglich.
Die hier angegebene Breite muss den Platz für eventuelle Trennzeichen oder Trennlinien am Anfang der Spalte enthalten.

't' Trennzeichen am Spaltenanfang.
Hier ist *t* ein einzelnes Zeichen, das am Anfang der jeweilige Spalte als Randzeichen zu drucken ist. Diese Angabe ist nur wirksam, wenn das Statement PAGE wirksam ist.
CNTA besitzt als Grundeinstellung vorgegebene Randzeichen.
Diese können durch die Angabe LCHARS im Statement PAGE verändert werden.
Die Angabe *'t'* hat Vorrang vor der Grundeinstellung und vor den Randzeichen LCHARS.

Beispiel

```
XTAB  COL=TOTAL;C1.1;..3
      ROW=TOTAL;C2.1;..3
      TAB=12,10,8,' '8,' '8,8
```

Dieses Statement könnte die folgende Tabelle erzeugen:

	TOTAL	C1.1	C1.2	C1.3
TOTAL	2401	698	421	1312
C2.1	517	-	-	241
C2.2	889	3	2	433
C2.3	1022	4	2	389

Dabei haben die Angaben zu TAB folgende Wirkung:

Der Wert 12 sorgt dafür, dass die erste Tabellenspalte für die Zeilentexte 12 Druckpositionen breit ist. Da kein Randzeichen angegeben wurde, wird als linke Begrenzung der ersten Tabellenspalte das Standardzeichen verwendet, hier der senkrechte Strich. Das Standard-Randzeichen lässt sich im Statement PAGE mit der Angabe LCHARS verändern.
Das Begrenzungszeichen wird stets in der Spaltenbreite mitgezählt.

Der folgende Wert 10 sorgt dafür, dass die zweite Spalte für die Angaben TOTAL aus 10 Druckpositionen besteht. Da das Randzeichen fehlt, wird als linke Begrenzung dieser Spalte wieder der senkrechte Strich verwendet.

Der nächste Wert 8 liefert 8 Druckstellen für die Spalte C1.1 mit dem senkrechten Strich als linkes Randzeichen.

Die beiden folgenden Angaben ' '8 geben den Spalten C1.2 und C1.3 eine Breite von 8 Druckstellen. Als Randzeichen ist, in Hochkommas eingeschlossen, eine Leerstelle angegeben, die anstelle des senkrechten Striches ausgegeben wird.

Die letzte Angabe 8 bei TAB liefert die Spaltenbreite für alle eventuell noch folgenden Spalten. Weiter sorgt sie dafür, dass wieder der senkrechte Strich als linkes Randzeichen verwendet wird. Diese Angabe sorgt insbesondere für den senkrechten Strich am rechten Tabellenrand.

Abkürzungen

Wenn in TAB direkt hintereinander mehrmals die gleichen Spaltenangaben verwendet werden, so ist die Angabe nur einmal erforderlich, mit der gewünschten Anzahl dahinter in Klammern.

Statt: TAB=20, 10, 10, 10, 10, 8
genügt TAB=20, 10(4), 8

Statt: TAB=20, 10, ' '10, ' '10, ' '10, 8
genügt: TAB=20, 10, ' '10(3), 8

XTAB: TITLE

```
TITLE  | = | 't'  <, | L | > |
TITLES |   |         | R |   |
                      | C |
               | NO            |
```

Diese Angabe liefert Überschriftstexte, die über den Tabellen vor den Filterangaben gedruckt werden.

TITLES Diese Angabe liefert Texte für das laufende und alle folgenden XTAB-Statements, bis sie durch ein neues TITLES ersetzt wird.

TITLE Diese Angabe gilt nur für das laufende XTAB-Statement.
Eventuelle Angaben TITLES aus früheren XTAB-Statements werden für das laufende XTAB unwirksam.

't' Der ein- oder mehrzeilige Text, der als Titel gedruckt werden soll; wie im Abschnitt *Textzeilen* beschrieben.

L R C Mit diesen Angaben wird der Text über der Tabelle ausgerichtet:
L = linksbündig, R = rechtsbündig, C = zentriert.
Zum Beispiel stellt

```
TITLE=' Frage 24' , C
```

die Überschrift mittig über die Tabelle.
Fehlt die Angabe L, R oder C, so wird der Text linksbündig ausgegeben.

NO setzt die Angabe TITLES aus früheren XTAB-Statements außer Kraft.

Beispiel

```
XTAB  ROW=TOTAL;C3
      COL=TOTAL;C2
      SUPPRESS=FILTER,SPACES
      TITLE='Frage 25: Welche dieser Artikel haben Sie'/
            'in den vergangenen 2 Wochen gekauft?'
```

Dieses Statement könnte folgende Tabelle erzeugen:

Frage 25: Welche dieser Artikel haben Sie
in den vergangenen 2 Wochen gekauft?

	TOTAL	C2.1	C2.2	C2.3
TOTAL	24	5	9	10
C3.1	5	5	-	-
C3.2	9	-	9	-
C3.3	10	-	-	10

XTAB: TMARK

TMARK
TMARKS | **=** ' Markierungszeichen '

Für t-Tests werden die Zeilen oder Spalten der Tabellen mit den Buchstaben (a) (b) (c) ... (z) markiert, um auf signifikante Unterschiede hinweisen zu können. Mit der Angabe TMARK lassen sich andere Zeichen in frei wählbarer Reihenfolge für diesen Zweck vorgeben.

TMARKS Diese Angabe gilt für das laufende und alle folgenden XTAB-Statements, bis sie durch eine neue TMARK-Angabe ersetzt wird.

TMARK Diese Angabe gilt nur für das laufende XTAB-Statement.
Eventuelle TMARK-Angaben aus früheren XTAB-Statements werden für das laufende XTAB unwirksam.

Markierungszeichen

Hier lassen sich bis zu 200 Zeichen zur Markierung der Zeilen oder Spalten vorgeben. Sie werden in der hier angegebenen Reihenfolge verwendet. Zum Beispiel:

```
XTAB TMARK='zyxwvuäöüßαβγδ'
```

Dabei ist zu beachten, dass PRINT=SIG mit mehreren Signifikanzschwellen die schärferen Bedingungen durch Großbuchstaben kennzeichnet. Deshalb sollten hier nur Kleinbuchstaben angegeben werden, die sich von ihrem Gegenstück in Großschrift unterscheiden.

XTAB: VTAB

VTAB **VTABS**	=	n s n **NO**

Diese Angabe sorgt dafür, dass Tabellen mit Randlinien auf einer vorgegebenen Höhe enden, auch wenn sie weniger Zeilen mit Zählergebnissen besitzen. Dazu werden die Tabellen hinter der letzten Datenzeile mit Leerraum aufgefüllt.

VTABS Diese Angabe legt das Tabellenende für das laufende und alle folgenden XTAB-Statements fest, bis sie durch ein neues VTABS ersetzt wird.

VTAB Diese Angabe gilt nur für das laufende XTAB-Statement.
Eventuelle Angaben VTABS aus einem früheren XTAB-Statement werden für das laufende XTAB unwirksam.

n Mit der Angabe *n* wird die Höhe der Tabelle festgelegt. Zum Beispiel endet eine Tabelle mit
VTAB=60
60mm unterhalb der Blattüberschriften.

Bei Ausgabe auf Seitendruckern durch das Statement PAGEP ist *n* als 1 ... 500 Millimeter oder in einer der anderen Maßeinheiten anzugeben, die beim Statement PAGEP beschrieben sind.
Wird die Tabelle über das Statement PAGE auf Zeilendruckern ausgegeben, so ist die Höhe *n* als 1 ... 32 767 Zeilen anzugeben.

Besitzt eine Tabelle mehr Zeilen als die Angabe VTAB erlaubt, so wird VTAB außer Kraft gesetzt und die Tabelle normal bis zu ihrem Ende ausgegeben.
Ist *n* größer als die Seite, so wird der untere Tabellenrand am Seitenende ausgegeben.

s Mit der Angabe *s* wird die Position festgelegt, ab der die Höhenangabe *n* berechnet werden soll.
Diese Positionsangabe kann auch fehlen, dann wird ab dem Seitenanfang gerechnet.
Die folgenden drei Buchstaben sind hier möglich:

S Die Höhe *n* des Tabellenendes wird vom Seitenanfang aus gerechnet. Dies ist die erste Position der Seite, in der eine Tabelle beginnen kann, also unterhalb eventueller Blattüberschriften und Randlinien. Zum Beispiel
```
VTAB=S60
```

E Die Höhe *n* des Tabellenendes wird vom Seitenende aus gerechnet. Dies ist die letzte Position der Seite, in der eine Tabelle enden kann, also oberhalb eventueller Randtexte oder Randlinien. Zum Beispiel
```
VTAB=E60
```

T Die Höhe n des Tabellenendes wird vom Tabellenanfang aus gerechnet.
TITLE- und FILTER-Angaben sind dabei in der Tabellenhöhe enthalten. Zum Beispiel
```
VTAB=T60
```

NO setzt die Angabe VTABS aus früheren XTAB-Statements außer Kraft.

Beispiel

```
XTAB   COL=TOTAL;C10
       ROW=TOTAL;C2030
       VTAB=T80
```

Dieses Statement kann folgende Tabelle erzeugen:

	Gesamt	Männer	Frauen
Gesamt	4594	992	3602
Schulabschluss			
Volksschule, Hauptschule	110	15	95
Weiterführende Schule ohne Abitur	621	155	466
Abitur	579	149	430

Die Angabe *VTAB=T80* sorgt für eine Gesamthöhe der Tabelle von 80mm.

XTAB: t-Tests zur Markierung signifikanter Unterschiede

Die Funktionen MTT... in ROW oder COL vergleichen die Werte innerhalb ihrer Zeile oder Spalte paarweise miteinander. Wertepaare die sich im Sinne des t-Tests signifikant voneinander unterscheiden, werden mit Buchstaben markiert. Zusätzlich zu den Signifikanzniveaus lassen sich Schwellenwerte für Effektstärken nach J. Cohen vorgeben.

Eine Tabelle mit t-Test-Markierungen könnte so aussehen:

	Gesamt (a)	Design (b)	Ausstattung (c)	Volumen (d)
Basis	1204	368	216	65
Alter				
bis 29 Jahre	19%	17%	d 23%	c 11%
60 Jahre und älter	d 8%	d 8%	d 9%	abc 2%

a ... d = signifikante Unterschiede auf 5%-Niveau

Dies sind die Statements dazu:

```
XTAB ROW=TOTAL:'Basis';MTT{C2.1;.4}
     COL=TOTAL:'Gesamt';C1
     PRINT=SIG,5
     PRINT1=PROW1,%
     BOTTOM='a...d = signifikante Unterschiede auf 5%-Niveau'
```

Die Funktionen MTT... in ROW oder COL vergleichen die Werte innerhalb ihrer Zeile oder Spalte paarweise miteinander. Wertepaare die sich im Sinne des t-Tests signifikant voneinander unterscheiden, werden mit Buchstaben markiert.

Die Angaben PRINT ...=SIG bestimmen die Position der Markierungen in den Tabellen sowie die Signifikanzschwelle für den t-Test.

Die Angaben MTT... dürfen innerhalb einer Staffelung erscheinen, aber nur auf der unteren Stufe. Zum Beispiel:

```
ROW={TOTAL;N10;C0}{MTT{TOTAL;C2.1;..6;MEAN(C2)}}
```

Sie dürfen aber auch eine ganze Staffelung umfassen:

```
ROW=MTT{{TOTAL;N10;C0}{TOTAL;C2.1;..6;MEAN(C2)}}
```

Die t-Tests lassen sich gleichzeitig auf mehreren Signifikanzniveaus durchführen. Zum Beispiel:

```
PRINT1=C(2...5)SIG,5,10
```

Damit werden Wertepaare, die die stärkere Bedingung auf dem 5-Prozent-Niveau erfüllen, durch Großbuchstaben A ... Z gekennzeichnet. Für die schwächere Bedingung auf dem 10-Prozent-Niveau dienen die Kleinbuchstaben a ... z. Zu diesem Zweck sind die Spalten mit (a) und (b) markiert. Mit Hilfe der Angabe TMARK im Statement XTAB lassen sich andere Zeichen hierfür vorgeben.

Sind nicht alle Werte einer Zeile oder Spalte miteinander zu vergleichen, so lässt sich der Wirkungsbereich von SIG durch davor stehende Angaben R... oder C... einschränken. Zum Beispiel bewirkt

```
PRINT=C(1..4)SIG,1 C(5..6)SIG,2
```

einen Signifikanz-Vergleich auf dem 1%-Niveau für die Spalten 1 bis 4 und einen Vergleich auf dem 2%-Niveau für die Spalten 5 bis 6.

Es ist auch möglich, die zu vergleichenden Werte einer Zeile oder Spalte in Gruppen einzuteilen, wobei die Werte der einen Gruppe mit denen der anderen verglichen werden, jedoch nicht die Werte innerhalb der Gruppen. Zum Beispiel vergleicht

```
PRINT1=C(2..3)SIG(4..5),1
```

die Werte der Spalten 2 und 3 mit denen der Spalten 4 und 5.

Folgende Varianten des t-Test stehen zur Verfügung:

MTT{MEAN(Nn) ... **N**n ... **MEAN(C**n**)** ... **C**n.a ... **C**n.**Z**i ... **TOTAL** ... **CIDi** ... **ADD** ... **UPD** ... **OLD** ... **KOMP}**

Prüfung von Wertepaaren auf signifikante Untersdchiede bei unabhängigen Stichproben gleicher Varianz.
MTT in ROW vergleicht die Werte innerhalb der Zeilen und MTT in COL die Werte innerhalb der Spalten.
Eine Tabelle mit MTT könnte so aussehen:

	TOTAL	Männer (a)	Frauen (b)
TOTAL	399	198	201
Monatseinkommen, Mittelwert	3230	3243 b	3217 a
Bewertung Marke A	2,7	2,3 b	3,0 a
Marke A bekannt	200	66 b	134 a

a...c = signifikante Unterschiede auf 1%-Niveau

Dies sind die Statements dazu:

```
XTAB ROW=TOTAL;MTT{MEAN(N4);MEAN(C2);C3.1}
     COL=TOTAL;C1
     PRINT1=C(2..3)SIG,1
     BOTTOM='a...c = signifikante Unterschiede auf 1%-Niveau'
```

Die Angabe MTT hinter ROW legt die Zeilen fest, deren Werte durch die t-Tests miteinander zu vergleichen sind.
PRINT1 bestimmt die Position der Markierungen, wählt die Spalten 2 und 3 für die t-Tests aus und gibt dafür eine Signifikanzschwelle von 1% vor.

Die t-Tests verwenden folgende Mittelwerte, unabhängig von den PRINT-Angaben abhängig nur von den Angaben hinter MTTU:

- Für MEAN(Nn) und MEAN(Cn) entsprechen die Mittelwerte den durch PRINT=ABS ausgewiesenen Daten.
- Nn verwendet die gleichen Mittelwerte wie MEAN(Nn).
- ROWS=MTT{Cn.a ... Cn.Zi ... TOTAL ... CIDi ... ADD ... UPD ... OLD ... KOMP} verwendet als Mittelwerte die Prozentwerte von PRINT=PROWi mit einer TOTAL-Zeile i als Basis. Dies entspricht dem z-Test für dichotome Variable.
- COLS=MTT{Cn.a ... Cn.Zi ... TOTAL ... CIDi ... ADD ... UPD ... OLD ... KOMP} arbeitet analog mit den Prozentwerten aus PRINT=PCOLi mit einer TOTAL-Spalte i als Basis.

Fälle, deren N-Variable den Wert Z0 (keine Angabe) besitzt, nehmen nicht an der Bildung der Mittelwerte teil.
Bei MEAN(Cn) wird jedes erfüllte Merkmal wie ein eigener Fall behandelt, Fälle ohne erfülltes Merkmal werden nicht ausgewertet.

Die t-Werte t und die Freiheitsgrade f werden auf folgende Weise berechnet:

$$t = \frac{|m_1 - m_2|}{\sqrt{\frac{n_1 + n_2}{n_1 \cdot n_2} \cdot \frac{(n_1 - 1)\, s_1^2 + (n_2 - 1)\, s_2^2}{n_1 + n_2 - 2}}} \qquad f = n_1 + n_2 - 2$$

m_i = Mittelwerte aus den zu vergleichenden Zeilen oder Spalten.

n_i = Anzahl Fälle in den zu vergleichenden Zeilen oder Spalten.

s_i = Standardabweichungen aus den zu vergleichenden Zeilen oder Spalten.

MTTU{MEAN(Nn) ... **N**n ... **MEAN(C**n**)** ... **C**n.a ... **C**n.**Z**i ... **TOTAL** ... **CIDi** ... **ADD** ... **UPD** ... **OLD** ... **KOMP}**

Dieser t-Test arbeitet analog zu MTT, verzichtet aber auf gleiche Varianz der Variablenwerte.
MTTU in ROW vergleicht die Werte innerhalb der Zeilen und MTTU in COL die Werte innerhalb der Spalten.
Eine Tabelle mit MTTU könnte so aussehen:

	Gesamt (a)	Design (b)	Ausstattung (c)	Volumen (d)
Basis	1204	368	216	65
Alter				
bis 29 Jahre	19%	17%	d 23%	c 11%
60 Jahre und älter	d 8%	d 8%	d 9%	abc 2%

a ... d = signifikante Unterschiede auf 5%-Niveau

Dies sind die Statements dazu:

```
XTAB ROW=TOTAL:'Basis';MTTU{C2.1;.4}
     COL=TOTAL:'Gesamt';C1
     PRINT=SIG,5
     PRINT1=PROW1,%
     BOTTOM='a...d = signifikante Unterschiede auf 5%-Niveau'
```

PRINT=SIG,5 bestimmt die Position der Markierungen und gibt dafür eine Signifikanzschwelle von 5% vor.
Die Angabe MTTU hinter ROW aktiviert den t-Test und bestimmt die Zeilen, deren Werte miteinander zu vergleichen sind.

Die t-Tests verwenden folgende Mittelwerte, unabhängig von den PRINT-Angaben abhängig nur von den Angaben hinter MTTU:

- Für MEAN(Nn) und MEAN(Cn) entsprechen die Mittelwerte den durch PRINT=ABS ausgewiesenen Daten.
- Nn verwendet die gleichen Mittelwerte wie MEAN(Nn).
- ROWS=MTT{Cn.a ... Cn.Zi ... TOTAL ... CIDi ... ADD ... UPD ... OLD ... KOMP} verwendet als Mittelwerte die Prozentwerte von PRINT=PROWi mit einer TOTAL-Zeile i als Basis. Dies entspricht dem z-Test für dichotome Variable.
- COLS=MTT{Cn.a ... Cn.Zi ... TOTAL ... CIDi ... ADD ... UPD ... OLD ... KOMP} arbeitet analog mit den Prozentwerten aus PRINT=PCOLi mit einer TOTAL-Spalte i als Basis.

Fälle, deren N-Variable den Wert Z0 (keine Angabe) besitzt, nehmen nicht an der Bildung der Mittelwerte teil.
Bei MEAN(Cn) wird jedes erfüllte Merkmal wie ein eigener Fall behandelt, Fälle ohne erfülltes Merkmal werden nicht ausgewertet.

Der t-Wert t und die Freiheitsgrade f werden auf folgende Weise berechnet:

$$t = \frac{|m_1 - m_2|}{\sqrt{|e_1^2 + e_2^2|}} \qquad f = \frac{\left(e_1^2 + e_2^2\right)^2}{\frac{e_1^4}{m_1 - 1} + \frac{e_2^4}{m_2 - 1}}$$

m_i = Mittelwerte aus den zu vergleichenden Zeilen oder Spalten.

n_i = Anzahl Fälle in den zu vergleichenden Zeilen oder Spalten.

e_i = Standardfehler aus den zu vergleichenden Zeilen oder Spalten.

MTTX{Cn.a ... **C**n.**Z**i ... **TOTAL** ... **CIDi** ... **ADD** ... **UPD** ... **OLD** ... **KOMP}**

Vergleich von Mittelwerten bei unabhängigen Stichproben mit gleicher Varianz:
MTTX in ROW vergleicht wie MTT die Werte innerhalb der Zeilen und MTTX in COL die Werte innerhalb der Spalten.
Für MTTX in ROW werden die Mittelwerte jedoch aus den COL-Variablen gebildet; MEAN ist nur in COL möglich. Für MTTX in COL werden die Mittelwerte aus den ROW-Variablen gebildet; MEAN ist nur in ROW möglich.

Eine Tabelle mit MTTX könnte so aussehen:

	TOTAL	Bewertung vorher (a)		Bewertung nachher (b)	
TOTAL	399	2,8	-	2,8	-
Männer	198	2,6	b	2,9	a
Frauen	201	3,0	b	2,7	a

a...b = signifikante Unterschiede auf 1%-Niveau

Dies sind die Statements dazu:

```
XTAB COL= TOTAL;MEAN(C1);MEAN(C2)
     ROW=MTTX{TOTAL;C10}
     PRINT1=C(2 3)SIG,1
     BOTTOM='a...b = signifikante Unterschiede auf 1%-Niveau'
```

Die Angabe MTTX hinter ROW legt die Zeilen fest, deren Werte durch die t-Tests miteinander zu vergleichen sind. PRINT1 bestimmt die Position der Markierungen, wählt die Spalten 2 und 3 für die t-Tests aus und gibt dafür eine Signifikanzschwelle von 1% vor.

Die t-Tests verwenden folgende Mittelwerte, unabhängig von den PRINT-Angaben. Stattdessen hängen sie bei MTTX in ROW von den COL-Angaben ab und bei MTTX in COL von den ROW-Angaben:

- Für MEAN(Nn) und MEAN(Cn) entsprechen die Mittelwerte den durch PRINT=ABS ausgewiesenen Daten.
- Nn verwendet die gleichen Mittelwerte wie MEAN(Nn).
- ROWS=Cn.a ... Cn.Zi ... TOTAL ... CIDi ... ADD ... UPD ... OLD ... KOMP mit MTTX in COLverwendet als Mittelwerte die Prozentwerte von PRINT=PROWi mit einer TOTAL-Zeile i als Basis. Dies entspricht dem z-Test für dichotome Variable.
- COLS=Cn.a ... Cn.Zi ... TOTAL ... CIDi ... ADD ... UPD ... OLD ... KOMP mit MTTX in ROW arbeitet analog mit den Prozentwerten aus PRINT=PCOLi mit einer TOTAL-Spalte i als Basis.

Der t-Wert t und die Freiheitsgrade f werden auf folgende Weise ermittelt:

$$t = \frac{|m_1 - m_2|}{\sqrt{\frac{n_1 + n_2}{n_1 \cdot n_2} \cdot \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}} \qquad f = n_1 + n_2 - 2$$

m_i = Mittelwerte aus den zu vergleichenden Zeilen oder Spalten.

n_i = Anzahl Fälle in den zu vergleichenden Zeilen oder Spalten.

s_i = Standardabweichungen aus den zu vergleichenden Zeilen oder Spalten.

MTTUX{Cn.a ... **C**n.**Z**i ... **TOTAL** ... **CIDi** ... **ADD** ... **UPD** ... **OLD** ... **KOMP}**

Dieser t-Test arbeitet analog zu MTTX, verzichtet aber auf gleiche Varianz der Variablenwerte.
MTTUX in ROW vergleicht wie MTT die Werte innerhalb der Zeilen und MTTUX in COL die Werte innerhalb der Spalten.
Für MTTUX in ROW werden die Mittelwerte jedoch aus den COL-Variablen gebildet; MEAN ist nur in COL möglich. Für MTTUX in COL werden die Mittelwerte aus den ROW-Variablen gebildet; MEAN ist nur in ROW möglich.

Eine Tabelle mit MTTUX könnte so aussehen:

	TOTAL	Bewertung vorher (a)	Bewertung nachher (b)
TOTAL	399	2,8 -	2,8 -
Männer	198	2,6 b	2,9 a
Frauen	201	3,0 b	2,7 a

a...b = signifikante Unterschiede auf 1%-Niveau

Dies sind die Statements dazu:

```
XTAB COL=TOTAL;MEAN(C1);MEAN(C2)
     ROW=MTTUX{TOTAL;C10}
     PRINT1=C(2 3)SIG,1
     BOTTOM='a...b = signifikante Unterschiede auf 1%-Niveau'
```

Die Angabe MTTUX hinter ROW legt die Zeilen fest, deren Werte durch die t-Tests miteinander zu vergleichen sind. PRINT1 bestimmt die Position der Markierungen, wählt die Spalten 2 und 3 für die t-Tests aus und gibt dafür eine Signifikanzschwelle von 1% vor.

Die t-Tests verwenden folgende Mittelwerte, unabhängig von den PRINT-Angaben. Stattdessen hängen sie bei MTTUX in ROW von den COL-Angaben ab und bei MTTUX in COL von den ROW-Angaben:

- Für MEAN(Nn) und MEAN(Cn) entsprechen die Mittelwerte den durch PRINT=ABS ausgewiesenen Daten.
- Nn verwendet die gleichen Mittelwerte wie MEAN(Nn).
- ROWS=Cn.a ... Cn.Zi ... TOTAL ... CIDi ... ADD ... UPD ... OLD ... KOMP mit MTTUX in COLverwendet als Mittelwerte die Prozentwerte von PRINT=PROWi mit einer TOTAL-Zeile i als Basis. Dies entspricht dem z-Test für dichotome Variable.
- COLS=Cn.a ... Cn.Zi ... TOTAL ... CIDi ... ADD ... UPD ... OLD ... KOMP mit MTTUX in ROW arbeitet analog mit den Prozentwerten aus PRINT=PCOLi mit einer TOTAL-Spalte i als Basis.

Der t-Wert t und die Freiheitsgrade f werden auf folgende Weise berechnet:

$$t = \frac{|m_1 - m_2|}{\sqrt{|e_1^{\,2} - e_2^{\,2}|}} \qquad f = \frac{(e_1^{\,2} + e_2^{\,2})^2}{\frac{e_1^{\,4}}{n_1 - 1} + \frac{e_2^{\,4}}{n_2 - 1}}$$

m_i = Mittelwerte aus den zu vergleichenden Zeilen oder Spalten.

n_i = Anzahl Fälle in den zu vergleichenden Zeilen oder Spalten.

e_i = Standardfehler für die zu vergleichenden Zeilen oder Spalten.

MTTD{Cn.a ... **C**n**.Z**i ... **TOTAL** ... **CIDi** ... **ADD** ... **UPD** ... **OLD** ... **KOMP}**

MTTD untersucht die Differenzen zweier Variablenwerte (abhängiger oder verbundener Test) für jeden statistischen Fall und prüft, ob sich der Mittelwert dieser Differenzen signifikant von 0 unterscheidet.

MTTD hinter ROW vergleicht die Werte innerhalb einer Zeile für jeden statistischen Fall miteinander. Dazu wird aus je zwei Werten der Zeile die Differenz gebildet und geprüft, ob sich der Mittelwert daraus signifikant von 0 unterscheidet. MEAN ist dabei nur in COL möglich.

MTTD hinter COL vergleicht analog die Werte innerhalb einer Spalte für jeden statistischen Fall miteinander. Hier wird aus je zwei Werten der Spalte die Differenz gebildet und geprüft, ob sich der Mittelwert daraus signifikant von 0 unterscheidet. MEAN ist dabei nur in ROW möglich.

Eine Tabelle mit MTTD könnte so aussehen:

	TOTAL	Bewertung vorher (a)	Bewertung nachher (b)
TOTAL	399	2,8	2,8
Männer	198	2,6 b	2,9 a
Frauen	201	3,0 b	2,7 a

a, b = signifkante Unterschiede auf 5%-Niveau

Dies sind die Statements dazu:

```
XTAB ROW=MTTD{TOTAL;C10.1;...5}
     COL=TOTAL;MEAN(C1);MEAN(C2)
     PRINT1=C(2..3)SIG,5
     BOTTOM='a, b = signifikante Unterschiede auf 5%-Niveau'
```

Die Angabe MTTD hinter ROW vergleicht die Werte innerhalb jeder Zeile für den t-Test.
PRINT1 beschränkt den Vergleich auf die Werte der Spalten 2 und 3 und legt das Signifikanzniveau auf 5% fest. Zu jedem statistischen Fall wird dazu die Differenz der Werte von C1 und C2 gebildet.

Besitzt eine der zu vergleichenden Variablen C1 und C2 im Kopf eine Angabe, die andere jedoch keine, so erscheint eine Fehlermeldung im Zählprotokoll und der statistische Fall wird vom t-Test ausgeschlossen. Das Gleiche geschieht, wenn eine oder beide Variablen Mehrfachnennungen besitzen.

Eine ähnlich aussehende Tabelle erzeugen die Statements:

```
XTAB ROW=MTTD{TOTAL;C10.1;.2}
     COL=TOTAL;C1.1;C1.2
     PRINT1=C(2..3)SIG,5
```

Hier werden für den t-Test die Differenzen der Werte 0 und 1 von C1.1 und C1.2 gebildet und geprüft, ob deren Mittelwert sich signifikant von 0 unterscheidet.

Der t-Wert t und die Freiheitsgrade f werden auf folgende Weise ermittelt:

$$t = \frac{\left|\sum \frac{d_i}{n}\right|}{\sqrt{\frac{\sum d_i^2 - \frac{\left(\sum d_i\right)^2}{n}}{n(n-1)}}} \qquad f = n - 1$$

d_i = Differenz aus den zu vergleichenden Werten eines jeden statistischen Falles.

n = Anzahl Fälle, die zur Bildung der Mittelwerte verwendet wurden.

Effektstärken für MEAN(Nn), MEAN(Cn) und Nn

Die Effektstärken beim Vergleich von zwei Mittelwerten nach J. Cohen werden auf folgende Weise berechnet:

$$d = \frac{m_1 - m_2}{\sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}}$$

m_i = Mittelwerte aus den zu vergleichenden Zeilen oder Spalten
Fälle deren N-Variable den Wert Z0 besitzt, nehmen nicht an der Bildung des Mittelwerts teil.

s_i = Standardabweichung für die zu vergleichenden Zeilen oder Spalten.

n_i = Anzahl Fälle in den zu vergleichenden Zeilen oder Spalten.

Effektstärken für Cn.a, Cn.Zi, TOTAL, CIDi.E, ADD, UPD und OLD

Die Effektstärken beim Vergleich von zwei Anteilswerten nach J. Cohen werden auf folgende Weise berechnet:

$$h = 2 \cdot arcsin\sqrt{\frac{x_1}{n_1}} - 2 \cdot arcsin\sqrt{\frac{x_2}{n_2}}$$

x_i = Anzahl Fälle mit erfüllter Bedingung (Wert 1) in den zu vergleichenden Zeilen oder Spalten.

n_i = Anzahl Fälle in den zu vergleichenden Zeilen oder Spalten.

Beispiel: Vergleich ausgewählter Spalten

```
XTAB  ROW=TOTAL;MTT{C2.1;..4 }
      COL=C1
      PRINT=ABS
      PRINT1=C(1)ABS  (2..3)SIG,1,5  (4..5)SIG,5,1
      BOTTOM='A...Z = Signifikanzniveau 1%'/'a...z=Signifikanzniveau 5%'
```

Dieses Statement könnte folgende Tabelle liefern:

	TOTAL	C1.1	C1.2	C1.3	C1.4
		(a)	(b)	(c)	(d) -
TOTAL	138	37 -	34 -	27 -	40 -
C2.1	26	9 -	8 -	- -	9 -
C2.2	39	9 -	10 -	8 d	12 c
C2.3	43	10 -	11 -	10 -	12 -
C2.4	30	9 B	5 A	9 d	7 c

A...Z = Signifikanzniveau 1%
a...z = Signifikanzniveau 5%

Die Angabe *MTT{ C2.1 ; ..4 }* liefert zunächst für die Zählung die gleichen Ergebnisse wie C2.1;..4, nämlich die Anzahl von Fällen, die die entsprechenden Merkmale erfüllen. Durch die Angabe PRINT=ABS wird erreicht, dass diese Fallzahlen als Absolutwerte ausgewiesen werden.

Die Angabe *PRINT1=C(1)ABS* sorgt zunächst dafür, dass in der ersten Spalte die Signifikanztests unterbleiben und nur die Zählergebnisse ausgewiesen werden.

Die darauf folgende Angabe *(2..3)SIG,1,5* bewirkt, dass zwischen den Spalten 2 und 3 t-Tests durchgeführt werden; dies sowohl auf dem Signifikanzniveau 1% als auch auf dem Niveau 5%. Spaltenpaare, die sich bereits auf dem 1% Niveau unterscheiden, werden dann mit Großbuchstaben markiert, solche auf dem 5% Niveau durch Kleinbuchstaben.

Durch die abschließende Angabe (4..5)SIG,1,5 wird das gleiche Verfahren auch auf die Spalten 4 und 5 angewendet.

XTAB: t-Tests zur Ausgabe von Irrtumswahrscheinlichkeiten

Die Funktionen TT, TTU und TTD vergleichen die Mittelwerte von zwei Variablen miteinander. Sie liefern in den Tabellen die Irrtumswahrscheinlichkeiten dafür, dass sich die Mittelwerte signifikant im Sinne von t-Tests voneinander unterscheiden.

Eine solche Auswertung könnte folgendermaßen aussehen:

t-Test: Unterscheiden sich die Produkt A und B innerhalb der Altersklassen signifikant voneinander?	TOTAL	Alter		
		20-40	41-60	61-80
Anzahl Fälle	200	100	50	50
Mittelwert der Ausgaben für Produkt A	1,81	1,91	1,72	1,70
Mittelwert der Ausgaben für Produkt B	1,78	1,60	1,78	2,13
Signifikanz für den Unterschied zwischen den Produkten A und B	61,1%	0,1%	55,6%	0,4%

Die Irrtumswahrscheinlichkeiten erscheinen in den Tabellen als Prozentwerte für den zweiseitigen Test. Kleine Werte weisen auf signifikant unterschiedliche Mittelwerte hin, große Werte lassen keine signifikanten Unterschiede vermuten.

Bei vorgegebenem 1%-Niveau unterscheiden sich die beiden Produkte signifikant in der ersten und letzten Altersklasse, nicht jedoch in der zweiten Klasse und in der Gesamtheit.

Die Statements zum obigen Beispiel könnten folgendermaßen aussehen:

```
XTAB    ROW= TOTAL; MEAN(N1); MEAN(N2);
            TT(N1,N2): 'Signifikanz für den Unterschied zwischen den Produkten A und B'
        COL=TOTAL;C1
        PRINT=R(1) ABS R(2 .. 3) ABS,2 R(4) ABS, 1, %
```

Die t-Werte und Freiheitsgrade selbst werden nicht ausgewiesen. Sie können durch Angabe der entsprechenden Formeln als numerische Ausdrücke hinter ROW oder COL ermittelt werden.

Bei t-Tests für Merkmale von C-Variablen entsprechen die Prozentwerte der Zählergebnisse den Mittelwerten. Für TT hinter ROW wird dabei auf die Werte der Total-Zeile prozentuiert, hinter COL auf die der Total-Spalte.

Für die Ausgabe solcher Signifikanzwerte ist in ROW oder COL eine der folgenden Funktionen anzugeben. Sie lassen sich auch als Teile von numerischen Ausdrücken verwenden, um weitere Verrechnungen durchzuführen:

TT(| **N**n | **,** | **N**m | **)**
| **C**n | | **C**m |
| **C**n.a | | **C**m.b |

Diese Funktion vergleicht die Mittelwerte von zwei Variablen. Die Variablenwerte sollten unabhängig voneinander sein und gleiche Varianz besitzen.

Der t-Wert t und die Freiheitsgrade f werden auf folgende Weise ermittelt:

$$t = \frac{|m_1 - m_2|}{\sqrt{\frac{n_1 + n_2}{n_1 \cdot n_2} \cdot \frac{(n_1 - 1)s_1^{\,2} + (n_2 - 1)s_2^{\,2}}{n_1 + n_2 - 2}}} \qquad f = n_1 + n_2 - 2$$

m_i = die zu vergleichenden Mittelwerte.
Für Cn.a und Cm.b werden diese aus den Werten 0 und 1 der Merkmale gebildet.
Fälle deren N-Variable den Wert Z0 besitzt, nehmen nicht an der Bildung des Mittelwerts teil. Gleiches gilt für C-Variable ohne erfülltes Merkmal.
Sind mehrere Merkmale einer C-Variablen gleichzeitig erfüllt, so wird jedes erfüllte Merkmal wie ein eigener Fall behandelt.

n_i = Anzahl Fälle, die zur Bildung der Mittelwerte verwendet wurden.

s_i = Standardabweichung.

TT(k **,** | **N**n | **)**
| **C**n |
| **C**n.a |

Diese Testfunktion vergleicht die Mittelwerte einer Variablen mit dem konstanten Wert k.

Der t-Wert t und die Freiheitsgrade f werden auf folgende Weise ermittelt:

$$t = \frac{|m - k|}{s}\sqrt{n} \qquad f = n - 1$$

m = Mittelwert der Variablen Cn oder Nn.
Für Cn.a wird dieser aus den Werten 0 und 1 des Merkmals errechnet.
Fälle deren N-Variable den Wert Z0 besitzt, nehmen nicht an der Bildung des Mittelwerts teil. Gleiches gilt für C-Variable ohne erfülltes Merkmal.
Sind mehrere Merkmale einer C-Variablen gleichzeitig erfüllt, so wird jedes erfüllte Merkmal wie ein eigener Fall behandelt.

s = Standardabweichung der Variablen.

n = Anzahl Fälle, die zur Bildung des Mittelwertes verwendet wurden.

TTU(| **N**n | **,** | **N**m | **)**
| **C**n | | **C**m |
| **C**n.a | | **C**m.b |

Diese Funktion arbeitet analog zu TT. Für die beiden Variablen ist jedoch nicht mehr gleiche Varianz erforderlich. Dafür wird mit den folgenden Näherungsformel für die t-Werte t und die Freiheitsgrade f gearbeitet:

$$t = \frac{|m_1 - m_2|}{\sqrt{|e_1^{\,2} - e_2^{\,2}|}} \qquad f = \frac{(e_1^{\,2} + e_2^{\,2})^2}{\frac{e_1^{\,4}}{n_1 - 1} + \frac{e_2^{\,4}}{n_2 - 1}}$$

m_i = die zu vergleichenden Mittelwerte.
Für Cn.a und Cm.b werden diese aus den Werten 0 und 1 der Merkmale gebildet.
Fälle deren N-Variable den Wert Z0 besitzt, nehmen nicht an der Bildung des Mittelwerts teil. Gleiches gilt für C-Variable ohne erfülltes Merkmal.
Sind mehrere Merkmale einer C-Variablen gleichzeitig erfüllt, so wird jedes erfüllte Merkmal wie ein eigener Fall behandelt.

e_i = Standardfehler.

n_i = Anzahl Fälle, die zur Bildung der Mittelwerte verwendet wurden.

TTD(| **N**n | **,** | **N**m | **)**
| **C**n | | **C**m |
| **C**n.a | | **C**m.b |

Bei dieser Testfunktion wird für jeden statistischen Fall die Differenz aus den Werten der beiden Argumente gebildet (abhängiger oder verbundener Test). Das Ergebnis ist die Irrtumswahrscheinlichkeit dafür, dass sich der Mittelwert der Paardifferenzen signifikant von 0 unterscheidet. Für diesen Test sollten die Paardifferenzen annähernd normalverteilt sein.

Der t-Wert t und die Freiheitsgrade f dazu werden auf folgende Weise ermittelt:

$$t = \frac{\left|\sum \frac{d_i}{n}\right|}{\sqrt{\frac{\sum d_i^2 - \frac{\left(\sum d_i\right)^2}{n}}{n(n-1)}}} \qquad f = n - 1$$

d_i = Differenz aus den Werten der beiden Variablen für jeden statistischen Fall.
Für Cn.a und Cm.b wird diese aus den Werten 0 und 1 der Merkmale gebildet.
Fälle deren N-Variable den Wert Z0 besitzt, nehmen nicht an der Bildung des Mittelwerts teil. Gleiches gilt für C-Variable ohne erfülltes Merkmal.
Sind mehrere Merkmale einer C-Variablen gleichzeitig erfüllt, so wird das Merkmal mit der niedrigsten Merkmalsnummer verwendet..

n = Anzahl Fälle, die zur Bildung der Mittelwerte verwendet wurden.

Beispiel

```
XTAB  SUPPRES=SPACES,FILTER
      COL=TOTAL;C1
      ROW=TOTAL;
          MEAN(C2):'Mittelwert C2';
          MEAN(C3):'Mittelwert C3';
          STD(C2):'Standardabw. C2';
          STD(C3):'Standardabw. C3';
          TT(C2,C3):'t-Test TT';
          TTU(C2,C3):'t-Test TTU'
          PRINT=R(1)ABS  (2..3)ABS,1  (4..5)ABS,2  (6..7)ABS,2,%
```

Dieses Statement könnte folgende Tabelle erzeugen:

	TOTAL	C1.1	C1.2	C1.3
TOTAL	260	79	84	97
Mittelwert C2	2.5	2.6	2.5	2.5
Mittelwert C3	3.0	3.0	3.0	2.9
Standardabw. C2	1.16	1.15	1.10	1.23
Standardabw. C3	1.40	1.42	1.43	1.37
t-Test TT	0.04%	4.77%	0.38%	8.44%
t-Test TTU	0.04%	4.78%	0.39%	8.44%

In den Zeilen dieser Tabelle wird zunächst durch TOTAL die Anzahl der Fälle ausgewiesen. Es folgen die Mittelwerte der Bedingungsvariablen C2 und C3 sowie die Standardabweichungen zu diesen Variablen.

In der vorletzten Zeile werden die Mittelwerte von C2 und C3 durch den t-Test TT miteinander verglichen und das Ergebnis als Irrtumswahrscheinlichkeit in Prozenten ausgewiesen.

Die letzte Zeile zeigt das Ergebnis des t-Tests TTU für Stichproben ungleicher Varianz. Da die Standardabweichungen - und damit auch die Varianzen - hier nicht nennenswert voneinander abweichen, ergeben sich zwischen den Funktionen TT und TTU praktisch keine Unterschiede.

XTAB: Zweidimensionaler Chi-Quadrat-Test

Dieser Chi^2-Test vergleicht Zeilen- und Spaltenbedingungen einer Tabelle oder Teiltabelle miteinander.
Eine entsprechende Tabelle könnte folgendermaßen aussehen:

	TOTAL	C2.1	C2.2	C2.3	C2.4	C2.5
TOTAL	192	45	33	37	34	43
C1.1	35	17,8% a)	24,2% a)	13,5% a)	26,5% a)	11,6% a)
C1.2	44	20,0% a)	24,2% a)	29,7% a)	20,6% a)	20,9% a)
C1.3	37	17,8% a)	12,1% a)	32,4% a)	5,9% a)	25,6% a)
C1.4	41	26,7% a)	15,2% a)	8,1% a)	29,4% a)	25,6% a)
C1.5	35	17,8% a)	24,2% a)	16,2% a)	17,6% a)	16,3% a)

a) Diese Zähler nehmen am Chi-Quadrat-Test teil.
Testergebnis: Chi-Quadrat = 19.27 Signifikanz = 25.50%

Die beteiligten Tabellenzähler werden in der Ausgabe mit frei wählbaren Zeichen markiert, hier mit a).
Als Ergebnis wird der Chi^2-Wert und der dazugehörige Signifikanzwert in Prozent ausgegeben und zwar hinter einem Fußnotentext unterhalb der Tabelle.

Der Signifikanzwert gibt den kleinsten Fehler an, bei dem die Hypothese *die Zeilen- und Spaltenbedingungen sind statistisch unabhängig* abzulehnen ist: Kleine Werte weisen auf signifikant abhängige Bedingungen hin, große lassen auf unabhängige Bedingungen schließen.

Dieser Chi-Quadrat -Test verlangt lediglich eine Eintragung hinter PRINT ... PRINT3 im XTAB-Statement:

PRINTi **= R(** ... **) C(** ... **) CHI2 <,'t'> <,Fi> <,FT**i**> <,G**i**>**

Mit den Angaben R(...) und C(...) sind die Zeilen und Spalten auszuwählen, die an dem Test teilnehmen sollen. Fehlen solche Angaben, so nehmen alle Zeilen oder Spalten teil.

Die am Test beteiligten Tabellenzellen werden mit den Zeichen aus der Druckmaske 't' gekennzeichnet.
Fehlt eine solche Druckmaske, so wird dafür das Zeichen x verwendet.

Der ermittelte Chi-Quadrat-Wert und der dazugehörige Signifikanzwert werden hinter der Fußnote Fi in der Form
Chi-Quadrat = x.xx Signifikanz = y.yy%
ausgegeben. Ist kein Fußnotentext Fi angegeben, so erscheint dafür der Text:
x) Chi-Quadrat-Test dieser Tabellenwerte: Chi-Quadrat = x.xx Signifikanz = y.yy%

Die Berechnung der Chi-Quadrat-Werte erfolgt auf folgende Weise:

$$\chi^2 = \sum_{i=Zeilen} \sum_{k=Spalten} \frac{\left(x_{ik} - \frac{z_i \, s_k}{n} \right)^2}{\frac{z_i \cdot s_k}{n}}$$

x_{ik} = Tabellenwert der Zeile i und Spalte k.
z_i = Summe der Tabellenwerte der Zeile i.
s_k = Summe der Tabellenwerte der Spalte k.
n = Summe aller beteiligten Tabellenwerte.

Zur Umrechnung der Chi-Quadrat-Werte in Signifikanzwerte S = 0.00 ... 1.00 werden folgende Regeln verwendet:

F =Freiheitsgrade= (Anzahl Zeilen - 1) (Anzahl Spalten - 1)

Für F < 1 oder $\chi^2 \le 0$: $S = 1.00$

Für F = 1: $S = 2\,Q\left(\sqrt{\chi^2}\right)$

Für F > 30: $S = Q\left(\frac{\sqrt[3]{\frac{\chi^2}{F}} - \left(1 - \frac{2}{9F}\right)}{\sqrt{\frac{2}{9F}}} \right)$

Für F = 2 ... 30 und $\chi^2 \ge 100$: $S = 0.00$

Für F = 2 ... 30, F gerade und $\chi^2 < 100$: Mit $R = \sum_{i=1}^{\frac{F-2}{2}} \frac{\left(\chi^2\right)^i}{2 \cdot 4 \cdot \ldots \cdot 2i}$

$$S = \exp\left(-\frac{\chi^2}{2}\right)(1 + R)$$

Für F = 2 ... 30, F ungerade und $\chi^2 < 100$: Mit $R = \sum_{i=1}^{\frac{F-1}{2}} \frac{\left(\chi^2\right)^{i-\frac{1}{2}}}{1 \cdot 3 \cdot \ldots \cdot (2i-1)}$

$$S = 2Q\left(\sqrt{\chi^2}\right) + R\sqrt{\frac{2}{\pi}} \cdot \exp\left(-\frac{\chi^2}{2}\right)$$

Dabei ist Q(x) der Signifikanzwert der Standard Normalverteilung x, ermittelt auf folgende Weise:

$$Q(x) = 0.5\{1 + Z(a_1 + Z(a_2 + Z(a_3 + Z(a_4 + Z(a_5 + Za_6)))))\}^{-16}$$

mit:

$Z = 0.7071067812\,|x|$ für $|x| \le 14.14$

$Z = 10$ für $|x| > 14.14$

a_1 = 0.070523078 a_2 = 0.0422820123 a_3 = 0.0092705272
a_4 = 0.0001520143 a_5 = 0.0002765672 a_6 = 0.0000430638

Beispiel: Standard-Ausgabe

```
XTAB   ROW=TOTAL:ABS;C1
       COL=TOTAL:ABS;C2
       PRINT=R(2..-1)C(2..-1)PROW1,1,%
       PRINT1=R(2..-1)C(2..-1)CHI2
```

Dieses Statement könnte folgende Tabelle liefern:

	TOTAL	C2.1	C2.2	C2.3	C2.4	C2.5
TOTAL	192	45	33	37	34	43
C1.1	35	17,8% x	24,2% x	13,5% x	26,5% x	11,6% x
C1.2	44	20,0% x	24,2% x	29,7% x	20,6% x	20,9% x
C1.3	37	17,8% x	12,1% x	32,4% x	5,9% x	25,6% x
C1.4	41	26,7% x	15,2% x	8,1% x	29,4% x	25,6% x
C1.5	35	17,8% x	24,2% x	16,2% x	17,6% x	16,3% x

x) Chi-Quadrat-Test dieser Tabellenwerte: Chi-Quadrat = 19.27 Signifikanz = 25.50%

Die Angabe PRINT1 sorgt für einen Chi-Quadrat-Test der Werte der Zeilen 2 ... 6 und Spalten 2 ... 6. Die daran teilnehmenden Tabellenzellen werden in der Position PRINT1 mit dem Zeichen x markiert.

Das Ergebnis dieses Tests wird unterhalb der Tabelle als Fußnote ausgewiesen

Beispiel: Mit eigenem Fussnotentext

```
DEFS
-(3)  'a) ~S~Diese Zähler nehmen am Chi-Quadrat-Test teil.'/
      '~T~Testergebnis:'

XTAB  ROW=TOTAL:ABS;c1
      COL=TOTAL:ABS;C2
      PRINT=R(2..-1)C(2..-1)PROW1,1,%
      PRINT1=R(2..-1)C(2..-1)CHI2,F3,'a)'
```

Diese Statements könnten folgende Tabelle erzeugen:

	TOTAL	C2.1	C2.2	C2.3	C2.4	C2.5
TOTAL	192	45	33	37	34	43
C1.1	35	17,8% a)	24,2% a)	13,5% a)	26,5% a)	11,6% a)
C1.2	44	20,0% a)	24,2% a)	29,7% a)	20,6% a)	20,9% a)
C1.3	37	17,8% a)	12,1% a)	32,4% a)	5,9% a)	25,6% a)
C1.4	41	26,7% a)	15,2% a)	8,1% a)	29,4% a)	25,6% a)
C1.5	35	17,8% a)	24,2% a)	16,2% a)	17,6% a)	16,3% a)

a) Diese Zähler nehmen am Chi-Quadrat-Test teil.
Testergebnis: Chi-Quadrat = 19.27 Signifikanz = 25.50%

Die Angabe PRINT1 sorgt wie im vorigen Beispiel für einen Chi-Quadrat-Test.

Die Angabe 'a)' hinter PRINT1 markiert die am Test teilnehmenden Tabellenzellen mit den Zeichen *a)* .

Die Angabe F3 hinter PRINT1 wählt den Text des Textelements (3) als Fußnotentext aus, hinter dem die Ergebnisse des Chi-Quadrat-Tests erscheinen.

XTAB: Folgestatements zu XTAB

Diese Statements können die Zählergebnisse einer Tabelle vor ihrer Aufbereitung verändern.
Nur Absolutwerte lassen sich damit direkt bearbeiten. Prozente, Rangzahlen und Indexwerte werden später anhand der veränderten Absolutwerte ermittelt.

Zum Beispiel addiert das Folgestatement

```
-R(1)+=R(7)
```

die Zählerwerte der Zeile 7 einer Tabelle zu den entsprechenden Werten der Zeile 1.

Die Folgestatements werden ausgeführt, nachdem alle arithmetischen Ausdrücke des XTAB-Statements verrechnet und eventuelle XADD-Werte zur Tabelle hinzugefügt sind. Sie gelten nur für das davor liegende XTAB und werden in der eingegebenen Reihenfolge abgearbeitet.

Erst nachdem die Folgestatements ausgeführt sind, werden die Zeilen oder Spalten durch SORT sortiert oder durch SUPPRESS unterdrückt.

Diese Folgestatements beginnen mit einem Strich am Anfang der Zeile. Sie müssen direkt hinter dem zugehörigen XTAB und seinen XADD-Statements eingefügt werden.

Die Zeilen und Spalten von Tabellen lassen sich mit den Folgestatements von XTAB in besonderen Variablen V0 ... V99 zwischenspeichern. Spätere XTAB-Statements können dann auf die Werte dieser Variablen zugreifen.

Zu XTAB sind verschiedene Folgestatements möglich:

Wertezuweisungen

Mit diesen Wertezuweisungen lassen sich Zeilen und Spalten einer Tabelle durch andere Zeilen oder Spalten ersetzen oder mit ihnen verrechnen. Auch konstante Werte können unabhängig von Zählergebnissen in die Tabellen gestellt werden. Spezielle Variable übertragen Zeilen oder Spalten von Tabellen in spätere XTAB-Statements.

-**R(**z...**)** < **C(**s...**)** >	**+=**	**R(**z**)**
-**C(**s...**)** < **R(**z...**)** >	**=**	**C(**s**)**
-**V**i	**–=**	**T(**z,s**)**
	***=**	**V**i
	/=	k
		(k1 k2 ... kn **)**
		ABS(...**)**

C(s... **)** Diese Angabe überträgt Werte aus Tabellenspalten oder in Tabellenspalten. Zum Beispiel:

```
XTAB ...
-C(1)=C(7)
-V3=C(1)
```

Die Werte der Spalte 7 ersetzen die Werte der ersten Spalte. Danach werden die neuen Werte der Spalte 1 in die Variable V3 übertragen.

Auch negative Spaltennummern sind möglich: C(–1) ist die letzte Spalte der Tabelle, C(–2) die vorletzte usw.

Links vom Gleichheitszeichen sind mehrere Spalten möglich. Die Statements

```
-C(1 5..10)=99
-C(2 3)=C(-1)
```

füllen die Spalten 1 und 5 bis 10 mit dem Wert 99 und kopieren die Werte der letzten Spalte in die Spalten 2 und 3.

Auch rechts vom Gleichheitszeichen sind mehrere Spalten möglich. Die Statements

```
-C(6)=C(1...5)
```

bilden die Summen aus den Werten der Spalten 1 bis 5 und stellen sie in die Spalte 6.

Sind hinter COL einzelne Spalten mit Klassenbuchstaben A ... Z versehen, so lassen sich anstelle der Spaltennummern auch die Klassenbuchstaben A ... Z verwenden:

```
XTAB ROW=TOTAL;C1 COL=A{TOTAL};B{C2}
-C(B)=C(A)
```

Hier werden die Werte der ersten Spalte in die übrigen Spalten der Tabelle kopiert. Siehe Abschnitt *XTAB, ROW/COL/FILTER, Klassenbildung von Zähloperanden.*

Die Wertezuweisung in die Spalten links vom Gleichheitszeichen kann zusätzlich auf einzelne Zeilen beschränkt werden. Zum Beispiel:

```
-C(2..-1)R(3 4)=C(1)
```

Hier werden die Werte der Zeilen 3 und 4 der ersten Spalte in die Zeilen 3 und 4 der übrigen Spalten gestellt.

–R(z... **)** Diese Angabe überträgt Werte aus Tabellenzeilen oder in Tabellenzeilen. Zum Beispiel:

```
XTAB ...
-R(3)=R(1)
-V99=R(3)
```

Die Werte der ersten Zeile ersetzen die Werte der dritten Zeile. Danach werden die neuen Werte der dritten Zeile in die Variable V99 übertragen.

Auch negative Zeilennummern sind möglich: R(**–**1) ist die letzte Spalte der Tabelle, R(**–**2) die vorletzte usw.

Links vom Gleichheitszeichen sind mehrere Zeilen möglich. Die Statements

```
-R(1 5..10)=5.5
-R(2 3)=R(-1)
```

füllen die Zeilen 1 und 5 bis 10 mit dem Wert 5,5 und kopieren die Werte der letzten Zeile in die Zeilen 2 und 3.

Auch rechts vom Gleichheitszeichen sind mehrere Zeilen möglich. Die Statements

```
-R(6)=R(1...5)
```

bilden die Summen aus den Werten der Zeilen 1 bis 5 und stellen sie in die Zeile 6.

Sind hinter ROW einzelne Zeilen mit Klassenbuchstaben A ... Z versehen, so lassen sich anstelle der Zeilennummern auch die Klassenbuchstaben A ... Z verwenden:

```
XTAB ROW=B{TOTAL};A{C1} COL=TOTAL;C2
-R(A)=R(B)
```

Hier werden die Werte der ersten Zeile in die übrigen Zeilen der Tabelle kopiert. Siehe Abschnitt *XTAB, ROW/COL/FILTER, Klassenbildung von Zähloperanden.*

Die Wertezuweisung in die Zeilen links vom Gleichheitszeichen kann zusätzlich auf einzelne Spalten beschränkt werden. Zum Beispiel:

```
-R(2..-1)C(3 4)=R(1)
```

Hier werden die Werte der Spalten 3 und 4 der ersten Zeile in die Spalten 3 und 4 der übrigen Zeilen gestellt.

Vi In den Variablen V0 ... V99 lassen sich ganze Zeilen oder Spalten einer Tabelle speichern und in die gleiche oder in spätere Tabellen übertragen. Zum Beispiel:

```
XTAB ...
-V9=R(1)
XTAB ...
-R(7)=V9
```

Die Zeile 1 der ersten Tabelle wird in die Variable V9 kopiert. Anschließend werden diese Werte aus der Variablen V9 in die Zeile 7 der zweiten Tabelle gestellt.

T(z,s**)** Diese Angabe entnimmt den Wert aus Zeile z = 1 ... und Spalte s = 1 ... der Tabelle und verrechnet ihn mit den Zeilen und Spalten links vom Gleichheitszeichen. Zum Beispiel:

```
XTAB ...
-C(4)/=T(1,4)
```

Alle Werte der Spalte 4 werden durch den Wert aus Zeile 1 der Spalte 4 dividiert.

k Konstante Werte können in die Zeilen oder Spalten einer Tabelle gestellt oder mit ihnen verrechnet werden. Zum Beispiel:

```
XTAB ...
-R(7)*=1.438
```

Alle Werte der Zeile 7 werden mit der festen Zahl 1.438 multipliziert.
Eine solche Konstante darf bis zu 9 Ziffern besitzen und maximal 8 Nachkommastellen.
Dezimalzeichen ist der Punkt.

(k_1 k_2 ... k_m **)**

Auch unterschiedliche konstante Werte lassen sich in eine Zeile oder Spalte stellen. Zum Beispiel:

```
XTAB ...
-R(7)=(1.4 2.9 4.8 9.4)
```

Hier werden vier verschiedene Zahlen direkt in die Zeile 7 einer Tabelle gestellt.
Solche Zahlen dürfen bis zu 9 Ziffern besitzen und maximal 8 Nachkommastellen. Dezimalzeichen ist der Punkt. Die konstanten Werte sind durch Leerzeichen voneinander zu trennen.

ABS(... **)** Diese Funktion wandelt die Werte einer Zeile, einer Spalte oder einer Variablen Vi in ihre absoluten Beträge um. Enthält zum Beispiel die Variable V3 die Werte

```
(5  -2  4  -9)
```

so füllt das Statement

```
-V4=ABS(V3)
```

die Variable V4 mit den Werten

```
(5  2  4  9)
```

= Das Gleichheitszeichen ersetzt die Werte einer Zeile oder Spalte durch andere Werte.

+= – = *= /=

Diese Angaben verrechnen die Werte einer Zeile oder Spalte mit anderen Werten.
Mit += wird addiert, mit – = subtrahiert, mit *= multipliziert und mit /= dividiert. Division durch 0 liefert das Ergebnis 0. Zum Beispiel:

```
XTAB ...
-R(1)+=R(-1)
-V10=R(1)
-V10*=100
```

Die Zählergebnisse der letzten Zeile werden zu den entsprechenden Werten der ersten Zeile hinzuaddiert. Diese Summen werden danach in die Variable V10 übertragen und dort mit 100 multipliziert.

Einfache Rundung: ROUND

Diese Folgestatements von XTAB runden die Zählergebnisse ganzer Tabellen oder ausgewählter Zeilen und Spalten mit beliebigen Rundungsfaktoren. Dabei werden nur die Absolutwerte verändert. Prozente, Rangzahlen und Indexwerte werden anhand der neuen Absolutwerte ermittelt.

Zum Beispiel sorgen die Statements

```
XTAB ...
-ROUND=100
```

dafür, dass alle Zählergebnisse der Tabelle auf 100 gerundet werden, also mit 00 enden.

–ROUND < **(** < **C(**s ...**)** > < **R(**z ...**)** > **)** > **=** r

C(s ...**)** Zeilen und Spalten, deren Werte zu runden sind. Es sind die Zeilen- und Spaltennummern vor
R(z ...**)** eventueller Sortierung durch SORT oder Unterdrückung durch SUPPRESS anzugeben.

Zum Beispiel rundet das Statement

```
-ROUND(R(1) C(3 9))=10
```

die Zählergebnisse der Zeile 1 in den Spalten 3 und 9 so dass als letzte Stelle 0 erscheint. Die sonstigen Werte bleiben unverändert.

Fehlen die Angaben R und C, so wird die ganze Tabelle bearbeitet. Zum Beispiel rundet

```
-ROUND=100
```

die ganze Tabelle auf 100.

Fehlt nur die Angabe R, so werden alle Zeilen gerundet und fehlt nur C, werden alle Spalten bearbeitet. Zum Beispiel gilt

```
-ROUND(R(2..5))=50
```

für die Zeilen 2 bis 5 in allen Spalten.

Auch negative Zeilen und Spaltenangaben sind möglich: R(**–**1) ist die letzte Zeile einer Tabelle, C(**–**2) die vorletzte Spalte. So rundet

```
-ROUND(R(1..-2) C(-1))=10
```

die Werte der ersten bis vorletzten Zeile in der letzten Spalte.

Sind hinter ROW oder COL Zeilen oder Spalten mit Klassenbuchstaben A ... Z versehen, so lassen sich anstelle der Zeilen- oder Spaltennummern auch die Klassenbuchstaben A ... Z verwenden:

```
XTAB ROW=TOTAL;A{C1} COL=TOTAL;B{C2}
-ROUND(R(A),C(B))=100
```

Hier werden die Zeilen der Variablen C1 in den Spalten der Variablen C2 gerundet. Siehe Abschnitt *XTAB, ROW/COL/FILTER, Klassenbildung von Zähloperanden.*

Klassenangaben und Von-Bis-Werte lassen sich kombinieren. Zum Beispiel

```
-ROUND(R(8..7 A),C(B -1))=100
```

r Rundungsfaktor.
Die Zählergebnisse werden so gerundet, dass sie anschließend durch diesen Faktor ganzzahlig teilbar sind. Das Statement

```
-ROUND=25
```

erzeugt Zählergebnisse, die auf 00, 25, 50 oder 75 enden.

Auch Faktoren mit Nachkommastellen sind möglich. Zum Beispiel setzt

```
-ROUND=0.1
```

alle Nachkommastellen bis auf die erste auf 0. Als Dezimalzeichen ist dabei der Punkt anzugeben.

Anpassung an Summen: PROSUM

Diese Folgestatements verändern die Zählergebnisse vor ihrer Aufbereitung in den Tabellen:

- Summenanpassung durch Hochrechnung.
  Für ausgewählte Zähler einer Zeile oder Spalte wird geprüft, ob ihre Summe mit einem vorgegebenen Wert übereinstimmt. Andernfalls werden die Zähler mit einem vom Programm ermittelten Faktor multipliziert, so dass ihre Summe den gewünschten Wert liefert. Siehe *Beispiel PROSUM* am Ende dieses Abschnitts.
- Hochrechnung mit Rundung.
  Ausgewählte Zähler eine Zeile oder Spalte werden zunächst so hochgerechnet, dass ihre Summe gleich einem vorgegebenen Wert wird. Anschließend werden die Zähler gerundet, so dass sie in der Tabelle zum Beispiel mit 00 enden. Siehe *Beispiel PROSUM* am Ende dieses Abschnitts.
- Hochrechnung mit Rundung bei Erhalt von Prozentwerten.
  Die Rundung lässt sich auch so gestalten, dass sich Prozentwerte möglichst wenig ändern. Siehe *Beispiel PROSUM mit Erhalt von Prozentwerten* am Ende dieses Abschnitts.

–PROSUM-ROWS **–PROSUM-COLS**	< ***(*** < ***C(****s ...****)*** > < ***R(****z ...****)*** > ***)*** > **=**	**R(**i**)** **C(**i**)** **V**i	< **ROUND=** r >			
			< **ROUND=(**f1 f2 ...**)**	**R(**k**)** **C(**k**)** **V**k	t >	

–PROSUM–ROWS passt die Summe aus den Werten verschiedener Zeilen an vorgegebene Werte an.

–PROSUM–COLS passt die Summe aus den Werten verschiedener Spalten an vorgegebene Werte an.

C(s ...**)** Zeilen und Spalten, in denen die Anpassung erfolgen soll. Es sind die Zeilen- und Spaltennummern
R(z ...**)** vor eventueller Sortierung durch SORT und Unterdrückung durch SUPPRESS anzugeben.

Mit dem Statement

```
-PROSUM-ROWS(R(2..10) C(3 5))=R(1)
```

werden die Zählergebnisse der Zeilen 2 bis 10 in den Spalten 3 bis 5 so hochgerechnet, dass ihre Summe gleich dem Wert der ersten Zeile wird.

Fehlen die Angaben R und C, so gilt das Statement für die ganze Tabelle. Zum Beispiel:

```
-PROSUM-ROWS=V7
```

Fehlt nur die Angabe R, so werden alle Zeilen bearbeitet und fehlt nur C, werden alle Spalten bearbeitet. Zum Beispiel gilt

```
-PROSUM-ROWS(R(2..5))=R(1)
```

für die Zeilen 2 bis 5 in allen Spalten.

Auch negative Zeilen und Spaltenangaben sind möglich. −1 ist die letzte Zeile oder Spalte einer Tabelle, −2 die vorletzte usw. So bearbeitet

```
-PROSUM-ROWS(R(2..-2) C(-1))= ...
```

die Werte der ersten bis vorletzten Zeile in der letzten Spalte.

Sind in ROW oder COL Zeilen oder Spalten mit Klassenbuchstaben A ... Z versehen, so lassen sich anstelle der Zeilen- oder Spaltennummern auch die Klassenbuchstabe A ... Z verwenden:

```
XTAB ROW=TOTAL;A{C1} COL=TOTAL;B{C2}
-PROSUM-ROWS(R(A) C(B))= ...
```

Hier werden die Zeilen der Variablen C1 in den Spalten der Variablen C2 bearbeitet. Siehe Abschnitt *XTAB, ROW/COL/FILTER, Klassenbildung von Zähloperanden.*

Klassenangaben und Von-Bis-Werte lassen sich kombinieren. Zum Beispiel

```
-PROSUM-ROWS(R(1..7 A) C(B -1))=  ...
```

R(i **)** Diese Angabe legt für PROSUM-ROWS die Tabellenzeile fest, deren Werte die Summen erreichen sollen. Das Statement

```
-PROSUM-ROWS(R(2..5))=R(1)
```

verlangt, dass nach der Hochrechnung die Summen aus den Werten der Zeilen 2 bis 5 mit den Werten der Zeile 1 übereinstimmen. Dazu werden die Werte der Zeilen 2 bis 5 mit einem geeigneten Faktor multipliziert.

C(i **)** Diese Angabe legt für PROSUM-COLS die Tabellenspalte fest, deren Werte die Summen erreichen sollen. Das Statement

```
-PROSUM-COLS(C(2..5))=C(6)
```

verlangt, dass nach der Hochrechnung die Summen aus den Werten der Spalten 2 bis 5 mit den Werten der Spalte 6 übereinstimmen. Dazu werden die Werte der Spalten 2 bis 5 mit einem geeigneten Faktor multipliziert.

Vi Mit dieser Angabe entnimmt PROSUM-ROWS oder PROSUM-COLS die Zielwerte für die Summenbildung aus einer Variablen V0 ... V99. Diese muss in einer vorausgegangenen Wertezuweisung mit Daten gefüllt werden. Zum Beispiel:

```
XTAB ...
-V9=ROW1
XTAB ...
-PROSUM-ROWS(R(2..5))=V7
```

Die Zeile 1 der ersten Tabelle wird in die Variable V9 kopiert. Im nächsten XTAB werden diese Werte als Ziel für die Hochrechnung verwendet. Dazu werden die Werte der Zeilen 2 bis 5 mit einem geeigneten Faktor multipliziert.

ROUND = r Nach der Hochrechnung werden die beteiligten Zähler mit diesem Faktor gerundet. Die Angabe

```
... ROUND=25
```

erzeugt zum Beispiel Zählergebnisse, die auf 00, 25, 50 oder 75 enden.
Auch Faktoren mit Nachkommastellen sind möglich. Zum Beispiel setzt

```
... ROUND=0.1
```

alle Nachkommastellen ab der zweiten auf 0. Als Dezimalzeichen ist dafür der Punkt anzugeben.

Abschließend werden die gerundeten Werte so nachgebessert, dass ihre Summe wieder mit dem gerundeten Zielwert übereinstimmt. Der Zielwert selbst wird dabei nicht verändert.

ROUND = (f_1 f_2 ... **)** ... t

Hier werden mehrere Rundungsfaktoren angegeben. Für jeden dieser Faktoren wird das beim einfachen Faktor *ROUND = r* beschriebene Verfahren durchgeführt. Danach entscheidet sich das Programm für den größten Rundungsfaktor, bei dem sich die Prozentwerte vor und nach der Hochrechnung und Rundung um weniger als die Toleranzschwelle *t* unterscheiden.

Als Prozentuierungsbasis dienen die durch R(k), C(k) oder Vk vorgegebenen Werte. Zum Beispiel

```
-PROSUM-ROWS(R(2..4))=R(1)   ROUND=(100 50 10) V2 0.5
```

Die Werte der Zeilen 2 bis 4 werden vor und nach Hochrechnung und Rundung auf die Werte der Variablen V2 prozentuiert. Danach wird der größte der Rundungsfaktoren 100, 50, 10 gewählt, bei dem sich diese Prozentwerte um weniger als 0,5 verändert haben.

R(k **)** Diese Angabe macht die Zeile *k* zur Basis der Prozentuierung für den Prozente-Vergleich. Zum Beispiel:

```
-PROSUM-ROWS(R(2..4))=V1 ROUND=(100 10 5)R(1) 0.1
```

Die Werte der Zeilen 2 bis 4 werden vor und nach Hochrechnung und Rundung auf die Werte der Zeile 1 prozentuiert. Danach wird der größte der Rundungsfaktoren 100, 10, 5 gewählt, bei dem sich diese Prozentwerte um weniger als 0,1 verändert haben.

C(k **)** Diese Angabe macht die Spalte *k* zur Basis der Prozentuierung für den Prozente-Vergleich. Zum Beispiel:

```
-PROSUM-COLS(C(2..4))=V1 ROUND=(100 10 5)C(1) 0.1
```

Die Werte der Spalten 2 bis 4 werden vor und nach Hochrechnung und Rundung auf die Werte der Spalte 1 prozentuiert. Danach wird der größte der Rundungsfaktoren 100, 10, 5 gewählt, bei dem sich diese Prozentwerte um weniger als 0,1 verändert haben.

Vk Diese Angabe macht die Variable Vk zur Basis der Prozentuierung für den Prozente-Vergleich. Diese Variable muss in einer vorausgegangenen Wertezuweisung mit Daten gefüllt werden. Zum Beispiel:

```
XTAB ...
-V9=ROW1
XTAB ...
-PROSUM-ROWS(R(2..4))=R1 ROUND=(100 10 5)V9 0.1
```

Die Zeile 1 des ersten XTAB wird in die Variable V9 kopiert. Im zweiten XTAB werden die Werte der Zeilen 2 bis 4 vor und nach Hochrechnung und Rundung auf die Werte der Variablen V9 prozentuiert. Danach wird der größte der Rundungsfaktoren 100, 10, 5 gewählt, bei dem sich diese Prozentwerte um weniger als 0,1 verändern.

t Toleranzschwelle 0.0001 bis 100 um die sich die Prozentwerte vor und nach der Rundung maximal unterscheiden dürfen, damit ein Rundungsfaktor akzeptiert werden kann. Zum Beispiel:

```
-PROSUM-ROWS=V1  ROUND=(100 50 10 5) V2 0.5
```

Beispiel: PROSUM

Ein XTAB-Statement ohne PROSUM-Statement könnte folgende Tabelle liefern:

	Gesamt		Bayern		Niedersachsen	
Basis in Tsd	63 785		9 214		4 526	
Männer	1 164	44,7%	161	44,4%	81	44,3%
Frauen	1 438	55,3%	202	55,6%	102	55,7%

Die erste Zeile enthält Bevölkerungswerte in Tausend und die beiden folgenden Zeilen Werte aus der Stichprobe.

Das Folgestatement

```
-PROSUM-ROWS(R(2 3))=R(1)
```

zum gleichen XTAB erstellt daraus folgende Tabelle durch Hochrechnung:

	Gesamt		Bayern		Niedersachsen	
Basis in Tsd	63 785	100,0%	9 214	100,0%	4 526	100,0%
Männer	28 534	44,7%	4 087	44,4%	2 003	44,3%
Frauen	35 251	55,3%	5 127	55,6%	2 523	55,7%

In den Spalten 1 ... 3 werden die Summen der Zeilen 2 und 3 gebildet.
Die Werte der Zeilen 2 und 3 werden nun mit einem Faktor multipliziert, sodass ihre Summe mit dem Wert der ersten Zeile übereinstimmt. Für die erste Spalte ist dies der Faktor

24,514 = 63785 / 2602 = 63785 / (1164 + 1438)

Das Folgestatement

```
-PROSUM-ROWS(R(2 3))=R(1) ROUND=100
```

erzeugt dagegen eine Tabelle mit Hochrechnung und Rundung:

	Gesamt		Bayern		Niedersachsen	
Basis in Tsd	63 785	100,0%	9 214	100,0%	4 526	100,0%
Männer	28 500	44,7%	4 100	44,5%	2 000	44,2%
Frauen	35 300	55,3%	5 100	55,4%	2 500	55,2%

Zunächst wird auch hier die Hochrechnung wie in der vorigen Tabelle ausgeführt. Die neuen Werte werden jedoch zusätzlich auf 100 gerundet und so korrigiert, dass ihre Summe mit dem gerundeten Wert der ersten Zeile übereinstimmt. Diese Korrektur wird so vorgenommen dass sich in den Zeilen 2 oder 3 die Prozentwerte möglichst wenig ändern.

Mit den Statements

```
-PROSUM-ROWS(R(2 3))=R(1) ROUND=100
-ROUND(R(1))=100
```

wird auch die erste Zeile noch auf 100 gerundet:

	Gesamt		Bayern		Niedersachsen	
Basis in Tsd	63 800	100,0%	9 200	100,0%	4 500	100,0%
Männer	28 500	44,7%	4 100	44,6%	2 000	44,4%
Frauen	35 300	55,3%	5 100	55,4%	2 500	55,6%

Beispiel: PROSUM mit Erhalt von Prozentwerten

Ein XTAB-Statement ohne PROSUM-Statement könnte folgende Tabelle liefern:

	Gesamt		Bayern		Niedersachsen	
Basis in Tsd	63 785		9 214		4 526	
Männer	1 164	44,7%	161	44,4%	81	44,3%
Frauen	1 438	55,3%	202	55,6%	102	55,7%

Die erste Zeile enthält Bevölkerungswerte in Tausend und die beiden folgenden Zeilen Werte aus der Stichprobe.

Das Folgestatement

```
-PROSUM-ROWS(R(2 3))=R(1)
```

zum gleichen XTAB erstellt daraus eine Tabelle mit Hochrechnung:

	Gesamt		Bayern		Niedersachsen	
Basis in Tsd	63 785	100,0%	9 214	100,0%	4 526	100,0%
Männer	28 534	44,7%	4 087	44,4%	2 003	44,3%
Frauen	35 251	55,3%	5 127	55,6%	2 523	55,7%

Das Folgestatement

```
-PROSUM-ROWS(R(2 3))=R(1) ROUND=(100,50,10) R(1)0.2
```

liefert eine Tabelle mit Hochrechnung und Rundung, bei der versucht wird, Prozentwerte soweit wie möglich zu erhalten:

	Gesamt		Bayern		Niedersachsen	
Basis in Tsd	63 785	100,0%	9 214	100,0%	4 526	100,0%
Männer	28 540	44,7%	4 080	44,3%	2 010	44,4%
Frauen	35 250	55,3%	5 130	55,7%	2 520	55,7%

Zunächst wird auch hier die Hochrechnung wie in der vorigen Tabelle ausgeführt. Danach werden die neuen Werte nacheinander mit den Faktoren 100, 50 und 10 gerundet und so korrigiert, dass ihre Summe mit dem gerundeten Wert der ersten Zeile übereinstimmt.

Wegen der Angabe *ROUND=(100,50,10) R(1) 0.2* wird der größte der drei Rundungsfaktoren verwendet, bei dem die Prozentwerte sich um weniger als 0.2 von den Prozentwerten ohne Rundung unterscheiden. In diesem Beispiel wird der Rundungsfaktor 10 gewählt, um die prozentuale Abweichung unter der angegebenen Schwelle zu halten. Prozentuierungsbasis ist dabei wegen R(1) hinter ROUND die erste Zeile.

Anhang

CNTA

25. April 2018

CNTA-Handbuch
Stand 25. April 2018

Eulenkrugstraße 83
22359 Hamburg

Tel: (0 40) 6 03 05 62
E-Mail: info@wesselhoeft.de

Zeichenzuordnung ASCII, ANSI und EBCDIC

dezimal	oktal	hexa-dezimal	binär	Strg	ANSI		EBCDIC 1141	ASCII
0	000	00	00000000	^@	End Of Character String **\0**	NUL	NUL	NUL
1	001	01	00000001	^A	Start Of Heading	SOH	SOH	SOH
2	002	02	00000010	^B	Start Of Text	STX	STX	STX
3	003	03	00000011	^C	End Of Text	ETX	ETX	ETX
4	004	04	00000100	^D	End Of Transmission	EOT	SEL	EOT
5	005	05	00000101	^E	Enquiry	ENQ	HT	ENQ
6	006	06	00000110	^F	Acknowledge	ACK	RNL	ACK
7	007	07	00000111	^G	Bell **\a**	BEL	DEL	BEL
8	010	08	00001000	^H	Backspace **\b**	BS	GE	BS
9	011	09	00001001	^I	Horizontal Tab **\t**	HT	SPS	HT
10	012	0A	00001010	^J	Line Feed **\n**	LF	RPT	LF
11	013	0B	00001011	^K	Vertical Tab **\v**	VT	VT	VT
12	014	0C	00001100	^L	Form Feed **\f**	FF	FF	FF
13	015	0D	00001101	^M	Carriage Return **\r**	CR	CR	CR
14	016	0E	00001110	^N	Shift Out	SO	SO	SO
15	017	0F	00001111	^O	Shift In	SI	SI	SI
16	020	10	00010000	^P	Data Link Escape	DLE	DLE	DLE
17	021	11	00010001	^Q	Device Control 1 (XON)	DC1	DC1	DC1
18	022	12	00010010	^R	Device Control 2	DC2	DC2	DC2
19	023	13	00010011	^S	Device Control 3 (XOFF)	DC3	DC3	DC3
20	024	14	00010100	^T	Device Control 4	DC4	RES/ENP	¶
21	025	15	00010101	^U	No Acknowledge	NAK	NL	§
22	026	16	00010110	^V	Synchronous Idle	SYN	BS	SYN
23	027	17	00010111	^W	End Transmission Blocks	ETB	POC	ETB
24	030	18	00011000	^X	Cancel	CAN	CAN	CAN
25	031	19	00011001	^Y	End Of Medium	EM	EM	EM
26	032	1A	00011010	^Z	End Of Text File	SUB	UBS	SUB
27	033	1B	00011011	^[	Escape	ESC	CU1	ESC
28	034	1C	00011100	^\	File Separator	FS	IFS	FS
29	035	1D	00011101	^]	Group Separator	GS	IGS	GS
30	036	1E	00011110	^^	Record Separator	RS	IRS	RS
31	037	1F	00011111	^_	Unit Separator	US	IUS/ITB	US

dezimal	oktal	hexa-dezimal	binär	ASCII	ANSI	EBCDIC 1141
32	040	20	00100000	blank	blank	DS
33	041	21	00100001	!	!	SOS
34	042	22	00100010	"	"	FS
35	043	23	00100011	#	#	WUS
36	044	24	00100100	$	$	BIP/INP
37	045	25	00100101	%	%	LF
38	046	26	00100110	&	&	ETB
39	047	27	00100111	'	'	ESC
40	050	28	00101000	(	(	SA
41	051	29	00101001	)	)	
42	052	2A	00101010	*	*	SM/SW
43	053	2B	00101011	+	+	CSP
44	054	2C	00101100	,	,	MFA
45	055	2D	00101101	-	-	ENQ
46	056	2E	00101110	.	.	ACK
47	057	2F	00101111	/	/	BEL
48	060	30	00110000	0	0	
49	061	31	00110001	1	1	
50	062	32	00110010	2	2	SYN
51	063	33	00110011	3	3	IR
52	064	34	00110100	4	4	PP
53	065	35	00110101	5	5	TRN
54	066	36	00110110	6	6	NBS
55	067	37	00110111	7	7	EOT
56	070	38	00111000	8	8	SBS
57	071	39	00111001	9	9	IT
58	072	3A	00111010	:	:	RFF
59	073	3B	00111011	;	;	CU3
60	074	3C	00111100	<	<	DC4
61	075	3D	00111101	=	=	NAK
62	076	3E	00111110	>	>	
63	077	3F	00111111	?	?	SUB

dezimal	oktal	hexa-dezimal	binär	ASCII	ANSI	EBCDIC 1141
64	100	40	01000000	@	@	blank
65	101	41	01000001	A	A	
66	102	42	01000010	B	B	â
67	103	43	01000011	C	C	{
68	104	44	01000100	D	D	à
69	105	45	01000101	E	E	á
70	106	46	01000110	F	F	ã
71	107	47	01000111	G	G	å
72	110	48	01001000	H	H	ç
73	111	49	01001001	I	I	ñ
74	112	4A	01001010	J	J	Ä
75	113	4B	01001011	K	K	.
76	114	4C	01001100	L	L	<
77	115	4D	01001101	M	M	(
78	116	4E	01001110	N	N	+
79	117	4F	01001111	O	O	!
80	120	50	01010000	P	P	&
81	121	51	01010001	Q	Q	é
82	122	52	01010010	R	R	ê
83	123	53	01010011	S	S	ë
84	124	54	01010100	T	T	è
85	125	55	01010101	U	U	í
86	126	56	01010110	V	V	î
87	127	57	01010111	W	W	ï
88	130	58	01011000	X	X	ì
89	131	59	01011001	Y	Y	~
90	132	5A	01011010	Z	Z	Ü
91	133	5B	01011011	[	[	$
92	134	5C	01011100	\	\	*
93	135	5D	01011101	]	]	)
94	136	5E	01011110	^	^	;
95	137	5F	01011111	_	_	^

dezimal	oktal	hexa-dezimal	binär	ASCII	ANSI	EBCDIC 1141
96	140	60	01100000	`	`	-
97	141	61	01100001	a	a	/
98	142	62	01100010	b	b	Â
99	143	63	01100011	c	c	[
100	144	64	01100100	d	d	À
101	145	65	01100101	e	e	Á
102	146	66	01100110	f	f	Ã
103	147	67	01100111	g	g	Å
104	150	68	01101000	h	h	Ç
105	151	69	01101001	i	i	Ñ
106	152	6A	01101010	j	j	ö
107	153	6B	01101011	k	k	,
108	154	6C	01101100	l	l	%
109	155	6D	01101101	m	m	_
110	156	6E	01101110	n	n	>
111	157	6F	01101111	o	o	?
112	160	70	01110000	p	p	ø
113	161	71	01110001	q	q	É
114	162	72	01110010	r	r	Ê
115	163	73	01110011	s	s	Ë
116	164	74	01110100	t	t	È
117	165	75	01110101	u	u	Í
118	166	76	01110110	v	v	Î
119	167	77	01110111	w	w	Ï
120	170	78	01111000	x	x	Ì
121	171	79	01111001	y	y	`
122	172	7A	01111010	z	z	:
123	173	7B	01111011	{	{	#
124	174	7C	01111100	|	|	§
125	175	7D	01111101	}	}	‘
126	176	7E	01111110	~	~	=
127	177	7F	01111111	DEL	DEL	"

dezimal	oktal	hexa-dezimal	binär	ASCII	ANSI	EBCDIC 1141		UNICODE	hexa-dezuimal
128	200	80	10000000	€	€	Ø		€	20 AC
129	201	81	10000001	ü	•	a		•	00 81
130	202	82	10000010	é	‚	b		‚	20 1A
131	203	83	10000011	â	*ƒ*	c		*ƒ*	01 92
132	204	84	10000100	ä	„	d		„	20 1E
133	205	85	10000101	à	…	e		…	20 26
134	206	86	10000110	å	†	f		†	20 20
135	207	87	10000111	ç	‡	g		‡	20 21
136	210	88	10001000	ê	ˆ	h		ˆ	02 C6
137	211	89	10001001	ë	‰	i		‰	20 30
138	212	8A	10001010	è	Š	«		Š	01 60
139	213	8B	10001011	ï	‹	»		‹	20 39
140	214	8C	10001100	î	Œ	ð		Œ	01 52
141	215	8D	10001101	ì	•	ý		•	00 8D
142	216	8E	10001110	Ä	Ž	þ		Ž	01 7D
143	217	8F	10001111	Å	•	±		•	00 8F
144	220	90	10010000	É	•	°		•	00 90
145	221	91	10010001	æ	‘	j		‘	20 18
146	222	92	10010010	Æ	’	k		’	20 19
147	223	93	10010011	ô	“	l		“	20 1C
148	224	94	10010100	ö	”	m		”	20 1D
149	225	95	10010101	ò	•	n		•	20 22
150	226	96	10010110	û	–	o		–	20 13
151	227	97	10010111	ù	—	p		—	20 14
152	230	98	10011000	ÿ	˜	q		˜	02 DC
153	231	99	10011001	Ö	™	r		™	21 22
154	232	9A	10011010	Ü	š	ª		š	01 61
155	233	9B	10011011	ø	›	º		›	20 3A
156	234	9C	10011100	£	œ	æ		œ	01 53
157	235	9D	10011101	Ø	•	¸		•	00 9D
158	236	9E	10011110	×	ž	Æ		ž	01 7E
159	237	9F	10011111	*ƒ*	Ÿ	€		Ÿ	01 78

dezimal	oktal	hexa-dezimal	binär	ASCII	ANSI	EBCDIC 1141		UNICODE	hexa-dezimal
160	240	A0	10100000	á	NBSP	µ		NBSP	00 A0
161	241	A1	10100001	í	¡	ß		¡	00 A1
162	242	A2	10100010	ó	¢	s		¢	00 A2
163	243	A3	10100011	ú	£	t		£	00 A3
164	244	A4	10100100	ñ	¤	u		¤	00 A4
165	245	A5	10100101	Ñ	¥	v		¥	00 A5
166	246	A6	10100110	ª	¦	w		¦	00 A6
167	247	A7	10100111	º	§	x		§	00 A7
168	250	A8	10101000	¿	¨	y		¨	00 A8
169	251	A9	10101001	®	©	z		©	00 A9
170	252	AA	10101010	¬	ª	¡		ª	00 AA
171	253	AB	10101011	½	«	¿		«	00 AB
172	254	AC	10101100	¼	¬	Ð		¬	00 AC
173	255	AD	10101101	¡	SHY	Ý		SHY	00 AD
174	256	AE	10101110	«	®	Þ		®	00 AE
175	257	AF	10101111	»	¯	®		¯	00 AF
176	260	B0	10110000	Š	°	¢		°	00 B0
177	261	B1	10110001	š	±	£		±	00 B1
178	262	B2	10110010	™	²	¥		²	00 B2
179	263	B3	10110011	Ž	³	·		³	00 B3
180	264	B4	10110100	œ	´	©		´	00 B4
181	265	B5	10110101	Á	µ	@		µ	00 B5
182	266	B6	10110110	Â	¶	¶		¶	00 B6
183	267	B7	10110111	À	·	¼		·	00 B7
184	270	B8	10111000	©	¸	½		¸	00 B8
185	271	B9	10111001	Ç	¹	¾		¹	00 B9
186	272	BA	10111010		º	¬		º	00 BA
187	273	BB	10111011	Ÿ	»			»	00 BB
188	274	BC	10111100	›	¼	¯		¼	00 BC
189	275	BD	10111101	¢	½	¨		½	00 BD
190	276	BE	10111110	¥	¾	´		¾	00 BE
191	277	BF	10111111	‹	¿	×		¿	00 BF

dezimal	oktal	hexa-dezimal	binär	ASCII	ANSI	EBCDIC 1141		UNICODE	hexa-dezimal
192	300	C0	11000000	…	À	ä		À	00 C0
193	301	C1	11000001	Œ	Á	A		Á	00 C1
194	302	C2	11000010	—	Â	B		Â	00 C2
195	303	C3	11000011	•	Ã	C		Ã	00 C3
196	304	C4	11000100	–	Ä	D		Ä	00 C4
197	305	C5	11000101	ˆ	Å	E		Å	00 C5
198	306	C6	11000110	ã	Æ	F		Æ	00 C6
199	307	C7	11000111	Ã	Ç	G		Ç	00 C7
200	310	C8	11001000	”	È	H		È	00 C8
201	311	C9	11001001	“	É	I		É	00 C9
202	312	CA	11001010	†	Ê			Ê	00 CA
203	313	CB	11001011	†	Ë	ô		Ë	00 CB
204	314	CC	11001100	‰	Ì	¦		Ì	00 CC
205	315	CD	11001101	„	Í	ò		Í	00 CD
206	316	CE	11001110	-	Î	ó		Î	00 CE
207	317	CF	11001111	¤	Ï	õ		Ï	00 CF
208	320	D0	11010000	ð	Ð	ü		Ð	00 D0
209	321	D1	11010001	Ð	Ñ	J		Ñ	00 D1
210	322	D2	11010010	Ê	Ò	K		Ò	00 D2
211	323	D3	11010011	Ë	Ó	L		Ó	00 D3
212	324	D4	11010100	È	Ô	M		Ô	00 D4
213	325	D5	11010101	˜	Õ	N		Õ	00 D5
214	326	D6	11010110	Í	Ö	O		Ö	00 D6
215	327	D7	11010111	Î	×	P		×	00 D7
216	330	D8	11011000	Ï	Ø	Q		Ø	00 D8
217	331	D9	11011001	‘	Ù	R		Ù	00 D9
218	332	DA	11011010	’	Ú	¹		Ú	00 DA
219	333	DB	11011011	¦	Û	û		Û	00 DB
220	334	DC	11011100		Ü	}		Ü	00 DC
221	335	DD	11011101		Ý	ù		Ý	00 DD
222	336	DE	11011110	Ì	Þ	ú		Þ	00 DE
223	337	DF	11011111		ß	ÿ		ß	00 DF

dezimal	oktal	hexa-dezimal	binär	ASCII	ANSI	EBCDIC 1141		UNICODE	hexa-dezimal
224	340	E0	11100000	Ó	à	Ö		à	00 E0
225	341	E1	11100001	ß	á	÷		á	00 E1
226	342	E2	11100010	Ô	â	S		â	00 E2
227	343	E3	11100011	Ò	ã	T		ã	00 E3
228	344	E4	11100100	õ	ä	U		ä	00 E4
229	345	E5	11100101	Õ	å	V		å	00 E5
230	346	E6	11100110	µ	æ	W		æ	00 E6
231	347	E7	11100111	þ	ç	X		ç	00 E7
232	350	E8	11101000	Þ	è	Y		è	00 E8
233	351	E9	11101001	Ú	é	Z		é	00 E9
234	352	EA	11101010	Û	ê	²		ê	00 EA
235	353	EB	11101011	Ù	ë	Ô		ë	00 EB
236	354	EC	11101100	ý	ì	\		ì	00 EC
237	355	ED	11101101	Ý	í	Ò		í	00 ED
238	356	EE	11101110	¯	î	Ó		î	00 EE
239	357	EF	11101111	´	ï	Õ		ï	00 EF
240	360	F0	11110000		ð	0		ð	00 F0
241	361	F1	11110001	±	ñ	1		ñ	00 F1
242	362	F2	11110010		ò	2		ò	00 F2
243	363	F3	11110011	¾	ó	3		ó	00 F3
244	364	F4	11110100		ô	4		ô	00 F4
245	365	F5	11110101		õ	5		õ	00 F5
246	366	F6	11110110	÷	ö	6		ö	00 F6
247	367	F7	11110111	¸	÷	7		÷	00 F7
248	370	F8	11111000	°	ø	8		ø	00 F8
249	371	F9	11111001	¨	ù	9		ù	00 F9
250	372	FA	11111010	·	ú	³		ú	00 FA
251	373	FB	11111011	¹	û	Û		û	00 FB
252	374	FC	11111100	³	ü	]		ü	00 FC
253	375	FD	11111101	²	ý	Ù		ý	00 FD
254	376	FE	11111110		þ	Ú		þ	00 FE
255	377	FF	11111111	ž	ÿ			ÿ	00 FF

Zeichenverschlüsselung im UTF8-Code

Der von den CNT-Programmen verwendete UTF8-Code umfasst alle Zeichen des 16-Bit-UNICODE, praktisch alle international vorkommenden Zeichensätze.
UTF8 verschlüsselt die 16 Bits der UNICODE-Zeichen auf folgende Weise, wobei xxx ... für die Bits des UNICODE-Zeichens steht:

UNICODE hexadezimal	**UTF8 in Bits**	
0000 – 007F	0xxxxxxx	Diese Zeichen des UTF8-Codes stimmen mit denen des ASCII- und ANSI-Codes überein.
0080 – 07FF	110xxxxx 10xxxxxx	Im UTF8-Code beginnt das erste Byte hier stets mit 11 und alle folgenden mit 10.
0800 – FFFF	1110xxxx 10xxxxxx 10xxxxxx	

COLBIN-Spalten im K-Format

Diese Tabelle zeigt zu den hexadezimalen Werten zweier benachbarter Bytes die dazugehörigen Werte im COLBIN-Code des K-Formats. Werte, die nicht in dieser Tabelle vorkommen, sind keine gültigen COLBIN-Zeichen. Zum Beispiel entsprechen die hexadezimalen Werte 08 21 in den beiden Bytes einer COLBIN-Spalte den Werten 0, 4 und 9 .im K-Format.

linkes Byte			rechtes Byte		
hexadezimal	K-Format	binär	hexadezimal	K-Format	binär
00	. . . . .	0000 0000	00	. . . . .	0000 0000
01	 3	0000 0001	01	 9	0000 0001
02	. . . . 2 .	0000 0010	02	. . . . 8 .	0000 0010
03	. . . . 2 3	0000 0011	03	. . . . 8 9	0000 0011
04	. . . 1 . .	0000 0100	04	. . . 7 . .	0000 0100
05	. . . 1 . 3	0000 0101	05	. . . 7 . 9	0000 0101
06	. . . 1 2 .	0000 0110	06	. . . 7 8 .	0000 0110
07	. . . 1 2 3	0000 0111	07	. . . 7 8 9	0000 0111
08	. . 0 . . .	0000 1000	08	. . 6 . . .	0000 1000
09	. . 0 . . 3	0000 1001	09	. . 6 . . 9	0000 1001
0A	. . 0 . 2 .	0000 1010	0A	. . 6 . 8 .	0000 1010
0B	. . 0 . 2 3	0000 1011	0B	. . 6 . 8 9	0000 1011
0C	. . 0 1 . .	0000 1100	0C	. . 6 7 . .	0000 1100
0D	. . 0 1 . 3	0000 1101	0D	. . 6 7 . 9	0000 1101
0E	. . 0 1 2 .	0000 1110	0E	. . 6 7 8 .	0000 1110
0F	. . 0 1 2 3	0000 1111	0F	. . 6 7 8 9	0000 1111
10	. X	0001 0000	10	. 5	0001 0000
11	. X . . . 3	0001 0001	11	. 5 . . . 9	0001 0001
12	. X . . 2 .	0001 0010	12	. 5 . . 8 .	0001 0010
13	. X . . 2 3	0001 0011	13	. 5 . . 8 9	0001 0011
14	. X . 1 . .	0001 0100	14	. 5 . 7 . .	0001 0100
15	. X . 1 . 3	0001 0101	15	. 5 . 7 . 9	0001 0101
16	. X . 1 2 .	0001 0110	16	. 5 . 7 8 .	0001 0110
17	. X . 1 2 3	0001 0111	17	. 5 . 7 8 9	0001 0111
18	. X 0 . . .	0001 1000	18	. 5 6 . . .	0001 1000
19	. X 0 . . 3	0001 1001	19	. 5 6 . . 9	0001 1001
1A	. X 0 . 2 .	0001 1010	1A	. 5 6 . 8 .	0001 1010
1B	. X 0 . 2 3	0001 1011	1B	. 5 6 . 8 9	0001 1011
1C	. X 0 1 . .	0001 1100	1C	. 5 6 7 . .	0001 1100
1D	. X 0 1 . 3	0001 1101	1D	. 5 6 7 . 9	0001 1101
1E	. X 0 1 2 .	0001 1110	1E	. 5 6 7 8 .	0001 1110
1F	. X 0 1 2 3	0001 1111	1F	. 5 6 7 8 9	0001 1111
20	Y	0010 0000	20	4	0010 0000
21	Y 3	0010 0001	21	4 9	0010 0001
22	Y . . . 2 .	0010 0010	22	4 . . . 8 .	0010 0010
23	Y . . . 2 3	0010 0011	23	4 . . . 8 9	0010 0011
24	Y . . 1 . .	0010 0100	24	4 . . 7 . .	0010 0100
25	Y . . 1 . 3	0010 0101	25	4 . . 7 . 9	0010 0101
26	Y . . 1 2 .	0010 0110	26	4 . . 7 8 .	0010 0110
27	Y . . 1 2 3	0010 0111	27	4 . . 7 8 9	0010 0111
28	Y . 0 . . .	0010 1000	28	4 . 6 . . .	0010 1000
29	Y . 0 . . 3	0010 1001	29	4 . 6 . . 9	0010 1001
2A	Y . 0 . 2 .	0010 1010	2A	4 . 6 . 8 .	0010 1010
2B	Y . 0 . 2 3	0010 1011	2B	4 . 6 . 8 9	0010 1011
2C	Y . 0 1 . .	0010 1100	2C	4 . 6 7 . .	0010 1100
2D	Y . 0 1 . 3	0010 1101	2D	4 . 6 7 . 9	0010 1101
2E	Y . 0 1 2 .	0010 1110	2E	4 . 6 7 8 .	0010 1110
2F	Y . 0 1 2 3	0010 1111	2F	4 . 6 7 8 9	0010 1111
30	Y X	0011 0000	30	4 5	0011 0000
31	Y X . . . 3	0011 0001	31	4 5 . . . 9	0011 0001
32	Y X . . 2 .	0011 0010	32	4 5 . . 8 .	0011 0010
33	Y X . . 2 3	0011 0011	33	4 5 . . 8 9	0011 0011
34	Y X . 1 . .	0011 0100	34	4 5 . 7 . .	0011 0100
35	Y X . 1 . 3	0011 0101	35	4 5 . 7 . 9	0011 0101
36	Y X . 1 2 .	0011 0110	36	4 5 . 7 8 .	0011 0110
37	Y X . 1 2 3	0011 0111	37	4 5 . 7 8 9	0011 0111
38	Y X 0 . . .	0011 1000	38	4 5 6 . . .	0011 1000
39	Y X 0 . . 3	0011 1001	39	4 5 6 . . 9	0011 1001
3A	Y X 0 . 2 .	0011 1010	3A	4 5 6 . 8 .	0011 1010
3B	Y X 0 . 2 3	0011 1011	3B	4 5 6 . 8 9	0011 1011
3C	Y X 0 1 . .	0011 1100	3C	4 5 6 7 . .	0011 1100
3D	Y X 0 1 . 3	0011 1101	3D	4 5 6 7 . 9	0011 1101
3E	Y X 0 1 2 .	0011 1110	3E	4 5 6 7 8 .	0011 1110
3F	Y X 0 1 2 3	0011 1111	3F	4 5 6 7 8 9	0011 1111

Zeichenverschlüsselung im Quantum-Format

Diese Tabelle zeigt die im Quantum-Format verwendete Zuordnung der ASCII-Zeichen zu den Wertekombinationen des K-Formats. Sie stammt aus den Lochkartencodes IBM-029.

```
    _____________________________________________________________________________
    /&-0123456789ABCDEFGHIJKLMNOPQR/STUVWXYZ:#@'>"¢.<(+|!$*);¬ ,%_>?abcdefghijklmnopqrstuvwxyz
12 / Y           YYYYYYYYY                    YYYYYY           YYYYYYYYYYYYYYYYYY
11|   X                   XXXXXXXXX                   XXXXXX              XXXXXXXXXXXXXXXXX
 0|    0                           000000000                 000000000000000         00000000
 1|     1        1        1        1                                1        1
 2|      2        2        2        2       2     2     2     2      2        2       2
 3|       3        3        3        3       3     3     3     3      3        3       3
 4|        4        4        4        4       4     4     4     4      4        4       4
 5|         5        5        5        5       5     5     5     5      5        5       5
 6|          6        6        6        6       6     6     6     6      6        6       6
 7|           7        7        7        7       7     7     7     7      7        7       7
 8|            8        8        8        8 888888888888888888888888       8        8       8
 9|             9        9        9        9                                9        9       9
  |_____________________________________________________________________________
```

Quantum-Datensätze beginnen mit einem Datenbyte pro Spalte.
Für die Wertekombinationen der obigen Tabelle wird das entsprechende ASCII-Zeichen in das Datenbyte gestellt.
Die einzelnen Datensätze werden durch Zeilenende-Zeichen abgeschlossen und Leerzeichen am Ende eines Datensatzes nicht ausgegeben.
Enthält ein Datensatz Zeichenkombinationen, die nicht in der obigen Tabelle enthalten sind, so wird hinter die letzte nicht leere Datenspalte das hexadezimale Byte 7F gestellt.
Dahinter erscheinen die nicht passenden Zeichenkombinationen im COLBIN-Format mit zwei Bytes pro Spalte. Das ursprüngliche Byte enthält dafür einen Stern *. Die Ausgabe der COLBIN-Spalten geschieht in der Reihenfolge der dazugehörigen Sterne im vorderen Teil der Datensätze.
Ist ein Stern als Datenwert auszugeben, so wird dieser ebenfalls im COLBIN-Format in den hinteren Teil des Datensatzes gestellt.

Zeichenverschlüsselung im COLBIN- und EBCDIC-Format

EBCDIC	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
				space	&	–											
00					space	&	–									0	0
01							/		a	j			A	J		1	1
02									b	k	s		B	K	S	2	2
03									c	l	t		C	L	T	3	3
04									d	m	u		D	M	U	4	4
05									e	n	v		E	N	V	5	5
06									f	o	w		F	O	W	6	6
07									g	p	x		G	P	X	7	7
08									h	q	y		H	Q	Y	8	8
09									i	r	z		I	R	Z	9	9
0A					¢	!		:									2 8
0B					.	$	,	#									3 8
0C					<	*	%	@									4 8
0D					(	)	_	'									5 8
0E					+	;	>	=									6 8
0F					|	¬	?	"									7 8
	Y 9	X 9	0 9		Y	X	0		Y 0	Y X	0 X		Y	X	0		COLBIN

Liste der Fehlermeldungen

001 Fehler in den Statements.

Es wurden Fehler in den Statements festgestellt, die zu einem vorzeitigen Programmende führten. Mit Hilfe des Zählprotokolls können die Fehler beseitigt und der Programmlauf wiederholt werden.

002 Programmabbruch durch Benutzer.

Das Programm wurde durch einen Benutzereingriff vorzeitig beendet.

003 Fehlergrenze überschritten.

In diesem Datendurchlauf wurden zu viele Fehler- oder PRT-Meldungen gedruckt.
Die Verarbeitung wurde vorzeitig beendet.
Mit der Angabe ELIMIT im Statement OPTIONS lässt sich die Anzahl der zulässigen Fehlermeldungen verändern.

004 Eingabedatei für Statements nicht gefunden.

Mögliche Ursachen:
Die angegebene Datei existiert nicht.
Der Name der Datei wurde falsch geschrieben.
Die Datei befindet sich auf einem anderen Laufwerk oder in einem anderen Verzeichnis.

005 Kein ausführbares Statement in der Eingabedatei

Es wurde kein ausführbares Statement gefunden, höchstens Kommentare oder Makros.

006 Datei INPUT-EXT oder FETCH-FILE wurde entfernt.

Auf diese Datei kann nicht mehr zugegriffen werden obwohl sie zu Beginn der Verarbeitung noch vorhanden war. Mögliche Ursachen:

- Die Datei wurde gelöscht oder umbenannt.
- Der Datenträger mit der Datei wurde entfernt.
- Der Zugriff auf die Datei über ein Netzwerk wurde unterbrochen.

007 Datenfehler bei der Verarbeitung eines WEIGHT Statements

Mögliche Ursachen:

- Es wurde eine Variable Cn aus den Wichtungsbedingungen mit fehlenden Angaben oder mit Mehrfachnennungen gefunden.
- Es wurde ein statistischer Fall mit negativem Fallgewicht OLD gefunden.
- Zu diesen Fehlern enthält das Verarbeitungsprotokoll genauere Angaben.

008 Warnung: Sollwerte nicht erreicht.

Die Fallgewichte für WEIGHT konnten nicht genau zu den vorgegebenen Sollwerten ermittelt werden.
Die Gewichte werden trotzdem in die Zielvariable NEW oder NEWFACT ausgegeben. Mögliche Ursachen:

- Die Zielvariable NEW oder NEWFACT besitzt zu wenig Dezimal- oder Nachkommastellen.
- Die Angabe MIN ist zu groß oder MAX zu klein gewählt.
- Die vorgegebenen Sollwerte weichen stark von den Ist-Werten.
- Die Fallzahl ist zu klein, um das Ziel zu erreichen.

009 Warnung: Nicht besetzte Zellen

Bei der Ermittlung der Istwerte für ein WEIGHT Statement wurden leere Wichtungszellen gefunden. Die betroffenen Variablen und deren Merkmale werden hinter dieser Meldung aufgeführt. Die Verarbeitung wird fortgesetzt, indem die Sollwerte der leeren Zellen nachträglich den Wert 0 erhalten.

010 Statementeingabe und Zählprotokoll sind gleich

Die Statementdatei und die Protokolldatei besitzen den gleichen Dateinamen.

011 Falsche Konsoldatei

Beim Aufruf von CNTA wurde als dritter Parameter eine Datei zur Ausgabe der Betriebsmeldungen angegeben. Bei der Einrichtung dieser Datei wurde einer der folgenden Fehler festgestellt:

- Der beim Aufruf des Programms angegebene Dateiname ist falsch.
- Die Datei wird von einem anderen Programm bearbeitet.
- Der Plattenspeicher ist erschöpft.
- Der Zugriff auf die Datei über ein Netzwerk wurde unterbrochen.

012 Name der Bildschirmdatei mehrfach verwendet

Beim Aufruf von CNTA wurde eine Datei zur Aufnahme der Bildschirmmeldungen angegeben. Der Name dieser Datei ist aber identisch mit der Datei der Eingabestatements oder der Datei für die Druckausgabe.

013 Datenbereich für transponierte Variablenwerte erschöpft

In OUTPUT-INT ist TRANS oder TRANSONLY angegeben. Die transponierten Variablenwerte benötigen über 40.000.000.000 Bytes in der Datei. Das Problem lässt sich nur durch weniger Variable, Variable mit kleineren Datenwerten oder eine geringere Fallzahl in der Ausgabedatei lösen.

014 Zu wenig Hauptspeicher im heap

Für den aktuellen Verarbeitungslauf konnte nicht genügend Hauptspeicher beschafft werden. Vermutlich ist die Aufgabe zu umfangreich und sollte reduziert werden.

015 Hauptspeicher zu klein für die Auswertungsphase

Der verfügbare Hauptspeicher reicht nicht aus, um das Programm für den nächsten Datendurchlauf aufzunehmen. Mögliche Ursachen:

- es existieren zu viele Folgestatements zu ADD-PROC, MOD-PROC oder UPD-PROC,
- es wurden zu viele Variable definiert,
- viele Variablen wurden zu groß definiert,
- die Datensätze aus INPUT-EXT, INPUT-INT oder INPUT-SEC sind zu lang,
- in der Angabe RC von INPUT-EXT besitzt zu viele Satz-Kennzeichen,
- der Zählbereich für OPTIONS ESUM oder ESUMS ist zu groß.
- der verfügbare Hauptspeicher ist über die Angabe MAXWORK in den Dateien CNTA.INI oder CNTW.INI vergrößern.

016 Satzlänge in INPUT-EXT falsch

Im Statement INPUT-EXT liegt keine Angabe SIZE mit EOL vor. Dann muss die Dateilänge ein ganzes Vielfaches der Satzlänge sein. Das ist hier aber nicht der Fall.
Mögliche Ursachen:

- Es wurde keine oder eine falsche Satzlänge SIZE angegeben.
- Es liegt eine ASCII-Datei mit Zeilenende-Zeichen vor (CR und LF); dann ist aber die Angabe EOL bei SIZE erforderlich.

017 Zu wenig Hauptspeicher

Der Hauptspeicher reicht nicht für einen minimalen Auswertungslauf aus.
Durch Änderungen an den Statements lässt sich das Problem nicht beheben.
Stattdessen sollte die Angabe MAXWORK in den Dateien CNTA.INI oder CNTW.INI verändert werden.

018 Zu wenig Hauptspeicher für ein XTAB-Statement

Der verfügbare Hauptspeicher ist zu klein, um das laufende XTAB Statement zu bearbeiten.
Vermutlich gibt es zu viele Angaben in ROW, COL oder FILTER. Außerdem lässt sich der verfügbare Hauptspeicher über die Angabe MAXWORK in den Dateien CNTA.INI oder CNTW.INI vergrößern.

019 Zu wenig Hauptspeicher für die Aufbereitung der Auswertungsergebnisse

Der verfügbare Hauptspeicher ist zu klein für die Aufbereitung der Auswertungsergebnisse. Vielleicht wurden im Statement PAGE zu große Werte für POS oder LINES angegeben: Im Hauptspeicher wird stets Platz für ein volles Blatt reserviert. Außerdem lässt sich der verfügbare Hauptspeicher über die Angabe MAXWORK in den Dateien CNTA.INI oder CNTW.INI vergrößern.

020 Datensatz in INPUT-INT, INPUT-SEC oder FETCH falsch

Die Eingabedatei INPUT-INT, INPUT-SEC oder FETCH enthält einen zerstörten Datensatz. Diese Datei lässt sich nicht mehr verarbeiten. Sie muss neu erstellt werden.

021 Zu wenig Hauptspeicher für das Statement FUSION

Der verfügbare Hauptspeicher reicht nicht aus, um die Verbindung zwischen Spender- und Empfänger-Fällen herzustellen. Abhilfe könnte ein zusätzliches Paar von Verbindungsvariablen mit der Angabe EQ schaffen. Außerdem lässt sich der verfügbare Hauptspeicher über die Angabe MAXWORK in den Dateien CNTA.INI oder CNTW.INI vergrößern.

022 Platte für OUTPUT-EXT voll

Die Speicherkapazität der Magnetplatte oder Diskette für die Datei OUTPUT-EXT ist erschöpft.

023 Zu viele Dateien eröffnet

Im MS/DOS ist die Datei CONFIG.SYS zu überprüfen. Dort ist eine
Angabe FILES=n erforderlich, mit einem Wert n größer oder gleich 20.

024 INCLUDE-Datei verloren

Während des Einlesens der Statements ist eine INCLUDE-Datei verloren gegangen. Vielleicht wurde während der Verarbeitung eine Diskette herausgenommen.

025 Zählprotokoll nicht eröffnet

Die Datei zur Aufnahme des Zählprotokolls konnte nicht eröffnet werden.

- Der beim Aufruf des Programms angegebene Dateiname ist falsch.
- Die Datei wird von einem anderen Programm bearbeitet.
- Der Plattenspeicher ist erschöpft.
- Der Zugriff auf die Datei über ein Netzwerk wurde unterbrochen.

026 Arbeitsdatei STRFILE ist zu groß

Die Arbeitsdatei *STRFILE* zur Speicherung von Zwischenergebnissen der laufenden Auswertung ist zu groß geworden. Entweder ist die Speicherkapazität der Platte erschöpft oder die Datei wurde größer als 4 Gigabyte.

027 Zu wenig Hauptspeicher

Der verfügbare Hauptspeicher reicht nicht aus, den nächsten Datendurchlauf auszuführen.
Vielleicht hilft es, ein zu umfangreiches Statement (XTAB, CODEBOOK, HOLECOUNT ...) in mehrere Teile zu zerlegen.

028 Zu wenig Hauptspeicher

Der Hauptspeicher ist zu klein, um das laufende Statement korrekt verarbeiten zu können.
Vielleicht lässt sich das Problem durch einen größeren Wert für MAXWORK in CNTA.INI oder CNTW.INI lösen.

029 Schutzstecker fehlt

Das Programm verlangt hier einen geeigneten Schutzstecker (Dongle) auf der parallelen Druckerschnittstelle. Ein solcher Stecker wurde nicht gefunden. Ohne diesen ist eine Verarbeitung nicht möglich. Vielleicht liegt auch ein Kontaktproblem vor: Stecker abziehen, wieder aufsetzen und gut verschrauben.

030 Zu wenig Hauptspeicher

Der Hauptspeicher reicht nicht aus, um die Cross Reference Liste zu XREF aus dem OPTIONS Statement auszugeben. Vielleicht lässt sich das Problem durch einen größeren Wert für MAXWORK in CNTA.INI oder CNTW.INI lösen.

031 Falscher Schutzstecker oder falsche Zentraleinheit

Die Verarbeitung ist nicht möglich. Es liegt eine der folgenden Ursachen vor:

- Das verwendete Programm ist für den Betrieb mit einem Schutzstecker (Dongle) eingerichtet. Ein solcher Stecker wurde gefunden, er ist aber nicht für dieses Programm geeignet.
- Das verwendete Programm ist für den Betrieb auf bestimmten Computern eingerichtet. Hier wurde versucht, das Programm auf einem anderen Computer zu betreiben.

032 Falsche Version einer Drucksteuerdatei

In einem Statement PAGEP wird durch die Angabe PRINTER = ... ein Seitendrucker angefordert. Die dazugehörige Drucksteuerdatei wie POSTSCRI.DRV, KYOCERA.DRV, . . . ist aber nicht auf dem neuesten Stand.

033 Schutzstecker ohne Spannung

An der Druckerschnittstelle des Schutzsteckers befindet sich ein Drucker, der die Steuerleitungen der Schnittstelle spannungsfrei geschaltet hat.

034 Fehler bei der Erstellung einer CSV-, DIF-, PDF- oder XML-Datei

Es wurde versucht, eine CSV-, DIF-, PDF- oder XML-Datei für ein Statement PAGE oder PAGEP einzurichten. Diese konnte mangels Schreibberechtigung nicht erstellt werden. Vermutlich greift zurzeit ein anderes Programm darauf zu.

035 Speichermangel im Verzeichnis CNTWWORK

Der Plattenspeicher für das Arbeitsverzeichnis CNTWWORK ist erschöpft.

036 Fehlerhaftes Logo

An dieser Stelle soll ein Logo ausgegeben werden.
Das Logo konnte aus einem der folgenden Gründen nicht verarbeitet werden:

- Zu der Logo-Nummer wurde weder ein fest definiertes Logo im Druckertreiber gefunden noch eine passende PCX- oder JPG-Datei.
- Die Logo-Datei besitzt kein geeignetes PCX- oder JPG-Format.
- Die Logo-Datei ist wegen einer sehr hohen Auflösung zu groß für das Programm.

Ist die Logo-Datei vorhanden, so sollte sie mit einem geeigneten Bildbearbeitungs-Programm in das JPG-Format umgeformt werden.

037 Fehler beim Lesen transponierter Daten

Die Eingabedatei INPUT-INT wurde mit der Angabe TRANS erstellt. Beim Lesen der transponierten Variablenwerte wurde ein Fehler festgestellt. Die Datei ist defekt und muss neu erstellt werden.

038 Die Druckdatei *.~P~ lässt sich nicht öffnen

Dies ist eine Datei zur Zwischenspeicherung der Druckdaten für CNTW.
Vielleicht wird sie gerade von einem anderen Programm verwendet.

039 Der Druckertreiber NULL.DRV lässt sich nicht öffnen

Diese Datei muss im gleichen Verzeichnis wie das laufende Programm.
Vielleicht wurde sie zerstört. Sie ist auf der Installationsdiskette von CNTA und CNTW enthalten.

040 Kein Zugriff auf den Schutzstecker möglich

Der Treiber für den Stecker fehlt oder ist nicht aktiviert.
Wahrscheinlich wurde das Programm nicht vollständig installiert.

041 Satzlänge in FETCH-FILE falsch

Im Statement FETCH-FILE liegt keine Angabe SIZE mit EOL vor. Dann muss die Dateilänge ein ganzes Vielfaches der Satzlänge sein. Das ist hier aber nicht der Fall.
Sind mehrere FETCH-FILE-Dateien vorhanden, so lässt sich die betroffene Datei aus der Fehlermeldung 520 im Zählprotokoll ermitteln. Mögliche Ursachen:

- Es wurde keine oder eine falsche Satzlänge SIZE angegeben.
- Es liegt eine ASCII-Datei mit Zeilenende-Zeichen vor (CR und LF); dann ist aber die Angabe EOL bei SIZE erforderlich.

042 PDF-Ausgabe nicht möglich

Die im Statement PAGE oder PAGEP angeforderte PDF-Datei wurde nicht erstellt.
Das dazu vorgesehene Programm *Ghostscript* wurde nicht gefunden oder ließ sich nicht starten. Die Angabe MAKEPDF in der Datei CNTA.INI oder CNTW.INI sollte überprüft werden.

043 Verarbeitung mit Fehler beendet

Ein von CNTW aufgerufenes externes Programm ist auf einen Fehler gestoßen.
Details dazu enthält das Zählprotokoll.

044 Zu viele transponierte Variable

Die Datei INPUT-INT enthält die Variablenwerte nur in transponierter Form (TRANSONLY).
Damit lassen sich maximal 1 000 Variable auswerten.

045 Fehler beim Transponieren von Variablen für OUTPUT-INT

Fehler beim Erstellen der temporären Datei *trans.~T~* im Arbeitsverzeichnis CNTWWORK.
Die Datei lässt sich nicht öffnen oder der Speicherplatz im Verzeichnis ist erschöpft.

046 Verarbeitung durch END ALL beendet

Die Verarbeitung wurde vorzeitig durch das Statement -END ALL beendet.

050 Statementdatei lässt sich nicht öffnen

Die Statementdatei *.~S~ konnte nicht geöffnet werden.

100 Ungültiges Statement

Die erste Eintragung in diesem Statement ist falsch.

101 Falsche Reihenfolge der Statements

Dieses Statement muss weiter vorne liegen. Siehe Abschnitt *Reihenfolge der Statements*.

102 Statement mehrmals vorhanden

Dieses Statement darf in einem Verarbeitungslauf nur einmal verwendet werden.

103 Hochkomma fehlt

An dieser Stelle wird ein Hochkomma erwartet.

104 Fehler vor dem Unterstreichungsstrich

Das Programm hat einen Fehler in den Statements gefunden, kann ihn aber nicht genauer beschreiben.

105 Dieses Statement endet vorzeitig

Es fehlen Angaben in dem Statement.
Vielleicht fehlt das abschließende Hochkomma hinter einem Text.

106 Hochkomma fehlt

Es wird ein in Hochkommas ' eingeschlossenes Satz-Kennzeichen erwartet.

107 Datenfeld RC liegt hinter Datensatz

Das Datenfeld für die Satz-Kennzeichen RC ragt über das Ende der Datensätze hinaus. Vielleicht fehlt auch die Angabe SIZE, oder sie ist zu klein.

108 Datenfeld RC, ID oder KEY zu lang

Für RC oder ID wurde ein zu langes Datenfeld angegeben. Die maximale Länge ist beim:

- A-Feld 127 Bytes oder Spalten
- D-Feld 18 Bytes oder Spalten
- F- und G-Feld 8 Bytes
- P-Feld 10 Bytes

109 IDSHORT in INPUT-INT oder INPUT-EXT falsch

IDSHORT ist nur sinnvoll, wenn die Statements INPUT-INT und INPUT-EXT gleichzeitig vorhanden sind.

110 ID ... ID4 oder KEY ... KEY4 nicht lückenlos

Es fehlt eine der Eintragungen ID ... ID4 oder KEY ... KEY4. Beginnend bei ID oder KEY müssen diese lückenlos bis zur letzten gewünschten Stufe ID1 ... ID4 vorliegen.

111 Datenfeld ID oder KEY liegt hinter Datensatz

Ein Datenfeld für ID ... ID4 oder KEY ... KEY4 ragt über das Ende der Datensätze hinaus.
Vielleicht fehlt auch die Angabe SIZE, oder sie ist zu klein.

112 Zu viele ID-Stufen in INPUT-EXT

Das Statement INPUT-EXT enthält mehr Stufen von Fall-Identifikationen als die Datei INPUT-INT. Die ID-Stufen aus INPUT-INT werden im Zählprotokoll hinter der Statementliste angezeigt.

113 NAME zu lang

Der Kontrollname der internen Datei ist zu lang, oder es fehlt das abschließende Hochkomma. Der Fehlerstrich zeigt die Stelle an, an der die zulässige Länge überschritten wurde.

114 Zu viele ID-Stufen in INPUT-INT

Die Datei INPUT-INT enthält mehr Stufen von Fall-Identifikationen als das Statement INPUT-EXT. Die ID-Stufen von INPUT-INT werden im Zählprotokoll hinter der Statementliste ausgedruckt.

115 Interne Datei defekt

Bei einem früheren Verarbeitungslauf wurden Daten in diese Datei geschrieben. Anschließend wurde das Programm nicht korrekt beendet.
Es sollte versucht werden, diese interne Datei neu zu erstellen.

116 NAME falsch

Der bei NAME angegebene Kontrollname stimmt nicht mit dem Kontrollnamen der eingelesenen internen Datei überein.
Der Name aus der Datei wird im Zählprotokoll hinter der Statementliste ausgedruckt.

117 Falsche interne Datei

Mögliche Ursache:

- Die mit FNAME angegebene Datei ist keine interne CNT-Datei,
- wurde nicht vollständig ausgegeben,
- verlangt ein neueres CNT-Programm,
- sie wurde durch Veränderung zerstört.

118 NSORT in INPUT-EXT falsch

Es sind die Statements INPUT-EXT und INPUT-INT vorhanden. Mit einer unsortierten externen Datei ist eine gemeinsame Verarbeitung mit internen Datenbeständen nicht möglich. Daher ist die Angabe NSORT in INPUT-EXT unzulässig. Durch die Angabe SORT in INPUT-EXT lässt sich ein unsortierter externer Datenbestand während der laufenden Verarbeitung sortieren.

119 Falsche Nummer hinter FETCH-FILE oder REPORT-FILE

Es sind nur die Statements FETCH-FILE1 ... FETCH-FILE9 und
REPORT-FILE1 ... REPORT-FILE9 möglich.

120 Verschiedene ID-Stufen in INPUT-INT und INPUT-EXT

Es sind die Statements INPUT-EXT und INPUT-INT vorhanden.
Um die statistischen Fälle aus beiden Dateien zusammenführen zu können, muss in beiden Dateien die gleiche Anzahl von Fall-Identifikations-Stufen vorhanden sein.
Die ID-Stufen aus INPUT-INT werden im Zählprotokoll hinter der Statementliste ausgedruckt.
Vielleicht ist die Angabe IDSHORT in einem der INPUT-Statements erforderlich.

121 Variable oder Textelement nicht definiert

Die hier verwendeten Variablen oder Textelemente müssen bereits definiert sein. Eine Variable wurde weder unter DEFS definiert noch in der internen Datei gefunden.

122 Variable bereits vorhanden

Eine hier neu eingeführte Variable wurde bereits in einem vorangehenden Statement definiert oder aus einer internen Datei eingelesen.

123 Fall-Identifikation oder Key falsch

Hier sind nur die Angaben ID, ID1 ... ID4, CID, CID1 ... CID4 oder KEY, KEY1 ... KEY4 erlaubt.

124 Text zu lang

Eine Textzeile ist zu lang oder wird nicht durch ein Hochkomma ' beendet.
Beim Fehlerstrich ist die maximale Länge überschritten.

125 Numerischer Ausdruck in FILTER

Hinter FILTER sind weder Funktionen wie MEAN, MED, MTT noch Rechenoperationen wie + - * zulässig.

126 Angabe mehrfach vorhanden

Diese Angabe erscheint mehrfach im Statement, darf aber nur einmal angegeben werden.
Vielleicht liegen auch zwei Angaben vor, die sich gegenseitig ausschließen.

127 Es fehlt die C-Variable

Es wird eine Angabe .a, .Zi oder .S erwartet. Dazu fehlt aber eine direkt vorausgehende C-Variable.

128 Nummer zu klein

Diese Nummer ist kleiner als die Nummer der davor stehenden Variablen oder des vorigen Textelements.

129 Falsches Separatorzeichen

Hier sind nur die ersten 127 Zeichen des ANSI-Codes möglich. Das sind die Zeichen mit den dezimalen Werten 0 ... 126 der Tabelle: *Zeichenzuordnung ASCII, ANSI und EBCDIC*).
Außerdem müssen sich Separator- und Texterkennungszeichen unterscheiden.

130 Variablen- oder Textelementnummer falsch

Hier wird eine Nummer zwischen 0 und 999 999 999 erwartet.

131 Wert zu lang

Dieser numerische Wert besitzt mehr als 18 Dezimalziffern.

132 Falscher Feldtyp

Bei Dateien im COLBIN-Format sind nur externe Datenfelder vom Typ A I K L M U D F und G zulässig.
Das QUANTUM-Format erlaubt nur Datenfelder vom Typ A K L M U und D.
In den Formaten BINARY1 und BINARY2 sind nur Datenfelder vom Typ B I F und G möglich.

133 IDSHORT und IDi gleichzeitig

Im Statement OUTPUT-INT darf nicht gleichzeitig IDSHORT und ID ... ID4 angegeben werden.

134 Ungültiges Datenfeld

Die Angabe rechts vom Gleichheitszeichen ist fehlerhaft oder hier nicht zulässig.
Bei MOD-PROC sind externe Datenfelder unzulässig. Dort sind nur Variablenwerte möglich.
Externe Datenfelder lassen sich nur hinter ADD-PROC, UPD-PROC und OUTPUT-EXT verarbeiten.

135 Unzulässige Feldlänge

Die Gesamtlänge eines I- oder M-Feldes muss durch die Länge der einzelnen Merkmalswerte teilbar sein.
Ein K-Feld muss hier eine gerade Anzahl von Bytes besitzen.

136 Numerischer Wert passt nicht zum Datenfeld

Ein numerischer Wert ist zu groß für das Zielfeld. Bei D-Feldern sind auch Vorzeichen und Dezimalzeichen zu bedenken. G-Felder vertragen keine negativen Werte.

137 INPUT-SEC ohne INPUT-INT und INPUT-EXT

INPUT-SEC Statements setzen ein Statement INPUT-INT oder INPUT-EXT voraus.

138 Zu viele geöffnete Klammern (

Der Ausdruck enthält an dieser Stelle mehr als 255 geöffnete Klammern.

139 Zu viele schließende Klammern)

Der Ausdruck enthält an dieser Stelle mehr schließende Klammern) als vorher durch (geöffnet wurden.

140 Falsche Klassengrenze

Als Klassengrenze ist ein gültiger numerischer Wert erforderlich, der größer als sein Vorgänger sein muss.

141 Zu viele Operanden

Dieser Ausdruck enthält mehr als 255 Operanden (Angaben Cn.a, Nn ...).

142 Zu viele Operatoren

Dieser Ausdruck enthält mehr als 255 Operatoren (Angaben +, *, |, & ...).

143 Merkmalsangabe fehlt

Die Bedingungsvariable Cn vor dem Fehlerstrich benötigt noch ein Merkmal .a eine Anzahlbedingung .Zi oder einen Summenoperanden .S

144 Abkürzung unzulässig

Hier ist eine Abkürzung mit ... nicht zulässig. Der davor stehende Operator ist nicht dafür geeignet.

145 Operand fehlt

Vor diesem Vergleichsoperator =, <, >, . . . fehlt ein numerischer Operand oder ein Textoperand.

146 Zu viele Variable

Es wurden zu viele Variable definiert, oder einige Variable benötigen zu viel Platz. Zur Speicherung von Variablenwerten stehen 10 000 000 Bytes zur Verfügung.
Das Problem kann durch Entfernen von überflüssigen Variablen gelöst werden oder durch Verringerung des Speicherbedarfs für einzelne Variable:

C-Variable:	jeweils 8 Merkmale belegen ein Byte
N-Variable:	1 oder 2 Ziffern belegen 1 Byte, 3 oder 4 Ziffern belegen 2 Bytes, 5 bis 6 Ziffern belegen 3 Bytes, 7 bis 9 Ziffern belegen 4 Bytes, 10 bis 11 Ziffern belegen 5 Bytes, 12 bis 14 Ziffern belegen 6 Bytes, 15 bis 16 Ziffern belegen 7 Bytes, 17 bis 18 Ziffern belegen 8 Bytes
T-Variable:	jedes Zeichen belegt 2 Bytes

147 Zu viele oder zu lange Satz-Kennzeichen

Der Bereich zur Aufnahme der Satz-Kennzeichen ist übergelaufen.

- In RC wurden zu viele Kennzeichen angegeben.
- Die Kennzeichen in RC sind zu lang.

148 Unzulässige Angabe in OUTPUT-EXT

Für die SPSS-Ausgabe ohne VNAME sind die Angaben RC und SIZE nicht zulässig. Zu ID ... ID4 ist lediglich ein Variablenname in Hochkommas möglich.

149 Statistischer Fall zu groß

Der Bereich zur Aufnahme der Datensätze eines statistischen Falles übersteigt 10 000 000 Bytes oder Spalten. Die Länge dieses Bereichs ermittelt sich auf folgende Weise: Satzlänge aus SIZE multipliziert mit der Anzahl zu verarbeitender Datensätze aus RC.

150 Das Zeichen = fehlt

An dieser Stelle ist ein Gleichheitszeichen erforderlich.

151 Schließende Klammer fehlt

An dieser Stelle ist ein logischer oder numerischer Ausdruck beendet.
Es sind aber nicht alle vorher durch (geöffneten Klammern auch wieder durch) geschlossen worden. Der Fehler kann auch dadurch zustande kommen, dass die Auswertung des Ausdrucks wegen ungültiger Angaben vorzeitig beendet wird.

152 ID-Stufe falsch

Hier wird eine Fall-Identifikation ID...ID4 angesprochen, die weder in INPUT-EXT noch in INPUT-INT oder INPUT-SEC definiert wurde.

153 FETCH-FILE oder REPORT-FILE Statement mehrfach

Ein Statement FETCH-FILE ... FETCH-FILE9 oder REPORT-FILE ... REPORT-FILE9 ist mehrmals vorhanden.

154 Alphanumerische ID als Zähloperand unzulässig

In diesem Ausdruck in XTAB wird eine Fall-Identifikation von alphanumerischem Typ aufgerufen. Das ist nicht zulässig.

155 Operand numerisch statt logisch

Links vom Fehlerstrich befindet sich ein numerischer Operand. Hier ist aber ein logischer Operand erforderlich. Vielleicht sind auch Klammern falsch gesetzt.

156 Textoperand statt logischem Operanden

Links vom Fehlerstrich befindet sich ein Textoperand. Hier ist jedoch ein logischer Operand erforderlich. Vielleicht sind auch Klammern falsch gesetzt.

157 Textoperand statt numerischem Operanden

Links vom Fehlerstrich befindet sich ein Textoperand. Hier ist jedoch ein numerischer oder logischer Operand erforderlich. Vielleicht sind auch Klammern falsch gesetzt.

158 XADD hinter Folgestatements

Zwischen diesem XADD und dem davorliegenden XTAB befinden sich Folgestatements mit einem Strich - am Anfang der Zeile. Diese sind aber erst hinter den XADD-Statements möglich.

159 Falsches FETCH-Statement

Es sind nur die Angaben FETCH oder FETCH1 ... FETCH29 möglich.

160 Textoperand statt logischem Operanden

Links vom Fehlerstrich befindet sich ein Textoperand Tn oder kkk....
Davor steht ein logischer Operator &, | oder . Dieser verlangt aber logische Operanden. Vielleicht sind auch Klammern falsch gesetzt.

161 Textoperand statt numerischem Operanden.

Links vom Fehlerstrich befindet sich ein Textoperand Tn oder kkk....
An dieser Stelle ist aber ein numerischer Operand erforderlich. Vielleicht sind auch Klammern falsch gesetzt.

162 Textoperand fehlt.

Links vom Fehlerstrich befindet sich ein logischer oder numerischer Operand. Davor steht ein Vergleichsoperator <, >, = ... der statt dessen einen Textoperanden Tn oder kkk... verlangt.

163 Teilausdruck numerisch statt logisch.

Vor dem Fehlerstrich endet ein numerischer Teilausdruck. Dort wird aber ein logischer Teilausdruck erwartet. Vielleicht sind auch Klammern falsch gesetzt.

164 Textoperand fehlt.

Links vom Fehlerstrich befindet sich ein logischer oder numerischer Teilausdruck. Davor steht ein Vergleichsoperator <, >, = ... der statt dessen einen Textoperanden Tn oder kkk... verlangt.

165 Zu viele IF-Statements geschachtelt.

Vor diesem IF liegen mindestens zehn weitere IF-Statements, die noch nicht durch ein EIF abgeschlossen sind.
Dieser Fehler kann auch durch ein IF in einer DO-Schleife entstehen, das nicht vor dem nächsten EDO mit EIF beendet wird.

166 EIF-Statement fehlt.

Das Ende von ADD-PROC, MOD-PROC, UPD-PROC oder OUTPUT-EXT ist erreicht.
Es sind aber noch nicht alle IF-Statements durch EIF abgeschlossen.

167 IF-Statement fehlt.

An dieser Stelle ist kein IF-Statement wirksam. Alle eventuell davor liegenden IF-Statements sind bereits durch EIF abgeschlossen. Daher sind weder EIF- noch ELSE-Statements zulässig.

168 EIF oder ELSE auf falscher DO-Stufe.

Dieses Statement und das dazugehörige IF befinden sich nicht im Wirkungsbereich des gleichen DO-Statements:

- Ein IF-Statement vor einem DO-Statement ist hinter dem entsprechenden EDO durch ELSE oder EIF zu beenden.
- Ein IF-Statement hinter einem DO-Statement ist vor dem entsprechenden EDO durch ELSE oder EIF zu beenden.

169 SAMPLE-Wert falsch.

Hier ist eine Prozentangabe zwischen 0 und 100 erforderlich.

170 Startwert falsch.

Als Startwert für die Angabe SAMPLE ist eine Zahl zwischen 1 und 32 767 erforderlich.

171 Falsche Zeile oder Spalte.

Als Zeilen- oder Spaltennummer sind nur die Werte 1 ... 32 767 und -1 ... -32 767 erlaubt. Jede hier angegebene Zeile oder Spalte muss weiter zum Ende der Tabelle hin liegen als ihr Vorgänger.
Auf eine negative Zeile oder Spalte (Nummerierung vom Ende der Tabelle) darf keine positive folgen (Nummerierung vom Anfang der Tabelle).

172 Falsches Trennzeichen

Hier ist ein einzelnes Druckzeichen als Trennzeichen für die Tabellenspalten anzugeben. Es muss in Hochkommas eingeschlossen sein.

173 Falsche Spaltenbreite

Hier ist als Breite einer Tabellenspalte ein numerischer Wert zwischen 3 und 128 erforderlich, wenn PAGE wirksam ist.
Bei PAGEP ist eine Spaltenbreite zwischen 4mm und 250mm anzugeben.

174 Zu viele TAB-Spalten

Es sind maximal 32 767 TAB-Spalten zu den Tabellen zulässig.

175 Falsche Druckmaske.

Eine Maske zur Druckaufbereitung von Zählergebnissen muss 1 ... 16 Zeichen lang sein und darf an höchstens einer Stelle das Zeichen & oder eine zusammenhängende Zeichenfolge &&... enthalten.

176 Staffelanfang { unzulässig

An dieser Stelle ist eine vorausgehende Staffelung { offen und noch nicht durch } abgeschlossen. Dann darf keine neue Staffel mit { eröffnet werden. So führt der Ausdruck

```
row={n1}{c1;{n2}{c2}}
```

zum Fehler. Dagegen ist folgender Ausweg möglich:

```
row={n1}{c1};{n1}{n2}{c2}
```

177 Zu viele Staffelungen

An dieser Stelle befinden sich über 10 Staffelungen {...} direkt hintereinander.

178 Zu viele Klammern }

An dieser Stelle ist keine Klammer { offen. Deshalb darf auch keine schließende Klammer } angegeben werden.

179 FETCH-FILE nicht vorhanden

Vor diesem FETCH-Statement liegt kein passendes FETCH-FILE-Statement oder die FETCH-Datei war nicht zugänglich.

180 Zu viele Verbindungsvariable

Das Statement FUSION erlaubt bis zu 50 Verbindungsvariable der Form Cn=Cm, Nn=Nm oder Tn=Tm.

181 Ungültiger Vorgänger

Eine Abkürzung .. oder ... ist nur zulässig, wenn der vorausgehende Zählausdruck eine N-Variable oder eine C-Variable enthält.

182 Ungültige Fußnotennummer.

Eine Fußnotennummer muss zwischen 1 und 99 999 liegen.

183 Hintergrund hier nicht möglich

An dieser Stelle ist keine Tonfläche Gi oder COi möglich.

184 Nachkommastellen ohne ABS

Hier wurden hinter : Nachkommastellen zur Druckaufbereitung angegeben.
Das ist aber nur sinnvoll, wenn gleichzeitig die Angabe ABS vorliegt.

185 Ungültige Fortsetzungszeile.

Hier wird ein neues Statement erwartet, mit einer Angabe in der ersten Position der Zeile. Insbesondere muss hinter den Statements INCLUDE, PROCESS ON, PRINT, EDO und EIF ein neues Statement mit einer Angabe in der ersten Position der Zeile beginnen.

186 GROUP fehlt

Bei FILTER, ROW oder COL ist ein Zähloperand mit G- gekennzeichnet.
Dann muss aber auch GROUP angegeben werden.

187 ROW oder COL fehlen

In diesem Statement ist eine der Angaben ROW, COL und FILTER wirksam. Dann müssen gleichzeitig Angaben zu ROW und COL vorhanden sein.
Eine davon fehlt hier aber.

188 ROW oder COL fehlt

Eine der beiden Angaben ROW oder COL ist wirksam, die andere nicht.
ROW und COL können nur gleichzeitig angegeben werden. Vielleicht ist auch ROWS oder COLS in einem vorigen XTAB Statement fehlerhaft.

189 CONT ohne ROW und COL

Die Angabe CONT ist nur zulässig, wenn im XTAB-Statement auch Tabellen auszugeben sind, also ROW und COL wirksam sind.

190 ROW, COL oder FILTER zu lang

Zu ROW, COL oder FILTER liegen zu viele oder zu aufwendige Eintragungen vor: Jede der Angaben ROW, COL, FILTER verträgt bis zu 32 767 Zeilen, Spalten oder Tabellen.
Deren Summe darf 65 535 nicht übersteigen.
Funktionen wie MEAN, MED, PERCTL, STD ... benötigen mehrere Zeilen oder Spalten.

191 Zu große Spaltennummer in PRINT ... PRINT3

In einer der Angaben PRINT ... PRINT3 wird eine Spalte angesprochen, die es in den Tabellen dieses XTAB Statements nicht gibt.

192 Zu große Zeilennummer in PRINT ... PRINT3

In einer der Angaben PRINT ... PRINT3 wird eine Zeile angesprochen, die es in den Tabellen dieses XTAB Statements nicht gibt.

193 Sortierbasis zu groß

In diesem Statement ist die Angabe SORT=ROWSn oder SORT=COLSn wirksam. Die Zeile oder Spalte n, nach der sortiert werden soll, liegt außerhalb der Tabelle.

194 Zu wenige Ausgabezeilen in LINES

Nach Abzug der Kopf- und Fußzeilen verbleiben weniger als 6 Ausgabezeilen auf den Seiten.

195 Text zu lang

Der hier eingegebene Text ist zu lang, oder es fehlt das abschließende Hochkomma.

196 Rand zu groß

Der Rand in HOR oder VER darf im PAGE-Statement höchstens 99 Zeilen oder Positionen besitzen. Im PAGEP-Statement sind maximal 50mm zulässig.

197 Spaltenbreite HOR falsch

Hier wird mit HOR das Blatt in Spalten eingeteilt. Als Spaltenbreite sind im Statement PAGE 12 ... 255 Positionen zulässig, im Statement PAGEP sind es 20mm bis 120mm.

198 Zu viele Angaben

Zu HOR oder VER wurden zu viele durch Kommas getrennte Angaben gemacht. Es können maximal 20 Werte vorgegeben werden.

199 Seitenhöhe VER falsch.

Hier wird durch VER die Höhe der Teilseite eines Blattes angegeben.
Dafür sind im PAGE-Statement 8 ... 999 Zeilen zulässig, im PAGEP-Statement sind es 10 bis 500 mm.

200 Blatt zu schmal.

Die zu HOR angegebenen Teilseiten passen nicht nebeneinander auf das laufende Blatt. Vielleicht ist auch die Angabe POS falsch oder das falsche Format FORM wirksam.

201 Blatt nicht hoch genug.

Die zu VER angegebenen Teilseiten passen nicht übereinander auf das laufende Blatt. Vielleicht ist die Angabe LINES falsch oder das falsche Format FORM wirksam. Weiter ist zu bedenken, dass die Kopfzeilen HEAD und Fußzeilen FOOT noch Zeilen aus LINES kosten.

202 XADD ohne XTAB.

Zu diesem XADD fehlt das dazugehörige XTAB Statement.
Zwischen einem XTAB und seinen XADD Statements dürfen höchstens Kommentarzeilen, Makrodefinitionen oder DO-Statements liegen.
Nicht erlaubt sind dagegen Statements, die Druckausgaben erzeugen oder beeinflussen, wie PAGE, CODEBOOK, OUTPUT-EXT usw.
Vielleicht fehlen aber auch nur die Angaben ROWS oder COLS im vorausgehenden XTAB.

203 Zeilennummer falsch.

Diese Zeilennummer liegt außerhalb der Tabelle des zugehörigen XTAB Statements.

204 Spaltennummer falsch.

Diese Spaltennummer liegt außerhalb der Tabelle des zugehörigen XTAB Statements.

205 ROW oder COL fehlen.

In einem XADD Statement müssen ROW und COL vorhanden sein.
Dazu genügt auch die Angabe ROWS oder COLS aus einem früheren XADD Statement.

206 Zu viele FILTER-Operanden.

Dieses XADD Statement besitzt mehr FILTER-Angaben als das dazugehörige XTAB Statement.

207 Zu viele ROW-Operanden.

Die ROW-Angaben dieses XADD Statements reichen über die Tabelle des dazugehörigen XTAB Statements hinaus.
Vielleicht ist auch die Zeilenangabe von ASTART zu groß.

208 Zu viele COL-Operanden.

Die COL-Angaben dieses XADD Statements reichen über die Tabelle des dazugehörigen XTAB Statements hinaus.
Vielleicht ist auch die Spaltenangabe von ASTART zu groß.

209 Falsche Zeilenanzahl.

Hier ist eine Zeilenanzahl 1 ... 255 erforderlich.

210 Datei lässt sich nicht erstellen.

Diese Ausgabedatei konnte nicht geöffnet werden. Mögliche Ursachen:

- der Dateiname ist falsch geschrieben,
- die Datei wird zur Zeit von einem anderen Programm blockiert,
- das Laufwerk oder Verzeichnis sind nicht zugänglich,
- die Datei existiert bereits und ist vor dem Schreibzugriff geschützt.

211 Falscher Wichtungsfaktor

Als Wichtungsfaktor ist hier ein numerischer Wert ungleich 0 und Z0 erforderlich.

212 Merkmalslänge falsch

In M-Feldern dürfen Merkmale 1 ... 4 Bytes oder Spalten lang sein.
In I-Feldern dürfen Merkmale 1 ... 4 Bytes lang sein (BINARY2: 1 ... 2 Felder und COLBIN: 1 ... 2 Spalten).

213 INPUT-EXT fehlt

Dann ist das Statement HOLECOUNT nicht zulässig.

214 Satznummer zu klein

Diese zweite Satznummer muss größer oder gleich der vorausgehenden Satznummer sein.

215 Kein gültiger logischer Ausdruck

Ein TST-Statement muss mit einem logischen Ausdruck beginnen.

216 Keine gültige Zahl

An dieser Stelle wird ein numerischer Wert erwartet. Er darf maximal 9 Ziffern besitzen, davon bis zu 8 Nachkommastellen. Vorzeichen + und - sind möglich. Dezimalzeichen ist der Punkt.

217 Falsches Signifikanzniveau

Hier wird ein Signifikanzniveau als Prozentwert zwischen 0.01 und 50.00 erwartet. Sind mehrere Signifikanzen hinter SIG angegeben, so müssen diese unterschiedlich sein. Mehr als drei Signifikanzniveaus sind nicht möglich.

218 Falsche SEP-Angabe bei COPY

Im Statement OUTPUT-EXT liegt die Angabe COPY vor. Dann muss INPUT-EXT die gleiche SEP-Angabe besitzen.

219 Dateiname mehrfach vorhanden

Der Dateiname in diesem Statement erschien bereits in einem davor liegenden Statement: FETCH-FILE, INPUT-EXT, INPUT-INT, INPUT-SEC, PAGE, PAGEP, OUTPUT-EXT, OUTPUT-INT oder REPORT-FILE.

220 INPUT-EXT fehlt für COPY

Im Statement OUTPUT-EXT liegt die Angabe COPY vor. Dazu muss aber ein Statement INPUT-EXT vorausgehen.

221 Falsches Dateiformat für COPY

Im Statement OUTPUT-EXT liegt die Angabe COPY vor.
Dazu sind die Dateiformate AUTO ... SPSS und TRIPLES nicht zulässig.
Außerdem müssen INPUT-INT und INPUT-EXT die gleichen INTEGER-Angaben besitzen.

222 Signifikanzniveau zu SIG fehlt

Die Angabe SIG in PRINT verlangt ein Signifikanzniveau als Prozentwert
zwischen 0.01 und 50.00. Vielleicht fehlt auch ein Komma vor dem Signifikanzniveau.

223 Satzlänge falsch für COPY

Im Statement OUTPUT-EXT liegt die Angabe COPY vor. Dann müssen die Satzlängen SIZE von INPUT-EXT und OUTPUT-EXT übereinstimmen.

224 Satzanzahl falsch für COPY

Im Statement OUTPUT-EXT liegt die Angabe COPY vor. Dann muss die durch RC festgelegte Anzahl von Datensätzen pro Fall für INPUT-EXT und OUTPUT-EXT gleich sein.

225 Falscher Feldtyp

Der Typ des Ausgabefeldes passt nicht zum Datentyp von ID aus INPUT-EXT oder INPUT-INT:
Bei numerischer ID aus der Eingabe sind nur Felder vom Typ D, DK, DS, F, G oder P möglich, bei alphanumerischer Eingabe nur Typ A, AK oder AS.

226 R und C hier unzulässig

Zur Beschränkung einer PRINT-Angabe auf einzelne Zeilen oder Spalten ist direkt hinter PRINT= eine Angabe R(...) oder C(...) erforderlich.
Zwischen zwei Angaben C(...) ist eine Aufbereitungsregel wie ABS, PROW, PCOL ... erforderlich.
Das Gleiche gilt für zwei Angaben R(...).

227 Falsche Farbangabe

Für die Farben rot, grün und blau sind hier drei Farbwerte zwischen 0 und 255 erforderlich.

228 Operand fehlt oder ungültig

An dieser Stelle wird in einem Ausdruck ein gültiger Operand verlangt; ein solcher Ausdruck darf nicht mit einem Operator wie & + usw. beendet werden.

229 Schließende Klammer } fehlt

Hier enden die Angaben zu ROW, COL oder FILTER. Eine öffnende Klammer { ist nicht durch } abgeschlossen. Vielleicht fehlt aber auch ein Semikolon, oder es liegt eine ungültige Angabe vor.

230 MTT... geschachtelt

Hier wird in ROW oder COL ein t-Test durch MTT... angefordert, obwohl eine davor stehende Angabe MTT... noch nicht durch } abgeschlossen ist.

231 Klasseneinteilung A ... Z{ } geschachtelt

Hier wird eine Klasse A ... Z{ } neu begonnen, obwohl eine vorausgehende Klasse zum gleichen Buchstaben A ... Z noch nicht durch } abgeschlossen ist.

232 Staffelung noch nicht beendet

An dieser Stelle wird eine Staffel durch } beendet. Direkt dahinter darf weder eine Klasse A ... Z{ } noch ein t-Test MTT... beginnen. Vielleicht fehlt ein ; am Ende der Staffel.

233 Staffel falsch begonnen

An dieser Stelle beginnt mit { eine neue Staffel. Direkt davor wurde durch die Klammer } eine Klasse A...Z{ } oder ein t-Test MTT... beendet. Direkt dahinter ist keine neue Staffel möglich. Vielleicht fehlt ein Semikolon?

234 Zähloperand bei MTT... unzulässig

An dieser Stelle ist ein t-Test MTT ... wirksam. Dann sind folgende Zähloperanden nicht möglich:

- Fallidentifikation IDi
- Summenoperand Cn.S
- alle Funktionen, mit Ausnahme von MEAN bei MTT und MTTU
- numerische Ausdrücke
- numerische Konstante

235 C-Variable mit zu vielen Merkmalen

Die Funktion PERCTL kann zurzeit nur C-Variable mit maximal 255 Merkmalen auswerten.

236 MTT... in oberer Staffel

Die gerade durch } beendete Staffel enthält einen t-Test MTT...
Ein solcher darf aber nur in der letzten, unteren Staffel verwendet werden.
Hier wird die Staffelung aber durch { fortgesetzt.

237 Fall-ID ohne Feldangabe

Zu einer Angabe ID ... ID4 fehlt das Datenfeld rechts vom Gleichheitszeichen.

238 ID oder RC überlappen in OUTPUT-EXT

Zwei Angaben ID... oder RC belegen gemeinsame Positionen in einem Datensatz oder besitzen den gleichen Schlüssel.

239 MTT... in XTAB und XADDB passen nicht zusammen

In den Statements XTAB und XADDB müssen die Angaben MTT... in ROW und COL übereinstimmen und in der gleichen Zeile oder Spalte stehen.

240 Merkmal oder Position falsch

Hier wird eine Merkmalsnummer oder Positionsangabe erwartet.

241 Merkmal oder Position zu klein

Diese Merkmalsnummer oder Positionsangabe muss größer als die vorangehende sein.

242 Merkmal oder Position zu groß

Diese Merkmalsnummer oder Positionsangabe ist größer als in der Definition der Variablen erlaubt wurde.

243 SPACE oder Li hier nicht möglich

Hinter INSERT=C(...) sind SPACE oder Linien nicht zulässig.

244 Mehrere ELSE-Statements

Zu jedem IF-Statement darf nur ein ELSE-Statement angegeben werden.

245 MTT... in XADD

Die t-Tests MTT... dürfen nur in den Statements XTAB und XADDB vorkommen, nicht jedoch in XADD.

246 Mehrere Texte angegeben

Zu dieser Variablen oder Variablengruppe wurden zu viele Texte eingegeben.

247 Satznummer für RC unzulässig

Vor der Klammer (hinter RC ist keine Satznummer zulässig.

248 Merkmal mehrfach

Zu diesem Merkmal liegen im gleichen Statement bereits Angaben vor.

249 Mehrere Variable zum Kopieren von Texten

Im gleichen Statement liegt bereits eine Variable vor, aus der Texte übernommen werden sollen.

250 Position falsch

Hier wird eine Positionsangabe in einem externen Datensatz erwartet.
Diese muss ein numerischer Wert größer Null sein.

251 Position zu groß

Diese Positionsangabe in einem externen Datensatz ist zu groß.
Sie liegt nicht mehr innerhalb der Länge des Datensatzes.

252 Unverträgliche Angaben in einer XTAB-Komponente

Diese Angabe enthält einen logischen Operator & | ^ > < = und gleichzeitig TOTAL oder eine der Funktionen MEAN, MIN, MAX, MED, PERCTL, MODL, MODU, STD, STE, EC, ECC, ECR, ECRC, TT, TTU oder TTD.

253 Bis-Position zu klein

Diese Bis-Position liegt vor der voranstehenden Von-Position im externen Datensatz.

254 Längenangabe falsch

Hier wird eine Länge erwartet. Sie muss numerisch und größer Null sein.

255 Die Länge eines externen Feldes ist zu groß

Die Längenangabe oder Bis-Position eines externen Feldes ist zu groß oder das Feld reicht über das Ende des Datensatzes hinaus. Abhängig vom Datenfeld sind folgende Längen möglich:

A-Felder bis zu 8 000 Bytes oder 4 0003 Spalten
B-Felder bis zu 10 000 Merkmale
D-Felder bis zu 30 Zeichen
F- und G-Felder bis zu 8 Bytes
I-Felder bis zu 10 000 Merkmale
K-Felder bis zu 10 000 Merkmale
L-Felder bis zu 20 Bytes
M-Felder bis zu 12 000 Zeichen
P-Felder bis zu 10 Bytes
U-Felder bis zu 9 999 Zeichen

256 Zu viele Satz-Kennzeichen RC

Hinter RC sind maximal 2 000 Satz-Kennzeichen möglich.

257 Satz-Kennzeichen zu lang

Ein Satz-Kennzeichen hinter RC ist zu lang für das Zielfeld.
Satz-Kennzeichen vom Typ A dürfen höchstens so lang sein, wie das durch RC=A(a .. b) definierte Datenfeld und müssen durch ein Hochkomma beendet werden. In AS- und AK-Feldern sind maximal 128 Zeichen möglich.

258 Falsches Satz-Kennzeichen

Für die Satz-Kennzeichen RC wird hier ein numerischer Wert erwartet.

259 Satz-Kennzeichen mehrfach

Dieses Satz-Kennzeichen wurde mehrmals angegeben.
Jedes Kennzeichen darf hinter RC nur einmal angegeben werden.

260 EOL nicht zulässig

Bei den Datenformaten COLBIN, AUTOCOLBIN, AUTOEBCDIC, SPSS, SPSSUTF8, TRIPLES, BINARY1 und BINARY2 sind die Angaben EOL und EOLF hinter SIZE nicht zulässig.

261 Falscher Feldtyp für ID, RC oder KEY

Dateien im Format COLBIN oder AUTOCOLBIN erlauben für ID, RC und KEY nur Felder vom Typ A, D, F und G.
Die Formate QUANTUM, AUTOQUANTUM, SPSS, SPSSUTF8 und TRIPLES erlauben dazu nur Felder vom Typ A und D.
In den Formaten BINARY1 und BINARY2 sind für ID, RC und KEY nur Felder vom Typ F und G zulässig.

262 Zu viele Angaben ROW oder COL

Hinter FONT sind maximal zehn Angaben ROW und zehn Angaben COL möglich.

263 Klammer (fehlt

Hier wird eine öffnende Klammer (erwartet.

264 Klammer) fehlt

Hier wird eine schließende Klammer) erwartet.

265 Falsche Satzlänge

Hier wird die Satzlänge für die Datei erwartet. Dazu ist ein numerischer Wert zwischen 1 und 10 000 000 anzugeben. Bei SPSS-Dateien sind die Werte 1... 32 000 möglich.

266 Klammer] fehlt

Hier wird eine schließende Klammer] erwartet.

267 Falscher SKIP-Wert

Hier wird die Anzahl 1 ... 255 der am Anfang der Datei zu überspringenden Datensätze erwartet.

268 IDSHORT und MULTIPLE

Diese beiden Angaben dürfen in INPUT-EXT nicht gleichzeitig vorliegen.

269 ID fehlt

Die Angaben SORT, ASORT, DSORT, NSORT und IDSHORT sind ohne eine Fall-Identifikation ID nicht sinnvoll.

270 FNAME fehlt

Es fehlt der Dateiname FNAME.

271 Zu viele Datensätze

In dieser Datei sind maximal 1 000 Datensätze pro Fall möglich.

272 Falsche Dateiversion

Die Interne Datei wurde von einer neueren Programmversion erstellt. Sie kann von der hier verwendeten älteren Version nicht mehr verarbeitet werden.

273 Falsche Anzahl von Merkmalen oder Positionen

Hier wird eine Anzahl von Merkmalen oder Positionen zwischen 1 und 9 999 erwartet.

274 Anzahl von Merkmalen oder Positionen zu groß.

Die Anzahl von Merkmalen oder Positionen übersteigt den höchsten definierten Wert einer Variablen.

275 Satznummer falsch.

Hier wird die Nummer 1 ... 1 000 eines Datensatzes erwartet.

276 Falsche Eingabe in FETCH.

Hier ist ein externes Datenfeld anzugeben. Variable, Ausdrücke, Konstanten oder IDs sind nicht möglich..

277 Satznummer zu groß.

Diese Satznummer ist größer als in der Angabe RC in dem entsprechenden Statement INPUT-EXT, OUTPUT-EXT oder FETCH-FILE zugelassen wurde. Fehlt dort die Angabe RC, so ist nur die Satznummer 1 erlaubt.
Diese Satznummern sind nicht zu verwechseln mit eventuellen Satz-Kennzeichen. Die Satznummern geben an, an welcher Stelle hinter RC das dazugehörige Kennzeichen angegeben wurde.

278 Ungültiges Merkmalskennzeichen.

Hier wird eine Merkmalsnummer beziehungsweise die Angabe X, -, Y oder & erwartet.

279 Merkmal zu klein.

Diese Merkmalsnummer ist zu klein, sie muss größer oder gleich der voranstehenden sein.

280 Falsche Merkmalsnummer.

Abhängig vom Typ des Datenfeldes sind folgende Merkmalsnummern zulässig:

B-Feld:	1 ... 8
I-Feld:	1 ... 255 bei ein Byte langen Merkmalen
	1 ... 65 535 bei Merkmalen von 2 ... 4 Byte Länge
K-Feld:	1 ... 12, 0 = 10, X = - = 11, Y = & = 12
L-Feld:	1 ... 12, 0 = 10, X = - = 11, Y = & = 12
M-Feld:	1 ... 9, 0 = 10 bei Merkmalslänge 1
	1 ... 99, 0 = 100 bei Merkmalslänge 2
	1 ... 999, 0 = 1 000 bei Merkmalslänge 3
	1 ... 9 999, 0 = 10 000 bei Merkmalslänge 4
MK-Feld:	1 ... 99 999
MS-Feld:	1 ... 99 999

281 Datenfeld zu lang.

Das vor dem Fehlerstrich stehende Datenfeld reicht über das Ende des Datensatzes hinaus.

282 Feldlänge fehlt.

Zum Datenfeld vor dem Fehlerstrich fehlt das Feldende oder die Feldlänge.

283 Falsche Anzahl Nachkommastellen.

Hier wird die Anzahl von Nachkommastellen für numerische Werte erwartet. Es sind nur die Angaben 1 ... 18 zulässig.

284 Ungültige Schriftart

- Der Schriftname fehlt oder ist länger als 50 Zeichen.
- Die Angabe N, I, B oder BI für normal, kursiv, fett oder fett kursiv fehlt.
- Zu dem Schriftnamen ist keine passende Fontdatei *.ttf zu finden.

285 Eine Angabe fehlt

An dieser Stelle ist eine weitere Angabe erforderlich.

286 Falsche Dateierweiterung

Die hier angegebene Dateierweiterung passt nicht zu dieser Datei.

287 INPUT-EXT nicht vorhanden.

Es wurde bisher kein gültiges INPUT-EXT Statement gefunden. Dann sind externe Datenfelder an dieser Stelle nicht zulässig.

288 INC-Wert falsch.

Hier wird ein numerischer Wert 1 ... 32 767 erwartet.

289 INC-Wert zu groß.

Die Eingabefelder reichen über das Ende des Datensatzes hinaus:

- Die Schrittweite des INC-Wertes ist zu groß.
- Es sind zu viele Von-Bis-Variable links vom Gleichheitszeichen angegeben. Dadurch entstehen zu viele INC-Schritte.
- Die Satzlänge SIZE in INPUT-EXT oder FETCH-FILE ist zu klein.

290 ID-Stufe nicht alphanumerisch.

Hier wird eine Fall-Identifikation ID ... ID4 angesprochen, die entweder nicht definiert ist oder numerische Werte besitzt. Hier werden aber alphanumerische Werte verlangt.

291 ID-Stufe nicht numerisch.

Hier wird eine Fall-Identifikation ID ... ID4 angesprochen, die entweder nicht definiert ist oder alphanumerische Werte besitzt. Hier werden aber rein numerische Werte verlangt.

292 Ausdrücke in XADDB stimmen nicht mit XTAB überein.

Die numerische Ausdrücke im Statement XADDB müssen mit den entsprechenden Ausdrücken von XTAB übereinstimmen. Beide Ausdrücke müssen die gleichen Operatoren (+ - * ...) und Funktionen (MEAN, MED...) besitzen, ihre Operanden (Cn, Nn, TOTAL ...) dürfen sich jedoch unterscheiden. Vielleicht ist auch die Angabe ASTART falsch.

293 Wert nicht positiv.

Hinter TOTAL ist die Gesamtzahl der gewichteten Fälle anzugeben. Diese Anzahl muss größer als 0 und von Z0 verschieden sein.

294 Zu viele Wichtungsränder in WEIGHT.

Es dürfen höchsten 100 Wichtungsränder der Form Ca*Cb* ... in einem Statement vorkommen.

295 Mehr als 10 Variable in einer Wichtungsangabe.

In einer Randbedingung Ca*Cb* ... zu WEIGHT dürfen höchstens 10 Variable vorkommen.

296 Sollwert negativ oder Z0.

Als Sollwerte in WEIGHT dürfen keine negativen Werte oder Z0 (keine Angabe) verwendet werden. Null ist als Sollwert dagegen möglich.

297 Anzahl der Sollwerte falsch

Zu den Wichtungsbedingungen Ca*Cb* ... vor dem Fehlerstrich wurde eine falsche Anzahl von Sollwerten angegeben. Zu jeder Kombination von Merkmalen der Variablen Ca*Cb* ... muss ein Sollwert vorliegen.

298 Falsche Textstufe

Hinter STYLE=ROW oder STYLE=COL sind die Textstufen 1 ... 20 oder -1 ... -20 möglich.
Bei Von-Bis-Angaben muss die erste Stufe vor der zweiten liegen, also ROW(1...4) oder ROW(-2...-1) und nicht ROW(4...1) oder ROW(-1...-2).

299 ADD-PROC oder UPD-PROC ohne INPUT-EXT

In diesem Verarbeitungslauf wurde kein INPUT-EXT Statement gefunden.
Ohne eine externe Eingabedatei sind die Statements ADD-PROC und UPD-PROC sinnlos.

300 Zu viele INCLUDE-Statements

Vor diesem INCLUDE liegen mindestens neun weitere INCLUDE-Statements, die alle noch nicht abgearbeitet sind.

301 Dateiname zu lang

Dieser Name ist zu lang, oder es fehlt das abschließende Hochkomma.
Der Fehlerstrich zeigt die Stelle, an der die zulässige Länge überschritten wird.

302 Datei nicht zugänglich

Die Datei konnte nicht geöffnet werden. Mögliche Ursachen:

- Die Datei existiert nicht.
- Der Dateiname ist falsch geschrieben.
- Die Datei wird zurzeit von einem anderen Programm blockiert.
- Die Datei befindet sich in einem anderen Laufwerk oder Verzeichnis.

303 INCLUDE-Datei mehrfach aufgerufen

Diese Datei wurde in einem vorausgehenden INCLUDE-Statement schon einmal aufgerufen und ist noch nicht ganz eingelesen oder:
Die Statementdatei selbst wurde in einem INCLUDE-Statement noch einmal aufgerufen.
Wenn eine INCLUDE-Datei ganz eingelesen ist, darf sie sehr wohl noch einmal mit INCLUDE aufgerufen werden.

304 Zu viele INPUT-SEC Statements

In einem Verarbeitungslauf sind höchstens 20 dieser Statements möglich.

305 Falscher Makroname

Ein Makroname muss mit dem Zeichen # beginnen und darf aus den Buchstaben A ... Z, a ... z, Ä, ä, Ö, ö, Ü, ü, ß, den Ziffern 0 ... 9 und dem Unterstreichungsstrich _ bestehen.
Hinter # sind bis zu 10 Zeichen möglich.
In der Makrodefinition ist der Makroname durch eine Leerstelle zu beenden. Bei der späteren Verwendung eines Makros muss hinter dem Makronamen eine Leerstelle oder ein vom Unterstreichungsstrich _ verschiedenes Sonderzeichen stehen.

306 Makroname zu lang

Ein Makroname darf aus höchstens 10 Zeichen bestehen.

307 Zu wenig Argumente hinter einem Makro

An dieser Stelle werden die Argumente eines Makros mit der Klammer) beendet.
Davor befinden sich aber weniger durch voneinander getrennte Argumente, als die Definition des Makros vorsieht.

308 Dieses Makro besitzt keinen Wert

Durch die Angabe # ... wird ein Makro aufgerufen, das in den vorangehenden Statements noch keinen Wert erhalten hat.

309 Falsche Argumentnummer

Eine Zahl zwischen zwei $-Zeichen wird als Makroargument verstanden.
Dafür sind nur die Werte 1 ... 99 möglich.

310 Keine gültige Zahl

Hier wird ein numerischer Wert größer 0, ohne Nachkommastellen und ohne Vorzeichen erwartet.

311 Geschachtelte Argumente im Makro

Hier wird ein Makro mit Argumenten in Klammern verwendet. Vor dem Fehlerstrich befindet sich nun ein Makro in einem solchen Argument, das selbst wieder Argumente in Klammern verlangt. Eine solche Schachtelung von Argumenten ist aber nicht zulässig.

312 Es fehlt) hinter den Argumenten eines Makros

Die Argumente des Makros sollten hier durch eine schließende Klammer beendet werden. Vielleicht wurden auch mehr Argumente mit $$ dazwischen angegeben, als in der Definition des Makros vorgesehen sind.

313 Es fehlen die Argumente zu einem Makro

Dieses Makro wurde mit Argumenten in der Form n definiert. Deshalb müssen bei seiner Verwendung Argumente in Klammern angegeben werden.

314 Widersprüchliche Zeilen- oder Spaltenangaben in PRINT

In PRINT werden durch die Angaben R(...) oder C(...) einer Tabellenzeile oder Tabellenspalte mehrmals Druckanweisungen zugewiesen. Vielleicht ist auch eine Klassenangabe A...Z{ } falsch gesetzt.

315 Widersprüchliche Zeilen- oder Spaltenangaben in PRINT1

In PRINT1 werden durch die Angaben R(...) oder C(...) einer Tabellenzeile oder Tabellenspalte mehrmals Druckanweisungen zugewiesen. Vielleicht ist auch eine Klassenangabe A...Z{ } falsch gesetzt.

316 Widersprüchliche Zeilen- oder Spaltenangaben in PRINT2

In PRINT2 werden durch die Angaben R(...) oder C(...) einer Tabellenzeile oder Tabellenspalte mehrmals Druckanweisungen zugewiesen. Vielleicht ist auch eine Klassenangabe A...Z{ } falsch gesetzt.

317 Widersprüchliche Zeilen- oder Spaltenangaben in PRINT3

In PRINT3 werden durch die Angaben R(...) oder C(...) einer Tabellenzeile oder Tabellenspalte mehrmals Druckanweisungen zugewiesen. Vielleicht ist auch eine Klassenangabe A...Z{ } falsch gesetzt.

318 Benachbarte Vergleichsoperatoren ohne Klammern

Hier stehen zwei Vergleichsoperatoren <, =, > ... nebeneinander. Dann muss aber durch Klammern festgelegt werden, welcher dieser Operatoren zuerst ausgewertet werden soll.
Vielleicht fehlt aber auch ein logischer Operator | oder &.

319 MTT... in XTAB passt nicht zu XADD

Das zu diesem Statement gehörende XTAB enthält Angaben MTT... für t-Tests. Danach sind keine XADD-Statements möglich, sondern nur XADDB.

320 Kein DO zu diesem EDO.

An dieser Stelle ist kein DO Statement wirksam.
Alle eventuell davor liegenden DO Statements sind bereits durch EDO abgeschlossen.

321 EDO Statement fehlt.

An dieser Stelle müssen alle davor liegenden DO Statements beendet sein.
Zu einem dieser DO Statements fehlt aber noch ein EDO.

322 Zu viele DO Statements geschachtelt.

Vor diesem DO liegen mindestens zehn weitere DO Statements, die noch nicht durch ein EDO abgeschlossen sind.

323 Fehlerhafte Angaben zu MTT⋯

Einer der folgenden Fehler wurde gefunden:

- Dieses Statement enthält verschiedene Varianten der t-Tests MTT⋯
- ROW und COL enthalten gleichzeitig t-Test -Angaben MTT⋯
- COL enthält MTTD, MTTX oder MTTUX und ROW einen numerischen Ausdruck.
- COL enthält MTT oder MTTU und ROW die Funktion MEAN.
- ROW enthält MTTD, MTTX oder MTTUX und COL einen numerischen Ausdruck.
- ROW enthält MTT oder MTTU und COL die Funktion MEAN.

324 Indexnummer falsch.

Hier wird eine neue Indexnummer erwartet.
Das muss eine Zahl zwischen 0 und 99 in Klammern < > sein.
Eine solche Indexnummer darf in einem DO Statement nur einmal definiert werden.
Sie darf auch nicht in einem davor liegenden DO-Statement vorkommen, das noch nicht mit EDO abgeschlossen ist.

325 Zu viele DO Schritte.

Der laufende Index <...> in diesem DO Statement besitzt mehr Schrittwerte als ein vorangehender Index im gleichen Statement. Der Fehlerstrich steht hinter dem Wert, bei dem der Fehler festgestellt wurde.

326 Zu wenige DO Schritte.

Der Index <...> vor dem Fehlerstrich enthält weniger Schrittwerte als ein vorangehender Index im gleichen Statement. Dieser Fehler kann auch durch ein ungültiges Zeichen oder einen zu langen Schrittwert entstehen.

327 Schrittweite im DO Index falsch.

In einer Abkürzung von der Form a...b oder a...b(s) in einem DO Statement lässt sich der Wert b nicht durch eine ganze Anzahl von Schritten von a aus erreichen. Ist kein Fehlerstrich vorhanden, so ist die letzte Eintragung im Statement gemeint.

328 Indexnummer nicht definiert.

Hier wird innerhalb einer DO-Schleife ein Index <n> verwendet, der nicht in einem vorausgehenden DO-Statement definiert wurde.

329 Zu viele Soll-Werte.

Der Wichtungsrand vor dem Fehlerstrich besitzt mehr als 32 767 Wichtungsbedingungen: Es werden zu viele Variable oder Variable mit zu vielen Merkmalen miteinander kombiniert.

330 MIN oder MAX falsch

In einem WEIGHT-Statement muss der Wert MIN größer als 0 und der Wert MAX größer oder gleich MIN sein. Es sind maximal 9 Ziffern möglich. Fehlende MIN- und MAX-Werte werden aus der Definition der Zielvariablen NEW oder NEWFACT ermittelt.

331 Kein IDSHORT in den INPUT-Statements

Das Statement OUTPUT-INT enthält die Angabe IDSHORT. Das ist aber nur sinnvoll, wenn IDSHORT auch in INPUT-EXT oder in INPUT-INT vorhanden ist.

332 Variablentypen passen nicht zusammen

Der Datentyp rechts vom Gleichheitszeichen lässt sich nicht in den Datentyp der linken Seite umwandeln.

333 Unverträgliche Angaben zu ID ... ID4

Alphanumerische ID-Werte aus INPUT-SEC vertragen sich nicht mit numerischen aus INPUT-INT oder INPUT-EXT. Alphanumerische ID-Werte aus INPUT-EXT passen nicht zu numerischen Werten aus INPUT-INT.

334 Unzulässiger Variablenname bei ID ... ID4

Zu einer ID-Angabe wurde in Hochkommas ein Variablenname angegeben. Dies ist nur in den Dateiformaten SPSS, TRIPLES und AUTO... zulässig.

335 Zu viele ID-Stufen in INPUT-SEC

Diese Datei INPUT-SEC besitzt mehr ID-Stufen als die Dateien INPUT-INT oder INPUT-EXT.

336 Zu wenige ID-Stufen in INPUT-SEC

Diese Datei INPUT-SEC besitzt weniger ID-Stufen als die Datei INPUT-EXT oder INPUT-INT. Vielleicht wurde die Angabe IDSHORT vergessen.

337 Falsche Anzahl von Variablen

Die Anzahl der passenden Variablen rechts und links vom Gleichheitszeichen ist verschieden oder es trifft eine mit Index definierte Variable auf eine ohne Index definierte.

338 IDSHORT gleichzeitig in INPUT-INT und INPUT-EXT

Die Angabe IDSHORT darf nicht in beiden INPUT Statements gleichzeitig vorkommen.

339 Interne Datei lässt sich nicht verarbeiten

Diese interne Datei wurde auf einem anderen Rechnertyp erstellt und lässt sich auf dieser Maschine nicht verarbeiten (falsches Integer-Format). Die interne Datei muss auf dieser Maschine neu erstellt werden.

340 Index oder Positionsangabe falsch

Als Index einer Variablen und als Position eines Datenfeldes sind nur numerische Werte größer oder gleich 1 erlaubt. Bei Von-Bis-Angaben der Form v ... b darf der Bis-Wert nicht kleiner als der Von-Wert sein.

341 Klammer] fehlt hinter Variablenindex

Hinter einem Variablen-Index fehlt die schließende Klammer] .

342 Indexfehler

Zur Variablen vor dem Fehlerstrich ist ein unzulässiger Index [i] angegeben oder es fehlt ein solcher Index.

343 Unzulässige Abkürzung

Variablennamen mit Wildcardzeichen ? und * erlauben keine Abkürzung n ... m.
Indexangaben [i] sind nicht möglich hinter Variablennamen mit Wildcard-Zeichen * und ?, hinter nicht definierten Variablen und hinter einer Abkürzung n ... m.
Variablennamen, die nicht mit einer Zahl enden, erlauben keine Abkürzung ... m.

344 Von-Bis-Index [i] ... j bei Neudefinition

Hier werden Variable neu definiert. Dann darf nur ein einziger Index [i] angegeben werden, der die Index-Dimension der Variablen festlegt. Eine Von-Bis-Angabe der Form [i] ... j ist dabei nicht sinnvoll.

345 Index oder Positionsangabe zu groß

- Der Index einer Variablen ist größer als die Definition erlaubt oder
- die Positionsangabe für ein Datenfeld ist zu groß oder
- das Feld reicht über das Ende des Datensatzes hinaus.

346 Variablenname oder Textelement fehlerhaft

Dies ist kein hier zulässiger Variablenname oder ein ungültiges Textelement.
Variablennamen müssen mit den Buchstaben C, N oder T beginnen. Sie dürfen aus den Buchstaben A ... Z, den Ziffern 0 ... 9 und dem Zeichen _ bestehen und bis zu 10 Zeichen lang sein.

347 Variable als Index [Ni] hier nicht möglich

An dieser Stelle ist eine N-Variable nicht als Index [Ni] einer anderen Variablen zulässig.

348 Von-Bis-Angabe ([Ni] ... b) nicht möglich

Dieses Datenfeld beginnt bei einer variablen Von-Position.
Dann lässt sich keine Bis-Position in der Form (v ... b) angeben, sondern nur eine Feldlänge in der Form (v , l) .

349 Von-Bis-Abkürzung [Ni] ... j nicht möglich

Eine solche Abkürzung für einen Variablenindex j ist bei vorausgehendem variablem Index [Ni] nicht möglich.

350 DUP ist nur bei neu definierten Variablen möglich

Hier wird eine bereits definierte Variable mit neuen Angaben versehen.
DUP ist nur bei der Neudefinition von Variablen möglich (Doppelpunkt : hinter der Variablennummer).

351 Falsche Variable aus der Eingabedatei.

Die rechts vom Gleichheitszeichen stehende Variable ist fehlerhaft:

- Sie ist nicht definiert oder
- sie besitzt eine ungültige Indexangabe oder
- sie besitzt einen anderen Variablentyp als die zugehörige Variable vor dem Gleichheitszeichen oder
- es fehlt eine Indexangabe.

352 Makrotext zu lang

Die Werte der Makros in diesem Ausdruck sind zu lang (zusammen mehr als 2 000 Zeichen). Damit kann der Ausdruck nicht mehr ausgewertet werden.

353 Falscher Rundungsfaktor

Hier sind maximal 10 Rundungsfaktoren erforderlich, durch Kommas voneinander getrennt.
Jeder Faktor muss größer oder gleich 0.000001 und kleiner als sein Vorgänger sein.

354 PROCESS ON fehlt vor diesem EDO

In der DO-Schleife vor diesem EDO liegt ein PROCESS-Statement mit einem Makroausdruck. Vor dem EDO muss die weitere Verarbeitung unabhängig vom Makroausdruck durch PROCESS ON oder PROCESS OFF festgelegt sein.

355 Zeile oder Spalte hinter SORT liegt außerhalb der Tabelle

Hinter SORT befindet sich eine Verrechnungsregel PROW, PCOL, IROW oder ICOL.
Eine dabei verwendete Basiszeile oder -spalte liegt außerhalb der Tabelle.

356 Zeile oder Spalte hinter SORT liegt außerhalb der Tabelle

Hinter SORT befinden sich Klammern (...) oder [...]. Eine Zeile oder Spalte darin liegt außerhalb der Tabelle.

357 SAME hier unzulässig

In diesem XTAB-Statement befindet sich die Angabe SAME. Das ist nicht möglich, wenn im gleichen Statement ROW, COL, FILTER oder GROUP vorhanden sind. Weiter muss vorher ein XTAB-Statement mit ROW- und COL-Angaben vorhanden sein.

358 SAME im vorigen XTAB

Das zu diesem XADD oder XADDB gehörende XTAB-Statement enthält die Angabe SAME. Dazu sind aber XADD und XADDB nicht zulässig.

359 Falsche Kontaktanzahl

Hier wird eine Kontaktanzahl 1 ... 65 535 erwartet.
Ist davor bereits eine Kontaktanzahl angegeben, so muss die laufende Anzahl größer als die vorausgehende sein.

360 Falsches Kontaktgewicht

Hier wird ein Kontaktgewicht 0.0 ... 1.0 erwartet.
Dieses Gewicht darf maximal 8 Nachkommastellen besitzen.

361 EC, ECC, ECR, ECRC oder .F in einem Ausdruck

Diese Angaben dürfen nicht in einem Ausdrucks mit anderen Angaben kombiniert werden.

362 N-Variable mit Nachkommastellen.

Diese N-Variable wurde mit Nachkommastellen definiert. Ihre Werte sind dann nicht als Fall-Identifikation geeignet.

363 Wert des Makros nicht numerisch

Hier soll ein Makro mit Hilfe der Rechenoperationen #+ #- #* oder #/ einen neuen Wert erhalten. Dieses Makro ist noch nicht definiert oder besitzt keinen zulässigen numerischen Wert. Zulässige numerische Werte sind Zahlen mit maximal 9 Ziffern.
Vorzeichen + und - sowie ein Dezimalpunkt mit bis zu 8 Nachkommastellen sind dabei möglich.

364 Falscher Makrowert

Hier ist ein Makro-Ausdruck auszuwerten. Dabei wurde ein unzulässiger Wert gefunden.
Zulässige Werte sind Zahlen mit maximal 9 Ziffern.
Vorzeichen + und - sowie ein Dezimalpunkt mit bis zu 8 Nachkommastellen sind dabei möglich.

365 Ungültiges Zeichen im Namen für ein Arbeitsblatt

Die Zeichen ? * / \ [ ] : werden von Excel hier nicht akzeptiert.

366 Numerische Ausdrücke in zwei Staffeln

In der Staffel, die vor dem Fehlerstrich durch } beendet wird, befindet sich ein numerischer Ausdruck. Davor steht eine weitere Staffel { ... } ebenfalls mit einem numerischen Ausdruck. Solche Ausdrücke können aber nicht in einer Staffelung kombiniert werden. (Siehe auch im Abschnitt *Ausdrücke* bei XTAB unter COL/ROW/FILTER).

367 Konstante und numerischer Ausdruck in einer Staffelung

In der Staffel, die vor dem Fehlerstrich durch } beendet wird, befindet sich eine konstante Zahl. Davor steht eine weitere Staffel {...} mit einem numerischen Ausdruck. Solche Ausdrücke können aber nicht in einer Staffelung mit einer Konstanten auf der letzten Stufe kombiniert werden. (Siehe auch im Abschnitt *Ausdrücke* bei *XTAB, COL/ROW/FILTER*).

368 Keine gültige SPSS-Variable

Variablennamen VAR= für SPSS müssen aus Buchstaben (einschließlich Umlauten und ß) bestehen sowie aus den Ziffern 0 bis 9 und den Sonderzeichen . @ _ $ #
Als erstes Zeichen ist ein Buchstabe erforderlich, Leerzeichen sind nicht möglich.

369 Satznummer hinter ID oder KEY zu groß

Zu ID ... ID4 oder KEY ... KEY4 wurde für das Datenfeld eine Satznummer r (...) vor der Klammer angegeben. Diese Satznummer ist aber höher als die höchste durch RC zugelassene Nummer. Fehlt die Angabe RC in diesem Statement, so ist nur Satznummer 1 erlaubt.

370 Satznummer hinter ID oder KEY ungeeignet

Zu ID ... ID4 oder KEY ... KEY4 wurde für das Datenfeld eine Satznummer r (...) vor der Klammer angegeben. Diese Satznummer zeigt aber auf einen Datensatz, der nicht für jeden statistischen Fall vorhanden sein muss. (Angabe O: ... oder S: ... hinter RC). Fall-Identifikationen müssen aber für jeden statistischen Fall vollständig vorliegen.

371 Angaben S: ... liegen vor R: ... oder O: ...

Die Angaben R: ... und O: ... müssen vor den Angaben S: ... erscheinen.

372 Linie zu schmal oder zu breit

Für die Stärke einer Linie sind Werte zwischen 0.01 und 50.0 mm möglich.

373 Abstand zum Text zu groß

Der Abstand zwischen einer Linie und dem Text darf höchstens 20 mm betragen

374 Schriftgröße, Zeilenhöhe, Schriftbreite oder Absatzdistanz sind falsch.

Schriftgröße, Zeilenhöhe und Schriftbreite müssen zwischen 1 und 20 mm liegen.
Für die Absatzdistanz sind 0 bis 20 mm möglich.

375 Zu viele Spenderdateien

In einem Verarbeitungslauf sind beliebig viele FUSION-Statements zulässig aber höchstens 10 verschiedene Spenderdateien in FUSION FNAME= ...

376 Falsche Angabe nach einem IF

Nach IF hinter START-CNTD sind weder Erfassungsstatements noch BLK und EBLK möglich.

377 Mehrfaches BLK-Statement

Im Bereich eines BLK darf kein neues BLK begonnen werden.

378 Trennzeichen hinter TAB unzulässig

An dieser Stelle ist ein PAGEP-Statement wirksam. Dann können hinter TAB keine Trennzeichen x angegeben werden.

379 Falsches Schriftenkommando

In der Textzeile vor dem Fehlerstrich befindet sich eine Tilde ~.
Diese leitet immer ein Schriftenkommando ein, das hier fehlerhaft angegeben ist. Siehe Abschnitt *Textzeilen*.

380 Falsche Startposition

Ist ein Statement PAGEP wirksam, so müssen die Anfangspositionen START zwischen 0 und 500 000mm liegen. Ist dagegen PAGE wirksam, so sind Anfangspositionen zwischen 0 und 32 767 zulässig. Dies gilt auch, wenn weder PAGE noch PAGEP angegeben wurde.

381 TXT oder FNAME im Statement PAGEP

Diese Angaben zur Erstellung einer TXT-Datei sind nur im Statement PAGE nicht aber in PAGEP möglich.

382 Falsche Feldlänge

In COLBIN- und BINARY2-Dateien dürfen F- und G-Felder nur 1 ... 4 Spalten oder Felder (2 ... 8 Bytes) lang sein. Für I-Felder sind nur 1 ... 2 Spalten oder Felder (2 ... 4 Bytes) möglich.

383 Zeilen- oder Spaltenangabe fehlt

An dieser Stelle ist mindestens eine Zeilenangabe R(...) oder Spaltenangabe C(...) erforderlich.

384 Falscher Grauton

Als Grauton ist ein Prozentwert zwischen 0 (weiß) und 100 (schwarz) anzugeben.

385 Falscher Linientyp

Als Linientyp kann nur 1 für einfache Linien und 2 für doppelte Linien angegeben werden.

386 Rand zu breit

Ein in PAGEP hinter MARGINS angegebener Rand ist zu breit. Die nutzbare Breite des Blattes ist kleiner als 20mm oder die nutzbare Höhe kleiner als 10mm. Vielleicht wurde auch mit FORM ein falsches Formular angefordert.

387 Makrotext zu lang

Der Eingabetext für das hier definierte Makro ist länger als 65 535 Zeichen.

388 Variable mit Index in der Abkürzung

Vor dem Fehlerstrich befindet sich eine Abkürzung der Form Cn; ...m oder Nn; ...m. Eine in dieser Abkürzung enthaltene Variable wurde mit Index Cn[i] oder Nn[i] definiert. Eine Variable mit Index darf in einer solchen Abkürzung aber nicht vorkommen. Bei Variablen mit Index sind nur die Abkürzungen Cn[i]; ...[j] und Nn[i]; ...[j] möglich.

389 Zeilen- oder Spaltennummer falsch

Hier ist eine Zeilen- oder Spaltennummer erforderlich.
Sie muss zwischen 1 und 32 767 oder -1 und -32 767 liegen.

390 Rotationswinkel falsch

Als Rotationswinkel sind nur die Werte 0, 90 und -90 zulässig.
Außerdem ist eine solche Angabe nur für Spaltentexte möglich.

391 Mehrdeutige Wertezuweisung

Es handelt sich um ein Statement der Form:

```
- N1=N5 >0 . . .
```

Das Programm versteht den Anfang dieses Statements als Wertezuweisung

```
- N1=N5
```

mit der Prüfregel

```
>0
```

dahinter. So führt zum Beispiel das Statement

```
- N1 = N5>0 & <10
```

zum diesem Fehler 391 mit dem Fehlerstrich unter dem Zeichen &.
Damit wird angezeigt, dass >0 als Prüfregel für die davor stehende Wertezuweisung verstanden wird, zu der das Und-Zeichen & nicht passt.

Sollen die Angaben auf der rechten Seite des Gleichheitszeichens als logischer Ausdruck verarbeitet werden, so sind sie in Klammern zu setzen:

```
- N1 = (N5>0 & <10)
```

Jetzt erhält N1 den Wert 1 zugewiesen, wenn N5 größer als 0 und kleiner als 10 ist, andernfalls den Wert 0.

392 INPUT-SEC ist mit TRANSONLY nicht möglich

Diese interne Datei wurde durch OUTPUT-INT mit der Angabe TRANSONLY erstellt. Sie kann daher nicht als Eingabedatei für INPUT-SEC verwendet werden.
Gegebenenfalls muss die interne Datei neu erstellt werden, ohne die Angabe TRANSONLY.

393 Ungültiger Ausdruck in XTAB

Dieser Ausdruck liefert ein numerisches Ergebnis. Dann sind die Operatoren & | ^ < > = nicht möglich.

394 INPUT-SEC mit IDSHORT und ADD oder ADDP.

Das Statement INPUT-SEC enthält die Angabe IDSHORT. Diese verträgt sich nicht mit der gleichzeitig vorhandenen Angabe ADD oder ADDP: Wegen IDSHORT liefert INPUT-SEC nur Zusatzdaten zu vollständigen Fällen aus anderen Dateien, während ADD und ADDP vollständige Fälle aus INPUT-SEC erstellen.

395 Falsche Zeilen- oder Spaltennummer

Hier wird eine Zeilen- und Spaltennummer zwischen 1 und 32.767 oder zwischen -1 und -32.767 erwartet.
Außerdem müssen später angegebene Zeilen oder Spalten hinter den davorstehenden liegen.

396 Falsche Anzahl oder Schrittweite hinter INSERT

Eine Anzahlangabe (a) oder Schrittweite (s) hinter INSERT muss zwischen 1 und 255 liegen.

397 Falsche externe Datenfelder für C-Variable

Im Statement OUTPUT-EXT sind zu CSTD und in den Folgestatements nur folgende Angaben möglich:

- AUTOCOLBIN: I, K, L, M und U.
- AUTOQUANTUM: K, L, M und U.
- SPSS und SPSSUTF8: M, U und UM.
- TRIPLES: M.
- Sonstige Dateien mit SEP: M und U.
- Sonstige Dateien ohne SEP: B, I, K, L, M und U.

398 Falsche externe Datenfelder für N-Variable

Im Statement OUTPUT-EXT sind zu NSTD und in den Folgestatements nur folgende Angaben möglich:

- AUTOCOLBIN: D, F und G.
- AUTOQUANTUM: D.
- SPSS, SPSSUTF8: D.
- Sonstige Dateien mit SEP: D.
- Sonstige Dateien ohne SEP: D, F, G und P.

399 VNAME fehlt

Im Statement OUTPUT-EXT mit dem Format TRIPLES muß eine Datei VNAME für die Variablenbeschreibungen angegeben werden.

400 VNAME, CSTD, NSTD oder VSTD passen nicht zum Dateiformat

Es gelten folgende Regeln:
CSTD nicht mit ANSI, ASCII, COLBIN, EBCDIC, QUANTUM, TRIPLES.
MSTD nicht mit TRIPLES.
NSTD nicht mit ANSI, ASCII, COLBIN, EBCDIC, QUANTUM, XLSX, AUTOXLSX, TRIPLES.
VSTD nur mit SPSS, SPSSUTF8, TRIPLES und AUTO ...
VNAME nur mit SPSS, SPSSUTF8,TRIPLES und AUTO ... ausgenommen AUTOXLSX.

401 EBLK ohne BLK

Zu diesem EBLK fehlt das dazugehörige BLK.

402 Ungültiges Zielfeld für die Datenerfassung

Vor der Angabe E ist nur das einfache Gleichheitszeichen möglich. Links davon muß eine einfache Variable stehen. Als Index ist keine N-Variable zulässig.

403 Datensätze überfüllt

Die Datensätze der Datei OUTPUT-EXT bieten nicht genügend Platz für die Werte aller Variablen. Mit den Angaben SIZE oder RC in OUTPUT-EXT lassen sich die Satzlänge oder die Anzahl Datensätze pro Fall vergrößern.

404 Zu wenige Datensätze pro Fall.

Die hinter RC angegebene Anzahl von Datensätzen pro Fall ist zu klein, um alle Variablen aufnehmen zu können. Vielleicht ist auch die Satzlänge SIZE zu klein.

405 Zu viele Merkmale für das L-Format

Hier soll eine C-Variable in OUTPUT-EXT im L-Format ausgegeben werden, die mehr als 12 Merkmale besitzt. Das L-Format kann aber nur die Merkmale 1 bis 12 aufnehmen.

406 Mehrere Merkmale zu füllen

Diese Wertezuweisung versieht mehrere Merkmale einer C-Variablen mit neuen Werten.
Rechts vom Gleichheitszeichen steht ein logischer Ausdruck, der stets nur ein Merkmal liefert.

407 Name zu lang

Der Name für Erfassungs- BLK-Statements darf nicht länger als 40 Bytes werden.

408 Falsche Zeilen- oder Spaltenangabe

Hier sind Zeilen R(...), Spalten C(...), Tabellenzellen T(...) oder Variable Vi anzugeben.
Für R(...), C(...) und T(...) sind Zeilen- oder Spaltennummern aus dem davor liegenden XTAB-Statements möglich. Für R(...) und C(...) sind auch Klassenbuchstaben A ... Z aus dem XTAB-Statement zulässig. Als Variable sind V0 bis V99 möglich.

409 Schrittwert zu lang

In diesem DO-Statement besitzt ein Schrittwert mehr als 255 Zeichen. Vielleicht fehlt auch ein Trennzeichen (Komma, Leerstelle, ...) hinter einem Wert.

410 Abkürzung ... hier nicht möglich

Vor dem Fehlerstrich befindet sich eine Abkürzung **...** oder **..** für die Schrittwerte eines DO-Statements. Der Wert vor diesen Punkten ist keine Zahl, daher ist eine solche Abkürzung nicht möglich.

411 Von-Bis-Makros falsch sortiert

Das Bis-Makro hinter der Abkürzung **...** liegt in der alphabetischen Sortierfolge vor dem davor stehenden Von-Makro.

412 NSORT in INPUT-EXT hier nicht möglich

Die Statements INPUT-INT und INPUT-SEC können nur mit sortiert eingelesenen externen Dateien korrekt zusammenarbeiten.

413 Falsche Gruppennummer

Es sind maximal 9 Gruppenstufen mit den Werten 1 ... 255 zulässig.
Bei Von-Bis-Angaben mit **...** darf der zweite Wert nicht kleiner als der erste sein.

414 Text fehlt

Zu jeder Variablengruppe ist ein Gruppentext zwingend erforderlich, der in Hochkommas anzugeben ist. Textelemente (n) dürfen dabei nicht verwendet werden, um Verwechslungen mit Textelementen zu vermeiden, die der Variablengruppe zur Bildschirmanzeige zugeordnet werden sollen.

415 Vorige Gruppenstufe fehlt

Bei der Definition einer Gruppenstufe mit -G ... müssen alle Vorgänger bereits definiert sein: Zum Beispiel muss vor der Definition von -G7.3.1 die Stufe -G7.3 und vor -G7.3 die Stufe -G7 bereits vorhanden sein.

416 Satzlänge passt nicht zur Datei

Die Satzlänge SIZE aus FETCH-FILE passt nicht zur gefundenen Datei:
Die Gesamtlänge der Datei muss ein Vielfaches der Satzlänge sein.

417 Falsche Irrtumswahrscheinlichkeit

Hier wird eine Irrtumswahrscheinlichkeit zu den Grenzen CONFU oder CONFL eines Konfidenzintervalls als Prozentwert zwischen 0.01 und 50.00 erwartet.

418 Zeileneinzug falsch

Der Zeileneinzug muss numerisch sein und ist nur bei Zeilentexten möglich.

419 Ungültiger Key hinter FETCH

Ein Key hinter FETCH passt nicht zur Definition im Statement FETCH-FILE:

- Steht im zugehörigen Statement FETCH-FILE hinter KEY ein numerisches Feld, so sind hinter FETCH an dieser Stelle nur numerische Variable, numerische Konstanten oder numerische Fall-Identifikationen zulässig.
- Steht im zugehörigen Statement FETCH-FILE hinter KEY ein Feld für alphanumerische Werte, so sind hinter FETCH an dieser Stelle nur Textvariable, Textkonstanten in Hochkommas oder alphanumerische Fall-Identifikationen zulässig.

420 Key hinter FETCH mit falscher Länge

Der Schlüsselwert (Key) im Statement FETCH ist länger als die Angabe KEY im dazugehörigen Statement FETCH-FILE erlaubt.

421 Key mit Nachkommastellen hinter FETCH

Es wird eine numerische Variable oder Konstante mit Nachkommastellen als Key hinter FETCH verwendet. Es sind aber nur ganze Zahlen zulässig.

422 Schlüssel fehlt

In einem AK-, DK-, MK- oder UK-Feld ist ein Schlüssel in Hochkommas erforderlich.
Dieser Schlüssel darf nur fehlen, wenn auf der anderen Seite des Gleichheitszeichens eine Variable oder eine der Angaben ID, KEY oder RC steht.

423 Ein Wert passt nicht zu einer Variablen

Ein konstanter Wert passt nicht zu einer Variablen links vom Gleichheitszeichen.

424 Falsche Angabe zum externen Datenfeld

Bei externen Datenfeldern vom Typ AS, DS, AK, DK, MK und UK sind keine Längenangaben oder Bis-Werte möglich.

425 Von-Wert größer Bis-Wert

Der Von-Wert muss kleiner oder gleich dem Bis-Wert sein.

426 Falscher Feldtyp

Die Feldtypen AS, DS, MS, US, AK, DK, MK oder UK sind nur möglich mit der Angabe SEP oder für XLSX-Dateien.
Die Feldtypen A, D, F, G, P, B, I, K, L, M und U sind nicht möglich mit der Angabe SEP und für XLSX-Dateien.
Bei OUTPUT-EXT vertragen sich die Feldtypen AS, DS, MS und US nicht mit KEYLINE.

427 SEP mit falschem Dateiformat

Die Angabe SEP in diesem Statement verträgt sich nicht mit den Dateiformaten COLBIN, QUANTUM, XLSX, AUTOCOLBIN, AUTOQUANTUM, BINARY1, BINARY2, SPSS, SPSSUTF8, TRIPLES, XLSX und AUTOXLSX.

428 SKIP-Angabe zu groß

Diese Datei enthält weniger Datensätze als hinter SKIP angegeben.

429 Textjustage falsch

An dieser Stelle ist die Ausrichtung von Texten durch L, R , C oder T, B, M nicht zulässig.

430 Keine Datensätze angegeben

Dieses Statement enthält die Angabe RC. Dahinter sind aber keine Datensätze angegeben.

431 Datensätze für OUTPUT-INT zu lang

Es sind zu viele Variable auf OUTPUT-INT auszugeben, oder einige Variable benötigen zu viel Platz. Zur Speicherung von Variablenwerten stehen 10 000 000 Bytes für jeden statistischen Fall zur Verfügung. Das Problem kann nur durch Entfernen von überflüssigen Variablen gelöst werden oder durch Verringerung des Speicherbedarfs für einzelne Variable:

- Bei C-Variablen belegen jeweils 8 Merkmale ein Byte.
- N-Variable mit 1 oder 2 Ziffern belegen ein Byte, mit 3 oder 4 Ziffern zwei Bytes, mit 5 bis 7 Ziffern drei Bytes und mit mehr Ziffern vier Bytes.
- Bei T-Variablen belegt jedes mögliche Zeichen ein Byte.

432 Zu viele #IF-Statements geschachtelt

Vor diesem #IF liegen mindestens zehn weitere #IF-Statements, die noch nicht durch ein EIF abgeschlossen sind.
Dieser Fehler kann auch durch ein #IF in einer DO-Schleife entstehen, das nicht vor dem nächsten EDO mit EIF beendet wird.

433 #IF-Statement fehlt.

An dieser Stelle ist kein #IF-Statement wirksam. Alle eventuell davor liegenden #IF-Statements sind bereits durch #EIF abgeschlossen. Daher sind weder #EIF- noch #ELSE-Statements zulässig.

434 Schon ein #ELSE-Statements davor

Zu dem an dieser Stelle wirksamen #IF-Statement wurde bereits ein #ELSE-Statement angegeben.

435 #EIF-Statement fehlt

Am Ende der Statements fehlt das #EIF zum letzten davor liegenden #IF.

436 #EIF oder #ELSE auf falscher DO-Stufe

Dieses Statement und das dazugehörige #IF befinden sich nicht im Wirkungsbereich des gleichen DO-Statements:
Liegt ein #IF vor einem DO-Statement, so muss das dazugehörige #EIF oder #ELSE hinter dem entsprechenden EDO liegen.

437 #IF vor diesem EDO nicht beendet

Hinter dem letzten DO-Statement befindet sich ein #IF-Statement.
Dieses muss vor dem EDO mit #EIF beendet werden.

438 Falsches Argument einer Makrofunktion

Die Funktionen #NUM und #DEF verlangen als Argument eine Makro, das in Klammern eingeschlossen ist. Zum Beispiel #DEF(#ABC).

439 Falscher Sprachschlüssel

Es sind nur die Angaben LANG oder LANG1 ... LANG9 zulässig.

440 REPORT-FILE nicht vorhanden

Vor diesem REPORT-Statement liegt kein passendes REPORT-FILE-Statement oder die REPORT-Datei ist nicht zugänglich.

441 Chi-Quadrat-Test mit zu wenigen Zeilen oder Spalten

In einer Angabe PRINT ... PRINT3 wird ein Chi-Quadrat-Test CHI2 angefordert.
Zu diesem gibt es aber weniger als zwei Zeilen oder zwei Spalten.

442 Zeilen- oder Spaltenangabe falsch

Hier ist eine Zeile oder Spalte anzugeben in der Form 1 ... 32 767, -1 ... -32 767, A ... Z oder -A ... -Z.

443 Klasse fehlt in ROW oder COL

In einer der Angaben PRINT ... PRINT3 wird auf eine Klasse A{} ... Z{} prozentuiert.
Diese Klasse fehlt aber in ROW oder COL.

444 Schlüsselwort leer

Das in Hochkommas eingeschlossene Schlüsselwort enthält nur Leerzeichen.

445 Falsche Gruppenstufe

In den Folgestatements von OUTPUT-INT lassen sich Gruppen nur als -G1 bis -G255 selektieren.

446 Falsche Anzahlangabe

Die Anzahlangaben müssen größer als 0 sein. Folgende Höchstwerte sind möglich:

I-Format:	200 Nennungen
L-Format:	20 Nennungen
B- K- M- U-Format:	9 999 Nennungen
D- P-Format:	30 Ziffern
F- G-Format:	18 Ziffern
A-Format:	4 000 Zeichen

447 Doppelpunkt fehlt

An dieser Stelle ist ein Doppelpunkt : erforderlich.

448 Schlüssel zu lang

Der Schlüssel zu einem AK-, DK-, MK- oder UK-Feld darf höchstens 64 Zeichen lang sein. Vielleicht fehlt auch die Klammer) oder das Hochkomma hinter dem Schlüssel.

449 Falscher Prozentwert

Dieser Prozentwert muss zwischen 0.01 und 99.9 liegen.

450 Reihenfolgenummer nicht möglich

Links vom Gleichheitszeichen ist eine Reihenfolgenummer (...) nicht erlaubt.

451 Zu viele oder zu große Tabellen

Der Speicherbereich für die Zählergebnisse ist erschöpft. Es wurden zu viele oder zu große Tabellen angefordert.

452 KEYLINE ohne SEP

Die Angabe KEYLINE verlangt zwingend ein Separatorzeichen SEP.

453 Längenfehler im KEYLINE-Satz

Das Statement INPUT-EXT oder FETCH-FILE enthält die Angabe KEYLINE.
Entweder ist der entsprechende Datensatz länger als unter SIZE angegeben oder ein Schlüssel in diesem Satz ist länger als 20 Zeichen.

454 Schlüssel fehlt im KEYLINE-Satz

Das Statement INPUT-EXT oder FETCH-FILE enthält die Angabe KEYLINE.
Der Schlüssel dieses AK- DK- MK- oder UK-Feldes ist nicht in dem KEYLINE-Satz enthalten.

455 Anzahl auszulassender Datensätze hinter SKIP zu klein

Das Statement INPUT-EXT oder FETCH-FILE enthält die Angabe KEYLINE.
Dabei muss die Anzahl auszulassender Datensätze hinter SKIP mindestens so groß sein, wie die Anzahl der Datensätze pro Fall aus RC.

456 Angaben zu ID, RC oder KEY in verschiedenen Spalten

In den ersten Datensätzen dieser Datei mit KEYLINE-Angaben befinden sich die Schlüssel für ID, RC oder KEY in verschiedenen Spalten.

457 TRANS oder TRANSONLY mit START-CNTD

Die Angaben TRANS und TRANSONLY im Statement OUTPUT-INT vertragen sich nicht mit START-CNTD.

458 Falsche ID-Länge

Hier wird die Zeichenanzahl für die Eingabe der ID-Werte erwartet. Sie darf nicht größer sein als die entsprechende Angabe in INPUT-EXT, INPUT-INT oder OUTPUT-INT.

459 Text zu MSG ist zu lang

Diese Texte dürfen höchstens 200 Bytes lang sein.

460 GOTO-Statement ohne Sprungziel

In diesem GOTO-Statement fehlt die Zielangabe.

461 Falsche LANG-Angabe

Hier wird die Nummer 0 ... 9 des Sprachenschlüssels erwartet.

462 Text zu MSG fehlt

An dieser Stelle fehlt der Meldungstext.

463 Ungültige Von-Bis-Angabe hinter I= ...

Es liegt einer der folgenden Fehler vor:

- Von-Angabe ist größer als die Bis-Angabe.
- Zwei Von-Bis-Angaben überschneiden sich.
- Eine Angabe liegt außerhalb des Wertebereichs der Variablen.

464 Falsches Dateiformat für COPY in OUTPUT-EXT

Eingabedateien INPUT-EXT mit der Angabe SEP lassen sich nicht mit COPY in die Formate COLBIN, QUANTUM, BINARY1 und BINARY2 kopieren.
INPUT-EXT-Dateien ohne die Angabe SEP lassen sich nicht in Dateien mit der Angabe SEP kopieren und umgekehrt.
Eingabedateien im BINARY1- und BINARY2-Format lassen sich nicht in andere Formate umwandeln.
Umwandlungen zwischen UTF8- und anderen Formaten sind nur möglich, wenn INPUT-EXT die Angabe SEP enthält.

465 Eingabe mehrfach

Es liegt einer der folgenden Fehler vor:

- Der Name dieses Erfassungs- oder BLK-Statements wurde schon an anderer Stelle verwendet.
- Diese Eingabevariable wurde an anderer Stelle bereits verwendet.

466 Sprungziel ist nicht definiert

Die folgenden Sprungziele sind nicht definiert oder liegen in einem Block.

467 GOTO EBLK ungültig

Dieses Statement befindet sich nicht im Bereich eines BLK.

468 START-CNTD ungültig

Die Angabe DROP im Statement OUTPUT-INT erlaubt kein START-CNTD.

469 Ungültiges Sprungziel

Die als Sprungziel angegebene Variable oder Sprungmarke muss hinter dem Statement liegen, aus dem sie angesprungen werden soll. Entweder fehlt das Sprungziel oder es liegt vor dem laufenden Statement. Vielleicht wurde auch versucht, mit einem Sprung die DO-Stufe zu wechseln.

470 Ungültige Variable im IF-Ausdruck

Die Variablen in diesem logischen Ausdruck müssen zwischen den Statements CNTD-START und GOTO aufgeführt sein, um bei der Erfassung einen Wert zu erhalten.

471 Ungültiges Format der Statementdatei

An dieser Stelle befinden sich die drei BOM-Zeichen, mit denen eine UTF8-Datei beginnt. Die Datei enthält jedoch ungültige UTF8-Zeichen.

472 Ungültige Prüfziffernangabe

Die Angabe CD ist nur bei numerischen Fall-Identifikationen mit mindestens zwei Ziffern möglich.

473 Name des Arbeitsblattes ist zu lang

Es sind maximal 31 Zeichen möglich.

474 Ungültige Angabe hinter RC

Im Dateiformat SPSS, SPSSUTF8 und TRIPLES ist die Angabe O: für einen optionalen Datensatz nicht möglich.

475 Ungültige Angabe SEP

In den Statements PAGE und PAGEP ist die Angabe SEP nur zusammen mit der Angabe CSV zulässig.

476 Falsche Variable Vi oder Konstante

Eine Variable Vi oder Konstante (...) in diesem Statement enthält keine oder zu wenige Werte.

477 Zeilen und Spalten vermischt

Im Statement PROSUM-ROWS ist rechts vom Gleichheitszeichen die Angabe C( ) nicht möglich und im Statement PROSUM-COLS ist dort R( ) nicht erlaubt,
Beginnt eine Wertezuweisung mit R( ), so ist rechts vom Gleichheitszeichen C( ) nicht zulässig und eine Wertezuweisung mit C( ) am Anfang erlaubt rechts nicht R( ).

478 Falscher Toleranzwert

Hinter BASE ist ein Toleranzwert zur Auswahl der Rundungsfaktoren erforderlich.
Dieser muss zwischen 0.0001 und 100 liegen und ist durch ein Komma von den vorigen Angaben zu trennen. Als Dezimalzeichen ist der Punkt zu verwenden.

479 Falsche Farbangabe

Hier ist eine Farbe CO1 ... CO16, ein Grauton G1 ... G16 oder ein Grauton als Prozentwert zwischen 0 (weiß) und 100 (schwarz) erforderlich.

480 Falsche oder fehlende Logo-Datei

Hier ist eine Bilddatei im JPEG-Format mit der Dateierweiterung JPG erforderlich. Eine Datei ohne Pfadangabe wird der Reihe nach in folgenden Verzeichnissen gesucht:

- Im Verzeichnis der REP-Datei.
- In dem Verzeichnis, das in der Datei CNTW.INI oder CNTA.INI hinter LOGO angegeben ist.
- Im Programmverzeichnis CNTA oder CNTW.

481 Falscher Schlüssel

Der Von- oder Bis-Schlüssel endet nicht mit einer Zahl, die Zahl der Bis-Angabe ist kleiner als die der Von-Angabe oder der Bis-Schlüssel besteht aus mehr als 30 Zeichen.

482 Falsche Anzahl von Schlüsseln

In AK- und DK-Feldern ist nur ein Schlüssel möglich, in MK- und UK-Feldern sind maximal 1 000 Schlüssel zulässig.

483 MTT... oder PRINT=CORR passen nicht zu MEAN in oberen Staffeln

Die t-Tests MTT... sowie die Angabe PRINT=CORR verlangen, dass die Funktion MEAN in ROW und COL nur in der unteren (rechten) Staffel {...} vorkommt. Hier erscheint MEAN jedoch in einer oberen Staffel.

484 Fehler mit PRINT=CORR

Hier ist PRINT=CORR aus XTAB wirksam. Dann sind folgende Angaben unzulässig:

- Alle numerische Ausdrücke und Funktionen außer MEAN in ROW und COL.
- t-Tests mit MTT... in ROW und COL.
- Das Statement XADD hinter XTAB; XADDB hingegen ist möglich.

485 Interne Datei mit TRANSONLY

Die interne Datei FNAME enthält ihre Daten nur in transponierter Form. Sie wurde durch OUTPUT-INT mit der Angabe TRANSONLY erstellt. Sie lässt sich an dieser Stelle so nicht verarbeiten und müsste noch einmal ohne TRANSONLY erstellt werden.

486 Ungültige Angabe hinter COi oder Gi

Hier sind nur die Angaben L1, L2, L12 sowie Von-Bis-Zählerwerte der Form 0...100 möglich.

487 SELECT vor FNAME

Die Angaben SELECT und SELECTD sind hinter FNAME anzugeben, um die Variablen des logischen Ausdrucks verarbeiten zu können.

488 Wertebereich Spender- oder Empfängervariable falsch

Hier wird eine Angabe (a ... b) mit numerischen Werten erwartet.
Der Wert a muss kleiner oder gleich dem Wert b sein.
Der vorgegebene Bereich darf nicht außerhalb des Wertebereichs der Variablen liegen.

489 Spender- oder Empfängervariable vor FNAME

Die Angaben der Spender- und Empfängervariablen müssen hinter der FNAME-Angabe liegen.

490 Normierungsintervall ungültig

Normierungsintervalle sind nur für metrische (MET) Vergleiche zulässig.
Hier wird eine Angabe (a ... b) mit numerischen Werten erwartet, wobei a kleiner als b sein muss.

491 Ungültiger Gewichtungsfaktor

Gewichtungsfaktoren sind nur bei metrischen (MET) oder nominalen (NOM/NOMM) Vergleichsregeln zulässig. Die Faktoren müssen Werte größer 0 sein.

492 Verbindungsvariable passt nicht zu NOMM

Die Vergleichsregel NOMM ist nur für C-Variable möglich.
In den Klammerausdrücken hinter den Variablen sind nur positive ganzzahlige Werte möglich.
Die Wertebereiche von Spender und Empfänger müssen übereinstimmen; eventuell sind hinter den Variablen geeignete Bereiche in Klammern anzugeben.

493 Falsche Angabe für BESTFIT oder LIMIT

BESTFIT verlangt die Angabe ON oder Werte größer 0 und
LIMIT Werte zwischen 1 und 100.

494 Falsche Angaben in OUTPUT-EXT

- Z0= und MV= und die Label-Angaben I, V, IV und VI sind nur bei SPSS-Dateien möglich.
- *s='valuelabel'* ist nur bei C- und N-Variablen im U-Format mit IV oder VI für s = 0 oder 1 möglich.
- Z0= bei C-Variablen ist nur im M-Format zulässig.
- MV = ist nur bei C- und N-Variablen möglich.
- DATE ist nur bei SPSS-Dateien für N-Variable ohne Nachkommastellen möglich

495 Ungültige Angabe zu MV= oder Z0=

- Hinter MV= sind nur maximal drei Einzelwerte oder ein Einzelwert mit einer Von-Bis-Angabe möglich.
- Bei Z0= für T-Variable darf die Angabe maximal 100 Zeichen lang sein und die Länge A= nicht überschreiten.
- Bei Z0= für N-Variable darf die Konstante nicht länger als die durch D= vorgegebene Länge sein.

496 Ungültiger G-Operand

In logischen Ausdrücken von XTAB ist die Angabe G- nur am Anfang des Ausdrucks zulässig.

497 Arbeitsblatt nicht in der XLSX-Datei

Das Arbeitsblatt WORK ist nicht in der XLSX-Datei enthalten.

498 Ungültige XLSX-Datei

Dies ist keine gültige XLSX-Datei.

499 Falsche Zeilen- oder Spaltenangabe

Hinter STYLE=ROW oder STYLE=COL sind die Zeilen- oder Spaltenangaben 1, 2, 3 ... oder -1, -2, -3 ... möglich. Bei Von-Bis-Angaben muss die erste Angabe vor der zweiten liegen, also ROW( )(1..4) oder ROW( )(-2...-1) und nicht ROW( )(4...1) oder ROW( )(-1...-2).

500 Ungültiger COLBIN-Wert: xxxx = Xyyyy Stmt: ssss

Bei der Verarbeitung des Statements *ssss* wird ein Wert im COLBIN-Format erwartet. Stattdessen wurde ein ungültiger Wert gefunden:
Die beiden ersten (linken) Bits eines jeden Bytes im COLBIN-Format müssen auf 0 stehen; das ist hier nicht der Fall.
Die hexadezimalen Werte aller gültigen COLBIN-Werte sind in der Tabelle *Lochpositionen der COLBIN-Bytes* zu finden.
Die Angabe *yyyy* zeigt den gefundenen Wert in hexadezimaler Schreibweise. *xxxx* gibt Typ, Satz, Position und Länge des Eingabefeldes an.
Wahrscheinlich besitzt die Eingabedatei nicht das COLBIN-Format.

Der statistische Fall wird verarbeitet, das Statement mit der Nummer ssss aber ausgelassen. Verursacht jedoch eine Fall-Identifikation ID, ein Satz-Kennzeichen RC oder ein Schlüsselwert KEY den Fehler, so wird der ganze Fall unverarbeitet übergangen.

501 Ungültige Mehrfachlochung: xxxx = Xyyyy Stmt: ssss

Bei der Verarbeitung des Statements *ssss* wurde ein Wert im COLBIN-Format mit Mehrfachlochungen in den Positionen 1...7 gefunden. Das ist für das hier vorliegende Datenfeld aber unzulässig.
Die Angabe *yyyy* zeigt den gefundenen Wert in hexadezimaler Schreibweise. *xxxx* gibt Typ, Satz, Position und Länge des Eingabefeldes an.

Der statistische Fall wird verarbeitet, das betroffene Statement mit der Nummer ssss aber ausgelassen. Verursacht jedoch eine Fall-Identifikation ID, ein Satz-Kennzeichen RC oder ein Schlüsselwert KEY den Fehler, so wird der ganze Fall unverarbeitet übergangen.

502 Keine gepackte Zahl: xxxx = Xxxxx Stmt: ssss

Bei der Verarbeitung des Statements *ssss* wird in der Eingabe eine gepackte Dezimalzahl erwartet. Der gefundene Wert erfüllt diese Bedingung aber nicht.
Die Angabe *yyyy* zeigt den gefundenen Wert in hexadezimaler Schreibweise.
xxxx gibt Typ, Satz, Position und Länge des Eingabefeldes an.

Der statistische Fall wird verarbeitet, das betroffene Statement mit der Nummer ssss aber ausgelassen. Verursacht jedoch eine Fall-Identifikation ID, ein Satz-Kennzeichen RC oder ein Schlüsselwert KEY den Fehler, so wird der ganze Fall unverarbeitet übergangen.

503 Kein gültiger numerischer Wert: xxxx = yyyy Stmt: ssss

Bei der Verarbeitung des Statements *ssss* wird ein numerischer Eingabewert erwartet. Der gefundene Wert *yyyy* ist aber nicht numerisch oder hier nicht zulässig.
xxxx gibt Typ, Satz, Position und Länge des Eingabefeldes an.

Der statistische Fall wird verarbeitet, das betroffene Statement mit der Nummer ssss aber ausgelassen. Verursacht jedoch eine Fall-Identifikation ID, ein Satz-Kennzeichen RC oder ein Schlüsselwert KEY den Fehler, so wird der ganze Fall unverarbeitet übergangen.

504 Ungültiges Satz-Kennzeichen: xxxx = yy Stmt: ssss

Beim Einlesen eines Falles aus INPUT-EXT oder FETCH-FILE wurde ein Datensatz mit dem ungültigen Satz-Kennzeichen yy gefunden.
xxxx zeigt Typ, Position und Länge des Satz-Kennzeichens an, so wie bei der Angabe RC im Statement mit der Nummer ssss angegeben.

Der statistische Fall wird verarbeitet, jedoch ohne den ungültigen Datensatz.

505 Fehlender Satz: xxxx = yy Stmt: ssss

Beim Einlesen eines Falles aus INPUT-EXT oder FETCH-FILE wurde ein Datensatz mit dem Satz-Kennzeichen *yy* erwartet, aber nicht gefunden.
xxxx zeigt Typ, Position und Länge des Satz-Kennzeichens an, so wie bei der Angabe RC im Statement mit der Nummer *ssss* angegeben.

Wurde der fehlende Satz mit der Angabe R: hinter RC angefordert oder liegt eine ID/KEY-Stufe in diesem Satz, so wird der Fall unverarbeitet übergangen. Wurde zu dem Satz-Kennzeichen nicht R: angegeben, so wird der fehlende Datensatz durch Leerzeichen ersetzt und der Fall verarbeitet.

506 Mehrere Sätze: xxxx = yy Stmt: ssss

Beim Einlesen eines statistischen Falles von INPUT-EXT oder FETCH-FILE wurde mehr als ein Datensatz mit dem Satz-Kennzeichen yy gefunden.
xxxx zeigt Typ, Position und Länge des Satz-Kennzeichens an, so wie bei der Angabe RC im Statement mit der Nummer ssss angegeben.

Der statistische Fall wird verarbeitet. Von den mehrfach vorhandenen Datensätzen wird nur der zuletzt gelesene Satz verwendet.

507 Eingabewert zu groß oder zu lang: xxxx = yyyy Stmt: ssss

Bei der Verarbeitung des Statements ssss wurde der Wert yyyy gefunden, der für die weitere Verarbeitung zu groß ist.
xxxx gibt Typ, Satz, Position und Länge des Eingabefeldes an. Ist yyyy Ergebnis eines numerischen Ausdrucks, so erscheint xxxx nicht in der Fehlermeldung. Zur genaueren Untersuchung des Fehlers ist dann das Statement mit der Nummer ssss heranzuziehen.
Das Zielfeld oder die Zielvariable sind nicht groß genug, um den Wert aufzunehmen.

Der statistische Fall wird verarbeitet, das betroffene Statement mit der Nummer ssss aber ausgelassen. Verursacht jedoch eine Fall-Identifikation ID, ein Satz-Kennzeichen RC oder ein Schlüsselwert KEY den Fehler, so wird der ganze Fall unverarbeitet übergangen.

508 Falscher Wert: xxxx = yyyy Stmt: ssss

Bei der Verarbeitung des Statements ssss wurde der numerische Wert yyyy gefunden. Dieser widerspricht einer Prüfbedingung des Statements.
Der falsche Wert wird trotzdem verarbeitet.
xxxx gibt Typ, Satz, Position und Länge des Eingabefeldes an.

509 Falsche Anzahl von Merkmalen: xxxx = Zy Stmt: ssss

Die im Statement ssss geforderte Anzahl von Angaben wurde im externen Datensatz nicht gefunden. Stattdessen ist die Anzahlbedingung Zy erfüllt.
Der falsche Wert wird trotzdem verarbeitet.
xxxx gibt Typ, Satz, Position und Länge des Eingabefeldes an.

510 Reihenfolgefehler INPUT-EXT / FETCH-FILE Stmt: *ssss*

Voriger Fall: *xxxx*
Die Eingabedatei INPUT-EXT ist nicht nach ihren Fall-Identifikationen aufsteigend sortiert oder eine FETCH-Datei nicht nach ihren KEY-Angaben.
Der falsch einsortierte Fall wird nicht verarbeitet. Seine Fall-Identifikation erscheint über dieser Meldung. *xxxx* ist der letzte zuvor verarbeitete Fall.
Vielleicht liegen auch zwei Fälle mit gleicher Fall-Identifikation direkt hintereinander. Dabei ist die Bedingung CID.E für den verarbeiteten ersten Fall noch nicht gesetzt.
Zur Fehlerbeseitigung siehe im Statement INPUT-EXT die Angaben SORT, ASORT, DSORT, NSORT oder MULTIPLE.

511 Negatives Fallgewicht OLD in WEIGHT: x Stmt: ssss

In WEIGHT ist ein altes Fallgewicht OLD=Nn angegeben.
Die Variable Nn besitzt in dem laufenden statistischen Fall den ungültigen Wert x.
ssss ist die Statementnummer des betroffenen WEIGHT Statements.
Die Verarbeitung wird nach dem laufenden WEIGHT Statement abgebrochen.

512 Position oder Index ungültig: xxx Stmt: ssss

Das Statement mit der Nummer ssss enthält entweder

- eine Variable mit einem Index, der selbst eine N-Variable besitzt oder
- eine N-Variable als Positionsangabe mit ungültigem Wert
- Den Wert 0 oder negative Werte,
- Werte größer als der maximale Index der Variablen,
- Positionsangaben, die über den aktuellen Datensatz hinausreichen.

Der statistische Fall wird verarbeitet, das betroffene Statement mit der Nummer ssss aber ausgelassen.

513 Fehlende Datensätze: xx Stmt: ssss

Im Statement INPUT-EXT oder FETCH-FILE mit der Statementnummer ssss wurde mit der Angabe RC=n die Anzahl von Datensätzen pro Fall festgelegt. Zu dem vorliegenden Fall wurden aber nur xx Datensätze gefunden.

Der unvollständige Fall wird verarbeitet, die fehlenden Sätze durch Leerzeichen (Blanks) ersetzt.

514 Datensatz zu lang: xxxx Stmt: ssss

Im Statement INPUT-EXT oder FETCH-FILE mit der Statementnummer ssss wurde SIZE=EOL oder QUANTUM angegeben. Daher müssen alle Datensätze mit einem Zeilenende-Zeichen (CR=10 oder LF=13) beendet werden.
Der vorliegende Datensatz ist xxxx Bytes lang, also länger als die maximale Satzlänge aus SIZE. Fehlt die Angabe SIZE, so nimmt das Programm dafür den Wert 80 an.
Der falsche Satz wird verarbeitet, jedoch rechts abgeschnitten.

515 INPUT-SEC mit ADDP: Mehrfache Fall-ID Stmt: ssss

Das Statement **ssss** ist ein INPUT-SEC mit der Angabe ADDP.
Aus dieser Datei wurde ein Datensatz eingelesen, dessen Fall-Identifikation mit dem zuvor verarbeiteten Fall übereinstimmt.
Dieses ist nur eine Warnung. Der aus INPUT-SEC eingelesene Datensatz wird verarbeitet.

516 Zu viele Nennungen in xxxx = nnn Stmt: ssss

Bei der Wertezuweisung aus dem Statement *ssss* wurden nnn Nennungen im Feld *xxxx* gefunden. Das Zielfeld kann so viele Angaben nicht aufnehmen.

Es werden nur so viele Werte übernommen, wie das Zielfeld aufnehmen kann.
Der statistische Fall wird danach verarbeitet.

517 Ungültiger Eingabewert: xxxx = yyyy Stmt: ssss

Bei der Wertezuweisung aus dem Statement ssss wurde ein ungültiger Eingabewert gefunden. Das Zielfeld oder die Zielvariable xxxx können den Wert yyyy nicht aufnehmen.

Der statistische Fall wird verarbeitet, das betroffene Statement mit der Nummer ssss aber ausgelassen. Verursacht jedoch eine Fall-Identifikation ID, ein Satz-Kennzeichen RC oder ein Schlüsselwert KEY den Fehler, so wird der ganze Fall unverarbeitet übergangen.

518 Datensatz fehlt in FETCH-FILEn Stmt: ssss

Zu dem davor stehenden KEY fehlt der Datensatz in der Datei FETCH-FILEn.
Das Zielfeld oder die Zielvariable im Statement ssss wird nicht verändert.

Mit der Angabe NOM im Statement ssss lässt sich diese Fehlermeldung für das laufende Statement unterdrücken und mit MSG=FETCH:NO im Statement OPTIONS lässt sich diese Fehlermeldung für alle Statements unterbinden.

519 Datensatz übergelaufen: xx Stmt: ssss

Ein Datensatz in INPUT-EXT oder OUTPUT-EXT ist durch Ausgabe eines Datenwertes zu lang geworden. Dabei ist xx die Anzahl der verlorenen Zeichen und ssss die Statementnummer der Wertezuweisung, die den Überlauf verursachte.

Der Ausgabewert wird an der vorgesehenen Stelle eingefügt jedoch hinten abgeschnitten.
Der statistische Fall wird weiter verarbeitet.

520 Satzlänge in FETCH-FILEn falsch. Stmt: ssss

Im Statement FETCH-FILEn liegt keine Angabe SIZE mit EOL vor. Dann muss die Dateilänge ein ganzes Vielfaches der Satzlänge sein. Das ist hier aber nicht der Fall.
Mögliche Ursachen:

- Es wurde keine oder eine falsche Satzlänge SIZE angegeben.
- Es liegt eine ASCII-Datei mit Zeilenende-Zeichen vor (CR und LF); dann ist aber die Angabe EOL bei SIZE erforderlich.

521 Datensatz doppelt in FETCH-FILEn Stmt: ssss

Zu dem davor stehenden KEY gibt es in der Datei FETCH-FILEn mehrere Datensätze mit gleichem KEY und gleichem Satz-Kennzeichen RC.
Der letzte der mehrfachen Datensätze wird verarbeitet.

522 Eingabewert zu klein oder zu lang: xxxx = yyyy Stmt: ssss

Bei der Wertezuweisung mit dem Statement ssss wurde der Wert yyyy gefunden, der für die Verarbeitung zu klein ist.
xxxx gibt Typ, Satz, Position und Länge des Eingabefeldes an.
Ist yyyy Ergebnis eines numerischen Ausdrucks, so erscheint xxxx nicht in der Fehlermeldung. Zur genaueren Untersuchung des Fehlers ist dann das Statement mit der Nummer ssss heranzuziehen.
Das Zielfeld oder die Zielvariable sind nicht geeignet, um den Wert aufzunehmen.

Der statistische Fall wird verarbeitet, das betroffene Statement mit der Nummer ssss aber ausgelassen. Verursacht jedoch eine Fall-Identifikation ID, ein Satz-Kennzeichen RC oder ein Schlüsselwert KEY den Fehler, so wird der ganze Fall unverarbeitet übergangen.

523 Eingabewert zu lang: xxxx = yyyy Stmt: ssss

Bei der Verarbeitung des Statements ssss wurde der Wert yyyy gefunden, der für die weitere Verarbeitung zu lang ist.
xxxx gibt Typ, Satz, Position und Länge des Eingabefeldes an.
Das Zielfeld oder die Zielvariable kann den Wert nicht aufzunehmen.

Der statistische Fall wird normal verarbeitet, das betroffene Statement mit der Nummer ssss aber ausgelassen. Verursacht jedoch eine Fall-Identifikation ID, ein Satz-Kennzeichen RC oder ein Schlüsselwert KEY den Fehler, so wird der ganze Fall unverarbeitet übergangen.

524 Merkmal passt nicht zum Zielfeld: xxxx = yyyy Stmt: ssss

Bei der Wertezuweisung mit dem Statement ssss wurde im Datenfeld xxxx der Merkmalswert yyyy gefunden, der zu groß oder zu klein für das Zielfeld ist.

Der statistische Fall wird verarbeitet. Alle passenden Merkmale des Eingabefeldes werden in das Zielfeld übertragen. Diese Meldung erscheint nur nach der Angabe MSG=RANGE:YES im Statement OPTIONS.

525 Schlüssel nicht im Datensatz gefunden: xxxx Stmt: ssss

In der Wertezuweisung mit dem Statement ssss wurde der Schlüssel aus dem Datenfeld xxxx nicht im Datensatz gefunden.

Der statistische Fall wird verarbeitet, das Zielfeld aber nicht verändert. Diese Meldung erscheint nur nach der Angabe MSG=MISSING:YES im Statement OPTIONS.

526 Schlüssel nicht im Datensatz gefunden: xxxx Stmt: ssss

Im Statement *ssss* wurde der Schlüssel für ID, KEY oder RC aus dem Datenfeld *xxxx* nicht im Datensatz gefunden. Die Angabe Fall-ID zeigt auf den letzten korrekt gefundenen Fall.

Der betroffene Datensatz wird unverarbeitet übergangen.

527 Ungültige UTF8-Zeichen: xxxx = yyyy Stmt: ssss

Bei der Verarbeitung des Statements *ssss* wurde im Datenfeld *xxxx* der Wert *yyyy* gefunden, der ungültige UTF8-Zeichen enthält. Diese wurden durch ? ersetzt.

Der betroffene Datensatz wird unverarbeitet übergangen.

528 Ungültiges Datum: xxxx Stmt: ssss

Bei der Verarbeitung des Statements ssss wurde in einer N-Variablen der Wert xxxx gefunden.
Dies ist kein gültiges Datum der Form JJJJ MM TT, wie es die Funktionen WEEK( ), DAYW( ) oder DAYY( ) sowie in Angabe DATE bei der SPSS-Ausgabe verlangen.
Für SPSS darf das Datum nicht vor dem 1. 1. 1970 liegen.

Anstelle des ungültigen Datums wird der Wert Z0 (keine Angabe) ausgegeben.

529 Datensatz übergelaufen: xx Stmt: ssss

Ein Datensatz in OUTPUT-EXT mit XLSX enthält mehr als die von Excel akzeptierte Anzahl von 16 384 Spalten. Dabei ist xx die übergelaufene Spaltennummer und ssss die Statementnummer der Wertezuweisung, die den Überlauf verursachte.

Der Datenwert wird nicht ausgegeben, der statistische Fall jedoch weiter verarbeitet.

600 Zu viele Fußnoten.

Auf dieser Seite sollen mehr als 40 verschiedene Fußnoten gedruckt werden.
Die nächste und alle folgenden Fußnoten wurden entfernt.

601 CONT aus vorigem XTAB entfernt.

Bei der Druckaufbereitung dieser Tabelle war die Angabe CONT wirksam.
Da die Tabelle aber nicht mehr auf die alte Seite passt, wurde CONT nicht beachtet, sondern ein neues Blatt begonnen.

602 SUPPRESS=SPACES eingefügt.

Die Tabelle passt in der Höhe nicht auf eine Seite. Durch Einfügen von SUPPRESS=SPACES hat das Programm versucht, Platz zu schaffen.

603 Text in Position xx **nicht gedruckt.**

Die Seite ist zu schmal. Ein Text beginnt in Position xx rechts außerhalb der Seite. Er kann deshalb nicht gedruckt werden.

604 START verändert.

Die Tabelle lässt sich wegen Platzmangel nicht auf der Seite ausdrucken. Das Programm hat die Angabe START aus XTAB verändert, um Platz zu schaffen.

605 Fußnoten entfernt.

Die Tabelle lässt sich wegen Platzmangel nicht auf der Seite ausdrucken. Das Programm hat alle Fußnoten entfernt, um Platz zu schaffen.

606 RROW verändert.

Die Tabelle lässt sich wegen Platzmangel nicht auf der Seite ausdrucken. Das Programm hat die Angabe RROW aus XTAB verändert, um Platz zu schaffen.

607 RCOL verändert.

Die Tabelle lässt sich wegen Platzmangel nicht auf der Seite ausdrucken. Das Programm hat die Angabe RCOL aus XTAB verändert, um Platz zu schaffen.

608 BOTTOM entfernt.

Die Tabelle lässt sich wegen Platzmangel nicht auf der Seite ausdrucken. Das Programm hat die Angabe BOTTOM aus XTAB entfernt, um Platz zu schaffen.

609 TITLE entfernt.

Die Tabelle lässt sich wegen Platzmangel nicht auf der Seite ausdrucken. Das Programm hat die Angabe TITLE aus XTAB entfernt, um Platz zu schaffen.

610 SUPPRESS=COLTEXT eingefügt.

Die Tabelle passt in der Höhe nicht auf eine Seite. Durch Einfügen von SUPPRESS=COLTEXT hat das Programm versucht, Platz zu schaffen.

611 Tabelle in mehrere Teile zerlegt.

Die Tabelle passt nicht auf eine Seite. Sie wird in mehrere Teiltabellen zerlegt.

612 Tabelle leer, nicht gedruckt.

Durch die Angaben SUPPRESS=EMPTY, FROW, FCOL ... ist eine Tabelle ohne Zähler entstanden. Diese wird nicht gedruckt.

613 Tabelle zu groß, nicht gedruckt.

Die Tabelle passt nicht auf ein Blatt. Sie lässt sich auch nicht verkleinern oder vollständig auf mehrere Seiten verteilen. Sie wird gar nicht oder nur teilweise ausgegeben.

614 Ausgabewert in Spalte s **verkürzt.**

Ein Ausgabewert für die Spalte s der Tabelle ist zu breit für den vorhandenen Platz. Das Programm versucht, den Wert trotzdem auszugeben. Dazu werden eventuelle Druckmasken entfernt, die Anzahl von Nachkommastellen reduziert oder der Wert stärker an die Spaltenränder herangerückt.
Führt das alles nicht zum Erfolg, so wird anstelle des Wertes ein Stern * gedruckt.

615 Ausgabewert verkürzt.

Ein Ausgabewert passt nicht in den vorgesehenen Platz. Das Programm versucht, den Wert trotzdem auszugeben. Dazu werden eventuelle Nachkommastellen entfernt.
Führt das nicht zum Erfolg, so wird anstelle des Zahlenwerts ein Stern * gedruckt.

616 Zu viele Signifikanzmarkierungen in Zeile oder Spalte s.

Beim Vergleich von Werten mit der Angabe PRINT=SIG... wurden in der Zeile oder Spalte s zu viele signifikant unterschiedliche Wertepaare gefunden. Die Markierungen mit den höchsten Irrtumswahrscheinlichkeiten werden deshalb nicht ausgegeben.

617 Zu viele Zeilen oder Spalten zu vergleichen.

Hier sind die Werte aus über 78 Zeilen oder Spalten für Signifikanztests miteinander zu vergleichen. Sie können daher nicht mehr vollständig in der Form (a) ... (z") gekennzeichnet werden. Überstehende Zeilen oder Spalten werden mit (*) markiert.

618 Variablen- oder Merkmalstext fehlt.

Zu dem angegebenen XTAB Statement wird eine Variable ausgegeben, die keinen Variablentext oder keinen Merkmalstext besitzt. Stattdessen wird die Variablenbezeichnung *Cn*, *Nn* oder *Tn* selbst oder die Merkmalsnummer verwendet.

619 Ausgabewert passt nicht.

Ein Text oder ein Zahlenwert ist zu breit für den vorgesehenen Platz.
Anstelle des Zählerwerts wird das Zeichen * ausgegeben.
Zu lange Texte überschreiben andere Texte, Zahlen oder Linien.

620 Text in Position x **abgeschnitten.**

Eine Textzeile ist zu breit für den vorhandenen Platz. Der Text wird an Druckposition *x* abgeschnitten und diese Stelle mit den Zeichen *) gekennzeichnet.

621 Logo fehlerhaft oder nicht gefunden.

Durch das Statement PAGEP wurde die Druckausgabe für einen Seitendrucker aktiviert. Weiter wurde ein Logo in einem Text durch die Angabe ~n~ angefordert. Die dazugehörige Logo-Datei konnte nicht gefunden werden oder ist für das Programm nicht geeignet. Anstelle des Logos wird die Meldung *Fehler 36* gedruckt. Siehe bei dieser Fehlernummer für weitere Erläuterungen.

622 Erwartungswert in Zeile z **und Spalte** s **zu klein für korrekten Chi-Quadrat-Test.**

Der Erwartungswert ergibt sich aus:

Zellenwert * Zeilensumme * Spaltensumme

Er sollte für Chi-Quadrat-Tests nicht kleiner als 5 werden.

623 Klasse k aus PRINT*i* fehlt in ROW oder COL

Hinter PRINT*i* wird eine Prozentuierung PROWA ... PROWZ oder PCOLA ... PCOLZ angefordert. In ROW oder COL fehlt jedoch die entsprechende Klasse A{} ... {}.

624 Prozentuierungsbasis S*b* aus PRINT*i* liegt außerhalb der Tabelle

Hinter PRINT*i* wird eine Prozentuierung PROWS*b* oder PCOLS*b* angefordert.
Durch SUPPRESS-Angaben wurden so viele Zeilen oder Spalten aus der Tabelle entfernt, dass die Basiszeile oder -spalte *b* nicht mehr ausgegeben wird.

625 Eine PROSUM-Funktion konnte nicht ausgeführt werden

Folgende Ursachen sind möglich:

- Eine Summe aus SUM hat den Wert 0.
- Die Rundungsfaktoren sind zu groß.
- Die zu BASE angegebene Toleranzschwelle ist zu klein.
- Zu BASE oder SUM ist eine Zeile oder Spalte angegeben, die gleichzeitig in den Summanden vorkommt.

626 Text in Spalte s **abgeschnitten**

Die Tabelle hat zu viele oder zu breite Spalten und muss auf mehrere Seiten verteilt werden.
Für einen Text in Spalte *s* steht deshalb nicht genügend Platz zur Verfügung.

627 *n* **Fälle mit fehlenden Werten in Zeile** *a* **oder** *b***.**

In *n* Fällen enthält eine der Zeilen *a* oder *b* eine Angabe während sie in der anderen fehlt. Diese Meldung erscheint auch, wenn eine der Zeilen *a* oder *b* die Angabe MEAN(Cn) enthält und die Variable *Cn* Mehrfachnennungen besitzt. Solche Fälle werden bei der Ermittlung der Signifikanzen für MTTD nicht berücksichtigt.

628 *n* **Fälle mit fehlenden Werten in Spalte** *a* **oder** *b***.**

In *n* Fällen enthält eine der Spalten *a* oder *b* eine Angabe während sie in der anderen fehlt. Diese Meldung erscheint auch, wenn eine der Spalten *a* oder *b* die Angabe MEAN(Cn) enthält und die Variable *Cn* Mehrfachnennungen besitzt. Solche Fälle werden bei der Ermittlung der Signifikanzen für MTTD nicht berücksichtigt.

629 Gewünschte Schrift nicht gefunden.

Zu einer Schrift wurde die gewünschte Ausprägung *normal, kursiv, fett* oder *fett kursiv* nicht gefunden und durch eine andere ersetzt.

630 Zeichen wurden ersetzt

In der aktuellen Schrift nicht verfügbare Zeichen wurden durch @ ersetzt.

631 *n* **MTTD-Fehler in dieser Tabelle**

Zu dieser Tabelle wurden *n* MTTD-Datenfehler gefunden. Die entsprechenden Werte werden bei der Ermittlung der Signifikanzen nicht berücksichtigt. In *n* sind über alle statistischen Fälle alle fehlerhaften Wertepaare des MTTD-Vergleichs aufaddiert.

700 Falsche Satzlänge

Die Satzlänge SIZE passt nicht zur Datei FNAME.
Da hinter SIZE nicht EOL angegeben ist, muß die Dateilänge ein ganzes Vielfaches der Satzlänge sein.

701 CLEAN-Statement fehlt

Vor diesem ECLEAN-Statement existiert kein passenden CLEAN-Statement.

702 CLEAN auf falscher DO-Stufe

- Ein DO-Statement im Wirkungsbereich eines CLEAN-Statements ist vor dem nächsten CLEAN oder ECLEAN durch EDO zu beenden.
- Ein DO-Statement vor CLEAN-Statements ist erst hinter dem abschließenden ECLEAN mit EDO zu beenden.

703 ECLEAN-Statement fehlt

Das Ende von ADD-PROC, MOD-PROC, UPD-PROC oder OUTPUT-EXT ist erreicht.
Das letzte CLEAN-Statement ist noch nicht durch ECLEAN abgeschlossen.

704 CLEAN auf falscher IF-Stufe

- Ein IF-Statement im Wirkungsbereich eines CLEAN-Statements ist vor dem nächsten CLEAN oder ECLEAN durch EIF zu beenden.
- Ein IF-Statement vor CLEAN-Statements ist erst hinter dem abschließenden ECLEAN durch ELSE oder EIF zu beenden.
- Ein ELSE-Statement vor CLEAN-Statements ist erst hinter dem abschließenden ECLEAN durch ELSE oder EIF zu beenden.

705 Variable oder Textelement nicht definiert oder ungültig

Soll eine gültige Variable verändert werden, so muß diese vorher unter DEFS oder in der internen Datei INPUT-INT definiert sein.
Variablennamen dürfen bis zu 10 Zeichen lang sein, aus den Ziffern 0 ... 9, den Buchstaben a ... z und dem Zeichen _ bestehen. Umlaute ä, ö, ü sowie ß sind nicht möglich. Das erste Zeichen muss C, N oder T sein. Am Ende sind maximal neun Ziffern 0 ... 9 zulässig.

706 Falsche Wertezuweisung

Hier ist ein CLEAN-Statement wirksam. Dann sind nur Wertezuweisungen mit Variablen links vom Gleichheitszeichen möglich.

707 Ungültiges Zeichen

Für DCHAR, NCHAR und TCHAR sind nur die ersten 128 Zeichen des ANSI- und ASCII-Zeichensatzes möglich.

708 Ungültiges Zeichen

Die vor dem Fehlerstrich stehende Zeichenfolge zur Aufbereitung der Zählergebnisse enthält das im Statement PAGE oder PAGEP unter DCHAR angegebene Zeichen.

709 Falsche Label-Angabe zu einem U-Feld

Die Angaben I, V, VI und IV sind nur für das U-Format bei der Ausgabe von SPSS-Dateien möglich.

710 Fehlende Label-Angabe zu einem U-Feld

Hier wird eine der Angaben I, V, VI oder IV erwartet.

711 Datei *trans.~I~* nicht geöffnet

Diese Datei konnte im Arbeitsverzeichnis CNTWWORK nicht geöffnet werden.
Sie wird für OUTPUT-INT mit TRANSONLY benötigt.

712 Falscher Variablentyp

An dieser Stelle ist nur eine T-Variable möglich.

713 Ungültige Zeichenumsetzung

An dieser Stelle wird eine Zeichenumsetzung *'z' = wert* erwartet.
Für z sind nur die Zeichen A ... Z zulässig.
Die Angabe ist nur für C- oder N-Variable zulässig.
Pro Variable ist jeder Buchstabe z und jeder Wert nur einmal zulässig.

714 Unverträgliche Angaben zu HEAD oder FOOT

Die Angaben HEADL, HEADC, HEADR, FOOTL, FOOTC, FOOTR und ALTER vertragen sich in einem Verarbeitungslauf nicht mit den veralteten Angaben HEAD, FOOT, DATE, NODATE, TIME, NOTIME und NR. Nur NR=n mit einer Seitennummer n ist möglich.

715 Ungültige N = v ... b Angabe

Es liegt einer der folgenden Fehler vor:

- Die Von-Angabe v bei einer C-Variablen ist größer als die Anzahl der Merkmale dieser Variablen.
- Die Von-Angabe v ist ungültig. Für N- und T-Variable ist nur v=1 zulässig. Für C-Variable gilt v>0.
- Für N- oder T-Variable ist nur N=1 zulässig. Eine Bis-Angabe b ist nicht erlaubt.

716 AUTO hier nicht möglich

Es liegt einer der folgenden Fehler vor:

- Die unterste ID-Stufe ist alphanumerisch
- Die unterste ID-Stufe ist durch CD mit Prüfziffer definiert.

717 Effektstärken für MTTD, MTTX und MTTUX nicht möglich

t-Tests mit MTTD, MTTU und MTTUX erlauben keine Effektstärken *Es* hinter PRINT...=...SIG

718 Effektstärken mit mehreren Signifikanzwerten

Effektstärken *Es* hinter PRINT...=...SIG sind nur mit genau einem Signifikanzwert möglich.

719 Ungültige Effektstärke

Für die Effektstärken *Es* hinter PRINT...=...SIG sind nur Werte *s* zwischen 0.01 und 99.99 möglich.

720 EBLK-Statement fehlt

An dieser Stelle muß das letzte BLK-Statement beendet sein. Es fehlt aber noch ein EBLK.

721 DATE hier nicht möglich

Bei der Ausgabe in eine SPSS-Dateii mit DATE sind die Angaben MV, Z0 oder *s='valuelabel'* nicht zulässig.

722 Zu viele Spalten für OUTPUT-EXT mit XLSX

Die durch Excel vorgegebene Begrenzung auf 16 384 Spalten ist überschritten: Es liegt eine zu große Spaltennummer vor oder ein Datensatz enthält zu viele Felder in den Formaten AK, DK, MK oder UK.

723 OUTPUT-EXT mit CLEAN

CLEAN ist bei OUTPUT-EXT in den Formaten AUTO..., SPSS... und TRIPLES nicht möglich.

724 Klammer [ ] hinter ( )

Hinter der Angabe SORT von XTAB sind die Klammern [ ] vor den Klammern ( ) anzugeben.
Das gilt getrennt für ROWSi und COLSi, sofern beides verlangt wird.

725 Nachkommastellen fehlen

Um Rundungsdifferenzen zu vermeiden, müssen die Variablen NEW und NEWFACT mindestens eine Nachkommastelle besitzen.

726 Falscher Feldtyp für ID oder RC

Für OUTPUT-EXT mit AUTO ... und SEP verlangen ID und RC die Feldtypen AS und DS.

727 LIST hier unzulässig
Für OUTPUT-EXT mit SPSS oder TRIPLES ist diese Angabe nicht möglich.

728 Falsche Anzahl Zeilen oder Positionen
Hier sind numerische Werte zwischen 1 und 255 erforderlich.

729 Zu wenige Ausgabepositionen in POS
Nach Abzug der Breitenangaben MARGINS verbleiben weniger als 40 Ausgabepositionen auf den Seiten.

2000 Programmabbruch

Diese Meldung wird durch einen Programmfehler in CNTA verursacht und führt in jedem Fall zum vorzeitigen Abbruch der Verarbeitung. Der Fehler lässt sich vom Anwender nicht beheben.
Manchmal ist es möglich, den Abbruch durch kleinere Änderungen in den Statements oder Arbeitsabläufen zu umgehen. In jedem Fall sollte eine Korrektur des Programms angefordert werden.