SIEMENS

LEVEL II COBOL™

Sprachbeschreibung

SINIX

für Siemens PC

LEVEL II COBOL™ (SINIX)
Sprachbeschreibung

Ausgabe April 1985 (LEVEL II COBOL Version 2.0)

Bestell-Nr. U2052-J-Z95-1
Printed in the Federal Republic of Germany
9000 AG 7862. (11200)

Siemens Aktiengesellschaft

VORWORT

Dieses Handbuch beschreibt die Sprache LEVEL II COBOL zur Programmierung von Mikrocomputern. LEVEL II COBOL basiert auf dem ANSI COBOL Standard X3.23 (1974). Es beschreibt auch die zusätzlichen Eigenschaften von LEVEL II COBOL, die die Leistungsfähigkeit von Mikroprozessoren ganz ausnutzen.

Jede neue Ausgabe von LEVEL II COBOL ist durch einen zweistelligen Code gekennzeichnet in Form von:

xx.yy

xx = Versionsnummer
yy = Ausgabenummer innerhalb dieser Version.

Die beschriebene L/II COBOL-Version ist 2.0

Führende Nullen werden nicht geschrieben.

Anmerkung

Im folgenden Text dieses Handbuches wird der volle Produktname LEVEL II COBOL mit LII COBOL abgekürzt.

An wen wendet sich dieses Handbuch?
Welche Vorkenntnisse sind nötig?

Dieses Handbuch wendet sich an

- den Anfangsprogrammierer, der die Programmiersprache LII COBOL lernen will und nur wenige allgemeine Kenntnisse der Datenverarbeitung hat
- den erfahrenen Programmierer, der COBOL bereits von anderen Anlagen her kennt und damit über weitreichende COBOL-Kenntnisse verfügt.

Wie ist das Handbuch aufgebaut?

Das Kapitel 1 ist für den Anfangsprogrammierer gedacht.
Hier werden allgemeine Begriffe definiert, z.B.

— Schreibweise der Anweisungen

— LII COBOL-Wörter

— LII COBOL-Namen

— Syntaxübersicht
(mit farblicher Kennzeichnung der LII COBOL Erweiterungen zum ANSI Standard und den Teilen, die bei LII COBOL nur zur Dokumentation angegeben wurden; siehe 1.8).

Kapitel 2	erläutert die Teile (in COBOL DIVISIONs genannt) der COBOL-Programme und alle Anweisungen
Kapitel 3	beschreibt die Tabellenbearbeitung und
Kapitel 4-6	enthalten die Beschreibungen und die Anweisungen zu den Ein-/Ausgaben der drei Dateiformate sequentiell, relativ und indiziert.
Kapitel 7	benötigen Sie, wenn Sie Dateien sortieren oder mischen wollen.
Kapitel 8	beschreibt wie Sie ein Programm in Segmente unterteilen können.
Kapitel 9	zeigt, wie Sie mit anderen Programmen kommunizieren.

Kapitel 10	hier erfahren Sie alles über die COBOL-Bibliothek und in
Kapitel 11	wird die Testhilfe (Debug-Mode) beschrieben.
Kapitel 12	gibt zum Schluß allgemeine Hinweise zur Programmiertechnik und Programm-Dimensionierung.

Was wird nicht hier, sondern in der LII COBOL Bedienungsanleitung beschrieben?

Der Betrieb des COBOL-Übersetzers ist hier nicht enthalten, sondern im Handbuch COBOL-Bedienungsanleitung beschrieben. Dort erhalten Sie auch Hinweise, wie man COBOL in SINIX installiert und wie Sie mit Dateien in SINIX umgehen. Auch das Arbeiten am Bildschirm und die Menü-Schnittstelle werden erläutert.

Wie finden Sie sich in diesem Handbuch zurecht?

Die Kapitel sind mit einem Schnittregister versehen, damit Sie die einzelnen Kapitel schneller auffinden.
Zusätzlich steht rechts oben auf jeder Seite ein Hinweis, was in dem jeweiligen Kapitel behandelt wird.

Was können Sie tun, damit das Handbuch besser wird?

Viel! Ihre Kritik und Anregungen sind wichtig für uns.
Schreiben Sie uns! Teilen Sie uns bitte mit, an welchen Stellen das Handbuch Fehler hat, wo es unübersichtlich ist - oder auch, wenn Sie zufrieden sind! In jedem Fall herzlichen Dank!

Manualredaktion D ST PM2
München 83 Otto-Hahn-Ring 6

Der folgende Absatz wird üblicherweise vor COBOL-Sprachbeschreibungen aufgeführt. Wir tun das auch, und zwar in der englischen Originalfassung.

COBOL is an industry language and is not the property of any company or group of companies, or of any organization or group of organizations.

No warranty, expressed or implied, is made by any contributor or by the CODASYL Programming Language Committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or the committee, in connection herewith.

The authors and copyright holders of the copyrighted material used herein:

FLOW-MATIC (Trademark for Sperry Rand Corporation) Programming for the UNIVAC® I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation; IBM Commercial Translator Form No. F28-8013, copyrighted 1958 by IBM; FACT, DSI27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell.

have specifically authorized the use of this material in whole or in part, in the COBOL spezifications. Such authorization extends to the reproduction and the use of COBOL specifications in programming manuals or similar publications.

Inhalt

1 Einführung

1.1 Was ist LII COBOL?

COBOL (COmmon Business Oriented Language = allgemeine betriebswirtschftlich ausgerichtete Fachsprache) ist die am häufigsten gebrauchte Sprache für die Programmierung kommerzieller und administrativer Datenverarbeitung.

LII COBOL ist ein kompaktes, interaktives Standard-COBOL-Sprachsystem das für die Verwendung für Computer auf Mikroprozessorbasis und intelligenten Terminals ausgelegt ist.

Es basiert auf ANSI COBOL, wie es im "American National Standard Programming Language COBOL" (ANSI X3.23 1974) spezifiziert ist. Die folgenden Module sind im LEVEL II vollständig implementiert:

- Sprachenkern
- Tabellenbearbeitung
- Sequentielle Ein- und Ausgabe
- Relative Ein- und Ausgabe
- Indizierte Ein- und Ausgabe
- Sortieren/Mischen
- Segmentierung
- Bibliothek
- Programmkommunikation
- Testhilfe

Zusätzlich zur Implementierung des COBOL ANSI Standard enthält LII COBOL auch mehrere Spracherweiterungen, die speziell auf eine Mikrocomputerumgebung und für die Kompatibilität mit X 3.23n Großcomputeranlagen zugeschnitten sind. Damit ist es einem LII COBOL-Programm möglich, Bildschirmseiten für die Daten-Ein- bzw. -Ausgabe zu formatieren (ACCEPT und DISPLAY), Text-Dateien zu lesen und zu schreiben (READ/WRITE) und physikalische Dateinamen während des Ablaufs zu definieren.

Wenn Sie ein bereits bestehendes COBOL-Programm **mit Spracherweiterungen** übernehmen und mit LII COBOL ablaufen lassen wollen, müssen Sie prüfen, ob diese von LII COBOL unterstützt werden. Für diese Prüfung gibt es den Abschnitt 1.10.

Wollen Sie bei der Übersetzung ANSI COBOL-Eigenschaften oder LII COBOL-Erweiterungen kennzeichnen, können Sie dies mit einer 'Kompilierzeit-Flag'-Anweisung tun. Nähere Hinweise dazu in der LII COBOL-Bedienungsanleitung.

Der LII COBOL-Übersetzer ist dafür ausgelegt, Programme auf einem 64K Computer zu entwickeln. Der Übersetzer unterstützt sequentielle, relative und indizierte Dateien sowie den Dialogverkehr über ACCEPT und DISPLAY.

LII COBOL ist Teil einer Familie von Werkzeugen für die dialogorientierte Entwicklung von Anwendungsprogrammen:

* FORMS-2
  ermöglicht Ihnen, Bildschirmformate anhand eines Bildschirm-"Moduls" zu definieren und die Datenbeschreibung für ein LII COBOL-Programm zu erzeugen. Dies wird in der FORMS-2-Bedienungsanleitung beschrieben.

* ANIMATOR
  erweckt ein Programm auf dem Bildschirm zum Leben, indem es den Quellcode während der Ausführungszeit anzeigt. Dabei wird der Cursor (= Schreibmarke) von COBOL Quellanweisung zu Quellanweisung bewegt. ANIMATOR ist ein komplettes, symbolisches Dialog-Testhilfe-Werkzeug, damit können Sie

  - Haltepunkte setzen
  - Überprüfen und
  - Daten ändern
  - den Programmablauf ändern

LII COBOL-Programme können Sie mit Hilfe eines Texteditors, wie z.B. CED, erstellen. Der Übersetzer übersetzt die Programme, und das Laufzeitsystem wird mit dem übersetzten Programm verbunden und bildet dann ein ablauffähiges Anwenderprogramm. Während der Übersetzung wird durch den Übersetzer ein Protokollausdruck mit eventuellen Fehlermeldungen geliefert (siehe auch Kapitel 1 der LII COBOL Bedienungsanleitung).
Für Testzwecke stehen Dialog-Testhilfe-Werkzeuge zur Verfügung.

1.2 Schreibweise der COBOL-Anweisungen

Im gesamten vorliegenden Handbuch wird die folgende Notation (Schreibweise) zur Beschreibung des Aufbaus der COBOL-Anweisungen verwendet:

1. Alle Wörter, die in Großbuchstaben geschrieben und unterstrichen sind, müssen an der Stelle, wo sie im Format auftreten, angegeben werden. Wird die Funktion verwendet und die unterstrichenen Wörter fehlen oder sind falsch geschrieben, wird während der Übersetzung eine Fehlermeldung ausgegeben. Im COBOL-Quellprogramm ist selbstverständlich nicht zu unterstreichen.

   Beispiel

   PROCEDURE DIVISION.

2. Alle Wörter, die in Großbuchstaben geschrieben, jedoch nicht unterstrichen sind, werden lediglich zur Verbesserung der Lesbarkeit verwendet.

   Beispiel

   IF bedingung THEN ...

3. Alle Wörter, die in Kleinbuchstaben geschrieben sind, stehen stellvertretend für Namen, die vom Programmierer gewählt werden müssen.

   Beispiel

   literal-1

4. Wenn Text in geschweiften { } Klammern eingeschlossen ist, muß unter den untereinander stehenden Alternativen eine Wahl getroffen werden.
   Steht nur eine Angabe in diesen geschweiften Klammern, dient dies nur der Zusammenfassung für ein nachfolgendes Wiederholungssymbol.

   Beispiel

   ```
   {ALL     }
   {LEADING }
   {FIRST   }
   ```

   oder

   {datenname-1} ...

5. Wenn Text in eckige Klammern [] eingeschlossen ist, so zeigt dies an, daß der Text optional ist, d.h., er kann nach Bedarf angegeben oder ausgelassen werden.

   Beispiel

   ADD bezeichner-1 TO bezeichner-m [ROUNDED]

6. Wird Text in durchgestrichene eckige Klammern ⟊ ⟊ gesetzt, so weist dies darauf hin, daß dieser Text unbedingt erforderlich ist, wenn der ANSI-Schalter gesetzt ist (vgl. Kapitel 2), andernfalls jedoch weggelassen werden kann.

7. Wiederholungssymbol ...

   Ein im Format gezeigtes Wiederholungssymbol zeigt an, daß die unmittelbar vorausgehende Einheit nach einmaliger Angabe beliebig oft wiederholt werden kann, aber nicht muß. Eine wiederholbare Einheit ist entweder ein einziges Wort oder mehrere Wörter, die durch eckige oder geschweifte Klammern zusammengefaßt sind. Im letzten Fall folgt das Wiederholungssymbol unmittelbar auf eine schließende Klammer; die dazugehörige öffnende Klammer bestimmt den Anfang der Einheit, die wiederholt werden kann.

8. Die Zahlen eins (1) bis neun (9) werden im Text ausgeschrieben, z. B. fünf.

   Zahlen ab zehn (10) aufwärts werden im Text als Zahlen geschrieben, z.B. 12.

   Der Ausdruck "nur zur Dokumentation" im Text dieses Handbuches bedeutet, daß die dazugehörige Codierung syntaktisch vom Compiler akzeptiert, jedoch bei der Erzeugung des Ausführungsprogrammes ignoriert wird.

9. Die Sprachelemente sind unterteilt in:

   Funktion

   Enthält eine kurze allgemeine Beschreibung der einzelnen Sprachelemente.

   Format

   Gibt die vorgeschriebene Kombination von Zeichenfolgen und Trennsymbolen für eine Klausel, eine Anweisung oder einen Paragraphen an.

   Syntaxregeln

   Beschreiben die notwendigen Anforderungen und Einschränkungen für die Anwendung des jeweiligen Sprachelements.

   Allgemeine Regeln

   Geben allgemeine Hinweise zu Abhängigkeiten im Umfeld des beschriebenen Sprachelements.

1.3 Begriffsbestimmungen

1.3.1 Allgemeines

Die Begriffe in diesem Kapitel werden entsprechend der Bedeutung definiert, wie sie in diesem Handbuch LII COBOL verwendet werden. Sie haben bei anderen Sprachen eventuell nicht die gleiche Bedeutung. Diese Definitionen dienen sowohl der Einführung, als auch zum Nachschlagen gewünschter Begriffe. Sie sind als Übersicht gedacht. Aus diesem Grunde werden in den meisten Fällen diese Definitionen kurz gehalten. Sie enthalten auch keine ins Detail gehenden syntaktischen Regeln.

1.3.2 Definitionen

Abgekürzte zusammengesetzte Vergleichsbedingung:
Abbreviated Combined Relation Condition
Das bedeutet eine zusammengesetzte Bedingung, die eine Auslassung eines gemeinsamen Subjektes oder eines gemeinsamen Subjektes und eines gemeinsamen Vergleichsoperators in einer aufeinanderfolgenden Folge von Vergleichsbedingungen beinhaltet.

Absteigender Schlüssel:
Descending Key
Ein Schlüssel, nach dessen Werten Daten entsprechend den Regeln für den Vergleich von Datenfeldern absteigend geordnet werden, beginnend mit dem höchsten Wert des Schlüssels.

Aktueller Datensatz:
Current Record
Der Datensatz, der in dem mit der Datei verknüpften Datensatzbereich verfügbar ist.

Aktueller Satzzeiger:
Current Record Pointer
Ein Zeiger, der für die Anwahl des nächsten Datensatzes verwendet wird.

Alphabetisches Zeichen:
Alphabetic Character
Ein Zeichen aus der folgenden Buchstabenmenge:

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z und das Leerzeichen. Ebenso a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, die in die entsprechenden Großbuchstaben umgewandelt werden.

Alphabet-Name:
Alphabetname
Ein vom Programmierer definiertes Wort im SPECIAL-NAMES-Paragraph der ENVIRONMENT DIVISION, das einem bestimmten Zeichenvorrat und/oder einer Sortierfolge einen Namen zuordnet.

Alphanumerisches Datenfeld:
Nonnumeric Item
Ein Datenfeld, dessen Beschreibung es ermöglicht, daß dessen Inhalt aus irgendeiner Zeichenkombination aus dem Zeichenvorrat des Computers besteht.

Alphanumerisches Literal:
Nonnumeric Literal
Eine in Anführungszeichen gesetzte Zeichenfolge. Die Folge kann beliebige Zeichen aus dem Zeichenvorrat des Computers enthalten. Um ein einzelnes Anführungszeichen innerhalb eines alphanumerischen Literals darstellen zu können, müssen zwei Anführungszeichen hintereinander verwendet werden.

Alphanumerisches Zeichen:
Alphanumeric Character
Ein beliebiges Zeichen aus dem Zeichenvorrat des Computers.

Angabe:
Phrase
Eine Angabe ist eine geordnete Folge von einem oder mehreren aufeinanderfolgenden COBOL Zeichenfolgen, die einen Teil einer LII COBOL-Prozeduranweisung oder einer COBOL-Klausel bilden.

Angenommener Dezimalpunkt:
Assumed Decimal Point
Eine Dezimalpunktposition, die allerdings nicht ein tatsächliches Zeichen im Datenfeld voraussetzt. Der angenommene Dezimalpunkt hat eine logische Bedeutung, keine physische Darstellung.

Animator:
Ein COBOL-orientiertes Testhilfewerkzeug zur Verwendung mit LEVEL II COBOL Produkten.

Anweisung zur Steuerung des Übersetzers:
Compiler-Directing Statement
Eine Anweisung, die mit einem Verb zur Steuerung des Übersetzers beginnt. Sie veranlaßt den Übersetzer, während des Übersetzungsvorganges eine bestimmte Operation durchzuführen.

Anweisung:
Statement
Eine syntaktisch zulässige Kombination von Worten und Symbolen, die in der PROCEDURE DIVISION geschrieben sind, beginnend mit einem Verb.

Arithmetischer Ausdruck:
Arithmetic Expression
Ein arithmetischer Ausdruck kann ein Bezeichner oder ein numerisches Datenelement, ein numerisches Literal bzw. Bezeichner und Literale, die durch arithmetische Operatoren getrennt sind, oder zwei arithmetische Ausdrücke, die durch einen arithmetischen Operator getrennt sind oder ein in Klammern gesetzter arithmetischer Ausdruck sein.

Arithmetischer Operator:
Arithmetic Operator
Ein einzelnes Zeichen oder eine feste Kombination aus zwei Zeichen aus dem folgenden Zeichenvorrat:

Zeichen	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
**	Potenzierung

Arithmetisches Vorzeichen:
Operational Sign
Ein algebraisches Zeichen, das mit einem numerischen Datenfeld oder einem numerischen Literal verknüpft ist und anzeigt, ob dessen Wert positiv oder negativ ist.

AT END Bedingung:
At End Condition
Eine Bedingung, die bei einer der drei Umstände auftritt:

1. während der Ausführung einer READ-Anweisung für eine Datei, auf die sequentiell zugegriffen wird und für die kein nächster logischer Datensatz vorhanden ist;
2. während der Ausführung einer RETURN-Anweisung, wenn kein nächster logischer Datensatz für die zugehörige Sortier- oder Mischdatei vorhanden ist;
3. während der Ausführung einer SEARCH-Anweisung, wenn die Suchoperation beendet wird, ohne daß eine Bedingung erfüllt wurde, die nach der WHEN-Angabe steht.

Aufgerufenes Programm:
Called Program
Ein Programm, das in der CALL-Anweisung benannt und zur Ausführungszeit mit dem aufrufenden Programm verbunden wird, um eine Ausführungseinheit herzustellen.

Aufrufendes Programm:
Calling Program
Ein Programm, das eine CALL-Anweisung zu einem anderen Programm ausführt.

Aufsteigender Sortierschlüssel:
Ascending Key
Ein Schlüssel, nach dessen Werten Daten aufsteigend entsprechend den Regeln für den Vergleich von Datenfeldern geordnet werden, wobei mit dem niedrigsten Wert des Schlüssels begonnen wird.

Ausführungseinheit:
Run Unit
Ein oder mehrere Zwischencodeprogramme, die zur Ausführungszeit als eine Einheit funktionieren.

Ausgabemodus:
Output Mode
Der Zustand einer Datei nach Ausführung einer OPEN-Anweisung und vor der Ausführung einer CLOSE-Anweisung für diese Datei, wobei die OUTPUT- oder EXTEND-Angabe für diese Datei angegeben wurde.

Ausgabeprozedur:
Output Procedure
Eine Reihe von Anweisungen, an die die Steuerung während der Ausführung einer SORT-Anweisung gegeben wird nachdem die Sortierfunktion abgeschlossen ist, oder während der Ausführung einer MERGE-Anweisung, nachdem die Mischfunktion den nächsten Datensatz in der Reihenfolge bestimmt hat, die durch die MERGE-Anweisung vorgegeben wurde.

Ausgabe-Datei:
Output File
Eine Datei, die entweder im Ausgabemodus oder Erweiterungsmodus eröffnet wurde.

Bedingungsanweisung:
Conditional Statement
Eine Bedingungsanweisung gibt an, daß der Wahrheitswert einer Bedingung zu ermitteln ist und daß die nachfolgende Operation des Ausführungszeitprogramms von diesem Wahrheitswert abhängt.

Bedingungsausdruck:
Conditional Expression
Eine einfache in einer IF-, PERFORM- oder SEARCH-Anweisung angegebene Bedingung (vgl. einfache Bedingung und komplexe Bedingung).

Bedingungsnamen-Bedingung:
Condition-Name Condition
Aufgrund der Bedingung wird eine Bedingungsvariable geprüft, um zu entscheiden, ob ihr Wert gleich einem der Werte ist, die zu einem bestimmten Bedingungsnamen gehören.

Bedingungsname:
Condition Name
Ein vom Programmierer definiertes Wort, das dem Zustand eines herstellerdefinierten Schalters oder Gerätes oder einem bestimmten Wert aus dem einer Bedingungsvariablen zugehörigen Wertebereich zugeordnet ist.

Bedingungsvariable:
Conditional Variable
Ein Datenfeld, mit ein oder mehreren Werten, die einem Bedingungsnamen zugeordnet sind.

Bedingung:
Condition

Der Zustand eines Programms zur Ausführungszeit, für den ein Wahrheitswert bestimmt werden kann. Wenn der Begriff "Bedingung" (bedingung-1, bedingung-2,...) in den Sprachenspezifikationen in oder im Zusammenhang mit der "Bedingung" (bedingung-1, bedingung-2,...) eines allgemeinen Formats erscheint, so handelt es sich um einen Bedingungsausdruck, der entweder aus einer einfachen Bedingung besteht, die in Klammer stehen kann, oder aus einer zusammengesetzten Bedingung, die aus einer syntaktisch korrekten Kombination von einfachen Bedingungen, logischen Operatoren und Klammern besteht, wobei für diesen Bedingungsausdruck ein Wahrheitswert bestimmt werden kann.

Begrenzer:
Delimiter

Ein Zeichen (oder eine Zeichenfolge), das das Ende einer Zeichenfolge bestimmt oder das eine Zeichenfolge von einer nachfolgenden Zeichenfolge trennt. Ein Begrenzer ist nicht Bestandteil der Zeichenfolge.

Bestimmungsort:
Destination

Die symbolische Bezeichnung des Empfängers einer Übertragung von einer Warteschlange.

Bezeichner:
Identifier

Ein Datenname, der auch subskribiert oder indiziert bzw. gekennzeichnet sein kann.

Bezugsschlüssel:
Key of Reference

Der Schlüssel, der gerade verwendet wird, um auf Datensätze in einer indizierten Datei zuzugreifen.

Bibliotheksname:
Library-Name

Name eines Dateiverzeichnisses, das eine LII COBOL-Bibliotheksquelldatei enthält, die vom Übersetzer für die Übersetzung eines vorgegebenen Quellprogramms verwendet wird.

Bibliothekstext:
Library-Text

Eine alphanumerische Zeichenfolge und/oder Trennzeichen in einer COBOL-Bibliothek.

Bindewort:
Connective
Ein reserviertes Wort, das verwendet wird, um

1. einen Datennamen, Paragraphnamen, Bedingungsnamen oder Textnamen mit dem zugehörigen Kennzeichner zu verknüpfen;
2. zwei oder mehrere hintereinander geschriebene Operanden zu verbinden;
3. Bedingungen (logische Bindewörter) zu bilden (vgl. logischer Operator).

Block:
Eine physische Dateneinheit (das ist die durch jeweils eine physische Ein/Ausgabe transportierte Datenmenge), die normalerweise aus einem oder mehreren logischen Sätzen besteht. Bei Massenspeicher-Dateien kann ein Block einen Teil eines logischen Satzes enthalten. Die Größe eines Blocks hat keinen direkten Bezug zur Größe der Datei, in der der Block enthalten ist, oder zur Größe des bzw. der logischen Sätze, die entweder innerhalb des Blocks fortgesetzt werden oder die den Block überlappen. Der Begriff ist gleichbedeutend mit dem Begriff "physischer Satz".

CD-Name:
Ein vom Programmierer definiertes Wort das einen MCS-Schnittstellenbereich benennt, der in einer Kommunikationsbeschreibung in der COMMUNICATION SECTION der DATA DIVISION spezifiziert ist. Der Name kann angegeben werden, wird aber von LII COBOL nur syntaktisch geprüft und beim Ablauf nicht verarbeitet.

COBOL Wort:
COBOL word
Siehe "Wort"

COMMUNICATION SECTION:
Das Kapitel der DATA DIVISION, das die Schnittstellenbereiche zwischen dem MCS und dem Programm beschreibt. Es besteht aus einer oder mehreren CD-Beschreibungen. Die COMMUNICATION SECTION wird bei LII COBOL nur syntaktisch geprüft. Sie wird als Funktion nicht unterstützt.

Computer-Name:
Ein System-Name, der denjenigen Computer identifiziert, auf dem das Programm übersetzt wird oder abläuft.

Configuration Section:
Ein Kapitel der ENVIRONMENT DIVISION, in dem die allgemeinen Angaben des Quell- und Ausführungscomputers beschrieben sind.

CRT:
Der symbolische Ausdruck für die Datensichstation mit der Tastatur.

Cursor:
Die Schreibmarke auf einem Bildschirm, die die Zeile und die Zeichenposition markiert, auf die sich die Eingabe-/Ausgabe-Steuerung gerade bezieht.

Datei:
File
Eine Ansammlung von Datensätzen.

Dateibeschreibung:
File Description Entry
Ein Eintrag in der FILE SECTION der DATA DIVISION, der aus der Stufenbezeichnung FD, gefolgt von einem Dateinamen und einer Reihe von Dateiklauseln, sofern erforderlich, besteht.

Dateibezeichner:
File Idenfifier
Bezeichnet ein Datenfeld, das den Dateinamen im System (siehe externes Dateiname-literal), mit dem die LII COBOL Ein- oder Ausgabedatei verknüpft wird, enthält. Die Verknüpfung geschieht in der SELECT-Anweisung mit ASSIGN TO.

Dateiklausel:
File Clause
Eine Klausel, die als Teil eines der folgenden DATA DIVISION Einträge erscheint:

Dateibeschreibung (FD)

Sortier-Misch-Dateibeschreibung (SD)

Kommunikationsbeschreibung (CD) (wird als Funktion nicht unterstützt).

Dateiname:
File-Name
Ein vom Programmierer definiertes Wort, das eine Datei benennt, die in einer Dateierklärung oder in einer Sortier-Misch-Dateierklärung in der FILE SECTION der DATA DIVISION angegeben ist.

Dateiorganisation:
File Organization
Der logische Dateiaufbau, der zum Zeitpunkt der Erstellung einer Datei festgelegt wurde.

Datenelement:
Elementary Item
Ein Datenfeld, das so beschrieben ist, daß es nicht weiter logisch unterteilt werden kann.

Datenerklärung:
Data Description Entry
Ein Eintrag in der DATA DIVISION, bestehend aus einer Stufennummer, gefolgt von einem Datennamen, und von einer Reihe von Datenklauseln, sofern erforderlich.

Datenfeld:
Data Item
Ein Zeichen oder eine Reihe benachbarter Zeichen (ausgenommen Literale), die als eine Dateneinheit durch das LII COBOL-Programm definiert sind.

Datengruppe:
Group Item
Eine Menge benannter aufeinanderfolgender Datenelemente oder Datengruppen.

Datenklausel:
Data Clause
Eine Klausel, die in einer Datenerklärung in der DATA DIVISION erscheint und Information liefert, die ein bestimmtes Attribut eines Datenfeldes angibt.

Datenname:
Data-Name
Ein vom Programmierer definiertes Wort, das ein in einer Datenerklärung in der DATA DIVISION angegebenes Datenfeld benennt. Wenn datenname in den allgemeinen Formaten verwendet wird, so stellt dies ein Wort dar, das weder subskribiert, indiziert noch gekennzeichnet werden kann, sofern dies nicht ausdrücklich in dem Format erlaubt ist.

Datensatzbereich:
Record Area
Ein Speicherbereich, der für die Verarbeitung eines in einer Datensatzerklärung in der FILE SECTION definierten Datensatzes zugewiesen wird.

Datensatzbeschreibung:
Record Description Entry
Alle mit einem bestimmten Datensatz verknüpften Datenerklärungen.

Datensatzschlüssel:
Record Key
Ein Schlüssel, entweder der primäre Datensatzschlüssel oder ein alternativer Datensatzschlüssel, dessen Inhalt einen Datensatz in einer indizierten Datei identifiziert.

Datensatz-Name:
Record-Name
Ein vom Programmierer definiertes Wort, das einen Datensatz benennt, der in einer Datensatzerklärung in der DATA DIVISION definiert ist.

Datensatz:
Record
Siehe "Logischer Datensatz"

Datenverzeichnis:
Data Dictionary
Eine Datei mit dem Verzeichnis der vom Programmierer definierten Namen, die durch den Übersetzer erstellt wird und die die Anzahl von Bytes für jeden Eintrag enthält.

Direkter Zugriff:
Random Access
Eine Zugriffsmethode, in der der im Programm spezifizierte Wert eines Schlüsseldatenfeldes den logischen Datensatz identifiziert. Der Wert ist Teil der relativen oder indizierten Datei.

DIVISION-Überschrift:
Division Header
Eine Kombination von Wörtern, gefolgt von einem Punkt und einem Leerzeichen, durch die der Anfang einer DIVISION angezeigt wird. Es gibt folgende DIVISION-Überschriften:

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION [ USING datenname-1

                        [ datenname-2 ] ... ] .
```

DIVISION:
Eine Reihe von SECTIONs und Paragraphen (0 oder mehr), die entsprechend bestimmter Regeln gebildet bzw. zusammengesetzt werden, wird DIVISION genannt. Es gibt 4 DIVISIONs in einem LII COBOL-Programm:

- IDENTIFICATION DIVISION
- ENVIRONMENT DIVISION
- DATA DIVISION
- PROCEDURE DIVISION

Druckaufbereitungszeichen:
Editing Character

Ein einzelnes Zeichen oder eine feste Kombination aus zwei Zeichen aus dem gleichen Vorrat:

Zeichen	Bedeutung
B	Leerzeichen
0	Null
+	Plus
-	Minus
CR	Kredit
DB	Debit
Z	Nullunterdrückung
*	Summenschutz
$	Währungszeichen
,	Komma
.	Punkt (Dezimalpunkt)
/	Schrägstrich

Dynamischer Zugriff:
Dynamic Access
Eine Zugriffsmethode, mit dieser können:

- logische Sätze von einer Datei auf nichtsequentielle Art gelesen bzw. darauf geschrieben werden.

- die Sätze von einer Datei auf sequentielle Weise nach einer OPEN-Anweisung gelesen werden.

Einfache Bedingung:
Simple Condition
Eine einfache Bedingung aus der folgenden Menge:

Vergleichsbedingung
Klassenbedingung
Schalterstatusbedingung
Bedingungsname-Bedingung
Vorzeichenbedingung
(einfache Bedingung)

Eingabemodus:
Input Mode
Der Zustand einer Datei nach Ausführung einer OPEN-Anweisung und vor Ausführung einer CLOSE-Anweisung, wobei die INPUT-Angabe für diese Datei angegeben war.

Eingabe-Datei:
Input File
Eine Datei, die einem Quellprogramm Daten zur Verarbeitung liefert. Sie muß dazu im Eingabemodus geöffnet sein.

Eingabe-PROCEDURE:
Input Procedure
Eine Reihe von Anweisungen, die zu Beginn eines Sortiervorganges die zu sortierenden Sätze an die Sortierdatei übergibt.

Eingabe-/Ausgabe-Datei:
Input-Output File
Eine Datei, die im I-O Modus eröffnet worden ist.

Eingabe-/Ausgabe-SECTION:
Input-Output Section
Die SECTION in der ENVIRONMENT DIVISION, die die Dateien und die externen Geräte benennt, die von einem Programm verwendet werden. In dieser SECTION werden auch die Informationen bereitgestellt, die für Übertragung und die Bearbeitung von Daten während des Ablaufes des Objektprogramms notwendig sind.

Einstelliger Operator:
Unary Operator
Ein Pluszeichen (+) oder ein Minuszeichen (-), das vor einer Variablen oder einer linken Klammer in einem arithmetischen Ausdruck steht, wodurch der Ausdruck mit +1 bzw. -1 multipliziert wird.

Eintragssubjekt:
Subject of Entry
Ein Operand oder ein reserviertes Wort, das unmittelbar nach der Stufenbezeichnung oder der Stufennummer in einem Eintrag der DATA DIVISION erscheint.

Eintrag:
Entry
Eine Reihe aufeinanderfolgender Klauseln, die durch einen Punkt (.) beendet wird und in der IDENTIFICATION DIVISION, ENVIRONMENT DIVISION oder in der DATA DIVISION eines LII COBOL-Quellprogramms beschrieben wird.

Ende der PROCEDURE DIVISION:
End of Procedure Division
Die physische Position in einem LII COBOL-Quellprogramm, nach der keine weiteren Prozeduren erscheinen.

ENVIRONMENT-Klausel:
Environment Clause
Eine Klausel, die als Teil eines ENVIRONMENT DIVISION-Eintrags erscheint.

Eröffnungsmodus:
Open Mode
Der Zustand einer Datei nach Ausführung einer OPEN-Anweisung und vor Ausführung einer CLOSE-Anweisung für diese Datei. Der spezielle Eröffnungsmodus wird in der OPEN-Anweisung als INPUT, OUTPUT, I-O oder EXTEND angegeben.

Erweiterungsmodus:
Extend Mode
Ist EXTEND angegeben, befindet sich die Datei zwischen der OPEN- und der CLOSE-Anweisung im Erweiterungsmodus.

Externes-Bibliotheksname-Literal:
External-Library-Name-Literal
Ein alphanumerisches Literal, das den Namen eines Dateiverzeichnisses im SINIX-System angibt (siehe Bibliotheksname).

Externes-Dateiname-Literal:
External-File-Name-Literal
Ein alphanumerisches Literal, das den Dateinamen im System, mit dem die entsprechende LII COBOL Ein- oder Ausgabedatei verknüpft wird, angibt:

":CI:"	Standardeingabedatei
":CO:"	Standardausgabedatei
":CE:"	Standardfehlerdatei
":name:"	Ausgabe über SPOOL-System
"name"	Dateiname in SINIX inclusive Pfadnamen, sofern erforderlich

Die Verknüpfung geschieht in der SELECT-Anweisung mit ASSIGN TO.

Figurative Konstante:
Figurative Constant
Ein vom Übersetzer erzeugter Wert; diesen Wert definieren bestimmte reservierte Wörter.

FILE SECTION:
Die SECTION in der DATA DIVISION, die Dateierklärungen zusammen mit deren zugehörigen Datensatzerklärungen enthält.

FILE-CONTROL:
Der Name eines Paragraphen in der ENVIRONMENT DIVISION, in dem die Dateien für das Quellprogramm vereinbart werden.

Format:
Eine bestimmte Anordnung von Zeichenfolgen und Trennsymbolen einer Anweisung oder Klausel.

FORMS-2-Programm:
Ein Bildschirm-Formatierungsprogramm, das automatisch LII COBOL CRT Eingabe-/Ausgabe-Codierung von der aktuellen Bildschirmanzeige erzeugt.

Gekennzeichneter Datenname:
Qualified Data-Name
Ein Bezeichner, der aus einem Datennamen, gefolgt von einem oder mehreren Verknüpfern (entweder OF oder IN), gefolgt von einem Datennamen (Kennzeichner), besteht.

Herstellerwort:
Implementor-name
Ein System-Name, der auf ein ganz bestimmtes Merkmal Bezug nimmt, wird durch die Art der Anlage bestimmt.

Hintereinanderliegende Datenfelder:
Contiguous Items
Datenfelder, die durch aufeinanderfolgende Einträge in der DATA DIVISION beschrieben sind und die eine bestimmte hierarchische Beziehung zueinander haben.

Index-Datenfeld:
Index Data Item
Ein Datenfeld, worin der mit einem Index-Namen verknüpfte Wert gespeichert wird.

Index-Name:
Index-name
Ein vom Programmierer definiertes Wort, das einen Index benennt der mit einer bestimmten Tabelle verknüpft ist.

Index:
Eine Speicherposition oder ein Register, enthält die Bezeichnung eines bestimmten Elementes in einer Tabelle.

Indikatorbereich:
Indicator Area
Die siebte Stelle eines LII COBOL-Quelldatensatzes, die die Verwendung des Datensatzes anzeigt.

Indizierte Datei:
Indexed File
Eine Datei mit indizierter Organisation.

Indizierte Organisation:
Indexed Organization
Der logische Dateiaufbau, in dem jeder Datensatz durch den Wert von einem oder mehreren Schlüsseln in diesem Datensatz bestimmt wird.

Indizierter Datenname:
Indexed Data-Name
Ein Bezeichner, der aus einem Datennamen, gefolgt von einem oder mehreren in Klammern gesetzten Index-Namen besteht.

Integer bzw. Ganze Zahl:
Ein numerisches Literal oder ein numerisches Datenfeld, das keine Zeichenpositionen rechts des angenommenen Dezimalpunktes enthält. Wenn "Ganzzahl" in den allgemeinen Formaten erscheint, darf Ganzzahl kein numerisches Datenfeld sein und darf keine Vorzeichen besitzen bzw. Null sein, sofern dies nicht ausdrücklich durch die Regeln für das entsprechende Format erlaubt ist.

Interpunktionszeichen:
Punctuation Character
Ein Zeichen aus dem folgenden Vorrat:

Zeichen	Bedeutung
,	Komma
;	Strichpunkt
.	Punkt
"	Anführungszeichen
(	linke Klammer
)	rechte Klammer
␣	Leerzeichen
=	Gleichheitszeichen

I-O Modus:
I-O Mode
Der Zustand einer Datei nach Ausführung einer OPEN-Anweisung und vor der Ausführung einer CLOSE-Anweisung, wobei die I-O-Angabe für diese Datei angegeben war.

I-O-CONTROL:
Der Name eines Paragraphen in der ENVIRONMENT DIVISION, hier wird angegeben:

- Objektprogrammanforderungen für bestimmte Ein-/Ausgaben
- Wiederanlaufpunkte
- Belegung des gleichen Speicherbereiches durch verschiedene Dateien
- Mehrfachdateispeicherung auf einem einzelnen Ein-/Ausgabe-Gerät.

Kennzeichner:
Qualifier

1. Ein Datenname, der zusammen mit anderen Datennamen einer niedrigeren Stufe in der gleichen Hierarchie diese Datennamen eindeutig bezeichnet.
2. Ein SECTION-Name, der zusammen mit einem Paragraph-Namen, der in dieser SECTION angegeben ist, diesen eindeutig bezeichnet.
3. Ein Bibliotheksname, der zusammen mit einem Text-Namen, der mit dieser Bibliothek verknüpft ist, diesen eindeutig bezeichnet.

Klassenbedingung:
Class Condition
Die Aussage, für die ein Wahrheitswert bestimmt werden kann, ob der Inhalt eines Datenfeldes vollständig alphabetisch oder vollständig numerisch ist.

Klausel:
Clause
Eine Klausel ist eine geordnete Menge aufeinanderfolgender LII COBOL Zeichenfolgen, sie legt das Attribut eines Eintrags fest.

Kommentareintrag:
Comment Entry
Ein Eintrag in der IDENTIFICATION DIVISION, der aus einer beliebigen Zeichenkombination aus dem Zeichenvorrat des Computers bestehen kann.

Kommentarzeile:
Comment Line
Eine Quellprogrammzeile, dargestellt durch einen Stern (*) im Indikatorbereich der Zeile (das ist die Spalte 7 in der Programmzeile) und beliebigen Zeichen aus dem Zeichenvorrat des Computers im Bereich A und Bereich B dieser Zeile. Die Kommentarzeile dient nur zur Dokumentation in einem Programm.
Eine besondere Form der Kommentarzeile ist der Schrägstrich (/) im Indikatorbereich der Zeile. Er verursacht einen Seitenvorschub beim Ausdrucken des Quellprogramms vor Ausdruck des Kommentars, der wieder im Bereich A und Bereich B dieser Zeile stehen kann.

Kommunikationsbeschreibung:
Communication Description Entry
Ein Eintrag in der COMMUNICATION SECTION der DATA DIVISION, der aus der Stufenbezeichnung CD, gefolgt von einem CD-Namen und einer Reihe von Klauseln, sofern erforderlich, besteht. Dadurch wird die Schnittstelle zwischen dem Nachrichtenkontrollsystem (Message Control System = MCS) und dem COBOL-Programm beschrieben.
Wird von LII COBOL nicht unterstützt.

Kommunikationsgerät:
Communication Device
Ein Mechanismus (Hardware oder Hardware/Software), der Daten an eine Warteschlange sendet und/oder Daten von einer Warteschlange empfängt. Dieser Mechanismus kann ein Computer oder ein Peripheriegerät sein. Durch ein oder mehrere Programme, die Kommunikationbeschreibung enthalten und im gleichen Computer ablaufen, werden ein oder mehrere dieser Mechanismen definiert. Der Mechanismus wird bei LII COBOL nicht ünterstützt.

Kompilierzeit:
Compile Time
Die Zeit, in der ein LII COBOL-Quellprogramm durch den Übersetzer in ein LII COBOL-Zwischencodeprogramm übersetzt wird.

Komplexe Bedingung:
Complex Condition
Eine Bedingung, in der ein oder mehrere logische Operatoren eine oder mehrere Bedingungen miteinander verknüpfen. (siehe verneinte einfache Bedingung, zusammengesetzte Bedingung, verneinte zusammengesetzte Bedingung)

Laufzeitsystem:
Run-Time System
Die Software, die den durch den LII COBOL-Übersetzer erzeugten Zwischencode interpretiert und ermöglicht, daß dieser durch Bereitstellung von Schnittstellen zum Betriebssystem und dem Bildschirm ausgeführt werden kann.

Laufzeit:
Run-Time
Die Zeit, während der der durch den Übersetzer erzeugte Zwischencode durch das Ausführungszeitsystem ausgeführt wird.

LINKAGE SECTION:
Die SECTION in der DATA DIVISION des aufgerufenen Programms, die die Datenfelder beschreibt, die vom aufrufenden Programm übergeben werden. Auf diese Datenfelder kann sowohl das aufrufende als auch das aufgerufene Programm zugreifen.

Linksbündiges Ende:
High Order End
Das linksbündige Zeichen in einer Zeichenfolge.

Literal:
Eine beliebige Zeichenfolge.

Logischer Operator:
Logic Operator
Eines der reservierten Wörter AND, OR oder NOT. Bei der Bildung einer Bedingung kann AND oder OR als logisches Bindewort verwendet werden oder beide zusammen. NOT kann für die logische Verneinung verwendet werden.

Logischer Satz:
Logic Record
Das umfassendste Datenfeld eines Datensatzes. Die Stufennummer für ein solches Datenfeld ist 01.

Maschineneigene Sortierfolge:
Native Collating Sequence
Die herstellerdefinierte Sortierfolge. Die Verknüpfung mit dem Computer steht im OBJECT-COMPUTER-Paragraph.

Maschineneigener Zeichenvorrat:
Native Character Set
Der herstellerdefinierte Zeichenvorrat. Die Verknüpfung mit dem Computer steht im OBJECT-COMPUTER-Paragraphen.

MCS (= message control system = Nachrichtensteuerungssystem):
Siehe "Nachrichtensteuerungssystem"

Merkname:
Mnemonic Name
Ein vom Programmierer definiertes Wort, das in der ENVIRONMENT DIVISION mit einem bestimmten Herstellerwort verknüpft ist.

Mischdatei:
Merge File
Eine Ansammlung von Datensätzen, die durch eine MERGE-Anweisung gemischt werden sollen. Die Mischdatei wird durch die Mischfunktion erstellt und kann nur von dieser genutzt werden.

Nachrichtenindikatoren:
Message Indicators
EGI (Gruppenende-Indikator), EMI (Nachrichtenende-Indikator) und ESI (Segmentende-Indikator) sind Anzeigen, die dazu dienen, dem MCS mitzuteilen, daß eine bestimmte Bedingung besteht (Gruppenende, Nachrichtenende, Segmentende).
Für LII COBOL nicht relevant.

Nachrichtensegment:
Message Segment
Daten, die einen logischen Teil einer Nachricht bilden und die normalerweise mit einem Segmentende-Indikator verknüpft sind (vgl. Nachrichtenindikatoren).

Nachrichtensteuerungssystem (message control system = MCS):
Message Control System
Ein Kommunikationssteuerungssystem, das die Verarbeitung von Nachrichten unterstützt.

Nachrichtenzählung:
Message Count
Die Zählung der Anzahl der Nachrichten, die in der benannten Nachrichtenwarteschlange vorhanden sind.

Nachricht:
Message
Daten, die mit einem Nachrichtenende-Indikator oder einem Gruppenende-Indikator verknüpft sind (vgl. Nachrichtenindikatoren).

Nächste ausführbare Anweisung:
Next Executable Statement
Die nächste Anweisung, an die die Steuerung übertragen wird nachdem die Ausführung der aktuellen Anweisung vollständig abgeschlossen ist.

Nächster ausführbarer Satz:
Next Executable Sentence
Der nächste Satz, an den die Steuerung übertragen wird, nachdem die Ausführung der aktuellen Anweisung vollständig abgeschlossen ist.

Nächster Datensatz:
Next Record
Der Datensatz, der logisch nach dem aktuellen Datensatz einer Datei steht.

Nichtbenachbarte Datenfelder:
Noncontiguous Items
Elementare Datenfelder in der WORKING-STORAGE SECTION und der LINKAGE SECTION, die keine hierarchische Beziehung zu anderen Datenfeldern haben.

Numeral:
Literal, das nur Ziffern enthält (siehe Numerisches Literal).

Numerisches Datenfeld:
Numeric Item
Ein Datenfeld, dessen Zeichen aus den Ziffern 0 bis 9 bestehen. Wenn der Wert mit einem Vorzeichen versehen ist, kann das Datenfeld ebenfalls ein '+', '-' oder eine andere Darstellung eines Rechenvorzeichens enthalten.

Numerisches Literal:
Numeric Literal
Ein Literal, das aus einem oder mehreren numerischen Zeichen besteht und entweder einen Dezimalpunkt oder ein arithmetisches Vorzeichen oder auch beides enthalten kann. Der Dezimalpunkt darf nicht das rechtsbündige Zeichen sein; das arithmetische Vorzeichen muß, wenn vorhanden, das linksbündige Zeichen sein.

Numerisches Zeichen:
Numeric Character
Ein Zeichen, das zur folgenden Ziffernreihe gehört: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

OBJECT-COMPUTER:
Der Name eines Paragraphen in der ENVIRONMENT DIVISION, in dem die Umgebung des Computers beschrieben ist, auf dem das Objektprogramm abläuft.

Objekt-Programm:
Object Program
Das Ergebnis der Übersetzung eines COBOL-Quellprogramms durch einen COBOL-Übersetzer. Wo keine Verwechslungsgefahr besteht, kann auch statt Objekt-Programm einfach Programm gesagt werden.

Operand:
Die allgemeine Definition von Operand im Sinne dieses Handbuches ist "die Komponente, die bearbeitet wird". Jedes kleingeschriebene Wort (oder Wörter), das in einer Anweisung oder einem Format erscheint, ist ein Operand.

Optionales Wort:
Optional Word
Ein reserviertes Wort, das in einem bestimmten Format nur enthalten ist, um die Lesbarkeit der Sprache zu verbessern. Ob dieses Wort geschrieben wird, entscheidet der Programmierer, für den Ablauf des Programms ist es unerheblich.

Originaldiskette:
Issue Disk
Die Diskette, auf der die LII COBOL Software an die Anwender geliefert wird.

Paragraphenüberschrift:
Paragraph Header
Ein reserviertes Wort, gefolgt von einem Punkt und einem Leerzeichen, das den Anfang eines Paragraphen in der IDENTIFICATION DIVISION und ENVIRONMENT DIVISION anzeigt. Zulässige Paragraphenüberschriften sind:

In der IDENTIFICATION DIVISION:

```
PROGRAM-ID.
AUTHOR.
INSTALLATION.
DATE-WRITTEN.
DATE-COMPILED.
SECURITY.
```

In der ENVIRONMENT DIVISION:

```
SOURCE-COMPUTER.
OBJECT-COMPUTER.
SPECIAL-NAMES.
FILE-CONTROL.
I-O-CONTROL.
```

Paragraph-Name:
Ein vom Programmierer definiertes Wort, das am Anfang eines Paragraphen in der PROCEDURE DIVISION steht, von einem Punkt und einem Leerzeichen gefolgt wird und den Paragraphen identifiziert.

Paragraph:
Ein Paragraph-Name in der PROCEDURE DIVISION, gefolgt von einem Punkt und einem Leerzeichen und bei Bedarf von einem oder mehreren Sätzen. In der IDENTIFICATION DIVISION und der ENVIRONMENT DIVISION ist dies eine Paragraphenüberschrift, gefolgt von keinem, einem oder mehreren Einträgen.

Physischer Satz:
Physical Record
Siehe "Block"

Primärer Datensatzschlüssel:
Prime Record Key
Ein Schlüssel, dessen Inhalt einen Datensatz in einer indizierten Datei eindeutig identifiziert.

Programmierer Wort:
User-defined Word
Ein COBOL-Wort, das vom Anwender (Programmierer) angegeben werden muß, um das Format einer Klausel oder einer Anweisung zu erfüllen.

Programm-Name:
Program-Name
Ein vom Programmierer definiertes Wort, das ein COBOL-Quellprogramm identifiziert.

Programm-Satz:
Sentence
Eine Folge von einer oder mehreren Anweisungen, wobei die letzte durch einen Punkt, gefolgt von einem Leerzeichen, beendet wird.

Prozedur-Name:
Procedure-Name
Ein vom Programmierer definiertes Wort, das dazu verwendet wird, einen Paragraphen oder eine SECTION in der PROCEDURE DIVISION zu benennen. Es besteht aus einem Paragraph-Namen oder aus einem SECTION-Namen.

Prozedur:
Procedure
Ein Paragraph oder eine Gruppe logisch aufeinanderfolgender Paragraphen oder eine SECTION oder eine Gruppe logisch aufeinanderfolgender SECTIONs in der PROCEDURE DIVISION.

Pseudo-Textbegrenzer:
Pseudo-Text Delimiter
Zwei aufeinanderfolgende Gleichheitszeichen (=), die zur Begrenzung eines Pseudotextes verwendet werden.

Pseudo-Text:
Eine Zeichenfolge und/oder Trennzeichen, die durch Pseudo-Textbegrenzer begrenzt werden. Die Textbegrenzer selbst sind nicht Teil des Pseudo-Textes.

Quelle:
Source
Die symbolische Definition des Senders einer Übertragung an eine Warteschlange.

Quellprogramm:
Source Program
In diesem Handbuch auch oft Programm genannt. Eine syntaktisch korrekte Reihe von COBOL-Anweisungen, die mit einer IDENTIFICATION DIVISION beginnt (beginnen kann) und mit dem Ende der PROCEDURE DIVISION endet.

Rechtsbündiges Ende:
Low Order End
Das rechtsbündige Zeichen einer Zeichenfolge.

Referenzformat:
Reference-Format
Das Format, in dem die Beschreibung von COBOL Quellprogrammen erfolgen muß.

Relative Datei:
Relative File
Eine Datei mit einer relativen Organisation.

Relative Organisation:
Relative Organization
Der permanente logische Dateiaufbau, in dem jeder Datensatz durch einen ganzzahligen Wert größer als Null eindeutig identifiziert wird. Die logische Ordnungsstelle des Datensatzes in der Datei wird angegeben.

Relativer Schlüssel:
Relative Key
Ein Schlüssel, dessen Inhalt einen logischen Datensatz in einer relativen Datei identifiziert.

Reserviertes Wort:
Reserved Word
Ein COBOL-Wort, das in der Wortliste angegeben ist, die bei LII COBOL-Quellprogrammen verwendet werden kann. Es darf jedoch in den Programmen nicht als vom Programmierer definiertes Wort oder als System-Name erscheinen.

Routine-Name:
Ein vom Programmierer definiertes Wort, das eine Prozedur identifiziert, die nicht in LII COBOL geschrieben ist.

Schalterstatusbedingung:
Switch-Status Condition
Die Aussage, für die ein Wahrheitswert dahingehend bestimmt werden kann, ob ein herstellerdefinierter Schalter sich im EIN- oder AUS-Zustand befindet.

Schlüsselwort:
Key Word
Ein reserviertes Wort, das vorhanden sein muß, wenn das Format, in dem das Wort erscheint, in einem Quellprogramm verwendet wird.

Schlüssel:
Key
Ein Datenfeld, dessen Inhalt die Position eines Datensatzes in einer Datei identifiziert, oder eine Anzahl von Datenfeldern, die zur Bestimmung der Reihenfolge von Daten dient.

SECTION-Name:
Ein vom Programmierer definiertes Wort, das eine SECTION in der PROCEDURE DIVISION benennt.

SECTION-Überschrift:
Section Header
Eine Kombination aus Wörtern, gefolgt von einem Punkt und einem Leerzeichen, die den Anfang einer SECTION in den ENVIRONMENT, DATA und PROCEDURE DIVISIONs anzeigt. In der ENVIRONMENT DIVISION und DATA DIVISION besteht eine SECTION-Überschrift aus reservierten Wörtern, gefolgt von einem Punkt und einem Leerzeichen. Zulässige SECTION-Überschriften sind:

In der ENVIRONMENT DIVISION:

```
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
```

In der DATA DIVISION:

```
FILE SECTION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
```

In der PROCEDURE DIVISION besteht eine SECTION-Überschrift aus einem SECTION-Namen, gefolgt von dem reservierten Wort SECTION, gefolgt von einer Segmentnummer (optional), gefolgt von einem Punkt und einem Leerzeichen.

SECTION:
Keine, ein oder mehrere Paragraphen oder Einträge, SECTION-Rumpf genannt, wobei vor dem ersten Paragraphen oder Eintrag eine SECTION-Überschrift steht. Jede SECTION besteht aus der SECTION-Überschrift und dem zugehörigen SECTION-Rumpf.

Segmentnummer:
Segment-Number
Ein vom Programmierer definiertes Wort, das SECTIONs in der PROCEDURE DIVISION für die Segmentierung klassifiziert. Segmentnummern können nur die Ziffern 0 - 9 enthalten. Eine Segmentnummer wird als eine Zahl, bestehend aus entweder einer oder zwei Ziffern ausgedrückt, es wird nur die Syntax überprüft.

Sequentielle Datei:
Sequential File
Eine Datei mit sequentieller Organisation.

Sequentielle Organisation:
Sequential Organization
Der permanente logische Dateiaufbau, die Datensätze werden in fester Reihenfolge hintereinander in die Datei geschrieben.

Sequentieller Zugriff:
Sequential Access
Ein Zugriffsmodus, in dem logische Datensätze in eine Datei geschrieben oder dort gelesen werden. Die Sätze stehen hintereinander in der Datei und können nur in dieser Reihenfolge verarbeitet werden.

Sonderregister:
Special Register
Vom Übersetzer erzeugte Speicherbereiche, die für die Speicherung von solchen Informationen ausgelegt sind, die in Verbindung mit der Verwendung von bestimmten COBOL-Merkmalen erstellt wurden.

Sonderzeichenwort:
Special-Character Word
Ein reserviertes Wort, stellt einen arithmetischen Operator oder ein Vergleichszeichen dar.

Sonderzeichen:
Special Sign

Ein Zeichen aus dem folgenden Vorrat:

Zeichen	Bedeutung
+	Pluszeichen
-	Minuszeichen
*	Stern
/	Schrägstrich
=	Gleichheitszeichen
$	Währungszeichen
,	Komma (Dezimalpunkt)
;	Strichpunkt
.	Punkt (Dezimalpunkt)
"	Anführungszeichen
(	linke Klammer
)	rechte Klammer
>	größer-als-Symbol
<	kleiner-als-Symbol

Sortierdatei:
Sort File

Datensätze, die durch eine SORT-Anweisung sortiert werden sollen. Die Sortierdatei wird durch die Sortierfunktion erstellt und kann nur von dieser genutzt werden.

Sortierfolge:
Collating Sequence

Eine Folge von Zeichen, die in einer Datenverarbeitungsanlage beim Sortieren, Mischen und Vergleichen zur Bestimmung der Reihenfolge zugelassen sind.

SORT-MERGE Dateibeschreibung:
Sort-Merge File Description Entry

Ein Eintrag in der FILE SECTION der DATA DIVISION, der aus der Stufenbezeichnung SD, gefolgt von einem Dateinamen und, sofern erforderlich einer Reihe von Dateiklauseln besteht.

SOURCE-COMPUTER:

Der Name eines Paragraphen in der ENVIRONMENT DIVISION, in dem die Umgebung des Computers beschrieben ist, in dem das Quellprogramm übersetzt wird.

Spalte:
Column
Eine Zeichenposition in einer Druckzeile. Die Spalten werden von 1 an hochgezählt, wobei an der linksbündigen Zeichenposition der Druckzeile begonnen wird und durch Erhöhen von jeweils 1 bis zur rechtsbündige Zeichenposition der Druckzeile durchgezählt wird.

SPECIAL-NAMES:
Der Name eines Paragraphen in der ENVIRONMENT DIVISION, in dem Herstellerwörter den vom Programmierer angegebenen Merknamen zugeordnet werden.

Sprachenname:
Language Name
Ein System-Name, der eine bestimmte Programmiersprache angibt.

Standard-Datenformat:
Standard Data Format
In einer Liste mit Standardformat können bestimmte Daten der DVA - des Unterspeichers oder eines externen Gerätes - für den Leser übersichtlicher dargestellt werden als wenn sie nicht aufbereitet wären.

Stufenbezeichnung:
Level Indicator
Zwei alphabetische Zeichen, die einen bestimmten Dateityp oder eine hierarchische Position bezeichnen.

Stufennummer:
Level Number
Eine vom Programmierer definierte Zahl, die entweder die Position eines Datenfeldes in der hierarchischen Struktur eines logischen Datensatzes oder die speziellen Eigenschaften einer Datenerklärung anzeigt. Eine Stufennummer wird als eine Zahl ausgedrückt, die aus einer oder zwei Ziffern besteht.
Stufennummern im Bereich 1 bis 49 zeigen die Position eines Datenfeldes in der hierarchischen Struktur eines logischen Datensatzes an. Stufennummern im Bereich 1 bis 9 können entweder als einzelne Ziffern oder als 0 und eine bestimmte Ziffer geschrieben werden. Die Stufennummern 66, 77 und 88 bezeichnen spezielle Eigenschaften eines Dateneintrags.

Subskribierter Datenname:
Subscripted Data-Name
Ein Bezeichner, der aus einem Datennamen, gefolgt von einem oder mehreren in Klammern gesetzten Subskripten besteht.

Subskript:
Subscript
Eine Ganzzahl, deren Wert ein bestimmtes Element in einer Tabelle identifiziert.

Symbole in PICTURE-Zeichenketten:
Symbol Function
Die Verwendung bestimmter Zeichen in der PICTURE-Klausel zur Darstellung von Datentypen.

Syntax:
Die Reihenfolge, in der Elemente zusammengesetzt werden müssen, um ein Programm bilden zu können.

System-Name:
Ein COBOL-Wort, das dazu verwendet wird, mit der Rechnerumgebung zu kommunizieren.

Tabellenelement:
Table Element
Ein Datenfeld in einer Tabelle.

Tabelle:
Table
Eine Reihe logisch aufeinanderfolgender Datenfelder, die in der DATA DIVISION durch die OCCURS-Klausel definiert sind.

Tatsächlicher Dezimalpunkt:
Actual Decimal Point
Die physikalische Darstellung der Dezimalpunktposition in einem Datenfeld, wobei entweder das Dezimalpunktzeichen . (Punkt) oder , (Komma) verwendet wird.

Teilwarteschlange:
Subqueue
Eine logisch hierarchische Unterteilung einer Warteschlange.

Terminal:
Der Sender einer Übertragung an eine Warteschlange oder der Empfänger einer Übertragung von einer Warteschlange.

Testhilfezeile:
Debugging Line
Eine Testhilfezeile ist jede Zeile, die ein 'D' im Indikatorbereich der Zeile hat.

Testhilfe-SECTION:
Debugging Section
Eine Testhilfe-SECTION ist ein Kapitel, das eine USE FOR DEBUGGING-Anweisung enthält.

Text-Name:
Text Name
Name einer Datei in Großbuchstaben, die Bibliothekstext enthält.

Text-Wort:
Text-Word
Eine Zeichenfolge oder ein Trennzeichen, ausgenommen Leerzeichen, in einer COBOL-Bibliothek oder in Pseudo-Text.

Trennzeichen:
Separator
Ein Interpunktionszeichen zur Begrenzung von Zeichenfolgen.

Unbedingte Anweisung:
Imperative Statement
Eine Anweisung, die mit einem unbedingten Verb beginnt und die eine unbedingte Aktion bezeichnet. Eine unbedingte Anweisung kann aus einer Folge von mehreren unbedingten Anweisungen bestehen.

Unterprogramm:
Subprogram
Siehe "Aufgerufenes Programm"

Unzulässige Schlüsselbedingung:
Invalid Key Condition
Eine Bedingung zur Programmausführungszeit, die dadurch verursacht wird, daß ein bestimmter Wert eines Schlüssels, der mit einer indizierten oder relativen Datei verknüpft ist, als unzulässig festgestellt wird.

Variable:
Ein Datenfeld, dessen Wert durch Ausführung eines Objektprogramms geändert werden kann. Eine in einem arithmetischen Ausdruck verwendete Variable muß ein numerisches Datenelement sein.

Verb:
Ein Wort, das eine Aktion ausdrückt, die von einem COBOL-Übersetzer oder einem Ausführungszeitprogramm durchgeführt werden muß.

Vereinbarungen:
Declaratives
Eine oder mehrere SECTIONs mit speziellem Zweck, die am Anfang der PROCEDURE DIVISION geschrieben werden, wobei vor der ersten SECTION das Schlüsselwort DECLARATIVES und nach der letzten die Schlüsselworte END DECLARATIVES stehen. Eine Vereinbarung besteht aus einer SECTION-Überschrift, gefolgt von einer USE-Folge zur Steuerung des Übersetzers, gefolgt von einer Reihe zugehöriger Paragraphen (0 oder mehr).

Vereinbarungssatz:
Declarative-Sentence
Eine Übersetzeranweisung, die aus einer einzelnen USE-Anweisung besteht und mit dem Trennzeichen Punkt (.) abgeschlossen wird.

Vergleichsbedingung:
Relation Condition
Die Aussage, für die ein Wahrheitswert bestimmt werden kann. Sie bewirkt, daß zwei Operanden miteinander verglichen werden. Jeder dieser Operanden kann ein Bezeichner, Literal oder arithmetischer Ausdruck sein.

Vergleichsoperator:
Relational Operator
Ein reserviertes Wort, ein Vergleichszeichen, eine Gruppe aufeinanderfolgender reservierter Wörter oder eine Gruppe aufeinanderfolgender reservierter Wörter und Vergleichszeichen, die für den Aufbau einer Vergleichsbedingung verwendet werden. Zulässige Operatoren und ihre Bedeutung sind:

Vergleichsoperator	Bedeutung
IS [NOT] GREATER THAN IS [NOT] >	größer als oder [nicht größer als]
IS [NOT] LESS THAN IS [NOT] <	kleiner als oder [nicht kleiner als]
IS [NOT] EQUAL TO IS [NOT] =	gleich oder [nicht gleich]

Vergleichszeichen:
Relation Character
Ein Zeichen aus dem folgenden Vorrat:

Zeichen	Bedeutung
<	kleiner als
<	kleiner als
=	ist gleich

Vergleich:
Relation
Siehe "Vergleichsoperator"

Verneinte einfache Bedingung:
Negated simple Condition
Der logische Operator 'NOT', unmittelbar gefolgt von einer einfachen Bedingung.

Verneinte zusammengesetzte Bedingung:
Negated Combined Condition
Der logische Operator 'NOT', unmittelbar gefolgt von einer zusammengesetzten Bedingung in Klammern.

Vorzeichenbedingung:
Sign Condition
Die Aussage, für die ein Wahrheitswert dahingehend bestimmt werden kann, daß der algebraische Wert eines Datenfeldes oder eines arithmetischen Ausdruckes entweder kleiner als, größer als oder gleich 0 ist.

Wahrheitswert:
Truth Value
Das Ergebnis der Auswertung einer Bedingung, dargestellt durch einen der zwei Werte:

true / wahr
false / falsch

Währungssymbol:
Currency Symbol
Das durch die CURRENCY SIGN-Klausel im SPECIAL-NAMES-Paragraph definierte Zeichen. Wenn keine CURRENCY SIGN-Klausel in einem LII COBOL-Quellprogramm vorhanden ist, ist das Währungssymbol mit dem Währungszeichen identisch.

Währungszeichen:
Currency Sign
Das Zeichen '$' (Dollarzeichen) aus dem LII COBOL Zeichenvorrat.

Warteschlangenname:
Queue Name
Ein symbolischer Name, der dem MCS den logischen Weg anzeigt, auf dem auf eine Nachricht oder einen Teil einer vollständigen Nachricht in einer Warteschlange zugegriffen werden kann.

Warteschlange:
Queue
Eine logische Ansammlung von Nachrichten, die zur Übertragung oder Verarbeitung anstehen.

WORKING-STORAGE SECTION:
Die SECTION in der DATA DIVISION, die Arbeitsspeicher-Datenfelder beschreibt und entweder aus nichtbenachbarten Datenfeldern, aus Arbeitsspeicher-Datensätzen oder aus beidem besteht.

Wort:
Word
Eine Zeichenfolge von maximal 30 Zeichen, die ein vom Programmierer definiertes Wort, einen System-Namen oder ein reserviertes Wort bildet.

Zähler:
Counter
Ein Datenfeld, das für die Speicherung von Zahlen oder Zahlendarstellungen verwendet wird. Sie können durch den Wert einer anderen Zahl erhöht oder verringert werden oder auf einen beliebigen positiven oder negativen Wert (zurück-) gesetzt werden.

Zeichen:
Character
Die grundlegende unteilbare Einheit der Sprache.

Zeichenfolge:
Character String
Eine Anzahl aufeinanderfolgender Zeichen, die ein LII COBOL-Wort, ein Literal, eine PICTURE-Zeichenfolge oder eine Kommentareintragung bilden.

Zeichenposition:
Character Position
Eine Zeichenposition ist der physische Speicherplatz, der benötigt wird, um ein einzelnes Standard-Datenformat-Zeichen zu speichern, das als USAGE IS DISPLAY beschrieben ist.

Zeichenvorrat (LII COBOL):
Character Set (LII COBOL)

Der vollständige LII COBOL Zeichenvorrat besteht aus den nachfolgend aufgelisteten Zeichen:

Zeichen	Bedeutung
0...9	Ziffern
A...Z	Großbuchstaben
a...z	Kleinbuchstaben
+	Pluszeichen
-	Minuszeichen
*	Stern
/	Schrägstrich
=	Gleichheitszeichen
$	Währungszeichen
,	Komma
;	Strichpunkt
.	Punkt (Dezimalpunkt)
"	Anführungszeichen
(	linke Klammer
)	rechte Klammer
>	größer-als-Symbol
<	kleiner-als-Symbol

Zeilensequentielle Dateiorganisation:
Line Sequential File Organization

Eine sequentielle Datei, die Datensätze mit variabler Länge enthält, die durch das Zeichen X'0A' getrennt sind.

Ziffernposition:
Digit Position

Eine Ziffernposition ist die Menge physischen Speicherplatzes, die benötigt wird, um eine einzelne Ziffer zu speichern. Diese Menge hängt ab von der Verwendung des Datenfeldes, das die Ziffernposition beschreibt.

Zugriffsmodus:

Die Art, in der Datensätze einer Datei verarbeitet werden.

Zusammengesetzte Bedingung:
Combined Condition
Eine Bedingung, die durch Verbindung zwischen zwei oder mehreren Bedingungen mit den logischen Operatoren 'AND' oder 'OR' erzeugt wird.

Zwischencode:
Intermediate Code
Der Code, der vom LII COBOL Übersetzer aus dem eingegebenen Quellcode erzeugt wird. Diesen Zwischencode lädt das Laufzeitsystem für den Ablauf.

77-Stufenbeschreibung:
77 Level-Discription-Entry
Eine Datenerklärung, die ein nichtbenachbartes Datenfeld auf der Stufe 77 beschreibt.

1.4 Zeichenvorrat und Sortierfolge

ASCII	HEX	COBOL	ASCII	HEX	COBOL	ASCII	HEX	COBOL
NUL	00	x	/	2F			5E	x
SOH	01	x	0	30		–	5F	x
STX	02	x	1	31		«	60	x
ETX	03	x	2	32		a	61	
EOT	04	x	3	33		b	62	
ENQ	05	x	4	34		c	63	
ACK	06	x	5	35		d	64	
BEL	07	x	6	36		e	65	
BS	08	x	7	37		f	66	
HT	09	x	8	38		g	67	
LF	0A	x	9	39		h	68	
VT	0B	x	:	3A	x	i	69	
FF	0C	x	;	3B		j	6A	
CR	0D	x	<	3C		k	6B	
SO	0E	x	=	3D		l	6C	
SI	0F	x	>	3E		m	6D	
DLE	10	x	?	3F	x	n	6E	
DC1	11	x	@	40	x	o	6F	
DC2	12	x	A	41		p	70	
DC3	13	x	B	42		q	71	
DC4	14	x	C	43		r	72	
NAK	15	x	D	44		s	73	
SYN	16	x	E	45		t	74	
ETB	17	x	F	46		u	75	
CAN	18	x	G	47		v	76	
EM	19	x	H	48		w	77	
SUB	1A	x	I	49		x	78	
ESC	1B	x	J	4A		y	79	
FS	1C	x	K	4B		z	7A	
GS	1D	x	L	4C		{	7B	
RS	1E	x	M	4D		|	7C	
US	1F	x	N	4E		}	7D	
Blank	20		O	4F		~	7E	
!	21	x	P	50		DEL	7F	
"	22		Q	51				
#	23	x	R	52				
$	24		S	53				
%	25	x	T	54				
&	26	x	U	55				
'	27	x	V	56				
(	28		W	57				
)	29		X	58				
*	2A		Y	59				
+	2B		Z	5A				
,	2C		[	5B	x			
–	2D		\	5C	x			
.	2E		]	5D	x			

Ein x in Spalte COBOL bedeutet, daß das entsprechende Zeichen nicht zum Zeichenvorrat von LII COBOL gehört.

1.5 Liste der reservierten Wörter und Interpunktionszeichen

Hier sind alle reservierten Wörter für COBOL und LII COBOL aufgelistet. Die nur bei LII COBOL reservierten Wörter stehen im Kapitel 1.10.

ACCEPT
ACCESS
ADD
ADVANCING
AFTER
ALL
ALPHABITIC
ALSO
ALTER
ALTERNATE
AND
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR

BEFORE
BLANK
BLOCK
BOTTOM
BY

CALL
CANCEL
CD
CHARACTER
CHARACTERS
CLOCK-UNITS
CLOSE
COBOL
CODE-SET
COLLATING
COMMA
COMMAND-LINE
COMMIT
COMMUNICATION
COMP
COMP-3
COMPUTATIONAL
COMPUTATIONAL-3
COMPUTE
CONFIGURATION
CONSOLE
CONTAINS
COPY
CORR
CORRESPONDING
COUNT
CRT
CRT-UNDER
CURRENCY
CURSOR

DATA
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LINE
DEBUG-NAME
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DECIMAL-POINT
DECLARATIVES

DELETE
DELIMITED
DELIMITER
DEPENDING
DESCENDING
DESTINATION
DISABLE
DISPLAY
DIVIDE
DIVISION
DOWN
DUPLICATES
DYNAMIC

EGI
ELSE
EMI
ENABLE
END
END-OF-PAGE
ENTER
ENVIRONMENT
EOP
EQUAL
ERROR
ESI
EVERY
EXCEPTION
EXCESS-3
EXTEND

FD
FILE
FILE-CONTROL
FILLER
FIRST
FOR
FORMFEED
FROM

GIVING
GO
GREATER

HIGH-VALUE
HIGH-VALUES

I-O
I-O-CONTROL
ID
IDENTIFICATION
IF
IN
INDEX
INDEXED
INITIAL
INPUT
INPUT-OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
IS

JUST
JUSTIFIED

KEY
KEPT

LABEL
LEADING
LEFT
LENGTH
LESS
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINES
LINKAGE
LOCK
LOW-VALUE
LOW-VALUES

MEMORY
MERGE
MESSAGE
MODE
MODULES
MOVE
MULTIPLE
MULTIPLY

```
                ROLLBACK
                ROUNDED
                RUN
NATIVE          SAME
NEGATIVE        SD
NEXT            SEARCH
NO              SECTION             TRAILING
NOT             SECURITY            TYPE
NUMERIC         SEGMENT
                SEGMENT-LIMIT       UNIT
OBJECT-COMPUTER SELECT              UNSTRING
OCCURS          SEND                UNTIL
OF              SENTENCE            UP
OFF             SEPARATE            UPON
OMITTED         SEQUENCE            USAGE
ON              SEQUENTIAL          USE
OPEN            SET                 USING
OPTIONAL        SIGN
OR              SIZE                VALUE
ORGANIZATION    SORT                VALUES
OUTPUT          SORT-MERGE          VARYING
OVERFLOW        SOURCE
                SOURCE-COMPUTER     WHEN
PAGE            SPACE               WITH
PAGETHROW       SPACES              WORDS
PERFORM         SPECIAL-NAMES       WORKING-STORAGE
PIC             STANDARD            WRITE
PICTURE         STANDARD-1
POINTER         START               ZERO
POSITION        STATUS              ZEROES
POSITIVE        STOP                ZEROS
PROCEDURE       STRING
PROCEDURES      SUB-QUEUE-1         . (Punkt)
PROCEED         SUB-QUEUE-2         (
PROGRAM         SUB-QUEUE-3         -
PROGRAM-ID      SUBTRACT            *
                SWITCH              **
QUEUE           SYMBOLIC            )
QUOTE           SYNC                ;
QUOTES          SYNCHRONIZED        +
                SYSIN               /
RANDOM          SYSOUT              ,
RD                                  <
READ            TAB                 =
RECEIVE         TABLE               >
RECORD          TALLYING            "
RECORDS         TAPE
REDEFINES       TERMINAL
REEL            TEXT
REFERENCES      THAN
RELATIVE        THEN
RELEASE         THROUGH
REMAINDER       THRU
REMOVAL         TIME
RENAMES         TIMES
REPLACING       TO
RERUN           TOP
RESERVE
RETURN
REVERSED
REWIND
REWRITE
RIGHT
```

Reservierte Wörter Listprogramm

Der Listprogramm-Modul ist nicht in LEVEL II COBOL enthalten, da es für das High Level Zertifikat der GSA nicht erforderlich ist. Da die für den Listprogramm-Modul verwendeten reservierten Wörter in anderen COBOL-Implemetierungen reserviert sein könnten, werden sie nachfolgend aufgeführt. Durch einen Vergleich der reservierten Wörter können Sie eventuelle Fehler vermeiden.

CF
CH
CODE
COLUMN
CONTROL/S
DE
DETAIL
FINAL
FOOTING
GENERATE
GROUP
HEADING
INDICATE
INITIATE
LAST
LINE-COUNTER
NUMBER
PAGE-COUNTER
PF
PH
PLUS
PRINTING
REPORT/S
REPORTING
RESET
RF
RH
SUM
SUPPRESS
TERMINATE

1.6 Programmstruktur

Ein LII COBOL-Programm besteht aus vier DIVISIONs:

1. IDENTIFICATION DIVISION
   Erkennungsteil eines Programms
2. ENVIRONMENT DIVISON
   Beschreibung der Computerausstattung, die für die Übersetzung und Ausführung des Programms verwendet wird.
3. DATA DIVISION
   Beschreibung der Daten, die verarbeitet werden.
4. PROCEDURE DIVISION
   Eine Menge von Prozeduren, die die Operationen definieren, die mit den Daten ausgeführt werden sollen.

Bei LII COBOL müssen die ersten drei DIVISIONS nicht geschrieben werden. Sie können auch dann weggelassen werden, wenn Sie z.B. nur einen Teil der DATA DIVISION, die WORKING-STORAGE SECTION definieren.

Beispiel

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PROGR.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
.
.
.
Anweisungen
.
.
.
```

Jede DIVISION ist in SECTIONs aufgeteilt, die weiter in Paragraphen unterteilt sind, die wiederum aus Sätzen bestehen.

Innerhalb dieser Unterteilungen eines LII COBOL-Programms gibt es weitere Unterteilungen, die Klauseln bzw. Anweisungen genannt werden. Eine Klausel ist eine geordnete Menge von LII COBOL-Elementen, die ein Attribut einer Eintragung kennzeichnen. Eine Anweisung ist eine Kombination von Elementen in der PROCEDURE DIVISION, die mit einem LII COBOL-Verb beginnen und einen Programmbefehl bilden.

1.7 Formate und Regeln

1.7.1 Allgemeines Format

Ein allgemeines Format ist die spezielle Anordnung der Elemente einer Klausel oder einer Anweisung. Im vorliegenden Handbuch wird ein Format direkt anschließend an die Information gezeigt, die die Klausel oder die Anweisung definiert. Sofern mehr als nur eine bestimmte Anordung zulässig ist, wird die Anzahl der möglichen allgemeinen Formate durchnumeriert. Klauseln müssen in der in den allgemeinen Formaten vorgeschriebenen Folge geschrieben werden.

Wenn Sie optionale Klauseln verwenden, müssen diese ebenfalls in der angegebenen Folge erscheinen. In bestimmten Fällen, die in den Regeln beim entsprechenden Format ausdrücklich genannt sind, können die Klauseln in einer anderen Folge geschrieben werden als angezeigt. Anwendungsfälle, Erfordernisse oder Beschränkungen werden als Regeln angegeben.

Allgemeine Regeln

Eine allgemeine Regel ist eine Regel, die die Bedeutung oder die Beziehung von Bedeutungen eines Elements oder einer Elementengruppe definiert oder erläutert. Sie wird dazu verwendet, die Semantik einer Anweisung zu definieren oder zu erläutern sowie die Auswirkungen, die sie auf Ausführung oder Übersetzung hat.

Syntaxregeln

Syntaxregeln sind die Regeln, die die Reihenfolge definieren oder erläutern, in der Wörter oder Elemente angeordnet werden, um dadurch größere Elemente wie Angaben, Klauseln oder Anweisungen zu bilden. Die Syntaxregeln belegen auch einzelne Wörter oder Elemente mit Beschränkungen.

In diesen Regeln wird definiert oder erläutert, wie Anweisungen geschrieben werden müssen, d.h., die Reihenfolge der Elemente einer Anweisung mit eventuellen Einschränkungen.

Elemente

Elemente, die eine Klausel oder eine Anweisung bilden, bestehen aus groß- oder kleingeschriebenen Wörtern, Stufennummern, geschweiften Klammern, Bindewörtern und Sonderzeichen.

Quelltextformat

Das COBOL-Quelltextformat unterteilt jede LII COBOL-Quelltextzeile in 72 Spalten. Diese Spalten werden in folgender Weise verwendet:

Spalte 1 - 6	Folgenummer
Spalte 7	Indikationsbereich
Spalte 8 - 11	Bereich A
Spalte 12 - 72	Bereich B

Folgenummer

Eine Folgenummer, bestehend aus sechs Ziffern, kann verwendet werden, um eine Quellprogrammzeile zu identifizieren. Wenn in Spalte 1 ein Stern (*) oder in Spalte 1 oder 2 ein Seitenvorschub-Zeichen gefolgt von einem * steht, so wird die gesamte Zeile vom Übersetzer ignoriert und erscheint nicht in der List-Datei. Diese Einrichtung ermöglicht es, List-Dateien als Quell-Dateien zu verwenden.

Indikationsbereich

Ein Stern "*" in diesem Bereich kennzeichnet die Zeile als eine Kommentarzeile. Eine solche Kommentarzeile kann überall im Programm nach dem Kopfsatz der IDENTIFICATION DIVISION erscheinen. Jedes Zeichen aus dem ASCII-Zeichenvorrat kann im Bereich A oder Bereich B der Zeile stehen.

Ein Schrägstrich "/" im Indikationsbereich hat die gleiche Funktion wie die obenstehende Kommentarzeile, verursacht jedoch einen Seitenvorschub, ehe der Kommentar ausgedruckt wird.

Ein "D" im Indikationsbereich stellt eine Testhilfezeile dar. Die Bereiche A und B können jeden zulässigen LII COBOL-Satz enthalten.

Ein "-" im Indikationsbereich stellt die Weiterführung einer vorhergehenden Zeile ohne Zwischenräume dar bzw. die Fortsetzung eines nichtnumerisches Literals (vgl. Kapitel 2).

Bereiche A und B

SECTION-Namen und Paragraph-Namen beginnen im Bereich A und werden von einem Punkt und einem Leerzeichen abgeschlossen. Die Stufenbezeichnungen FD, 01 sowie 66, 77 und 88 beginnen im Bereich A und werden im Bereich B einer Datei- und Datensatz-Beschreibung abgeschlossen.

Programmsätze können überall im Bereich A und Bereich B beginnen.

Es ist zu beachten, daß TAB-Zeichen im LII COBOL Quelltext nicht erlaubt sind.

Das folgenden Beispiel zeigt das Quelltextformat eines typischen Programms.

```
* Level II COBOL  V2.1                STOCK9.CBL                    Page 0001
*
* Options:
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. LAGERSTAMM-DATEI.
000030 AUTHOR. MICRO FOCUS LTD, PROTEUS.
000040 ENVIRONMENT DIVISION.
000050 CONFIGURATION SECTION.
000060 SOURCE-COMPUTER. PROTEUS-80.
000070 OBJECT-COMPUTER. PROTEUS-80.
000075 SPECIAL-NAMES. CONSOLE IS CRT.
000080 INPUT-OUTPUT SECTION.
000090 FILE-CONTROL.
000100     SELECT LAGERSTAMM-DATEI ASSIGN "LAGER.IT"
000110     ORGANIZATION INDEXED
000120     ACCESS DYNAMIC
000130     RECORD KEY LAGER-SCHLUESSEL.
000140 DATA DIVISION.
000150 FILE SECTION.
000160 FD  LAGERSTAMM-DATEI; RECORD 32.
000170 01  LAGER-ARTIKEL.
000180     02  LAGER-SCHLUESSEL        PIC X(4).
000190     02  PRODUKT-BESCHR          PIC X(24).
000200     02  ABMESSUNGEN             PIC 9(4).
000210 WORKING-STORAGE SECTION.
000220 01  BILDSCHIRM-UEBERSCHRIFTEN.
000230     02  ASK-SCHLUESSEL          PIC X(26)
000231         VALUE "LAGER-SCHLUESSEL    <    >".
000240     02  FILLER                  PIC X(54).
000250     02  ASK-BESCHR              PIC X(21)
000251         VALUE "BESCHREIBUNG        <".
000260     02  SI-BESCHR               PIC X(25)
000261         VALUE "                      >".
000270     02  FILLER                  PIC X(34).
000280     02  ASK-GROESSE             PIC X(26)
000281         VALUE "ABMESSUNGEN         <    >".
000290 01 AENDERUNG-DAT REDEFINES BILDSCHIRM-UEBERSCHRIFTEN.
000300     02  FILLER                  PIC X(21).
000310     02  CRT-LAGER-SCHLUESSEL    PIC X(4).
000320     02  FILLER                  PIC X(76).
000330     02  CRT-PROD-BESCHR         PIC X(24).
000340     02  FILLER                  PIC X(56).
000350     02  CRT-ABMESSUNGEN         PIC 9(4).
000360     02  FILLER                  PIC X.
000370 PROCEDURE DIVISION.
000380 SR1.
000390     DISPLAY SPACE.
000400     OPEN I-O LAGERSTAMM-DATEI.
000410     DISPLAY BILDSCHIRM-UEBERSCHRIFTEN.
000420 NORMAL-EINGABE.
000430     MOVE SPACE TO AENDERUNG-DAT.
000440     DISPLAY AENDERUNG-DAT.
```

```
* Level II COBOL  V2.1                STOCK9.CBL                  Page 0002
*
000450 FEHLER-KORREKTUR.
000460     ACCEPT AENDERUNG-DAT.
000470     IF CRT-LAGER-SCHLUESSEL = SPACE GO TO ENDE.
000480     IF CRT-ABMESSUNGEN NOT NUMERIC GO TO FEHLER-KORREKTUR.
000490     MOVE CRT-PROD-BESCHR TO PRODUKT-BESCHR.
000500     MOVE CRT-ABMESSUNGEN TO ABMESSUNGEN.
000510     MOVE CRT-LAGER-SCHLUESSEL TO LAGER-SCHLUESSEL.
000520     WRITE LAGER-ARTIKEL; INVALID GO TO FEHLER-KORREKTUR.
000530     GO TO NORMAL-EINGABE.
000540 ENDE.
000550     CLOSE LAGERSTAMM-DATEI.
000560     DISPLAY SPACE.
000570     DISPLAY "PROGRAMM ENDE".
000580     STOP RUN.
* Level II COBOL  V2.1 REVISION 1                          URN LD/0001/BD
* Übersetzer   [ 1981, 1982  Micro Focus Ltd.
*
* ERRORS=00000 DATA=01024 CODE=00512 DICT=00510:61296/61806 GSA FLAGS =  OFF

Spalte     Spalte     Spalte     Spalte     Eingesetzt
1-6        7          8-11       12-72        durch
Folge-     Indika-    Bereich A  Bereich B  Übersetzer
nummer     torbereich
```

1.8 Syntaxübersicht

Alle Syntaxregeln für LII COBOL werden nachfolgend aufgelistet.
Grau unterlegte Stellen geben an, daß es sich hier für LII COBOL um eine Angabe zur Dokumentation handelt.
Grün zeigt eine Spracherweiterung gegenüber ANS74 COBOL an.

Allgemeines Format der IDENTIFICATION DIVISION

```
IDENTIFICATION DIVISION.
PROGRAM-ID.          programmname.
[AUTHOR.             [kommentar]...]
[INSTALLATION.       [kommentar]...]
[DATE-WRITTEN.       [kommentar]...]
[DATE-COMPILED.      [kommentar]...]
[SECURITY.           [kommentar]...]
```

Allgemeines Format der ENVIRONMENT DIVISION

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  übersetzungsanlage [WITH DEBUGGING MODE].
OBJECT-COMPUTER.  programmausführungsanlage
    [,MEMORY SIZE ganzzahl  {WORDS     }]
                            {CHARACTERS}
                            {MODULES   }
    [,PROGRAM COLLATING SEQUENCE IS alphabetname]
    [,SEGMENT-LIMIT IS segmentnummer].
```

```
[SPECIAL-NAMES.

    [ ,{SYSIN  } IS merkname-1 ]
       {SYSOUT }

    [, FORMFEED  IS merkname-2 ]

    [ SWITCH {0} [IS merkname] ON STATUS IS bedingungsname-1 ]
             {.}
             {7}
               [OFF STATUS IS bedingungsname-2]

    [,alphabetname IS
        {STANDARD-1                                          }
        {NATIVE                                              }
        {literal-1 [{THROUGH} literal-2                      ]}
        {          {THRU    }                                ]}
        {          [ALSO literal-3 [,ALSO literal-4] ...     ]} ...
        {  [literal-5 [{THROUGH} literal-6                   ]]}
        {             {THRU    }                             ]}
        {             [ALSO literal-7 [,ALSO literal-8]...  ]]...}

     [,CURRENCY SIGN IS literal-9]

     [,DECIMAL-POINT IS COMMA]

     [,CURSOR IS datenname-1]

     [,CONSOLE IS CRT]
```

```
[INPUT-OUTPUT SECTION.

  FILE-CONTROL.

       {dateisteuerungseintrag} ...

 I-O-CONTROL.

  ;RERUN  [ON {dateiname-1     }]
              {herstellername  }

          EVERY {{[END OF] {REEL}          } OF dateiname-2}
                {{         {UNIT}          }               } ...
                {{ganzzahl-1   RECORDS     }               }
                { ganzzahl-2   CLOCK-UNITS                 }
                { bedingungsname                           }

  [;SAME [RECORD    ] AREA FOR dateiname-3 {,dateiname-4} ...] ...
         [SORT      ]
         [SORT-MERGE]

  [; MULTIPLE FILE TAPE CONTAINS dateiname-5 [POSITION ganzzahl-3]

        [,dateiname-6 [POSITION ganzzahl-4] ] ...                ] ... . ]
```

Allgemeines Format für FILE-CONTROL

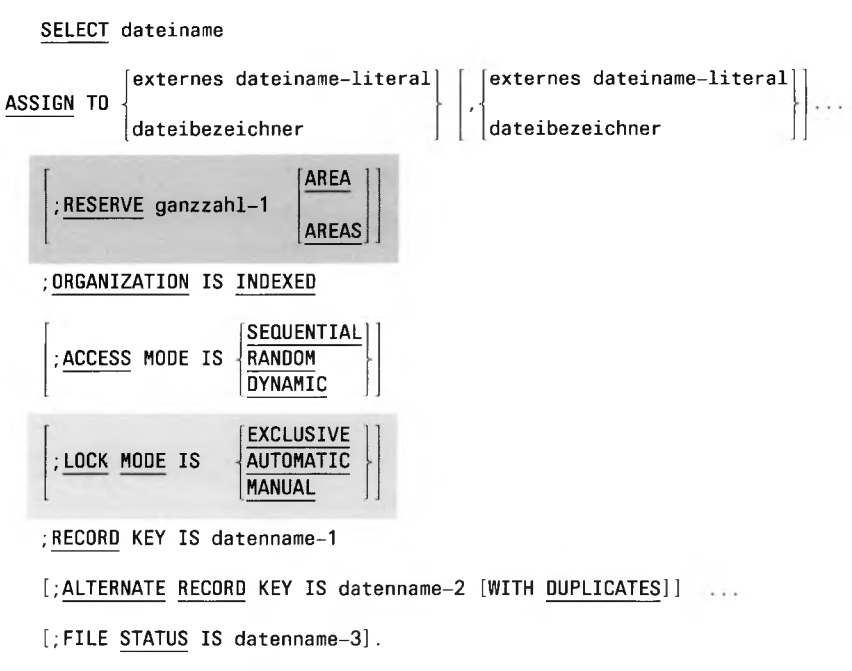
SELECT-Klausel bei sequentiellen Dateien

```
SELECT [OPTIONAL] dateiname

ASSIGN TO { externes dateiname-literal  } [ , { externes dateiname-literal  } ] ...
          { dateibezeichner             }       { dateibezeichner             }

[ ;RESERVE ganzzahl-1 [ AREA  ] ]
                      [ AREAS ]

[ ;ORGANIZATION IS { SEQUENTIAL      } ]
                   { LINE SEQUENTIAL }

[;ACCESS MODE IS SEQUENTIAL]

[;FILE STATUS IS datenname].
```

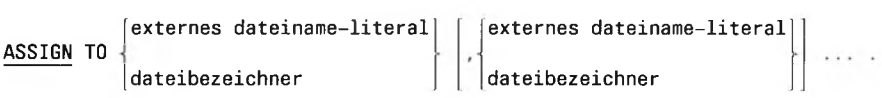
SELECT-Klausel bei relativen Dateien

```
SELECT dateiname

ASSIGN TO { externes dateiname-literal  } [ , { externes dateiname-literal  } ] ...
          { dateibezeichner             }       { dateibezeichner             }

[ ;RESERVE ganzzahl-1 [ AREA  ] ]
                      [ AREAS ]

ORGANIZATION IS RELATIVE

[ ;ACCESS MODE IS { SEQUENTIAL [,RELATIVE KEY IS datenname-1] } ]
                  { RANDOM       ,RELATIVE KEY IS datenname-1  }
                  { DYNAMIC                                    }

[;FILE STATUS IS datenname-2].
```

SELECT-Klausel bei indizierten Dateien



SELECT-Klausel bei SORT oder MERGE



Allgemeines Format der DATA DIVISION

```
DATA DIVISION.

[FILE SECTION.

[FD dateiname

[;BLOCK CONTAINS [ganzzahl-1 TO] ganzzahl-2 {RECORDS    }]
                                            {CHARACTERS}

[;RECORD CONTAINS [ganzzahl-3  TO ] ganzzahl-4  CHARACTERS]

 ;LABEL {RECORD IS  } {STANDARD}
        {RECORDS ARE} {OMITTED }

[;VALUE OF datenname-1 IS {datenname-2}
                          {literal-1  }

          [,datenname-3 IS {datenname-4}] ...
                           {literal-2  }  ]

    [;DATA {RECORD IS  } datenname-5 [,datenname-6] ...  ]
           {RECORDS ARE}

      [;LINAGE IS {datenname-7} LINES [,WITH FOOTING AT {datenname-8}]
                  {ganzzahl-5 }                         {ganzzahl-6 }

      [,LINES AT TOP {datenname-9}] [,LINES AT BOTTOM {datenname-10}]]
                     {ganzzahl-7 }                    {ganzzahl-8  }

    [;CODE-SET IS alphabetname] .

  [datensatzbeschreibung] ... ] ... ]
```

```
[SD dateiname
    [;RECORD CONTAINS [ganzzahl-1 TO] ganzzahl-2 CHARACTERS]
    [;DATA {RECORD IS  } datenname-1 [,datenname-2] ...]
           {RECORDS ARE}
    [datensatzbeschreibung] ... ] ... ]

[WORKING-STORAGE SECTION.
    [77-stufenbeschreibung]
    [datensatzbeschreibung] ...]

[LINKAGE SECTION.
    [77-stufenbeschreibung]
    [datensatzbeschreibung] ...]

[COMMUNICATION SECTION.
    [kommunikationsbeschreibung ]
    [[datensatzbeschreibung] ...] ...]
```

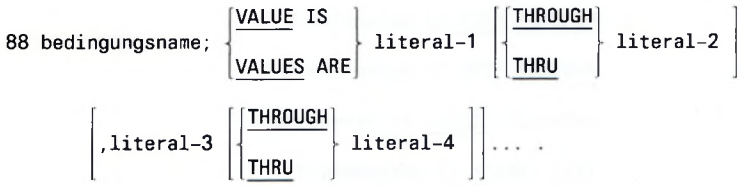
Allgemeines Format der Datenbeschreibung

Format 1

```
level-number { datenname-1 }
             { FILLER      }

[;REDEFINES datenname-2]

[ ;{ PICTURE } IS picture-zeichenfolge ]
  { PIC     }

[ ;[USAGE IS] { COMPUTATIONAL   } ]
              { COMP            }
              { COMPUTATIONAL-3 }
              { COMP-3          }
              { DISPLAY         }
              { INDEX           }

[ [;SIGN IS] { LEADING  } [SEPARATE CHARACTER] ]
             { TRAILING }

[ ;OCCURS { ganzzahl-1 TO ganzzahl-2 TIMES DEPENDING ON datenname-3 } ]
          { ganzzahl-2 TIMES                                        }

[ { ASCENDING  } KEY IS datenname-4 [,datenname-5] ... ] ...
  { DESCENDING }

[INDEXED BY indexname-1 [,indexname-2] ... ]

[ ;{ SYNCHRONIZED } { LEFT  } ]
   { SYNC         } { RIGHT }

[ ;{ JUSTIFIED } RIGHT ]
   { JUST      }

[;BLANK WHEN ZERO]

[ ;{ VALUE IS  } literal-1 [ { THROUGH } literal-2 ]
   { VALUES ARE }            { THRU    }

                  [ ,literal-3 [ { THROUGH } literal-4 ] ] ... ] .
                               { THRU    }
```

Format 2

Format 3



Allgemeines Format der Kommunikationsbeschreibung

Format 1

```
CD cd-name;
                        | [ [; SYMBOLIC SUB-QUEUE IS datenname-1 ]
                          |      [; SYMBOLIC SUB-QUEUE-1 IS datenname-2 ]
                          |      [; SYMBOLIC SUB-QUEUE-2 IS datenname-3 ]
                          |      [; SYMBOLIC SUB-QUEUE-3 IS datenname-4 ]
                          |      [; MESSAGE DATA IS datenname-5 ]
FOR [INITIAL] INPUT       |      [; MESSAGE TIME IS datenname-6 ]
                          |      [; SYMBOLIC SOURCE IS datenname-7 ]
                          |      [; TEXT LENGTH IS datenname-8 ]
                          |      [; END KEY IS datenname-9 ]
                          |      [; STATUS KEY IS datenname-10 ]
                          |      [; MESSAGE COUNT IS datenname-11 ]  ]
                          | [datenname-1, datenname-2, ... , datenname-11 ] | .
```

Format 2

```
CD cd-name; FOR OUTPUT

                    [; DESTINATION COUNT IS datenname-1 ]

                    [; TEXT LENGTH IS datenname-2 ]

                    [; STATUS KEY IS datenname-3 ]

                    [ ; DESTINATION TABLE OCCURS ganzzahl-2 TIMES          ]
                        [ ; INDEXED BY indexname-1 [, indexname-2] ...]    ]

                    [; ERROR KEY IS datenname-4 ]

                    [; SYMBOLIC DESTINATION IS datenname-5 ] .
```

Allgemeines Format der PROCEDURE DIVISION

DECLARATIVES-Format

```
PROCEDURE DIVISION [USING datenname-1 [,datenname-2]...].
DECLARATIVES.
{ sectionname  SECTION [segmentnummer] . vereinbarungssatz
  [paragraphenname. [programmsatz] ... ] ... } ...
END DECLARATIVES.
{ sectionname  SECTION [segmentnummer] .
  [paragraphenname . [programmsatz] ... ] ... } ...
```

Ohne DECLARATIVES

```
PROCEDURE DIVISION [USING datenname-1 [,datenname-2] ... ] .
{ sectionname  SECTION [segmentnummer].
    [paragraphenname. [programmsatz] ... ] ... } ...
```

Allgemeines Format der Anweisungen

```
ACCEPT datenname-1 [AT {datenname-2 }] FROM CRT
                       {literal-1   }

ACCEPT bezeichner [FROM {merkname }]
                        {CONSOLE  }

ACCEPT bezeichner FROM {DATE}
                       {DAY }
                       {TIME}

ACCEPT cd-name MESSAGE COUNT

ADD {bezeichner-1} [,bezeichner-2] ... TO bezeichner-m [ROUNDED]
    {literal-1   } [,literal-2   ]

    [,bezeichner-n [ROUNDED]]...[;ON SIZE ERROR unbedingte-anweisung]


ADD {bezeichner-1} [,bezeichner-2] [,bezeichner-3] ...
    {literal-1   } [,literal-2   ] [,literal-3   ]

          GIVING bezeichner-m [ROUNDED] [,bezeichner-n [ROUNDED]] ...

          [;ON SIZE ERROR unbedingte-anweisung]

ADD {CORRESPONDING} bezeichner-1 TO bezeichner-2 [ROUNDED]
    {CORR          }

          [;ON SIZE ERROR unbedingte-anweisung]

ALTER  prozedurname-1 TO [PROCEED TO] prozedurname-2
     [,prozedurname-3 TO [PROCEED TO] prozedurname-4] ...

CALL {bezeichner-1} [USING datenname-1 [,datenname-2] ... ]
     {literal-1   }

          [;ON OVERFLOW unbedingte-anweisung]
```

```
CANCEL { bezeichner-1 } [ ,bezeichner-2 ] ...
       { literal-1    } [ ,literal-2    ]
```

```
CLOSE dateiname-1 [ { REEL } [WITH NO REWIND] ]
                   { UNIT } [FOR REMOVAL     ]
                  [ WITH { NO REWIND } ]
                         { LOCK      }
 [ ,dateiname-2 [ { REEL } [WITH NO REWIND] ] ] ...
                  { UNIT } [FOR REMOVAL     ]
                 [ WITH { NO REWIND } ]
                        { LOCK      }
```

```
CLOSE dateiname-1 [WITH LOCK] [,dateiname-2 [WITH LOCK] ]...
```

```
COMMIT (siehe LII COBOL Bedienungsanleitung)
```

```
COMPUTE bezeichner-1 [ROUNDED] [,bezeichner-2 [ROUNDED]]...
    = arithmetischer ausdruck [;ON SIZE ERROR unbedingte-anweisung]
```

```
DELETE dateiname RECORD [;INVALID KEY unbedingte-anweisung]
```

```
DISABLE { INPUT [TERMINAL] } cd-name WITH KEY { bezeichner-1 }
        { OUTPUT           }                  { literal-1    }
```

```
DISPLAY { bezeichner-1 } [ ,bezeichner-2 ] ... [UPON { merkname }]
        { literal-1    } [ ,literal-2    ]         { CONSOLE  }
```

```
DISPLAY { datenname-1 } [ AT { datenname-2 } ] UPON { CRT       }
        { literal-3   }      { literal-4   }       { CRT-UNDER }
```

```
DIVIDE { bezeichner-1 } INTO bezeichner-2 [ROUNDED]
       { literal-1    }
    [,bezeichner-3 [ROUNDED]] ... [;ON SIZE ERROR unbedingte-anweisung]

DIVIDE { bezeichner-1 } { INTO } { bezeichner-2 } GIVING bezeichner-3 [ROUNDED]
       { literal-1    } { BY   } { literal-2    }
    [,bezeichner-4 [ROUNDED]] ... [;ON SIZE ERROR unbedingte-anweisung]

DIVIDE { bezeichner-1 } { INTO } { bezeichner-2 } GIVING bezeichner-3 [ROUNDED]
       { literal-1    } { BY   } { literal-2    }
      REMAINDER bezeichner-4 [;ON SIZE ERROR unbedingte-anweisung]

ENABLE { INPUT [TERMINAL] } cd-name WITH KEY { bezeichner-1 }
       { OUTPUT           }                  { literal-1    }

ENTER sprachenname [routinenname].

EXIT [PROGRAM].

GO TO [prozedurname-1]

GO TO prozedurname-1 {,prozedurname-2} ... DEPENDING ON bezeichner

IF bedingung; THEN { anweisung-1 } [;ELSE anweisung-2 ]
                   { NEXT SENTENCE } [;ELSE NEXT SENTENCE]
```

```
INSPECT bezeichner-1 TALLYING

  { bezeichner-2 FOR { , { { ALL     }  { bezeichner-3 } }
                           { LEADING }  { literal-1    }
                           CHARACTERS                    }

      [ { BEFORE }  INITIAL  { bezeichner-4 } ] } ... } ...
      [ { AFTER  }           { literal-2    } ]
```

```
INSPECT bezeichner-1  REPLACING

  {   CHARACTERS BY { bezeichner-6 }  [ { AFTER  }  INITIAL { bezeichner-7 } ]   }
  {                 { literal-4    }  [ { BEFORE }          { literal-5    } ]   }
  {                                                                              }
  { { , { ALL     } { , { bezeichner-5 }  BY { bezeichner-6 }                    }
  { {   { LEADING } {   { literal-3    }     { literal-4    }                    }
  { {   { FIRST   } {                                                            }
  {                 [ { AFTER  }  INITIAL { bezeichner-7 } ] } ... } ...         }
  {                 [ { BEFORE }          { literal-5    } ]                     }
```

```
INSPECT bezeichner-1 TALLYING {TALLY-Klausel} REPLACING {REPLACING-Klausel}
```

```
MERGE dateiname-1 ON { ASCENDING  } KEY datenname-1 [,datenname-2] ...
                     { DESCENDING }

               [ ON { ASCENDING  } KEY datenname-3 [,datenname-4] ... ] ...
               [    { DESCENDING }                                    ]

     [COLLATING SEQUENCE IS alphabet-name]

      USING dateiname-2, dateiname-3 [,dateiname-4]...

     { OUTPUT PROCEDURE IS sectionname-1  [ { THROUGH } sectionname-2 ] }
     {                                    [ { THRU    }               ] }
     {                                                                  }
     { GIVING dateiname-5                                               }
```

```
        ⎧bezeichner-1 ⎫
MOVE    ⎨             ⎬ TO bezeichner-2  [,bezeichner-3] ...
        ⎩literal      ⎭

        ⎧CORRESPONDING⎫
MOVE    ⎨             ⎬ bezeichner-1 TO bezeichner-2
        ⎩CORR         ⎭

            ⎧bezeichner-1⎫
MULTIPLY    ⎨            ⎬ BY bezeichner-2 [ROUNDED]
            ⎩literal-1   ⎭

    [,bezeichner-3 [ROUNDED] ] ...  [;ON SIZE ERROR unbedingte-anweisung]

            ⎧bezeichner-1⎫      ⎧bezeichner-2⎫
MULTIPLY    ⎨            ⎬ BY   ⎨            ⎬ GIVING bezeichner-3 [ROUNDED]
            ⎩literal-1   ⎭      ⎩literal-2   ⎭

    [,bezeichner-4 [ROUNDED] ] ...

    [;ON SIZE ERROR unbedingte-anweisung]

     ⎧                    ⎡REVERSED     ⎤ ⎡              ⎡REVERSED     ⎤⎤     ⎫
     ⎪INPUT dateiname-1   ⎢             ⎥ ⎢,dateiname-2  ⎢             ⎥⎥ ... ⎪
     ⎪                    ⎣WITH NO REWIND⎦ ⎣              ⎣WITH NO REWIND⎦⎦     ⎪
OPEN ⎨OUTPUT dateiname-3[WITH NO REWIND][,dateiname-4 [WITH NO REWIND]]...    ⎬ ...
     ⎪I-O dateiname-5  [,dateiname-6]...                                      ⎪
     ⎩EXTEND dateiname-7 [,dateiname-8] ...                                   ⎭
```

```
PERFORM prozedurname-1 [ [ THROUGH ] prozedurname-2 ]
                       [ [ THRU    ]               ]

PERFORM prozedurname-1 [ [ THROUGH ] prozedurname-2 ] [ bezeichner-1 ] TIMES
                       [ [ THRU    ]                 [ ganzzahl-1   ]

PERFORM prozedurname-1 [ [ THROUGH ] prozedurname-2 ] UNTIL bedingung-1
                       [ [ THRU    ]                 ]

PERFORM prozedurname-1 [ [ THROUGH ] prozedurname-2 ]
                       [ [ THRU    ]               ]

          VARYING [ bezeichner-2 ] FROM [ bezeichner-2 ]
                  [ indexname-1  ]      [ indexname-2  ]
                                        [ literal-1    ]

          BY      [ bezeichner-4 ] UNTIL bedingung-1
                  [ literal-2    ]

          [ AFTER [ bezeichner-5 ] FROM [ bezeichner-6 ]
                  [ indexname-3  ]      [ indexname-4  ]
                                        [ literal-3    ]

          BY      [ bezeichner-7 ] UNTIL bedingung-2
                  [ literal-4    ]

          [ AFTER [ bezeichner-8 ] FROM [ bezeichner-9 ]
                  [ indexname-5  ]      [ indexname-6  ]
                                        [ literal-5    ]

          BY      [ bezeichner-10 ] UNTIL bedingung-3 ] ]
                  [ literal-6     ]                     ]

READ dateiname [NEXT] RECORD [INTO bezeichner] [WITH [KEPT] LOCK]
        [;AT END unbedingte-anweisung]

READ dateiname RECORD [INTO bezeichner] [;KEY IS datenname]
     [WITH [KEPT] LOCK]    [;INVALID KEY unbedingte-anweisung]
                    (siehe LII COBOL Bedienungsanleitung)
```

```
RECEIVE cd-name { MESSAGE } INTO bezeichner-1 [;NO DATA unbedingte-anweisung]
                { SEGMENT }

RELEASE datensatzname [FROM bezeichner]

RETURN dateiname RECORD [INTO bezeichner] ;AT END unbedingte-anweisung

REWRITE datensatzname [FROM bezeichner]
        [;INVALID KEY unbedingte-anweisung]
```

ROLLBACK (siehe LII COBOL Bedienungsanleitung)

```
SEARCH bezeichner-1 [VARYING { bezeichner-2 }]
                             { indexname-1  }
    [; AT END unbedingte-anweisung-1]
     ;WHEN bedingung-1 { unbedingte-anweisung-2 }
                       { NEXT SENTENCE          }
    [;WHEN bedingung-2 { unbedingte-anweisung-3 }] ...
                       { NEXT SENTENCE          }
```

```
SEARCH ALL bezeichner-1 [;AT END unbedingte-anweisung-1]
       ;WHEN { datenname-1 { [IS EQUAL TO] } { bezeichner-3                    }
             {             { [IS =       ] } { literal-1                      }
             {                               { arithmetischer-ausdruck-1      }
             { bedingungsname-1                                               }

       [ AND { datenname-2 { [IS EQUAL TO] } { bezeichner-4                    } } ]...
             {             { [IS =       ] } { literal-2                      }
             {                               { arithmetischer ausdruck-2      }
             { bedingungsname-2                                               }

       { [unbedingte-anweisung-2] }
       { [NEXT SENTENCE         ] }
```

```
SEND cd-name FROM bezeichner-1
```

```
SEND cd-name [FROM bezeichner-1] { [WITH bezeichner-2] }
                                 { [WITH ESI         ] }
                                 {                     }
                                 { [WITH EMI         ] }
                                 { [WITH EGI         ] }

     [ { [BEFORE] } ADVANCING { { [bezeichner-3] } [LINE ] } ]
     [ { [AFTER ] }           { { [ganzzahl-1  ] } [LINES] } ]
     [                        {                           } ]
     [                        { { [merkname] }            } ]
     [                        { { [PAGE   ] }             } ]
```

```
SET { [bezeichner-1 [,bezeichner-2]...] } TO { [bezeichner-3] }
    { [indexname-1  [,indexname-2 ]...] }    { [ganzzahl-1  ] }
                                             { [indexname-3 ] }
```

```
SET { [indexname-4  [,indexname-5 ]...] } { [UP BY  ] } { [bezeichner-4] }
    { [bezeichner-5 [,bezeichner-6]...] } { [DOWN BY] } { [ganzzahl-2  ] }
                                                       { [indexname-6 ] }
```

```
SORT dateiname-1 ON { ASCENDING  } KEY datenname-1 [,datenname-2]...
                    { DESCENDING }

                  [ ON { ASCENDING  } KEY datenname-3 [,datenname-4 ] ... ] ...
                       { DESCENDING }

    [COLLATING SEQUENCE IS alphabet-name]

    { INPUT PROCEDURE IS sectionname-1 [ { THROUGH } sectionname-2 ] }
    { USING dateiname-2 ,[dateiname-3] ...                          }

    { OUTPUT PROCEDURE IS sectionname-3 [ { THROUGH } sectionname-4 ] }
    {                                     { THRU    }                 }
    { GIVING dateiname-4                                              }
```

```
                                  [ IS EQUAL TO                ]
                                  [ IS =                       ]
START dateiname [ KEY { IS GREATER THAN  } datenname ]
                                  [ IS >                       ]
                                  [ IS NOT LESS THAN           ]
                                  [ IS NOT <                   ]

    [;INVALID KEY unbedingte-anweisung]
```

```
STOP { RUN     }
     { literal }
```

```
STRING { bezeichner-1 } [ , { bezeichner-2 } ] ... DELIMITED BY { bezeichner-3 }
       { literal-1    }     { literal-2    }                     { literal-3    }
                                                                { SIZE         }

    [ , { bezeichner-4 } [ , { bezeichner-5 } ] ... DELIMITED BY { bezeichner-6 } ] ...
        { literal-4    }     { literal-5    }                     { literal-6    }
                                                                 { SIZE         }

    INTO bezeichner-7 [WITH POINTER bezeichner-8]

    [,ON OVERFLOW unbedingte-anweisung]
```

```
SUBTRACT {bezeichner-1} [ , {bezeichner-2} ] ... FROM bezeichner-m [ROUNDED]
         {literal-1   }     {literal-2   }

         [, bezeichner-n [ROUNDED]]...

         [;ON SIZE ERROR unbedingte-anweisung]
```

```
SUBTRACT {bezeichner-1} [ , {bezeichner-2} ] ... FROM {bezeichner-m}
         {literal-1   }     {literal-2   }           {literal-m   }

       GIVING bezeichner-n [ROUNDED][, bezeichner-o [ROUNDED] ] ...

       [;ON SIZE ERROR unbedingte-anweisung]
```

```
SUBTRACT {CORRESPONDING} bezeichner-1 FROM bezeichner-2 [ROUNDED]
         {CORR         }

[; ON SIZE ERROR unbedingte-anweisung]
```

```
UNSTRING bezeichner-1

    [DELIMITED BY [ALL] {bezeichner-2} [ ,OR [ALL] {bezeichner-3} ] ... ]
                        {literal-1   }             {literal-2   }

    INTO bezeichner-4 [,DELIMITER IN bezeichner-5] [,COUNT IN bezeichner-6]

[,bezeichner-7 [,DELIMITER IN bezeichner-8] [,COUNT IN bezeichner-9]]...

     [WITH POINTER bezeichner-10] [TALLYING IN bezeichner-11]

     [;ON OVERFLOW unbedingte-anweisung]
```

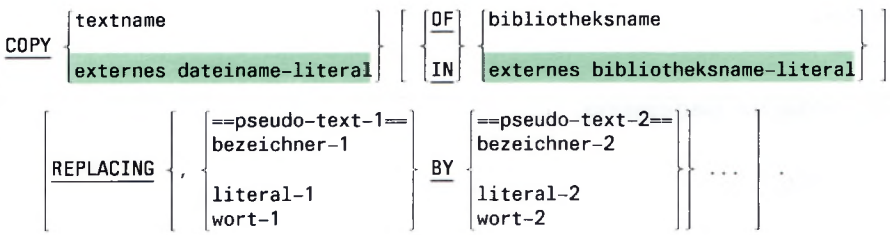
```
USE AFTER STANDARD {EXCEPTION} PROCEDURE ON {dateiname-1 [,dateiname-2] ...}.
                   {ERROR    }             {INPUT                          }
                                           {OUTPUT                         }
                                           {I-O                            }
                                           {EXTEND                         }
```

```
                          ┌ cd-name-1                              ┐
                          │[ALL REFERENCES OF] bezeichner-1         │
USE FOR DEBUGGING ON     {  dateiname-1                             }
                          │ prozedurname-1                          │
                          └ ALL PROCEDURES                         ┘

      ┌   cd-name-2                          ┐
      │   [ALL REFERENCES OF] bezeichner-2    │
      │ , dateiname-2                         │  ...  .
      │   prozedurname-2                      │
      └   ALL PROCEDURES                     ┘
```

```
WRITE datensatzname [FROM bezeichner-1]

          ┌                      ┌bezeichner-2 ┌LINE ┐┐┐
          │┌BEFORE┐              │ganzzahl     │     ││ │
          │{      } ADVANCING   {              │LINES┘} │
          │└AFTER ┘              │PAGE                  │
          └                      └merkname             ┘┘

          ┌     ┌END-OF-PAGE┐                         ┐
          │ ;AT {           } unbedingte-anweisung    │
          └     └EOP        ┘                         ┘
```

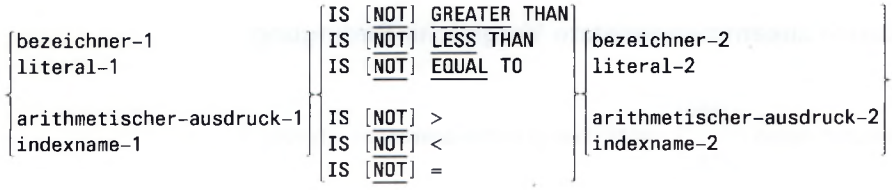
```
WRITE satzname [FROM bezeichner]

          [;INVALID KEY unbedingte-anweisung]
```

Allgemeines Format der COPY-Anweisung

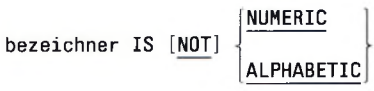


Allgemeines Format der Bedingungen

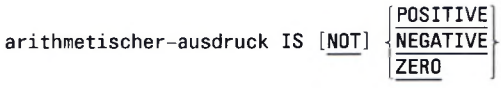
Vergleichs-Bedingung



Klassen-Bedingung:



Vorzeichen-Bedingung:



Bedingungsnamen-Bedingung:

bedingungsname

Schalterzustands-Bedingung:

```
bedingungsname
```

Negierte einfache Bedingung:

```
NOT einfache bedingung
```

Zusammengesetzte Bedingung:

```
bedingung { { AND } bedingung } ...
          { { OR  }           }
```

Abgekürzte zusammengesetzte Vergleichs-Bedingung:

```
vergleichsbedingung { { AND } [NOT] [vergleichs-operator] element } ...
                    { { OR  }                                     }
```

Vermischte Formate

Kennzeichnung:

```
{datenname-1     } [{OF} datenname-2] ...
{bedingungsname  } [{IN}            ]
```

```
paragraph-name [{OF} sectionname ]
               [{IN}             ]
```

```
textname [{OF} bibliotheksname]
         [{IN}                ]
```

Subskribierung:

```
{datenname      }
{bedingungsname } (subskript-1 [,subskript-2 [,subskript-3]])
```

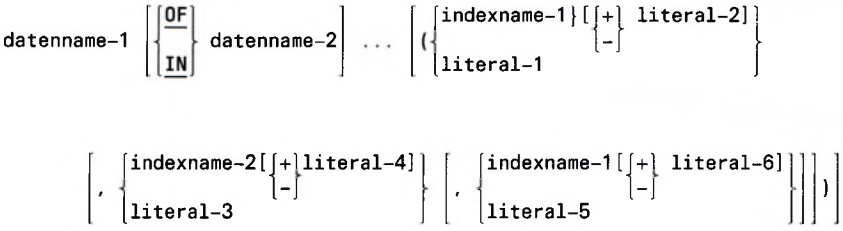
Indizierung:

```
{datenname } ( {indexname-1 [{+} literal-2]}
{bedingung }   {             {-}           }
               {literal-1                  }

  [, {indexname-2 [{+} literal-4]} [, {indexname-3 [{+} literal-6]}]] )
  [  {             {-}           } [  {             {-}           }]]
  [  {literal-3                  } [  {literal-5                  }]]
```

Bezeichner: Format 1

```
datenname-1 [{OF} datenname-2] ... [(subskript-1 [,subskript-2
            [{IN}            ]
                                          [, subskript-3] ] ) ]
```

Bezeichner: Format 2



1.9 Sprachkonzept

1.9.1 Zeichenvorrat

Die grundlegende und unteilbare Einheit der Sprache ist das Zeichen. Der Zeichenvorrat, der zur Bildung von LII COBOL Zeichenfolgen und Trennsymbolen verwendet wird, umfaßt die Buchstaben des Alphabets, Ziffern und Sonderzeichen. Der Zeichenvorrat besteht aus den nachfolgend aufgeführten Zeichen:

0 bis 9
A bis Z
a bis z (reservierte und Programmierer-Wortzeichen; zu lesen als: A bis Z)
Leerzeichen
\+ Pluszeichen
— Minuszeichen oder Gedankenstrich
* Stern
/ Schrägstrich
= Gleichheitszeichen
$ Dollarzeichen
. Punkt oder Dezimalpunkt
, Komma oder Dezimalpunkt
; Strichpunkt
" Anführungszeichen
(linke Klammer
) rechte Klammer
> Größer-als-Symbol
< Kleiner-als-Symbol

Die Sprache LII COBOL ist auf den obenstehenden Zeichenvorrat beschränkt; nichtnumerische Literale, Kommentarzeilen und Daten können jedoch alle Zeichen aus dem ASCII-Zeichenvorrat umfassen.

1.9.2 Sprachaufbau

Die einzelnen Zeichen der Sprache sind verkettet und bilden Zeichenfolgen und Trennsymbole. Ein Trennsymbol kann an ein anderes Trennsymbol oder an eine Zeichenfolge angefügt werden. An eine Zeichenfolge kann lediglich ein Trennsymbol angefügt werden.

Trennsymbole

Ein Trennsymbol besteht aus einem oder einer Folge von mehreren Interpunktionszeichen. Für die Bildung von Trennsymbolen gelten die folgenden Regeln:

1. Das Interpunktionszeichen ' ' (Leerzeichen/Blank) ist ein Trennsymbol. Überall, wo ein einzelnes Leerzeichen als Trennsymbol verwendet wird, kann auch mehr als ein Trennsymbol verwendet werden.

2. Die Interpunktionszeichen , (Komma), ; (Strichpunkt) und . (Punkt) sind Trennsymbole, wenn direkt nach ihnen ein Leerzeichen folgt. Diese Trennsymbole dürfen in einem COBOL Quellprogramm nur dort erscheinen, wo sie ausdrücklich durch die allgemeinen Formate, durch Formatregeln für die Interpunktion, durch die Definitionen des Aufbaus von Anweisungen oder den Referenzformatregeln ausdrücklich zugelassen werden,

3. Die Interpunktionszeichen () (rechte und linke Klammer) sind Trennsymbole. Rechte und linke Klammern dürfen nur paarweise erscheinen und trennen Indizes, Subskripte, arithmetische Ausdrücke oder Bedingungen.

4. Das Interpunktionszeichen " (Anführungszeichen) ist ein Trennsymbol. Unmittelbar vor einem eröffnenden Anführungszeichen muß ein Leerzeichen oder eine linke Klammer stehen; unmittelbar nach einem abschließenden Anführungszeichen muß eines der Trennsymbole Leerzeichen, Komma, Strichpunkt, Punkt oder rechte Klammer folgen.

   Anführungszeichen dürfen nur paarweise auftreten und begrenzen nichtnumerische Literale, außer wenn das Literal fortgesetzt wird. (vgl. Fortsetzung von Zeilen in diesem Kapitel).

5. Pseudo-Text-Begrenzer sind Trennsymbole. Unmittelbar vor einem eröffnenden Pseudo-Text-Begrenzer kann ein Leerzeichen stehen; unmittelbar nach einem abschließenden Pseudo-Text-Begrenzer muß eines der Trennsymbole Leerzeichen, Komma, Strichpunkt oder Punkt folgen.

   Pseudo-Text-Begrenzer dürfen nur paarweise auftreten und begrenzen Pseudo-Text (vgl. Kapitel 10 Bibliothek).

6. Das Trennsymbol ' ' (Leerzeichen/Blank) darf nicht vor den folgenden 3 Trennsymbolen stehen, sonst wenn gewünscht, kann es immer optional unmittelbar vor allen Trennsymbolen außer den folgenden stehen:

   - wenn es in Referenzformatregeln angegeben ist.
   - vor dem Trennsymbol '"' (**abschließendes** Anführungszeichen). In diesem Fall wird ein vorangehendes Leerzeichen als Teil des nichtnumerischen Literals angesehen und nicht als Trennsymbol.
   - vor einem eröffnenden Pseudo-Text-Begrenzer, bei dem ein vorangehendes Leerzeichen unbedingt erforderlich ist.
   - nach dem Trennsymbol '"' (eröffnendes Anführungszeichen).

7. Das Trennsymbol ' ' (Leerzeichen/Blank) kann unmittelbar nach jedem Trennsymbol stehen, außer dem '"' (**eröffnenden** Anführungszeichen). In diesem Fall wird ein nachfolgendes Leerzeichen als Teil des nichtnumerischen Literals angesehen und nicht als Trennsymbol.

   Jedes Interpunktionszeichen, das als Teil der Spezifikation einer PICTURE-Zeichenfolge oder eines numerischen Literals erscheint, wird nicht als Interpunktionszeichen angesehen, sondern als ein Symbol, das in der Spezifikation dieser PICTURE-Zeichenfolge oder dieses numerischen Literals verwendet wird. PICTURE-Zeichenfolgen werden nur durch die Trennsymbole Leerzeichen, Komma, Strichpunkt oder Punkt begrenzt.

   Die für die Bildung von Trennsymbolen aufgestellten Regeln gelten nicht für die Zeichen, die den Inhalt eines nichtnumerischen Literals, Kommentareintragungen oder Kommentarzeilen umfassen.

Zeichenfolgen

Eine Zeichenfolge besteht aus einem einzelnen Zeichen oder einer Folge benachbarter Zeichen, die ein LII COBOL-Wort, ein Literal, eine PICTURE-Zeichenfolge oder eine Kommentareintragung bilden. Eine Zeichenfolge wird durch Trennsymbole begrenzt.

1.9.3 LII COBOL-Wörter

Ein COBOL-Wort besteht aus einer Zeichenfolge mit maximal 30 Zeichen, welche ein Programmiererwort, einen System-Namen oder ein reserviertes Wort bildet. Innerhalb eines vorgegebenen Quellprogramms bilden diese Klassen unabhängige Mengen; ein COBOL-Wort kann immer nur zu einer dieser Klassen gehören.

Programmiererwörter

Ein Programmiererwort ist ein COBOL-Wort, das vom Anwender entsprechend dem Format einer Klausel oder einer Anweisung angegeben werden muß. Jedes Zeichen eines Programmiererwortes wird aus dem folgenden Zeichenvorrat gebildet: A', 'B', 'C', ..., 'Z', 'a', 'b', 'c', ... 'z' (als Großbuchstaben interpretiert) '0', ... '9' und '-', (mit der Ausnahme, daß das Zeichen '-' nicht als erstes oder letztes Zeichen erscheinen darf). Die Ausnahme zu dieser Regel bildet ein physikalisches Dateiname-Literal, das ein normales alphanumerisches Literal sein muß.

Es gibt folgende Programmiererworttypen:

Alphabetname
CD-Name
Bedingungsname
Datenname
Externes Dateiname-Literal
Dateiname
Indexname
Stufennummer
Bibliotheksname
Merkname
Paragraphname
Programmname
Datensatzname
Routinename
SECTION-Name
Segmentnummer
Textname

Innerhalb eines vorgegebenen Quellprogramms werden 13 dieser 16 Programmiererworttypen zu 11 disjunkten Gruppen zusammengefaßt. Diese disjunkten Datengruppen sind:

Alphabetname
CD-Namen
Bedingungsname, Datenname und Datensatzname
Dateiname
Indexname
Bibliotheksname
Merkname
Paragraphname
Programmname
Routinename
SECTION-Name
Textname

Außer Segmentnummern und Stufennummern, können alle Programmiererwörter immer nur zu genau einer dieser disjunkten Gruppen gehören. Ferner müssen alle Programmiererwörter innerhalb einer bestehenden disjunkten Gruppe eindeutig sein (vgl.in diesem Kapitel).

Mit Ausnahme von Paragraphnamen, SECTION-Namen, Stufennummern und Segmentnummern müssen alle Programmiererwörter mindestens ein alphabetisches Zeichen enthalten. Segmentnummern und Stufennummern müssen nicht eindeutig sein; die Angabe einer Segmentnummer oder einer Stufennummer kann mit irgendeiner anderen Segmentnummer oder Stufennummer identisch sein und kann auch mit einem Paragraphnamen oder einem SECTION-Namen übereinstimmen.

Bedingungsname

Ein Bedingungsname ist ein Name, dem ein bestimmter Wert, eine Wertemenge oder ein Wertebereich innerhalb der vollständigen Wertemenge eines Datenfeldes zugeordnet ist. Das Datenfeld selbst wird Bedingungsvariable genannt.

Bedingungsnamen können in der DATA DIVISION oder im Paragraph 'SPECIAL-NAMES' in der ENVIRONMENT DIVISION definiert werden, wobei ein Bedingungsname entweder dem ON STATUS und/oder OFF STATUS des Ausführungszeit-Schalters zugeordnet ist.

Ein Bedingungsname wird nur in der RERUN-Klausel oder in Bedingungen als Abkürzung für die Vergleichsbedingung verwendet; diese Vergleichsbedingung setzt voraus, daß die zugehörige Bedingungsvariable einem Wert aus der Wertemenge entspricht, die diesem Bedingungsnamen zugeordnet ist.

Merkname

Ein Merkname ordnet ein Programmiererwort einem Herstellerwort zu. Diese werden im Paragraph 'SPECIAL-NAMES' in der ENVIRONMENT DIVISION behandelt, (vgl. SPECIAL-NAMES in Kapitel 2).

Paragraph-Name

Ein Paragraph-Name ist ein Wort, das einen Paragraphen in der PROCEDURE DIVISION (Prozedurteil) benennt. Paragraph-Namen sind genau dann gleichbedeutend, wenn sie aus der gleichen Anzahl von Ziffern und/oder Zeichen in der gleichen Folge bestehen.

SECTION-Name

Ein SECTION-Name ist ein Wort, das ein Kapitel in der PROCEDURE DIVISION benennt. SECTION-Namen sind genau dann gleichbedeutend, wenn sie aus der gleichen Anzahl von Ziffern und/oder Zeichen in der gleichen Folge bestehen.

Andere Programmiererworte: Alle anderen Programmiererworttypen sind im Kapitel 1.3.2 beschrieben.

System-Namen

Ein System-Name ist ein COBOL-Wort, das zur Kommunikation mit dem arbeitenden System verwendet wird. Jedes zur Bildung eines SystemNamens verwendete Zeichen muß aus dem Zeichenvorrat 'A' ... 'Z', 'a' ... 'z', '0' ... '9' und '-' (mit der Ausnahme, daß das Zeichen '-' nicht als erstes oder letztes Zeichen erscheinen darf) bestehen.

Es gibt drei Typen von System-Namen:

1. Computer-Name
2. Herstellerwort
3. Sprachen-Name

Innerhalb einer vorgegebenen Implementierung bilden diese drei Typen von System-Namen disjunkte Mengen; ein vorgegebener System-Name kann immer nur zu einer dieser Mengen gehören.

Reservierte Wörter

Ein reserviertes Wort ist ein COBOL-Wort aus einer bestimmten Liste von Wörtern, die in COBOL-Quellprogrammen verwendet werden können, die jedoch nicht in den Programmen als Programmiererworte oder System-Namen auftreten dürfen. Reservierte Wörter können nur wie in den allgemeinen Formaten angegeben verwendet werden (siehe 1.5).

Es gibt sechs Arten von reservierten Wörtern:

1. Schlüsselwörter
2. Wahlwörter
3. Verknüpfer
4. Sonderregister
5. figurative Konstanten
6. Sonderzeichen-Wörter

Schlüsselwörter

Ein Schlüsselwort ist ein Wort, in einem Format. In jedem Format werden diese Wörter groß geschrieben und unterstrichen.

Es gibt drei Arten von Schlüsselwörtern:

1. Verben, wie ADD, READ und ENTER.
2. Erforderliche Wörter, die in Anweisungs- und Eintragungsformaten erscheinen.
3. Wörter, die eine bestimmte funktionelle Bedeutung haben, wie NEGATIVE, SECTION, usw.

Wahlwörter

In jedem Format werden nicht unterstrichene Wörter in Großbuchstaben Wahlwörter genannt und können nach Bedarf des Anwenders verwendet werden. Die Verwendung oder Nichtverwendung eines Wahlwortes ändert die Semantik des COBOL-Programmes nicht, in dem es verwendet wird.

Verknüpfungen

Es gibt drei Arten von Verknüpfungen.

1. Kennzeichnerverknüpfungen, die dazu verwendet werden, einen Datennamen, einen Bedingungsnamen und einen Text-Namen oder einen Paragraph-Namen mit seinem Kennzeichner zu verbinden: OF und IN sind Beispiele für Kennzeichnerverknüpfungen
2. Serien von Verknüpfungen, die zwei oder mehrere aufeinanderfolgende Operanden verbinden: ',' (Trennsymbol Komma) oder ';' (Trennsymbol Strichpunkt).
3. Logische Verknüpfungen, die zur Bildung von Bedingungen verwendet werden: AND, OR.

Figurative Konstanten

Bestimmte reservierte Wörter sind bestimmten Werten zugeordnet. Diese reservierten Wörter sind im Abschnitt 'Figurative Konstante' in diesem Kapitel angegeben.

Literale

Ein Literal ist eine Zeichenfolge, deren Wert durch eine geordnete Menge von Zeichen, aus der das Literal besteht, oder durch Angabe eines reservierten Wortes, das sich auf eine figuratives Konstante bezieht, vorgegeben ist. Literale sind entweder nichtnumerisch oder numerisch.

Nichtnumerische Literale:

Ein nichtnumerisches Literal ist eine Zeichenfolge, die an beiden Enden durch Anführungszeichen begrenzt ist und die aus den zulässigen Zeichen des Computers besteht. Zulässig sind nichtnumerische Literale mit einer Länge von 1 bis 128 Zeichen.
Um ein einzelnes Anführungszeichen als nichtnumerisches Literal darstellen zu können, müssen zwei direkt hintereinanderstehende Anführungszeichen verwendet werden. Der Wert eines nichtnumerischen Literals im Zielprogramm ist die Zeichenfolge selbst, außer:

1. die begrenzenden Anführungszeichen sind ausgeschlossen;
2. jedes Paar direkt hintereinanderstehender Anführungszeichen stellt ein einzelnes Anführungszeichen dar.

Alle anderen Interpunktionszeichen sind Teil des Wertes des nichtnumerischen Literals und keine Trennsymbole; alle nichtnumerischen Literale gehören der Kategorie alphanumerisch an.
Zusätzlich können hexadezimale Werte nichtnumerischen Literalen zugeordnet werden, indem das Literal in folgender Form geschrieben wird: X"nn", wobei 'n' ein hexadezimales Zeichen aus der Menge 0-9 A-F ist; nn kann bis zu 128 mal wiederholt werden, die Anzahl der hexadezimalen Ziffern muß jedoch gerade sein.

Numerische Literale:

Ein numerisches Literal ist eine Zeichenfolge. Die Zeichen können sein:

- '0' bis '9'
- das Pluszeichen
- das Minuszeichen
- und/oder der Dezimalpunkt.

Die Implementierung erlaubt numerische Literale mit einer Länge von eins bis 18 Ziffern. Für die Bildung numerischer Literale gelten die folgenden Regeln:

1. Ein numerisches Literal muß mindestens eine Ziffer enthalten.
2. Ein numerisches Literal darf nicht mehr als ein Vorzeichen enthalten. Wenn ein Vorzeichen verwendet wird, muß es als das am weitesten links stehende Zeichen des Literals erscheinen. Ein Literal ohne Vorzeichen ist positiv.
3. Ein numerisches Literal darf nicht mehr als einen Dezimalpunkt enthalten. Der Dezimalpunkt wird als angenommener Dezimalpunkt behandelt und kann irgendwo innerhalb des Literals erscheinen, außer als das am weitesten rechts stehende Zeichen. Wenn das Literal keinen Dezimalpunkt enthält, ist es eine Ganzzahl (Integer-Größe).

   Wenn ein Literal den Regeln zur Bildung eines numerischen Literals entspricht, jedoch in Anführungszeichen steht, so ist es ein nichtnumerisches Literal und wird als solches vom Übersetzer behandelt.
4. Der Wert eines numerischen Literals ist gleich den Zeichen, die algebraisch im numerischen Literal dargestellt sind. Jedes numerische Literal gehört der Kategorie numerisch an.

   Die Größe eines numerischen Literals aus Datenzeichen im Standardformat ist identisch mit der Anzahl der vom Anwender angegebenen Ziffern.

Werte einer figurativen Konstante

Der Wert einer figurativen Konstante wird vom Übersetzer erzeugt und durch Verwendung der untengenannten reservierten Wörter referenziert. Diese Wörter dürfen nicht durch Anführungszeichen begrenzt werden, wenn sie als figurative Konstante verwendet werden. Einzahl und Mehrzahl der figurativen Konstanten sind äquivalent und können beliebig angegeben werden.

Die Werte der figurativen Konstanten und die reservierten Wörter, die zur Referenzierung verwendet werden, sind in Tabelle 1-1 aufgelistet:

Figurative Konstante

Literal	Darstellung
ZEROS ZEROES '0'	Stellt den Wert '0' oder ein oder mehrere Zeichen entsprechend dem Zusammenhang dar.
SPACE SPACES	Stellt ein oder mehrere Zeichen ' ' (Leerzeichen/Blank) aus dem Zeichenvorrat des Computers dar.
HIGH-VALUE HIGH-VALUES	Stellt ein oder mehrere Zeichen dar, die den höchsten Wert in der Programm-Sortierfolge haben (Hex 7F für den ASCII-Zeichenvorrat)
LOW-VALUE LOW-VALUES	Stellt ein oder mehrere Zeichen dar, die den niedrigsten Wert in der Programm-Sortierfolge haben.(Hex 00 für den ASCII-Zeichenvorrat)
QUOTE QUOTES	Stellt ein oder mehrere Zeichen '"' (Anführungszeichen) dar. Das Wort QUOTE oder QUOTES kann nicht anstelle eines Anführungszeichen in einem Quellprogramm verwendet werden, um nichtnumerische Literale zu verbinden. Daher ist QUOTE ABD QUOTE falsch, wenn das nichtnumerische Literal "ABD" dargestellt werden soll.
ALL Literal	Stellt ein oder mehrere Zeichen aus der Zeichenfolge, die das Literal umfaßt, dar. Das Literal muß entweder ein nichtnumerisches Literal oder eine figurative Konstante sein, jedoch kein Literal ALL. Wenn eine figurative Konstante verwendet wird, so ist das Wort ALL überflüssig und wird nur aus Gründen der Lesbarkeit verwendet.

Tabelle 1-1 Werte der konfigurativen Konstanten

Wenn eine figurative Konstante ein Zeichen oder eine Folge von mehreren Zeichen darstellt, wird die Länge der Zeichenfolge vom Übersetzer aus dem Zusammenhang entsprechend den folgenden Regeln bestimmt:

1. Wird eine figurative Konstante mit anderen Datenfeldern verknüpft, und wenn die figurative Konstante in ein Datenfeld gespeichert werden soll oder mit einem Datenfeld verglichen wird, dann geschieht Folgendes: Der Wert der figurativen Konstanten wird zeichenweise nach rechts wiederholt, bis die Länge der daraus resultierenden Zeichenfolge der Länge des korrespondierenden Datenfeldes gleicht. Ist die JUSTIFIED-Klausel angegeben, wird dieser Vorgang unabhängig davon ausgeführt. (Z.B. in einer DISPLAY-, STRING-, STOP- oder UNSTRING-Anweisung), so beträgt die Länge der Zeichenfolge ein Zeichen.

2. Wenn eine figurative Konstante nicht in Verbindung mit einem anderen Datenfeld verwendet wird, wenn also die figurative Konstante in einer DISPLAY-, STRING-, STOP- oder UNSTRING-Anweisung erscheint, so beträgt die Länge der Zeichenfolge ein Zeichen.

   Ausnahme:

   DISPLAY SPACE im Format 2 der DISPLAY-Anweisung.

   Eine figurative Konstante kann überall dort verwendet werden, wo ein Literal in einem Format erscheint. Ausnahme: Sobald das Literal nur aus numerischen Zeichen besteht, ist ZERO (ZEROS, ZEROES) die einzig zulässige figurative Konstante.

   Wenn die figurativen Konstanten HIGH-VALUE(S) oder LOW-VALUE(S) im Quellprogramm verwendet werden, hängt das tatsächliche Zeichen, das mit der entsprechenden figurativen Konstanten verknüpft ist, von der angegeben Programm-Sortierfolge ab, (vgl. OBJECT-COMPUTER-Paragraph und SPECIAL-NAMES-Paragraph in Kapitel 2)

   Jedes reservierte Wort, das sich auf eine figurative Konstante bezieht, ist eine eindeutig bestimmte Zeichenfolge, mit Ausnahme von 'ALL Literal', das aus zwei eindeutigen Zeichenfolgen besteht.

PICTURE-Zeichenfolgen

Eine PICTURE-Zeichenfolge verwendet bestimmte Symbole. Die Erläuterung der PICTURE-Zeichenfolgen und der Regeln, die ihre Verwendung bestimmen, sind aus dem Abschnitt Die PICTURE-Klausel in Kapitel 2 ersichtlich.

Alle Interpunktionszeichen, die als Teil einer PICTURE-Zeichenfolge erscheinen, werden nicht als Interpunktionszeichen betrachtet, sondern als ein Symbol, das in dieser PICTURE-Zeichenfolge verwendet wird.

Kommentareintragungen

Eine Kommentareintragung ist eine Eintragung in der IDENTIFICATION DIVISION und kann aus einer beliebigen Zeichenkombination aus dem Zeichenvorrat des Computers bestehen.

1.9.4 Konzept der maschinenunabhängigen Dateibeschreibung

Um Daten so maschinenunabhängig wie möglich machen zu können, werden die Eigenschaften oder Merkmale der Daten in einem Standard-Datenformat beschrieben und nicht in einem anlageorientierten Format. Dieses Standard-Datenformat ist auf allgemeine Datenverarbeitungszwecke ausgelegt und verwendet das Dezimalsystem zur Darstellung von Zahlen (ohne Berücksichtigung des vom Computer verwendeten Zahlensystems) und die anderen Zeichen im LII COBOL Zeichenvorrat zur Darstellung nichtnumerischer Datenfelder.

Stufenkonzept

Ein Stufenkonzept ist Bestandteil der Struktur eines logischen Datensatzes. Dieses Konzept ist entstanden, um auch auf Teile eines Datensatzes bezugnehmen zu können. Auch wenn eine Unterteilung detailliert angegeben wird, kann diese weiter unterteilt werden, um noch detailliertere Datenzugriffe zu ermöglichen.

Die kleinsten Unterteilungen eines Datensatzes, d. h., diejenigen, die nicht weiter unterteilt werden, werden 'Datenelemente' genannt; daher besteht ein Datensatz aus einer Folge von Datenelementen bzw. der Datensatz selbst kann aus einem einzigen Datenelement bestehen.

Um auf eine Menge Datenelemente bezugnehmen zu können, werden diese zu Gruppen zusammengefaßt. Jede Gruppe besteht aus einer mit einem Namen versehenen Folge von einem oder mehreren Datenelementen. Gruppen wiederum können zu größeren Gruppen mit zwei oder mehreren Gruppen zusammengefaßt werden usw. Daher kann ein Datenelement zu mehr als nur zu einer Gruppe gehören.

Beispiel

```
01  ERSTENS.
    02  ZWEITENS.
        03  DRITTENS-1.
        03  DRITTENS-2.
```

In diesem Beispiel gehören DRITTENS-1 und DRITTENS-2 einmal zur Gruppe ZWEITENS und außerdem zur Gruppe ERSTENS.

Stufennummern

Ein System von Stufennummern zeigt die Organisation von Datenelementen und Datengruppen auf. Da Datensätze die umfassendsten Datenfelder sind, beginnen Stufennummern für Datensätze bei 01. Weniger umfassenden Datenfeldern werden höhere (nicht unbedingt aufeinanderfolgende) Stufennummern zugeordnet, deren Wert 49 nicht übersteigen darf. In einem Datensatz sind maximal 49 Stufen zulässig. Es gibt die Sonderstufennummern 66, 77 und 88, die eine Ausnahme von dieser Regel darstellen (siehe unten). Für jede verwendete Stufennummer werden separate Eintragungen in das Quellprogramm geschrieben.

Eine Gruppe umfaßt alle ihr nachfolgenden Gruppen und Datenelemente bis zu der nächsten Stufennummer, die kleiner oder gleich der Stufennummer dieser Gruppe ist. Alle Datenfelder, die einer vorgegebenen Datengruppe unmittelbar untergeordnet sind, müssen mit identischen Stufennummern versehen werden, die größer sind, als die für die Beschreibung dieser Datengruppe verwendete Stufennummer. Es ist zu beachten, daß Datenelemente nicht länger als 8191 Byte sein dürfen. Datengruppen können größer definiert werden (z. B. als übergeordnete Datengruppen für große Tabellen); in solchen Fällen darf auf sie jedoch nicht direkt zugegriffen werden.

Beispiel

Es gibt drei Arten von Eintragungen, für die es kein wirkliches Stufenkonzept gibt:

1. Eintragungen, die Datenelemente oder Datengruppen spezifizieren, die durch eine RENAMES-Klausel eingeführt wurden.
2. Eintragungen, die nicht benachbarte Arbeitsspeicher- und Verbindungs-Datenfelder bezeichnen.
3. Eintragungen, die Bedingungsnamen spezifizieren.

Die Sonderstufennummer 66 wird den Eintragungen zugeordnet, die Datenfelder mit Hilfe der RENAMES-Klauseln zur Umbenennung von Datenfeldern beschreiben.

Die Sonderstufennummer 77 wird den Eintragungen zugeordnet, die nicht benachbarte Datenfelder spezifizieren, die nicht Unterteilungen anderer Datenfelder sind und die selbst nicht unterteilt sind.

Die Sonderstufennummer 88 wird den Eintragungen zugeordnet, die Bedingungsnamen spezifizieren, die mit bestimmten Werten einer Bedingungsvariablen verknüpft werden sollen.

Konzept der Datenklassen

Die fünf Kategorien von Datenfeldern (vgl. Die PICTURE-Klausel in Kapitel 2) werden in drei Klassen zusammengefaßt: alphabetisch, numerisch und alphanumerisch. Die Klassen und Kategorien der alphabetischen und numerischen Datengruppe sind gleichbedeutend. Die alphanumerische Klasse umfaßt die Kategorien alphanumerisch aufbereitet, numerisch aufbereitet und alphanumerisch (ohne Aufbereitung). Jedes Datenelement, mit Ausnahme der Index-Datenfelder, gehört zu einer der Klassen und auch zu einer der Kategorien. Die Klasse einer Datengruppe wird zur Ausführungszeit als alphanumerisch angenommen, ohne Berücksichtigung der Klasse der Datenelemente, die dieser Datengruppe untergeordnet ist. Die folgende Aufstellung veranschaulicht die Beziehung zwischen Klassen und Kategorien der Datenfelder.

Stufen von Datenfeldern	Klasse	Kategorie
Elementare	alphabetisch	alphabetisch
	numerisch	numerisch numerisch druckaufbereitet
	alphanumerisch	alphanumerisch druckaufbereitet
nicht elementare Gruppe	alphanumerisch	alphabetisch numerisch numerisch druckaufbereitet alphanumerisch druckaufbereitet alphanumerisch

Auswahl der Zeichendarstellung und des Zahlensystems

Der Wert eines numerischen Datenfeldes kann entsprechend der verwendeten Anlage binär oder dezimal dargestellt werden. Zusätzlich gibt es verschiedene Möglichkeiten, die dezimale Form darzustellen. Da diese Darstellungen in Wirklichkeit Kombinationen von Bits sind, werden sie allgemein BCD-Formen (binary-coded decimal) genannt. Für das Abspeichern numerischer Daten unter LII COBOL werden die vier folgenden Standard-Formate verwendet:

1. Gespeichert als alphanumerische Zeichen; jeweils ein Zeichen pro Byte in der ASCII Darstellung.

2. Gespeichert als numerische Zeichen, wie in USAGE IS DISPLAY (vgl. USAGE-Klausel in Kapitel 2) definiert; ein Zeichen pro Byte in der ASCII Darstellung. Wenn sie mit einem Vorzeichen versehen sind und das Vorzeichen als INCLUDED angegeben ist, wird Bit 6 des führenden oder letzten Bytes des Feldes gesetzt, wenn die Zahl negativ ist, in Abhängigkeit von der Felddefinition. Wenn das Vorzeichen als SEPARATE angegeben ist (ASCII + oder - Zeichen), so wird ein Vorzeichen als führendes oder letztes Byte hinzugefügt. Wenn keine SIGN Klausel angegeben wird, so wird standardmäßig Bit 6 der letzten Ziffer gesetzt, um einen negativen Wert anzuzeigen.

3. Gespeichert als numerische Zeichen, wie in USAGE IS COMP oder COMPUTATIONAL definiert in rein binärer Form. Wenn das Feld mit einem Vorzeichen versehen ist, so wird die Zahl im Zweier -Komplement gespeichert. Der Speicherplatzbedarf hängt dann von der Anzahl der 9en in der PICTURE-Klausel ab (vgl. Die PICTURE-Klausel in Kapitel 2) und davon, ob das Feld mit einem Vorzeichen versehen ist oder nicht (vgl. Die SIGN-Klausel in Kapitel 2).

Die nächste Tabelle veranschaulicht den Speicherplatzbedarf für jede COMP(UTATIONAL) PICTURE-Klausel.

Erforderliche Anzahl Bytes	Anzahl der Zeichen	
	mit Vorzeichen	ohne Vorzeichen
1	1–2	1–2
2	3–4	3–4
3	5–6	5–7
4	7–9	8–9
5	10–11	10–12
6	12–14	13–14
7	15–16	15–16
8	17–18	17–18

Tabelle 1-2 Speicherplatz

4. Gespeichert als numerische Zeichen, wie in USAGE IS COMPUTATIONAL-3 oder USAGE IS COMP-3 definiert, in gepackter interner dezimaler Form. Die Speicherung hängt ab von der Anzahl der 9en in der PICTURE-Klausel. Die Dezimalzahlen werden als Zeichenfolge mit Vorzeichen mit einer variablen Länge von 1 bis 18 Ziffern gespeichert. Das Vorzeichen einer gepackten Dezimalzahl wird immer anstelle der vier rechten Bits des niederwertigsten Bytes gespeichert. Jedes Byte enthält zwei Dezimalstellen (vier Bits pro Ziffer) und die Ziffern (0-9) sind als BCD-Zahlen (0000-1001) verschlüsselt. Zahlen werden im Feld rechtsbündig mit einem + oder Vorzeichen, wie im nächsten Beispiel gezeigt, dargestellt. Die in arithmetischen Operanden maximal zulässige Anzahl von Ziffern ist 18.

Beispiel

Bei COMPUTATIONAL-3 und PICTURE 9999 würde die Zahl +1234 wie folgt gespeichert werden:

0	1	2	3	4	F
0000	0001	0010	0011	0100	1111

1 Byte

'F' stellt das nicht abdruckbare positive Vorzeichen dar.

Beispiel

Bei COMPUTATIONAL-3 und PICTURE S9999 würde die Zahl +1234 wie folgt gespeichert:

Speicherung würde bis auf das rechte Halbbyte wie im obigen Beispiel erfolgen, das rechte Halbbyte ist ein C, die Aussage für einen positiven Wert.

0	1	2	3	4	C
0000	0001	0010	0011	0100	1100

wie im Beispiel vorher, nur steht rechts das D, die Aussage für einen negativen Wert.

0	1	2	3	4	D
0000	0001	0010	0011	0100	1101

Beispiel

Bei COMPUTATIONAL-3 und PICTURE S9999 würde die Zahl -1234 wie folgt gespeichert:

Speicherung würde wie im ersten Beispiel oben erfolgen, mit der Ausnahme, daß das rechte Byte durch D (1101), das Minuszeichen, ersetzt wird.

Numerische Datenspeicherung für die COMPUTATIONAL-3 PICTURE-Klausel.

Erforderliche Anzahl Bytes	Anzahl der Ziffern (mit bzw. ohne Vorzeichen)
1	1
2	2- 3
3	4- 5
4	6- 7
5	8- 9
6	10-11
7	12-13
8	14-15
9	16-17
10	18

Algebraische Vorzeichen

Algebraische Vorzeichen werden in zwei Kategorien aufgeteilt:

- Rechenvorzeichen, die Teil von numerischen Datenfeldern oder numerischen Literalen sind,
- druckaufbereitete Vorzeichen, die beim Listendruck das Vorzeichen eines Datenfeldes anzeigen.

Die SIGN-Klausel ermöglicht es dem Programmierer, die Position des Rechenvorzeichens ausdrücklich festzulegen. Die Klausel ist optional; wenn sie nicht verwendet wird, werden Rechenvorzeichen dadurch dargestellt, daß das Bit 6 der letzten Ziffer bei ASCII-Zahlen gesetzt wird (siehe oben).

Druckaufbereitete Vorzeichen werden in ein Datenfeld durch die Vorzeichensteuersymbole der PICTURE-Klausel eingefügt.

Standardregeln für die Ausrichtung

Die Standardregeln für die Ausrichtung von Daten innerhalb eines Datenelements hängen von der Art der Empfangsdatenfelder ab:

1. Wenn das Empfangsdatenfeld als numerisch definiert ist:
   a) werden die Daten nach dem Dezimalpunkt ausgerichtet und in die Empfangszeichenposition durch Auffüllen mit Nullen oder Abschneiden je nach Vorgabe eingetragen.
   b) wenn ein angenommener Dezimalpunkt nicht ausdrücklich angegeben ist, wird das Datenfeld so behandelt, als hätte es einen angenommenen Dezimalpunkt direkt nach dem am weitesten rechts stehenden Zeichen und wird wie in a) beschrieben ausgerichtet.
2. Wenn das Empfangsdatenfeld ein numerisch druckaufbereitetes Datenfeld ist, werden die Daten, die in dieses aufbereitete Datenfeld übertragen werden, am Dezimalpunkt durch Auffüllen mit Nullen oder Abschneiden innerhalb der Empfangszeichenposition des Datenfeldes ausgerichtet.

Ausnahme:

Die Fälle in denen aus Druckaufbereitungsgründen der Ersatz der führenden Nullen erzwungen wird.

3. Wenn das Empfangsdatenfeld alphanumerisch (allerdings kein numerisch druckaufbereitetes Datenfeld), alphanumerisch drukkaufbereitet oder alphabetisch ist, so werden die Sendedaten in die Empfangszeichenposition übertragen auf die am weitesten links stehende Zeichenposition im Datenfeld ausgerichtet und nach rechts mit Leerzeichen aufgefüllt oder abgeschnitten, wie vorgegeben.

   Wenn für das Empfangsdatenfeld die JUSTIFIED-Klausel angegeben ist, so werden diese Standardregeln so abgeändert, wie in der JUSTIFIED-Klausel in Kapitel 2 beschrieben.

1.9.5 Eindeutigkeit der Referenz

Kennzeichnung

Jeder vom Anwender angegebene Name, der ein Element in einem COBOL-Quellprogramm definiert, muß eindeutig sein. Das heißt: Es darf kein zweiter gleicher Name vorkommen oder es muß durch Angaben einer höheren Namenshierarchie eine Eindeutigkeit hergestellt werden.
Die höheren Stufen werden 'Kennzeichner' genannt, und der Vorgang, durch den die Eindeutigkeit erreicht wird, 'Kennzeichnung'.

Beispiel

```
01 GRUPPE-A.
   02 GRUPPE1.
      03 NAME    PIC X(10).
      03 TEL     PIC 9(6).

01 GRUPPE-B.
   02 GRUPPE2.
      03 NAME    PIC X(10).
      03 TEL     PIC 9(6).

PROCEDURE DIVISION.

  MOVE NAME OF GRUPPE1 TO NAME OF GRUPPE2.
```

Es müssen genügend Kennzeichner angegeben werden, damit der Name eindeutig wird; es besteht jedoch nicht die Notwendigkeit, alle Stufen der Hierarchie anzugeben. Innerhalb der DATA DIVISION müssen alle Datennamen, die für die Kennzeichnung verwendet werden, mit einer Stufenbezeichnung oder einer Stufennummer verknüpft sein. Daher dürfen zwei identische Datennamen nicht unter einer Datengruppe erscheinen, sofern sie nicht durch Kennzeichnung eindeutig gemacht werden können.

In der PROCEDURE DIVISION dürfen zwei identische Paragraphennamen nicht in der gleichen SECTION erscheinen.

In der Hierarchie der Kennzeichnung sind die Namen, die mit einer Stufenbezeichnung verknüpft sind, die signifikantesten; dann folgen die Namen, die mit der Stufennummer 01 verknüpft sind, dann die Namen, die mit der Stufennummer 02, ... , 49 verknüpft sind. Ein SECTION-Name ist der höchste (und der einzige) Kennzeichner, der für einen Paragraphennamen zur Verfügung steht. Daher muß der signifikanteste Name in der Hierarchie eindeutig sein und kann nicht gekennzeichnet werden. Subskribierte oder indizierte Datennamen und Bedingungsvariable sowie Prozedur-Namen und Datennamen können durch Kennzeichnung eindeutig gemacht werden. Der Name einer Bedingungsvariablen kann als Kennzeichner für alle seine Bedingungsnamen verwendet werden. Ohne Berücksichtigung der verfügbaren Kennzeichnung kann kein Name Datenname und gleichzeitig Prozedur-Name sein.

Die Kennzeichnung findet dadurch statt, daß nach einem Datennamen, einem Bedingungsnamen, einem Paragraph-Namen oder einem Text-Namen eine oder mehrere Angaben folgen, die aus einem Kennzeichner bestehen, dem IN oder OF vorangestellt ist. IN und OF sind logisch gleichbedeutend.

Die allgemeinen Formate für Kennzeichnung sind die folgenden:

Format 1

```
{datenname-1      }  [ [{OF}] datenname-2 ] ...
{bedingungsname   }     {IN}
```

Format 2

```
paragraphname [ [{OF}] sectionname ]
              {IN}
```

Format 3

```
textname [ [{OF}] bibliotheksname ]
         {IN}
```

Kennzeichnung

Für die Kennzeichnung gelten die folgenden Regeln:

1. Jeder Kennzeichner muß auf jeweils höherer Stufe, aber innerhalb der gleichen Hierarchie stehen wie der Name, den er kennzeichnet.
2. Der gleiche Name darf nicht auf zwei verschiedenen Stufen einer Hierarchie erscheinen.
3. Wenn ein Datenname oder ein Bedingungsname mehr als nur einem Datenfeld in einem Quellprogramm zugeordnet ist, muß dieser Datenname oder Bedingungsname jedesmal gekennzeichnet werden, wenn in der PRODEDURE, ENVIRONMENT und DATA DIVISION darauf Bezug genommen wird, (mit Ausnahme in der REDEFINES-Klausel, in der Kennzeichnung unnötig ist und nicht verwendet werden darf).
4. Ein Paragraph-Name darf innerhalb einer SECTION nicht doppelt auftreten. Wenn ein Paragraph-Name durch einen SECTION-Namen gekennzeichnet wird, darf das Wort 'SECTION' nicht erscheinen. Ein Paragraph-Name braucht nicht gekennzeichnet zu werden, wenn er aus der gleichen SECTION heraus angesprochen wird.
5. Ein Datenname kann nicht subskribiert werden, wenn er als Kennzeichner verwendet wird.
6. Ein Name kann gekennzeichnet werden, obwohl er keine Kennzeichnung benötigt. Wenn es mehr als eine Kombination von Kennzeichnern gibt, die die Eindeutigkeit sicherstellen, dann kann eine beliebige dieser Kombinationen verwendet werden. Die Gesamtmenge aller Kennzeichner für einen Datennamen darf nicht gleichzeitig Untermenge der Kennzeichner eines anderen Datennamens sein.

   Gekennzeichnete Datennamen dürfen bis zu fünf Kennzeichner besitzen.
7. Wenn mehr als eine COBOL Bibliothek für den Übersetzer während des Übersetzungsvorganges zur Verfügung steht, muß ein Textname jedesmal, wenn er aufgeführt wird, gekennzeichnet werden.

Subskribierung

Subskripte können nur verwendet werden, wenn man sich auf ein einzelnes Element einer Liste oder Tabelle gleicher Elemente bezieht, denen keine individuellen Datennamen zugeordnet sind.(vgl. Die OCCURS-Klausel in Kapitel 3).

Das Subskript kann entweder durch ein ganzzahliges numerisches Literal oder durch ein ganzzahliges numerisches Datenelement dargestellt werden.

Das Subskript kann mit einem Vorzeichen versehen werden, das positiv sein muß. Der niedrigste Subskriptwert ist 1. Dieser Wert weist auf das erste Element der Tabelle. Die nächsten sequentiellen Elemente der Tabelle werden durch die Subskripte 2, 3, ... angegeben. Der höchste Subskriptwert ist die höchste Tabellenelementnummer des Datenfeldes, wie es in der OCCURS-Klausel festgelegt ist.

Die relative Subskribierung kann in ähnlicher Weise verwendet werden, wie die relative Indizierung, wenn der ANSI-Schalter nicht gesetzt ist.

Das Subskript oder die Menge von Subskripten, die das Tabellenelement identifiziert, wird durch das Trennsymbolpaar linke Klammer und rechte Klammer nach dem Tabellenelement-Datennamen eingerahmt. Ein Tabellenelement-Datenname, dem ein Subskript folgt, wird 'subskribierter Datenname' oder 'bezeichner' genannt!
Eine Tabelle kann auch mehrdimensional sein, dann müssen mehrere Subskripte geschrieben werden, die umfassenderen Subskripte werden jeweils **vor** den weniger umfassenden Subskripten geschrieben. In der Praxis gibt es eine Grenze für die Anzahl der Subskripte, die mit einem Datennamen verwendet werden können. Die maximale Anzahl ist 9. Wenn sie überschritten wird, gibt es einen internen Pufferüberlauf.

Das Format ist:

```
{datenname      }
{               }(subskript-1[ ,subskript-2[ ,subskript-3]...])
{bedingungsname }
```

Indizierung

Innerhalb einer Tabelle mit gleichen Elementen können einzelne Elemente durch eine Indizierung angesprochen werden. Dieser Tabellenstufe wird ein Index zugeordnet, in dem die Angabe INDEXED BY in der Definition der Tabelle verwendet wird.
Ein in der INDEXED BY-Angabe angegebener Name wird Indexname genannt und wird dazu verwendet, den ihm zugeordneten Index anzusprechen.
Der Wert eines Index entspricht der Tabellenelementnummer eines Elementes in der entsprechenden Tabelle oder in irgendeiner anderen Tabelle. Ein Indexname muß initialisiert werden, bevor er als Tabellenreferenz verwendet werden kann. Dem Indexnamen kann durch eine SET-Anweisung ein Anfangswert zugeordnet werden.

Direkte Indizierung verwendet einen Indexnamen in Form eines Subskripts. Relative Indizierung wird erreicht, wenn nach dem Indexnamen der Operator + oder folgt und danach ein ganzzahliges numerisches Literal ohne Vorzeichen (alles begrenzt durch das Trennsymbolpaar linke Klammer und rechte Klammer) nach dem Tabellenelement-Datennamen.
Die aus der relativen Indizierung hervorgehende Tabellenelementnummer wird errechnet durch Erhöhung (wenn der Operator + verwendet wird) oder Verringerung (wenn der Operator - verwendet wird) des Indexwertes um den Wert des Literals. Wenn mehr als ein Indexname erforderlich ist, so wird der umfassendere jeweils vor dem weniger umfassenden geschrieben.

Bei der Ausführung einer Anweisung, die sich auf ein indiziertes Tabellenelement bezieht, darf der Wert des Index weder kleiner als eins noch größer als die Tabellenelementnummer des letzten Elementes in der entsprechenden Tabelle werden. Diese Beschränkung gilt auch für den Wert, der aus der relativen Indizierung resultiert. In der Praxis gibt es eine Grenze für die Anzahl der Indexnamen, die zusammen mit einem Datennamen verwendet werden können. Die maximale Anzahl ist 9. Wenn sie überschritten wird, gibt es einen Laufzeitfehler 'interner Puferüberlauf'.

Das allgemeine Format für die Indizierung ist:

```
{datenname      } ({indexname-1  [{+}literal-2]  }
{bedingungsname}  {literal-1     [{-}          ]  }

    [ ,{indexname-2  [{+}literal-4]  } [ ,{indexname-3  [{+}literal-6]  }  ] ] )
       {literal-3    [{-}           ]  }   {literal-5    [{-}           }
```

Bezeichner

Ein Bezeichner ist ein Begriff, der einen nicht eindeutigen Dateinamen durch eine syntaktisch korrekte Kombination von Kennzeichnern, Subskripten oder Indizes eindeutig bestimmt.

Die allgemeinen Formate für Bezeichner sind:

Format 1

```
datenname-1 [ {OF} datenname-2 ] ... [(subskript-1[,subskript-2[,subskript-3]])]
              {IN}
```

Format 2

```
datenname-1 [ {OF} datenname-2 ] ... [ ({indexname-1  [{+}literal-2]  }
              {IN}                     {literal-1     [{-}          ]  }

    [ ,{indexname-2  [{+}literal-4]  } [ ,{indexname-3  [{+}literal-6]  }  ] ] ) ]
       {literal-3    [{-}           ]  }   {literal-5    [{-}           }
```

Es gibt folgende Beschränkungen für die Subskribierung und Indizierung:

1. Wenn dieser datenname als Index oder Subskript verwendet wird, darf er selbst nicht subskribiert oder indiziert werden.
2. Indizierung ist dort nicht erlaubt, wo Subskribierung auch nicht zulässig ist.
3. Ein Index darf nur durch die Anweisungen SET, SEARCH und PERFORM verändert werden.
4. literal-1, literal-3 und literal-5 im obengenannten Format müssen positive, numerische Ganzzahlen sein. literal-2, literal-4 und literal-6 müssen numerische Ganzzahlen ohne Vorzeichen sein.

Bedingungsnamen

Jeder Bedingungsname muß eindeutig sein oder durch Kennzeichnung und/oder Indizierung oder Subskribierung eindeutig gemacht werden. Wenn Kennzeichnung verwendet wird, um einen Bedingungsnamen eindeutig zu machen:

- kann die verknüpfte Bedingungsvariable als der erste Kennzeichner verwendet werden
- muß die Namenshierarchie, die mit der Bedingungsvariablen verknüpft ist, oder die Bedingungsvariable selbst verwendet werden.

Wenn die Auswahl einer Bedingungsvariablen Indizierung oder Subskribierung erforderlich macht, so müssen Referenzen zu irgendeinem ihrer Bedingungsnamen die gleiche Kombination von Indizierung oder Subskribierung erhalten.

Das Format und die Beschränkungen für die kombinierte Verwendung der Kennzeichnung, der Subskribierung und Indizierung von Bedingungsnamen entspricht genau denen für Bezeichner, mit der Ausnahme, daß datenname-1 durch bedingungsname-1 ersetzt wird.

In den allgemeinen Formaten ist mit Bedingungsname ein gekennzeichneter, indizierter oder subskribierter Bedingungsname gemeint, je nach Erfordernis.

Explizite und Implizite Spezifikationen

Es gibt drei Arten expliziter und impliziter Angaben, die in einem COBOL-Quellprogramm auftreten:

1. Explizite und implizite PROCEDURE DIVISION Referenzen.
2. Explizite und implizite Steuerungen des Programmablaufs.
3. Explizite und implizite Attribute.

Explizite und implizite PROCEDURE DIVISION Referenzen

Ein COBOL-Quellprogramm kann in Anweisungen in einer PROCEDURE DIVISION Datenfelder entweder explizit oder implizit ansprechen.

Explizit geschieht dies:

- der Name des Datenfeldes wurde in der Anweisung der PROCEDURE DIVISION geschrieben
- der Name des Datenfeldes wurde durch eine Anweisung in die PROCEDURE DIVISION kopiert.

Implizit geschieht dies:

- Ein Datenfeld wird in einer Anweisung der PROCEDURE DIVISION nicht mit seinem Namen angesprochen.
- bei einer PERFORM-Anweisung wird derjenige Index oder das Datenfeld initialisiert, geändert oder ausgewertet, der durch die Angaben VARYING, AFTER oder UNTIL mit dem Indexnamen oder Bezeichner in der PERFORM-Anweisung verknüpft wurde. Eine solche implizite Referenz tritt nur dann auf, wenn das Datenfeld an der Ausführung der Anweisungen mitwirkt.

Explizite und implizite Programmablaufsteuerung

Implizite Programmablaufsteuerung

Die Programmsteuerung gibt die Steuerung automatisch von Anweisung zu Anweisung in der Folge, wie sie im Quellprogramm geschrieben wurden, sofern eine explizite Programmablaufsteuerung diese Folge nicht unterbricht, oder bis keine nächste ausführbare Anweisung mehr vorhanden ist.
Die Programmablaufsteuerung von Anweisung zu Anweisung erfolgt ohne eine explizite Anweisung in der PROCEDURE DIVISION und ist daher eine **implizite** Programmablaufsteuerung.

Unter COBOL sind sowohl explizite als auch implizite Mittel für die Änderung der impliziten Programmablaufsteuerung verfügbar.

Zusätzlich zur impliziten Programmablaufsteuerung zwischen zwei aufeinanderfolgenden Anweisungen erfolgt auch dann ein impliziter Programmablauf, wenn der normale Ablauf ohne die Ausführung einer Prozedurverzweigungsanweisung geändert wird. Unter LII COBOL sind die folgenden Arten der impliziten Programmablaufänderungen möglich, die dem Programmablauf von Anweisung zu Anweisung übergeordnet sind:

1. Wenn ein Paragraph unter der Steuerung einer anderen LII COBOL-Anweisung ausgeführt wird (z.B. PERFORM, USE, SORT und MERGE) und wenn der Paragraph der letzte im angegebenen Bereich ist, dann erfolgt ein implizierter Programmablauf von der letzten Anweisung im Paragraphen an den Steuerungsmechanismus der zuletzt ausgeführten übergeordneten Anweisung.

   Ferner, wenn ein Paragraph unter der Steuerung einer PERFORM-Anweisung ausgeführt wird und wenn dieser Paragraph der erste in einer Reihe von Paragraphen in dieser PERFORM-Anweisung ist, erfolgt ein impliziter Programmablauf zwischen dem Steuerungsmechanismus, der mit dieser PERFORM-Anweisung verknüpft ist, und der ersten Anweisung in diesem Paragraphen für jede schrittweise Ausführung des Paragraphen.

2. Wenn eine SORT- oder MERGE-Anweisung ausgeführt wird, so erfolgt ein impliziter Programmablauf für jede auftretende Eingabe- oder Ausgabe-Prozedur.

3. Wenn irgendeine LII COBOL-Anweisung ausgeführt wird, die im Vereinbarungskapitel angegeben ist, geht die Steuerung implizit an das Vereinbarungskapitel über. Es ist zu beachten, daß nach Ausführung der Vereinbarungskapitel ein weiterer impliziter Programmablauf stattfindet, wie unter Punkt 1. oben beschrieben.

4. Bei jedem Dateizugriff (einschließlich OPEN und CLOSE) wird einer Datei eine implizite USE-Anweisung zugeordnet, wenn für diese Datei kein FILE STATUS Datenfeld vereinbart wurde und der Datei nicht explizit eine USE-Anweisung zugeordnet wurde. Die implizite USE-Prozedur ist äquivalent zu:

```
USE AFTER ERROR PROCEDURE ON dateiname.
IF statusschlüssel-1 = 9
   DISPLAY fehlermeldung UPON CONSOLE
   STOP RUN.
```

wobei 'fehlermeldung' in der LII COBOL Bedienungsanleitung angegeben ist.

Explizite Programmablaufsteuerung

Eine explizite Steuerung des Programmablaufs besteht aus einer Änderung des impliziten Programmablaufs durch die Ausführung einer Prozedurverzweigung oder einer Bedingungsanweisung. Eine explizite Programmablaufsteuerung kann nur durch die Ausführung einer Prozedurverzweigung oder einer Bedingungsanweisung verursacht werden. Die Ausführung der Prozedurverzweigungsanweisung ALTER selbst stellt keine explizite Programmablaufsteuerung dar, beeinflußt jedoch die explizite Programmablaufsteuerung. Diese explizite Programmablaufsteuerung setzt ein, wenn die verknüpfte GO TO-Anweisung ausgeführt wird. Die Prozedurverzweigungsanweisung EXIT PROGRAM führt zu einer expliziten Programmablaufsteuerung, wenn die Anweisung in einem aufgerufenen Programm ausgeführt wird.

In diesem Handbuch wird der Begriff 'nächste ausführbare Anweisung' verwendet, um die nächste COBOL-Anweisung zu bezeichnen, an die die Steuerung gemäß den obigen Regeln und den mit jedem Sprachelement in der PROCEDURE DIVISION verknüpften Regeln übergeben wird.

Keine nächste ausführbare Anweisung steht nach:

1. der letzten Anweisung in einem Vereinbarungskapitel, wenn der Paragraph, in dem sie erscheint, nicht unter der Steuerung irgendeiner anderen COBOL-Anweisung ausgeführt wird,
2. der letzten Anweisung in einem Programm, wenn der Paragraph, in dem sie erscheint, nicht unter der Steuerung irgendeiner anderen COBOL-Anweisung ausgeführt wird.

Explizite und implizite Attribute

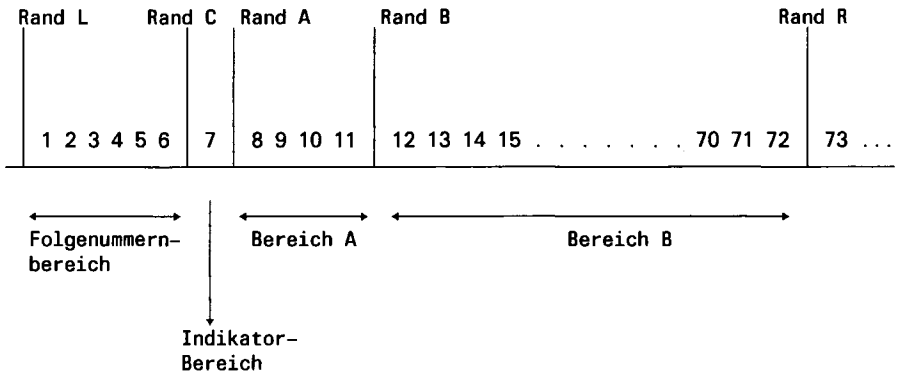
Attribute können implizit oder explizit angegeben werden. Jedes Attribut, das explizit angegeben wird, wird explizites Attribut genannt. Wenn ein Attribut nicht explizit angegeben wird, dann nimmt das Attribut den Standardwert an. Solch ein Attribut wird ein implizites Attribut genannt.

Der Verwendungszweck eines Datenfeldes beispielsweise muß nicht angegeben werden; in diesem Fall wird als Verwendungszweck des Datenfeldes DISPLAY angenommen.

1.9.6 Referenzformat

Das Referenzformat, das eine Standardmethode für die Beschreibung von COBOL-Quellprogrammen bietet, wird anhand von Zeichenpositionen in einer Zeile auf einem Eingabe-/Ausgabe-Gerät beschrieben. Der LII COBOL-Übersetzer akzeptiert Quellprogramme, die im Referenzformat geschrieben sind, und produziert ein Ausgabeprotokoll der Quellprogrammeingabe im Referenzformat (ein Beispiel eines Quellprogramms wird in der LII COBOL Bedienungsanleitung beschrieben).

Das Referenzformat für eine Zeile sieht folgendermaßen aus:



Der Rand L befindet sich links neben der am weitesten links stehenden Zeichenpostion einer Zeile.

Der Rand C befindet sich zwischen der 6. und 7. Zeichenposition einer Zeile.

Der Rand A befindet sich zwischen der 7. und 8. Zeichenposition einer Zeile.

Der Rand B liegt zwischen der 11. und 12. Zeichenposition einer Zeile.

Der Rand R liegt unmittelbar rechts neben der am weitesten rechts stehenden Zeichenposition einer Zeile.

Der Folgenummern-Bereich belegt sechs Zeichenpositionen (1-6) und liegt zwischen Rand L und Rand C.

Eine aufsteigende Reihenfolge der Zahlen wird bei LII COBOL nicht geprüft.

Der Indikatorbereich liegt auf der 7. Zeichenposition einer Zeile.

Bereich A belegt die Zeichenpositionen 8, 9, 10 und 11 und liegt zwischen Rand A und Rand B.

Der Bereich B belegt die Zeichenpositionen 12 bis 72 einschließlich; er beginnt unmittelbar rechts von Rand B und endet unmittelbar links von Rand R.

Fortsetzung von Zeilen

Benötigt ein Satz, eine Eintragung, eine Angabe oder eine Klausel mehr als eine Zeile, kann diese durch eine oder mehrere Folgenzeilen fortgesetzt werden. Diese Folgezeilen werden 'Fortsetzungszeilen' genannt. Die Zeile, die fortgesetzt wird, wird 'fortgesetzte Zeile' genannt. Alle Wörter oder Literale können so getrennt werden, daß ein Teil von ihnen auf einer Fortsetzungszeile erscheint.

Ein Bindestrich im Indikatorbereich einer Zeile weist darauf hin, daß das erste Zeichen ungleich Blank in Bereich B der aktuellen Zeile das Folgezeichen des letzten Zeichens ungleich Blank auf der vorangehenden Zeile ist, ohne daß Leerzeichen eingefügt werden.

Beispiel

```
7A   B       (Spalte/Bereich)
-----|-|-------------------------------------------------|
     | |                                                 |

 01  Feld PIC X(40) val
-      ue "Text          ".
```

Wenn jedoch eine fortgesetzte Zeile ein nichtnumerisches Literal ohne abschließendes Anführungszeichen enthält, so muß das erste Zeichen ungleich Blank in Bereich B der Fortsetzungszeile, ein Anführungszeichen sein und die Fortsetzung beginnt mit dem Zeichen, das unmittelbar nach dem Anführungszeichen steht.

Alle Leerzeichen am Ende der fortgesetzten Zeile werden als Teil des Literals aufgefaßt. Bereich A einer Fortsetzungszeile muß leer bleiben.

Beispiel

```
7A   B       (Spalte / Bereich)            Spalte72
-----|-|-------------------------------------------|
     | |                                           |

 01  Feld PIC X(40) value "  Die Zeilenfortsetzung
-      "ist hier".
```

Wenn kein Bindestrich im Indikatorbereich einer Zeile steht, so wird angenommen, daß dem letzten Zeichen in der vorangehenden Zeile ein Leerzeichen folgt.

Leere Zeilen

Eine leere Zeile ist eine Zeile, die von Rand C bis Rand R einschließlich leer ist. Eine leere Zeile kann außer unmittelbar vor einer Fortsetzungszeile überall im Quellprogramm erscheinen.

Pseudo-Text

Die Zeichenfolgen und Trennsymbole eines Pseudo-Textes können entweder in Bereich A oder Bereich B beginnen. Wenn jedoch ein Bindestrich im Indikatorbereich einer Zeile steht, die einem eröffnenden Pseudo-Text-Begrenzer folgt, so muß Bereich A der Zeile leer bleiben und die normalen Regeln für die Fortsetzung von Zeilen gelten für die Bildung von Textwörtern (vgl. Kapitel 10, Bibliothek).

1.10 Die Differenzen zwischen LII COBOL und COB1 / Erweiterungen zu ANSI COBOL

Die aufgeführten Differenzen beziehen sich auf den COB1-Stand I.84. Aktuelle Informationen und damit entstehende Differenzen zu LII COBOL entnehmen Sie bitte den entsprechenden Freigabemitteilungen.

1.10.1 Allgemeines

LII COBOL ist für Anwender geschrieben, die mit Mikro-Computern arbeiten und einen Bildschirm als Arbeitsplatz verwenden.

LII COBOL bietet daher Erweiterungen für interaktives Arbeiten, Programmsteuerung von Dateien, Textdateibearbeitung und schnelle Entwicklung und Prüfung. Diese Einrichtungen werden ausführlich in der LII COBOL Bedienungsanleitung beschrieben.

Für die ACCEPT-Anweisung gibt es ein zusätzliches Format:

```
ACCEPT datenname-1 [ AT { datenname-2 } ] FROM CRT
                       { literal-1   }
```

datenname-2 ermöglicht es, den Anfang der Zeilennumerierung des Bildschirmes dynamisch zu ändern. Es bezieht sich auf ein PIC 9999 Feld, wobei die beiden linken 99 eine Zeilenposition 1-25 ist und die beiden rechten 99 eine Zeichenposition von 1 bis 80 ist.

datenname-1 bezieht sich auf einen Datensatz, eine Datengruppe oder ein Datenelement, darf jedoch nicht mit Subskript versehen werden.

literal-1 ist ein numerisches Literal.

Die DISPLAY-Anweisung:

Für die DISPLAY-Anweisung gibt es ein zusätzliches Format:

```
DISPLAY { datenname-1 } [ [AT { datenname-2 }] ] [UPON { CRT        }]
        { literal-3   }       { literal-1   }          { CRT-UNDER }
```

literal-3 ist ein alphanumerisches Literal

datenname-1 bezieht sich auf einen Datensatz, eine Datengruppe oder ein Datenelement, darf jedoch nicht mit Subskript versehen werden.

datenname-2 definiert die linksbündige Position auf dem Bildschirm. Es bezieht sich auf ein PIC 9999 Feld, wobei die beiden linken 99 eine Zeilenposition 1-25 und die beiden rechten 99 eine Zeichenposition 1-80 ist.

Plattenspeicherdateien

Durch die LII COBOL Dateiverarbeitung werden die beiden folgenden Erweiterungen angeboten:

1. zeilensequentielle Dateien
2. Eingabe von Dateinamen zur Laufzeit

Wenn LINE SEQUENTIAL ORGANIZATION in der ORGANIZATION IS-Eintragung des FILE-CONTROL-Paragraphen angegeben ist, wird die Datei so behandelt, als ob sie aus Datensätzen mit variabler Länge bestünde, die durch ein dem Betriebssystem entsprechendes Zeilenbegrenzungszeichen getrennt sind. Bei Eingabe wird der Begrenzer gelöscht, und der Datensatzbereich wird mit Leerzeichen, sofern notwendig, aufgefüllt; bei Ausgabe werden nachfolgende Leerzeichen im Datensatzbereich gelöscht.

Eingabe von Dateinamen zur Laufzeit

Sehen Sie dazu bitte die LII COBOL Bedienungsanleitung.

Kleinbuchstaben

Alle Kleinbuchstaben von a bis z stehen unter LII COBOL zur Verfügung. Reservierte Wörter und Programmiererwörter werden so gelesen, als ob es sich um Großbuchstaben (A bis Z) handeln würde.

Hexadezimale Werte

Hexadezimale binäre Werte können nichtnumerischen Literalen unter LII COBOL zugeordnet werden, indem sie als x "xx" ausgedrückt werden, wobei x ein hexadezimales Zeichen aus dem Vorrat 0 - 9, A - F ist; xx kann bis zu 128mal wiederholt werden; die Anzahl der hexadezimalen Ziffern muß jedoch gerade sein.

1.10.2 Die Differenzen in den einzelnen Kapiteln

Die Unterschiede sind nach Kapiteln aufgelistet.
Es werden nur kurze Hinweise gegeben, was an den betreffenden Stellen zu beachten ist, nicht, wie Sie das jeweilige Problem lösen.

Sprachumfang

1. Das Kommunikationsmodul ist in COB1 nicht vorhanden.
2. Die Implementierung von Testhilfen in COB1 ist nicht in allem dem ANSI 1974 Standard angepaßt. Testhilfezeilen und die WITH DEBUGGING MODE-Angabe sind implementiert.
3. Das Modul REPORT WRITER ist nicht in LEVEL II enthalten.
4. Die SUBSCHEMA SECTION ist nicht in LEVEL II enthalten.

Quellformat

1. Vorhandene Zeilennummern in Spalten 1-6 werden in COB1 nach aufsteigender Reihenfolge sortiert, aber nicht in LII COBOL.

   Dieser Bereich muß bei COB1 numerisch oder Leerzeichen sein, während bei LEVEL II ein Stern in Spalte 1 oder ein Schrägstrich gefolgt von einem Stern in Spalte 1 und 2 bewirkt, daß die ganze Zeile ignoriert wird und nicht in der Dateiliste erscheint.
2. Stufennummern 66 und 88 müssen im Bereich A (Spalte 8-11) bei LEVEL II beginnen, während sie bei COB1 sowohl bei A als auch B (Spalte 12-72) beginnen können.
3. Programmsätze bei LEVEL II können im Bereich A oder B beginnen, in COB1 sind sie nur im Bereich B zugelassen.
4. Die Spalten 73-80 können in COB1 als (unbewerteter) Identifikationsbereich benutzt werden, bei LII COBOL müssen aber diese Spalten frei bleiben.

Kapitel 1 Allgemeine Begriffe

Zeichenmenge

Kleinbuchstaben werden vom COB1 nicht akzeptiert. In LEVEL II werden sie als Großbuchstaben interpretiert, soweit sie nicht Teil eines nichtnumerischen Literals sind.

Wörter, die vom Benutzer definiert werden müssen

1. Die folgenden vom Benutzer definierten Wörter gibt es bei LEVEL II aber nicht bei COB1:

Externes Dateiname-Literal	für Ein/Ausgabe
Routinename	für Nicht COBOL-Prozeduren

2. Die folgenden vom Benutzer definierten Worte gibt es bei COB1 aber nicht bei LEVEL II:

Eingangsname	für CALL und ENTER
Listenname	für REPORT WRITER

Reservierte Wörter in COB1 bzw LII

Wörter, die in COB1, aber nicht in LEVEL II reserviert sind:

ACTUAL
ALPHABET
APPLY

BEGINNING
BLOCK-DENSITY
CBL-CTR
CF
CH
CHANGED
CHECKING
CHECKPOINT
CODE
COLUMN
COMP(UTATIONAL)-1
COMP(UTATIONAL)-2
COMP(UTATIONAL)-4
CONTROL(S)
CONVERTING
CREATING
CSP
CURRENT-DATE
CYL-OVERFLOW
CYLINDER-OFLOW
C01,C02,...,C12

DAY-OF-WEEK
DE
DETAIL
DIRECT
DISC

ENDING
ENTRY
EXAMINE
EXHIBIT
EXTENDED

EXTENDED-SEARCH
FILE-LIMIT(S)
FINAL
FOOTING
FORM-OVERFLOW

GENERATE
GROUP

HEADING

INDICATE
INITIATE

LAST
LINE-COUNTER
LOG

MINUS
MORE-LABELS

NAMED
NO
NOMINAL
NOTE
NUMBER

ORDER
OTHERWISE

PAGE-COUNTER
PERCENT
PF
PH
PLUS
POSITIONING
PRINT-SWITCH
PRINTING
PROCESSING
PURGE

RECORD-OVERFLOW
RECORDING
REMARKS
REORG-CRITERIA
REPORT(S)
REPORTING
RESET
RESTRICTING
RESULT
RETURN-CODE
RF
RH
ROLL-BACK

SEEK
SORT-TAPE(S)
STANDARD-2
SUM
SUPPRESS
SYSIPT
SYSLST
SYSOPT
SYSPCH
SYSPUNCH
SYSRDR

TERMINATE
TALLY
TAPES
TRACE
TRACK-AREA
TRACKS
TRANSFORM
TRY
TSW-1=,...,TSW-31

UNITS
UPSI-0,...,UPSI-7
USW-0,...,USW-31

WRITE-ONLY
WRITE=VERIFY

Wörter reserviert in LEVEL II aber **nicht in COB1:**

COMMAND-LINE	DEBUG-CONTENS	EXCESS-3	ROLLBACK
COMMIT	DEBUG-ITEM		
CRT	DEBUG-LINE	FORMFEED	SWITCH
CRT-UNDER	DEBUG-NAME		
CURRENCY	DEBUG-SUB-1	KEPT	TAB
CURSOR	DEBUG-SUB-2		
	DEBUG-SUB-3	PAGETHROW	

Literale

1. Nichtnumerische Literale können bei LEVEL II 128 Zeichen lang sein, bei COB1 180 Zeichen lang.
2. Zwei aufeinanderfolgende Anführungszeichen werden in LEVEL II-Literalen benutzt, um ein Anführungszeichen darzustellen. In COB1 werden diese durch Apostroph dargestellt. (Der COBRUN-Parameter QUOTE1 vertauscht die Funktionen von Anführungszeichen und Apostrophen.)
3. Nichtnumerische Literale in LEVEL II können hexadezimale binäre Werte der Form X"nn" haben, wobei nn hexadezimal ist. Dies gilt nicht für COB1.
4. Gleitpunktliterale gibt es in LEVEL II nicht.

Datenformate

1. Die maximale Größe für ein Datenelement in COB1 ist 65.535 bytes, verglichen mit 8191 bytes in LEVEL II. In LII kann eine Datengruppe größer sein, aber dann nicht als ganze angesprochen werden.
2. USAGE COMPUTATIONAL-1 und COMPUTATIONAL-2 (interne Gleitpunktfelder) und COMPUTATIONAL-4 sind in LEVEL II nicht vorhanden.
3. Binäre Felder in COB1 sind in Halbworten, Worten oder Doppelworten gespeichert und können deshalb größer sein als dasselbe Feld in LEVEL II.
4. Für Felder mit mehr als 16 Ziffern und Werten größer (16**14) geht in COB1 die Genauigkeit der letzten Ziffern verloren.

ANSI-Schalter

Hinweis für COB1 Benutzer

Der ANSI-Schalter bei LEVEL II veranlaßt den Übersetzer, auf einer strikten ANSI-Interpretation zu beharren; er fordert solche ANSI Standardanweisungen, die normalerweise bei LEVEL II optional sind und signalisiert LEVEL II-Erweiterung zum Standard. Der Gebrauch dieses Schalters führt so dazu, daß der Code leichter mit COB1 vergleichbar ist.

Qualifikation

1. Datennamen können bis zu 5 Kennzeichner in LEVEL II und bis zu 49 in COB1 haben.
2. LEVEL II hat eine Einschränkung, die in COB1 nicht existiert, nämlich daß nicht ein Name gleichzeitig Datenname und Prozedurname sein darf.

Subskribierung und Indizierung

1. Relative Subskribierung oder Indizierung kann in LEVEL II nur dann benutzt werden, wenn der ANSI-Schalter nicht gesetzt ist.
2. Wenn der ANSI-Schalter für LEVEL II gesetzt ist, ist die größte Zahl an Subskripten und Indizes (und die Dimension einer Tabelle) 3, wie in COB1. Sonst können Tabellen bis zu 49 Dimensionen (beschränkt durch die maximale Größe der DATA DIVISION) haben und die maximale Zahl von Subskripten und Indizes ist im allgemeinen 9, abhängig von ihrer Länge.

Programmstruktur

1. Die folgenden Klauseln sind notwendig in COB1, aber in LEVEL II nur optional, außer wenn der ANSI-Schalter gesetzt ist:

   ```
   IDENTIFICATION/ID DIVISION
   PROGRAM-ID
   ENVIRONMENT DIVISION
   DATA DIVISION
   ```

2. In der DATA DIVISION eines LEVEL II-Programms darf man keine REPORT SECTION oder SUBSCHEMA SECTION angeben. COB1 erkennt die COMMUNICATION SECTION nicht.

3. Wenn der ANSI-Schalter nicht gesetzt ist, ist der erste Paragraphname in der PROCEDURE DIVISION optional bei LEVEL II. Dies gilt nicht für COB1.

4. Auf einen DECLARATIVES Kapitelnamen kann in LEVEL II eine Segmentnummer folgen, das gilt nicht in COB1.

Anweisungen

Anweisungen, die in LEVEL II, aber nicht in COB1 geschrieben werden können:

```
CANCEL
SEND
```

Anweisungen, die in COB1, aber nicht in LEVEL II bekannt sind:

```
ENTRY
EXAMINE
EXHIBIT
GENERATE
INITIATE
ON
SEEK
TERMINATE
TRACE
TRANSFORM
```

Kapitel 2 Sprachkonzept

IDENTIFICATION DIVISION

1. Die Abkürzung ID DIVISION in COB1 ist in LEVEL II nicht bekannt. IDENTIFICATION DIVISION und PROGRAM-ID Zeilen sind in LEVEL II optional, falls der ANSI-Schalter nicht gesetzt ist.
2. Die REMARKS-Klausel in COB1 ist in LEVEL II nicht bekannt.
3. Ein COB1-Programmname muß mit einem Buchstaben beginnen. In LEVEL II kann der Programmname mit Buchstaben oder Ziffern begonnen werden, aber er darf nicht mit einem Bindestrich beginnen oder enden. Im BS2000 müssen die ersten acht Buchstaben des Programmnamens eindeutig sein.

ENVIRONMENT DIVISION

1. Die Zeilen ENVIRONMENT DIVISION, CONFIGURATION SECTION und INPUT-OUTPUT-SECTION können bei LEVEL II weggelassen werden, auch dann, wenn Klauseln dieser Section vorhanden sind.
2. Die ursprüngliche COLLATING SEQUENCE in LEVEL II ist ASCII; in COB1 ist sie EBCDIC.
3. Die SEGMENT-LIMIT-Klausel in COB1 bewirkt, daß diejenigen Segmente überlagerbar sind, die eine größere Segmentnummer haben, als vom Limit erlaubt.
   In LEVEL II ist sie nur Dokumentation.

Sondernamen

1. Die Klauseln SWITCH IS, FORMFEED IS und CURSOR IS bei LEVEL II sind in COB1 nicht bekannt.
2. Die ALPHABET-Klausel in COB1 muß abhängig von COBRUN-Parametern mit dem Wort ALPHABET beginnen.
3. Der Alphabetname in COB1 IS STANDARD-2 (für ISO 7 bit Code) gilt nicht für LEVEL II.
4. In der CURRENCY SIGN IS-Klausel ist der Buchstabe E für LEVEL II als Literal erlaubt, aber nicht für COB1.

DATA DIVISION

Die PICTURE-Klausel

1. In LEVEL II dürfen die Zeichen '.' und ',' nicht die letzten in einer PICTURE-Zeichenfolge sein.
2. LEVEL II hat keine Gleitpunktfelder.
3. In LEVEL II muß das Währungszeichen das erste Zeichen sein, während in COB1 ein '+' oder '-' vorangestellt sein kann.
4. '*' in einer PICTURE-Klausel als Nullenunterdrückungssymbol hat Vorrang vor der BLANK WHEN ZERO-Klausel in COB1. In LEVEL II dürfen die beiden nicht in der selben Eintragung erscheinen.
5. COB1 hat die Einschränkung, daß auf der Stufe 01 im FD-Eintrag PICTURE-Klauseln nicht erlaubt sind.

Die REDEFINES KLausel

1. In COB1 muß auf die REDEFINES-Klausel unmittelbar der redefinierende Datenname folgen.
2. Bei mehrfachen Redefinitionen erlaubt COB1 eine Erleichterung des ANSI Standard, in der Redefinitionen den Datennamen von der vorhergehenden REDEFINES-Klausel genausogut benutzen können wie den Datennamen von der Originaldefinition (eine Warnmeldung erfolgt).

Die SYNCHRONIZED-Klausel

1. Die SYNCHRONIZED-Klausel, die in LEVEL II nur für ein Datenelement gegeben werden kann, kann in COB1 genauso mit einer Datengruppe verwendet werden.
2. Der LEFT- oder RIGHT-Befehl wird in COB1 als Kommentar behandelt.

Die USAGE-Klauseln

1. USAGE COMPUTATIONAL-1
   USAGE COMPUTATIONAL-2
   USAGE COMPUTATIONAL-4
   gibt es bei COB1, aber nicht bei LEVEL II.

PROCEDURE DIVISION

1. In LEVEL II kann einem arithmetischen oder unären Operator ein Leerzeichen vorangestellt werden und folgen. In COB1 muß einem arithmetischen Operator immer ein Leerzeichen vorangestellt sein und folgen. Einem unären Operator muß immer ein Leerzeichen oder Klammern vorangestellt sein. Es muß ein Leerzeichen folgen, falls ein Datenname vorangestellt war und es muß kein Leerzeichen folgen, wenn ein numerisches Literal vorangestellt war.

Bedingungen

1. COB1 läßt für eine Vergleichsbedingung nur Datenfelder mit gleichen USAGE-Angaben zu, ausgenommen, wenn beide Operanden numerisch sind.
   Für LII COBOL gilt dies nur, wenn der ANSI-Schalter gesetzt ist.
2. Der Bezeichner in einer Klassenbedingung, der in LEVEL II mit USAGE IS DISPLAY erklärt sein muß, kann in COB1 genauso COMPUTATIONAL-3 sein.
3. In einer Vorzeichenbedingung in LEVEL II muß der Operand ein arithmetischer Ausdruck sein. In COB1 kann es auch ein Bezeichner sein.
4. In COB1, nicht in LEVEL II, sind Abkürzungen in der einer zusammengesetzten Bedingung erlaubt, auch wenn Klammern vorhanden sind (bei arithmetischen Vergleichsbedingungen).

Die ACCEPT-Anweisung

ACCEPT-Anweisungen bei LEVEL II und COB1 enthalten grundlegende Unterschiede in Format und Interpretation des Verbs:

1. Die Formate
   ```
   ACCEPT datenname-1 [AT datenname-2] FROM CRT
   ```
   und
   ```
   ACCEPT datenname-1 AT datenname-2
   ```

   gibt es nur bei LEVEL II.

2. Die Formate
   ```
   ACCEPT bezeichner FROM SYSIN
                          TERMINAL
                          SYSRDR
                          SYSIPT
   ```
   und
   ```
   ACCEPT bezeichner  FROM DAY-OF-WEEK
   ```

   gibt es nur in COB1.

3. Für die Grundform 'ACCEPT bezeichner' ist die Standardannahme in LEVEL II 'FROM CONSOLE' (oder 'FROM CRT', wenn diese Option im SPECIAL NAMES-Paragraphen angegeben ist).
   'FROM SYSIPT' ist die Standardannahme bei COB1.

4. Bei dem Format ACCEPT bezeichner FROM TIME enthält bei COB1 das 8-Ziffern-Feld immer Nullen in den letzten zwei Ziffern.
   Bei LII enthält es hundertstel Sekunden.

Die DISPLAY-Anweisung

Wie bei der ACCEPT-Anweisung sind die LII- und COB1-Versionen von DISPLAY sehr verschieden in Sprache und Bedeutung. Anweisungen im Programm erfordern sicherlich Anpassungen bei der Übertragung ins andere System.

Die Hauptprobleme sind:

1. Die Formate

   ```
   DISPLAY datenname-1 [AT datenname-2] UPON CRT

   DISPLAY datenname-1 [AT datenname-2] UPON CRT-UNDER

   DISPLAY datenname-1  AT datenname-2

   gibt es nur in LEVEL II.
   ```

2. Die Formate

   ```
   DISPLAY ..... UPON SYSLST
                      SYSPUNCH
                      SYSOPT
                      TERMINAL
   ```

   gibt es nur in COB1.

3. Für die Grundform 'DISPLAY bezeichner' ist die Standardannahme in LEVEL II 'UPON CONSOLE' (oder 'UPON CRT', wenn diese Option im SPECIAL-NAMES-Paragraphen angegeben ist).
   Bei COB1 ist die Grundform 'UPON SYSLST'.

Die DIVIDE-Anweisung

Wenn ON SIZE ERROR ... in einer COB1 DIVIDE-Anweisung angegeben ist, dann wird kein Test auf Quotientenüberlauf, sondern nur auf Quotientenunterlauf (d.h. Division durch Null) gemacht. Daher kann ein Divisionsfehler in COB1 immer noch vorkommen, dagegen werden in LII die Zwischen- und Endergebnisse alle überprüft.

Die ENTER-Anweisung

Die ENTER-Anweisung wird in LEVEL II als Dokumentation behandelt. COB1 hat 4 Formate für diese Anweisung, (alle kein Standard) und alle außer ENTER sprachenname eingangsname sind zu LII unterschiedlich.

Die ENTRY-Anweisung

Diese COB1-Anweisung, die es nicht in LEVEL II gibt, wird in Verbindung mit der COB1 ENTER-Anweisung zur Unterprogrammtechnik benutzt.

Die EXAMINE-Anweisung

Diese Anweisung gibt es bei LEVEL II nicht.

Die GO TO-Anweisung

Bei dem Format GO TO ...DEPENDING ON bezeichner darf bei COB1 der Bezeichner in der PICTURE-Zeichenfolge nur max 4 Ziffern haben. In LII gibt es da keine Einschränkungen.

Die INSPECT-Anweisung

Bei LEVEL II, aber nicht bei COB1 sollten Datenfelder, die mit INSPECT angesprochen werden, innerhalb der ersten 10000 bytes des Zwischencodes liegen.

Die MOVE-Anweisung

1. In COB1, nicht in LEVEL II, kann eine MOVE CORRESPONDING-Anweisung mehr als ein Empfangsfeld ansprechen.
2. Die folgenden MOVE's sind in LEVEL II, nicht in COB1, erlaubt:

   a) Ein numerisch druckaufbereitetes in ein numerisches Datenfeld.
   b) HIGH-VALUE, LOW-VALUE oder QUOTES in ein alphabetisches, numerisches oder numerisch druckaufbereitetes Datenfeld.

3. Der folgende MOVE ist in COB1, nicht in LEVEL II, erlaubt:

   a) Ein numerisches Literal in ein alphabetisches Datenfeld.

Die PERFORM-Anweisung

1. In COB1, nicht in LEVEL II, können 2 oder mehr aktive PERFORM-Anweisungen einen gemeinsamen Ausgang haben.
2. Für Format 4,(PERFORM..VARYING..FROM..BY..) erlaubt COB1, daß jeder Bezeichner als nicht ganzzahliges numerisches Feld beschrieben wird. In LEVEL II, wenn ein Indexname in einer VARYING- oder AFTER-Angabe benutzt wird, muß der Bezeichner bei zugehörigen VARYING-, AFTER- und BY-Angaben ganzzahlig sein. Wird ein Indexname in einer FROM-Angabe benutzt, so müssen die Bezeichner der zugehörigen VARYING-, AFTER- und BY-Angaben ganzzahlig sein.
3. Wenn ein Indexname in einer VARYING oder AFTER-Angabe angegeben ist und ein Bezeichner in der dazugehörigen FROM-Angabe, dann muß bei LEVEL II das mit Bezeichner angesprochene Feld einen positiven Wert haben, während bei COB1 der ursprüngliche Wert Null oder negativ sein kann.
4. Eine Änderung des Wertes eines mit Indexnamen oder mit Bezeichner angesprochenem Feldes der FROM-Option während der Ausführung der PERFORM-Anweisung hat keine Auswirkung bei der Ausführung in COB1. In LEVEL II wird die nachfolgende Ausführung der PERFORM-Anweisung hingegen beeinflußt.

Die SET-Anweisung

Abgesehen vom Gebrauch der Anweisung in der Tabellenverarbeitung hat COB1 ein drittes Format, das es bei LEVEL II nicht gibt, und das gebraucht wird, um den Status von externen Schaltern zu verändern: SET Merkname ... TO ON/OFF ...

Die STRING-Anweisung

LEVEL II, aber nicht COB1, verlangt, daß das Empfangsfeld einer STRING-Anweisung ein Datenelement ist.

Die TRANSFORM-Anweisung

Die COB1-Anweisung TRANSFORM gibt es bei LII nicht.

Die UNSTRING-Anweisung

Die Bezeichner in COUNT-, POINTER- und TALLYING-Angaben müssen bei COB1 als ganzzahlige Datenfelder definiert sein. Bei LEVEL II müssen es ganzzahlige Datenelemente sein.

Kapitel 3 Tabellenverarbeitung

1. Ist der ANSI-Schalter für LEVEL II gesetzt, ist die maximale Zahl an Tabellendimensionen 3 (wie in COB1), sonst ist sie 49. Die maximale Zahl von Subskripten oder Indizes ist im allgemeinen 9, je nach Länge.
2. Relative Subskripte oder Indizes können in LEVEL II nur dann benutzt werden, wenn der ANSI-Schalter nicht gesetzt ist.
3. In COB1 kann ein spezielles TALLY-Register ein Subskript ersetzen.
4. Wenn auf einen Indexnamen eine ' + ganze Zahl' oder '- ganze Zahl' folgt, muß in COB1 ein Leerzeichen vorangestellt sein und eins folgen.
5. Während sich in COB1 der Index nur auf die Tabelle bezieht, der er zugewiesen ist, kann ein LEVEL II-Index mit jeder anderen Tabelle benutzt werden.

Die OCCURS-Anweisung

1. Im Format 2 kann bei COB1 'ganzzahl-1 TO' in der OCCURS-Anweisung (OCCURS ganzzahl-1 TO ganzzahl-2 TIMES DEPENDING ON datenname-1..) weggelassen werden.
2. Bei COB1 darf datenname-1 im Format 2 nicht indiziert werden.
3. Bei' KEY IS datenname-1, datenname-2 ...' hat COB1 die Einschränkung, daß datenname-2 nicht angegeben werden darf, wenn datenname-1 das Subjekt der Klausel ist.

4. Während in LEVEL II auf eine Datenerklärung, die eine Format 2 OCCURS-Klausel enthält, nur in einer Datensatzerklärung Einträge folgen können, die diesen untergeordnet sind, erlaubt COB1, daß bis zu 10 aufeinanderfolgende OCCURS...DEPENDING-Angaben irgendwo in einer Datensatzerklärung erscheinen, aber nicht ineinandergeschachtelt sein dürfen.

5. Bei LEVEL II ist die OCCURS-Klausel auf den Stufen 01, 66, 77 oder 88 erlaubt, wenn der ANSI-Schalter nicht gesetzt ist. Bei COB1 ist die OCCURS-Klausel mit Stufe 66 gestattet, aber nicht mit den Stufen 01, 77 und 88.

Die USAGE-Klausel

Im Unterschied zu LEVEL II erlaubt COB1 die SYNCHRONIZED-Klausel im Gebrauch mit der USAGE IS INDEX-Klausel.

PROCEDURE DIVISION

Die SET-Anweisung

1. Bei COB1 muß ein Indexname in einer INDEXED BY-Option einer OCCURS-Klausel angegeben werden. Bei LEVEL II werden Indexnamen nur relativ zu ihrer angegebenen Tabelle betrachtet, wenn der ANSI-Schalter gesetzt ist.

2. ganzzahl in 'SET bezeichner TO ganzzahl' muß bei LEVEL II positiv sein, während COB1 auch den Wert Null erlaubt.

3. Format 2, 'SET ..UP BY...' kann in COB1 nur mit einem oder mehreren Indexnamen als Subjekt benutzt werden und mit einem ganzzahligen Bezeichner oder einem Literal als Objekt. LEVEL II erlaubt ebenfalls ganzzahlige Bezeichner oder Literale als Subjekte und einen Indexnamen als Objekt. Weiterhin muß in COB1, nicht in LEVEL II, das Literal als Objekt ungleich Null sein.

Kapitel 4 Sequentielle Eingabe und Ausgabe

FILE STATUS

Zusätzlich zu den Werten des Statusbyte der beiden Übersetzer hat COB1 die folgenden:

04 erfolgreiches READ, aber Datensatz länger als Datensatzbereich
05 erfolgreiches OPEN, aber OPTIONAL-Datei existiert nicht
15 erfolgloses READ, OPTIONAL-Datei existiert nicht
16 erfolgloses READ, AT END vorher durchgeführt
34 erfolgloses WRITE, außerhalb Systemdateibeschränkungen
35 erfolgloses OPEN, Datei existiert nicht, keine OPTIONAL-Klausel
38 erfolgloses OPEN, Datei vorher mit WITH LOCK geschlossen
39 erfolgloses OPEN, Konflikt in der Dateierklärung
41 OPEN für bereits geöffnete Datei
42 CLOSE für nicht geöffnete Datei
43 Mit Dateieröffnung I-O: DELETE oder REWRITE nicht von einem READ gefolgt
44 Überschreitung der Bereichsgrenzen
46 Kein nächster gültiger Datensatz für READ
47 READ für Datei nicht im Eingabe- oder E-A-Modus
48 WRITE für Datei nicht im Ausgabe-, E-A- oder EXTEND-Modus
49 REWRITE für Datei nicht im E-A-Modus

Für die Werte, die mit 9 beginnen, hat LEVEL II eine Laufzeitfehlerbezeichnung als 2. Zeichen und COB1 hat die Werte:

90 Systemfehler: keine weiteren Informationen
91 Systemfehler: Systemaufruf beendet unnormal

FILE CONTROL-Paragraph

1. COB1 hat die Einschränkung, die es in LEVEL II nicht gibt, daß die ASSIGN-Klausel unmittelbar auf die SELECT-Klausel folgen muß. Im BS2000 müssen die ersten acht Zeichen des Dateinamen in der SELECT-Klausel eindeutig sein.
2. Die Klausel ORGANIZATION IS LINE SEQUENTIAL gibt es bei COB1 nicht.
3. Datennamen in der FILE STATUS-Klausel in LEVEL II müssen alphanumerisch definiert sein. In COB1 können sie numerisch oder alphnumerisch sein.
4. Die RESERVE NO ALTERNATE AREA-KLausel existiert nicht in LEVEL II und die ganze RESERVE-Klausel wird in LEVEL II nur als Dokumentation behandelt.
5. Die TRACK-AREA, FILE-LIMIT und PROCESSING-MODE-Klauseln, die in COB1 aus Kompatibilitätsgründen existieren und als Kommentare geführt werden, gibt es in LEVEL II nicht.

Die ASSIGN-Klausel

Die COB1- und LEVEL II-Formate der ASSIGN-Klausel sind nicht mehr vergleichbar. Während in LEVEL II die Dateiidentifikation im Programm enthalten ist, erfolgt die Information in COB1 zur Laufzeit über ein Systemkommando.

Das einfachste COB1 Format ist

```
ASSIGN TO xx-S-SYS nnn
```

wobei nnn eine ganze Zahl kleiner gleich 099 und xx ein Datenfeld mit dem Inhalt:

UT für Band-oder sequentiellen Plattenzugriff
DA für direkten Zugriff ist.

Für den Drucker (oder Leser usw.) kann ein beliebiger Name angegeben werden z.B.

```
ASSIGN TO US-PRINTERANS-S-SYS nnn
```

oder eine Systemnamenklausel kann benutzt werden z.B.

```
ASSIGN TO SYSLST.
```

Das LEVEL II Format ist

```
ASSIGN TO externes dateiname-literal
oder  ASSIGN TO dateibezeichner
```

wobei ein Dateibezeichner ein vom Benutzer definiertes Wort ungleich dem Dateinamen (beim SELECT) ist, das den Wert des Literals 'externer Dateiname' enthält. Die Angabe FOR MULTIPLE REEL/UNIT, die in COB1 als Kommentar benutzt wird, wird von LEVEL II nicht erkannt.

Näheres dazu siehe LII COBOL Bedienungsanleitung.

I-O-CONTROL-Paragraph

1. Die RETURN- und MULTIPLE FILE TAPE-Klauseln werden in LEVEL II als Dokumentation behandelt.
2. Die COB1 APPLY-Klausel ist in LEVEL II nicht bekannt.

FILE SECTION

1. Die LABEL-RECORDS-KLausel ist in COB1 notwendig, während sie in LEVEL II optional ist, wenn der ANSI-Schalter nicht gesetzt ist. Die Form LABEL RECORD(S) datenname-1 datenname-2 .. ist in COB1 erlaubt, aber nicht in LEVEL II. Die Klausel wird in LEVEL II als Kommentar behandelt.
2. Die RECORDING MODE-Klausel existiert bei LEVEL II nicht.
3. Die BLOCK CONTAINS-Klausel dient in LEVEL II nur zur Dokumentation, wird aber bei COB1 zusammen mit der RECORDING MODE-Klausel gebraucht, um die maximale Blocklänge zu bestimmen.

4. Die RECORD-Klausel

   'RECORD...VARYING...DEPENDING...'

   gibt es nur bei COB1, bei LEVEL II sind sie nur Dokumentation. Der COB1-Übersetzer braucht diese Klausel zum Vergleich mit der Datensatzerklärung.

5. Zusätzlich zum Format der VALUE OF-Klausel bei LEVEL II hat COB1 ein Format (VALUE OF ID[ENTIFICATION] IS) für ergänzende Dateiinformation.

6. Die CODE SET-KLausel wird in LEVEL II als Dokumentation behandelt.

PROCEDURE DIVISION

Die USE-Prozeduren, die in COB1 für Kennsatzverarbeitung in Verbindung mit OPEN, CLOSE, READ und WRITE-Anweisungen gebraucht werden, gibt es in LEVEL II nicht.

Die CLOSE-Anweisung

1. Das Format 'CLOSE dateiname FOR REMOVAL' (ohne REEL/UNIT) ist in COB1 erlaubt, aber nicht in LEVEL II.

2. Eine 'CLOSE WITH NO REWIND'-Anweisung in LEVEL II bewirkt, daß die aktuelle reel/unit in ihrer aktuellen Position bleibt, während es in COB1 bewirkt, daß die aktuelle reel/unit an das logische Ende der Datei dieser Einheit positioniert wird.

Die OPEN-Anweisung

1. Bei LEVEL II bewirkt die OPEN-Anweisung eine Überprüfung des ASSIGN-Namens, bei COB1 wird der zu benutzende Dateiname extern angegeben.
2. Bei einer OPEN-, I-O- oder EXTEND-Anweisung verlangt COB1 eine existiernde Datei, während LEVEL II eine anlegt, falls keine existiert.
3. In LEVEL II wird die NO REWIND-Angabe als Dokumentation behandelt. In COB1 hat sie bei Plattendateien dieselbe Funktion wie OPEN EXTEND.
4. In COB1 ist die REVERSED-Angabe nur für Einzelspuldateien erlaubt, mit Standardkennsätzen oder Nicht-Standardkennsätzen gefolgt von einer Bandmarke.
5. COB1 erlaubt, im Gegensatz zu LEVEL II, eine WRITE-Anweisung für eine im I-O-Modus eröffnete Datei.

Die USE-Anweisung

1. Diese Anweisung hat in COB1 zwei getrennte Formate. Das erste gibt die Routinen für die Behandlung der Benutzerkennsätze der Dateien an.
2. Das zweite Format ist identisch mit dem LEVEL II-Format, allerdings erlaubt der COB1-Übersetzer einige Abweichungen von der Regel. Diese sind:

   Die USE-Prozedur kann mit einem GO TO oder EXIT-PROGRAM enden und ein CALL enthalten. Prozedurnamen in einer USE-Anweisung können durch GO TO angesprochen werden.
3. Im Format 2 erlaubt COB1 nicht mehr als 2 USE-Anweisungen für mehrere Dateinamen, egal ob implizit oder explizit.

Die WRITE-Anweisung, allgemein

1. COB1 besitzt die Einschränkung, daß der Datensatzname in einer Anweisung nicht mit einem Dateinamen bezeichnet werden darf.
2. In COB1 kann, im Gegensatz zu LEVEL II, eine WRITE-Anweisung für die Dateieröffnung im Ein-Ausgabemodus gegeben werden.

Die WRITE-Anweisung, Druckerdateien

1. Das optionale Wort 'POSITIONING' in der 'BEFORE'- oder 'AFTER'-Angabe wird in LEVEL II nicht erkannt.
2. Die LEVEL II-Angaben

   BEFORE/AFTER ADVANCING FORMFEED

   gibt es in COB1 nicht.
3. Die Interpretation von (und Einschränkung zu) den verschiedenen Formen der WRITE-Anweisung für Druckdateien in COB1 ist abhängig von Format der ASSIGN-Klausel, die diese Datei betrifft. Das Format ASSIGN TO UR-PRINTERANS-S-SYS nnn macht die WRITE-Anweisung gleichbedeutend mit der LEVEL II-Anweisung (neben den anderen hier aufgelisteten Einschränkungen) und ist darüberhinaus das einzige Format, das mit der LINAGE-Angabe in COB1 erlaubt ist. Für die anderen Formate, die SYSLST, PRINTER und PRINTERDOD benutzen gelten die folgenden Regeln:

   a) das erste Zeichen dieses Satzes wird als Druckkontrollzeichen benutzt, außer bei SYSLST Datei ohne ADVANCED-Option. Im Falle von PRINTERDOD liefert der Benutzer das zugehörige Zeichen vor der Ausführung, während bei SYSLST und PRINTER-Datei das Leerzeichen reserviert sein muß.
   b) Für SYSLST und PRINTER-Dateien ist der vermischte Gebrauch von WRITE-Anweisungen mit und ohne ADVANCING nicht erlaubt.

Kapitel 5 Relative Ein- und Ausgabe

FILE STATUS

Ergänzend zu den Statusbytes, die beiden Übersetzern bekannt sind, hat COB1 die folgenden:

04 erfolgreiches READ, aber Datensatz länger als Datensatzbereich
14 erfolgloses READ, RELATIVE KEY Überlauf
16 erfolgloses READ, AT END vorher angegeben
35 erfolgloses OPEN, Datei existiert nicht
38 erfolgloses OPEN, Datei vorher WITH LOCK geschlossen
39 erfolgloses OPEN, Konflikt in der Datensatzerklärung
41 OPEN für bereits geöffnete Dateien
42 CLOSE für nicht geöffnete Dateien
43 Sequentieller Zugriff: DELETE und REWRITE geht kein READ voraus
46 Kein nächster gültiger Dateisatz fürs READ
47 READ oder START für Datei nicht im Eingabe- oder Ein-Ausgabe-Modus ist
48 WRITE für Datei, die nicht im Ausgabe-,Ein-Ausgabe- oder EXTEND-Modus ist
49 DELETE oder REWRITE für Dateien, die nicht im Ein-Ausgabe-Modus sind

Für die Werte, die mit 9 beginnen hat LEVEL II eine Laufzeitfehlerbezeichnung als 2. Zeichen und COB1 hat die Werte:

90 Systemfehler: keine weiteren Informationen
91 Systemfehler: Systemaufruf terminiert unnormal
93 Systemfehler mit asynchronen Dateiprozessen

FILE CONTROL Paragraph

1. COB1 besitzt die Einschränkung, die es bei LEVEL II nicht gibt, daß auf die SELECT-Klausel unmittelbar die ASSIGN-Angabe folgen muß. Beim BS2000 müssen die ersten acht Zeichen eindeutig sein.

2. In LEVEL II ist die RELATIVE KEY-Klausel abhängig von der ACCESS MODE-Klausel und kann nicht ohne sie vorkommen (die RELATIVE KEY-Klausel ist optional beim ACCESS SEQUENTIAL, zugelassen für ACCESS RANDOM und DYNAMIC). COB1 hat diese Einschränkungen nicht.

3. Datennamen müssen in LEVEL II in der FILE STATUS-Klausel alphanumerisch definiert werden. In COB1 numerisch oder alphanumerisch.

4. Die Klausel RESERVE NO ALTERNATE AERA existiert in LEVEL II nicht und die ganze RESERVE-Klausel wird in LEVEL II nur zur Dokumentation benutzt.

5. Die TRACK-AREA, FILE-LIMIT UND PROCESSING-MODE-Klauseln, aus Kompatibilitätsgründen bei COB1 zugelassen, werden als Kommentar behandelt. Bei LII gibt es sie gar nicht.

Die ASSIGN-Klausel

Die COB1- und LEVEL II-Formate dieser Anweisung sind nicht gleich, wobei in LEVEL II die Dateiidentifikation ins Programm eingeschlossen ist und in COB1 die Information durch Dateikommandos während der Laufzeit angegeben wird.

Das einfachste COB1 Format ist

```
ASSIGN TO UT-R-SYSnnn
```

wobei nnn eine ganze Zahl nicht größer als 099 ist.

Das LEVEL II Format ist

```
ASSIGN TO externes dateiname-literal
```

oder

```
ASSIGN TO dateibezeichner,
```

wobei dateibezeichner ein vom Benutzer definiertes Wort ungleich des Dateinamens (im SELECT) ist.

I-O CONTROL-Paragraph

Die RERUN-Klausel wird in LEVEL II als Dokumentation behandelt.

FILE SECTION

1. Die LABEL RECORDS-Klausel ist in COB1 notwendig, aber in LEVEL II nur optional, wenn der ANSI-Schalter nicht gesetzt ist.

   Die Form LABEL RECORD(S) OMITTED ist in LEVEL II erlaubt, aber nicht in COB1 bei relativen Dateien. Die Klausel wird in LEVEL II als Dokumentation behandelt.

2. Die RECORDING MODE-Klausel existiert nicht in LEVEL II.

3. In der BLOCK CONTAINS-Klausel in COB1 ist nur CHARACTERS bei der relativen Ein-Ausgabe erlaubt. Die Klausel ist für Dokumentationszwecke in LEVEL II, aber in COB1 wird sie zusammen mit der RECORDING MODE-Klausel benutzt, um die maximale Blocklänge festzulegen.

4. RECORD CONTAINS ganzzahl-1 TO ganzzahl-2 CHARACTERS wird in LEVEL II akzeptiert, während COB1 nur Dateisätze mit fester Länge von relativen Dateien erlaubt. Wo mehr als eine Art von Dateisätzen angegeben ist, kann das Format verschieden, aber die Länge muß gleich sein. Diese Klausel wird in LEVEL II als Dokumentation behandelt, aber im COB1-Übersetzer wird sie benutzt, um sie mit der folgenden Datensatzerklärung zu vergleichen.

5. Die VALUE OF-Klausel ist in COB1 für relative Dateien nicht erlaubt.

PROCEDURE DIVISION

Die DELETE-Anweisung

Für sequentielle COB1-Dateien gilt:
RELATIVE KEY darf nicht zwischen der vorher angegebenen READ-Anweisung und der DELETE-Anweisung geändert werden. Das gilt nicht für LEVEL II.

Die OPEN-Anweisung

1. Bei LEVEL II bewirkt die OPEN-Anweisung eine Überprüfung des ASSIGN-Namens, bei COB1 wird der zu benutzende Dateiname extern angegeben.
2. Bei einer OPEN-, I-O- oder EXTEND-Anweisung verlangt COB1 eine existierende Datei, während LEVEL II eine Datei anlegt, falls keine existiert.

Die REWRITE-Anweisung

COB1 hat die Einschränkung, daß das 1. Byte eines Datensatzes, wenn es nochmal beschrieben wird, nicht den Wert X'FF' haben darf, weil es sonst eine Löschung des Datensatzes bewirken würde.

Die USE-Anweisung

Diese Anweisung hat in COB1 zwei getrennte Formate.

1. Das erste Format gibt die Benutzerkennsatzprogramme der Dateien an.
2. Das zweite COB1 Format ist identisch mit dem LEVEL II Format, allerdings erlaubt der COB1 Übersetzer einige Abweichungen von der Regel. Diese sind:

   Die USE-Prozedur kann mit einem GO TO oder EXIT PROGRAM enden und ein CALL enthalten. Auf Prozedurnamen der USE-Anweisung kann mit GO TO verzweigt werden.

3. Im Format 2 erlaubt COB1 nicht mehr als 2 USE-Anweisungen für mehrere Dateinamen, egal ob implizit oder explizit.

Die WRITE-Anweisung

1. COB1 besitzt die Einschränkung, daß das 1. Byte eines Datensatzes nicht den Wert X'FF' haben darf.

2. Genauso darf in COB1 der Datensatzname in der Anweisung nicht mit einem Dateinamen bezeichnet werden.

Kapitel 6 Indizierte Ein- und Ausgabe

Dateistatus

Zusätzlich zu den Werten, die beiden Übersetzern bekannt sind, hat COB1 die folgenden:

04 erfolgreiches READ, aber Datensatz länger als Datensatzbereich
16 erfolgloses READ, AT END vorher angegeben
35 erfolgloses OPEN, Datei existiert nicht
38 erfolgloses OPEN, Datei vorher WITH LOCK geschlossen
39 erfolgloses OPEN, Konflikt in der Datensatzerklärung
41 OPEN für bereits eröffnete Datei
42 CLOSE für nicht geöffnete Datei
43 sequentieller Zugriff: DELETE oder REWRITE geht kein READ voraus
44 WRITE oder REWITE überschreitet die Bereichsbegrenzung
46 Kein nächster gültiger Datensatz fürs READ
47 READ für Datei nicht im INPUT- oder E-A-Modus
48 füer Datei nicht im OUTPUT-, E-A- oder EXTEND-Modus
49 REWRITE für Datei nicht im E-A-Modus

LEVEL II hat zusätzlich den Wert:

02 erfolgreiche Beendigung, doppelter Schlüssel

Für Statusbyte 1 = 9 gilt:
Bei COB1:

90 Systemfehler: keine weiteren Informationen

91 Systemfehler: Systemaufruf beendet abnormal

93 Systemfehler mit asynchronem Dateiprozeß

Bei LEVEL II:

Statusbyte 1	Statusbyte 2	Fehlermeldung	ohne FILE STATUS-Angabe: Abbruch des LZS mit der Fehlernummer:
9	A	gesamte Datei gesperrt	065
9	C	Datei nicht eröffnet	067
9	D	Satz ist gesperrt	068
9	E	unzulässiges Argument für ISAM-Modul	069
9	F	zu viele Dateien offen	070
9	G	fehlerhaftes ISAM-Dateiformat	071
9	H	Dateiende	072
9	J	Kein aktueller Satzanzeiger	074
9	K	Dateiname zu lang	075
9	M	Interner ISAM-Fehler	077
9	N	unzulässiger Datensatzschlüssel	078
9	O	unzulässiger Primärer Datensatzschlüssel	079
9	P	Kein exklusiver Zugriff	080

Statusbyte 2 und zugehörige Fehlermeldungen

FILE CONTROL-Paragraph

1. COB1, nicht LEVEL II, besitzt die Einschränkung, daß auf die SELECT-Klausel unmittelbar die ASSIGN-Klausel folgen muß. Im BS2000 müssen die ersten acht Zeichen der SELECT-Klausel eindeutig sein.
2. Den COB1 ACCESS MODE 'EXTENDED' gibt es in LEVEL II nicht.
3. Die ALTERNATE RECORD KEY-Klausel gibt es in COB1 nicht.
4. Datennamen müssen in LEVEL II in der FILE STATUS-Klausel als alphanumerisch definiert werden. In COB1 numerisch oder alphanumerisch, können aber nur einen Kennzeichner haben.
5. Der Datenname in der RECORD KEY-Klausel in COB1 kann einen Bezeichner haben, darf aber nicht subskribiert oder indiziert sein. In LEVEL II muß er alphanumerisch sein.
6. Die RESERVE...ALTERNATE AREA-Klausel existiert nicht in LEVEL II und die ganze RESERVE-Klausel wird hier nur für Dokumentationszwecke benutzt.
7. Die TRACK-AREA, FILE-LIMIT und PROCESSING-MODE-Klauseln sind in COB1 enthalten, werden als Kommentare geführt und existieren in LEVEL II gar nicht.

Die ASSIGN-Klausel

Die COB1 und LEVEL II-Formate dieser Anweisung sind ungleich, wobei in LEVEL II die Dateiidentifikation ins Programm eingeschlossen ist und in COB1 die Information durch Dateikommandos während der Laufzeit ausgegeben werden.
Das einfachste COB1 Format ist

```
ASSIGN TO DA-I-SYSnnn
```

wobei nnn eine ganze Zahl nicht größer als 099 ist.
Das LEVEL II Format ist

```
ASSIGN TO externes dateiname-literal
```

oder

```
ASSIGN TO dateibezeichner,
```

wobei dateibezeichner ein vom Benutzer definiertes Wort ungleich des Dateinamen (im SELECT) ist.
Die Angabe FOR MULTIPLE UNIT wird von LEVEL II nicht erkannt.

I-O CONTROL-Paragraph

1. Die RETURN-Klausel wird in LEVEL II als Dokumentation behandelt.
2. Die APPLY-Klausel aus COB1 gibt es in LEVEL II nicht.

FILE SECTION

1. Die LABEL RECORDS-Klausel ist in COB1 notwendig, aber in LEVEL II nur optional, wenn der ANSI-Schalter nicht gesetzt ist (siehe Kapitel 2) Die Form LABEL RECORD(S) OMITTED ist in LEVEL II erlaubt, aber nicht in COB1 bei indizierten Dateien. Die Klausel wird in LEVEL II als Dokumentation behandelt.
2. Die RECORDING MODE-Klausel existiert nicht in LEVEL II.
3. Die BLOCK CONTAINS-Klausel ist für Dokumentationszwecke in LEVEL II, aber in COB1 wird sie zusammen mit der RECORDING MODE-KLausel benutzt, um die maximale Blocklänge festzulegen.
4. Die RECORD-Klausel der Form 'RECORD...VARYING...DEPENDING...' gibt es in COB1, nicht aber in LEVEL II. Die Klausel ist bei LII nur Dokumentation. Der COB1-Übersetzer benutzt sie als Vergleich mit der folgenden Datensatzerklärung.
5. Die VALUE OF-Klausel ist in COB1 nicht für indizierte Dateien erlaubt.

PROCEDURE DIVISION

Die DELETE-Anweisung

Für Dateien im sequentiellen Zugriffsmodus besitzt COB1, nicht LEVEL II, die Einschränkung, daß RELATIVE KEY nicht zwischen der vorher angegebenen READ-Anweisung und der DELETE-Anweisung geändert werden darf.

Die OPEN-Anweisung

1. Bei LEVEL II bewirkt die OPEN-Anweisung eine Überprüfung des ASSIGN-Namens, bei COB1 wird der zu benutzende Dateiname extern angegeben.
2. Bei einer OPEN-, I-O- oder EXTEND-Anweisung verlangt COB1 eine existierende Datei, während LEVEL II eine Datei anlegt, falls keine existiert.

Die READ-Anweisung

1. In LEVEL II darf die INTO-Angabe nicht für solche Dateien benutzt werden, die mit verschiedenen Satzlängen beschrieben sind.
2. Im Format 2 der Anweisung gibt es bei COB1 die optionale Angabe 'KEY IS datenname' nicht, da es 'ALTERNATE RECORD KEY' auch nicht gibt.

Die REWRITE-Anweisung

1. Für Dateien in sequentiellem Zugriff existiert eine COB1 Einschränkung, nämlich daß der RECORD KEY nicht zwischen dem vorher geforderten READ und dem REWRITE geändert werden darf.

Die START-Anweisung

Während in COB1 der Datenname der START-Anweisung der Name des RECORD KEY für die Datei sein muß, kann es in LEVEL II auch ein zu diesem Schlüssel untergeordnetes Feld sein, vorausgesetzt, das Feld beginnt mit dem am weitesten links stehenden Zeichen vom Schlüsseldatenfeld.

Die USE-Anweisung

Diese Anweisung hat in COB1 zwei getrennte Formate.

1. Das erste Format gibt die Routinen zur Behandlung der Benutzerkennsätze der Dateien an.
2. Das zweite COB1 Format ist identisch zum LEVEL II Format, allerdings erlaubt der COB1 Übersetzer einige Abweichungen von der Regel. Diese sind:
   Die USE-Prozedur kann mit einem GO TO oder EXIT PROGRAM enden und ein CALL enthalten.
   Auf Prozedurnamen der USE-Anweisung kann mit GO TO verzweigt werden.
3. Im Format 2 erlaubt COB1 nicht mehr als 2 USE-Anweisungen für mehrere Dateinamen, egal ob implizit oder explizit.

Die WRITE-Anweisung

COB1 hat die Einschränkung, daß ein Datensatzname in der Anweisung nicht von einem Dateinamen bezeichnet werden darf.

Kapitel 7 Sortieren und Mischen

FILE CONTROL-Paragraph

1. In LEVEL II ist das Format einer ASSIGN-Klausel für eine Sortier-Mischdatei identisch mit anderen Dateien. In COB1 sind folgende Formate außerdem noch vorhanden für Sortier-Mischdatei:

   SELECT dateiname ASSIGN TO [ganzzahl] SORT TAPES bzw. DISC.

   Die COB1 SORT-Routine weist den nötigen Arbeitsbereich zu.

2. Für ein COB1-Programm, das SORT benutzt, sollen die folgenden Dateinamen nicht in der SELECT-Klausel benutzt werden:
   ```
   SORTIN
   SORTIN xx (xx=01,...,15)
   SORTOUT
   SORTWK
   SORTWK xx (xx=01,...,15)
   SORTCKPT
   ```

I-O CONTROL-Paragraph

1. Die COB1-Klausel

   APPLY CHECKPOINT ON sortiername

   und

   RERUN ON dateiname EVERY SORT ...

   gibt es in LEVEL II nicht.

2. LII COBOL stellt bei Ausführung der SAME SORT[-MERGE] AREA-Klausel einen Bezug zu den in der Klausel genannten Dateien her.

FILE SECTION

1. Die RECORDING MODE-Klausel existiert in LEVEL II nicht.
2. Für eine Sortierdatei ist die LABEL RECORDS-Klausel optional in COB1 (Fehler 'OMITTED'), in LEVEL II gibt es sie nicht.

Die MERGE-Anweisung

1. In LEVEL II sollte eine MERGE-Anweisung nicht in einer Ein-Ausgabeprozedur erscheinen, die mit einer SORT-oder MERGE-Anweisung verbunden ist. Die entsprechende Einschränkung in COB1 enthält nur, daß eine MERGE-Anweisung nicht in der Ausgabeprozedur, die zu der MERGE-Anweisung gehört, erscheinen soll.
2. In COB1 gibt es folgende Einschränkungen für Sortierschlüssel:

   a) Ein Maximum von 12 sollte für jede Einzeldatei angelegt werden.
   b) Die gesamte Länge aller Schlüssel darf 256 nicht überschreiten.
   c) Schlüssel, die in einem gegebenen Sortierlauf sortiert werden, müssen vom Beginn des Dateisatzes an gleiche Abstände haben. Der Abstand darf 4092 Bytes nicht überschreiten.
   d) Ein gepackter dezimaler Sortierschlüssel sollte nicht mehr als 16 Ziffern haben.

3. In LEVEL II sollen die Dateinamen in der MERGE-Anweisung nicht mehr wiederholt werden. Wenn die Dateien bei der USING-Angabe nicht so eingerichtet sind wie in der KEY-Klausel beschrieben, sind die Ergebnisse nicht vorhersagbar.

4. COB1, nicht LEVEL II, erlaubt ALTER-, GO TO- und PERFORM-Anweisungen in einer Ausgabeprozedur, um auf Prozedurnamen außerhalb dieser Prozedur hinzuweisen und erlaubt das Verzweigen von außen in diese Prozedur, vorausgesetzt die Verzweigung greift nicht in eine RERUN-Anweisung oder das Ende der Prozedur ein.
5. Für segmentierte Programme fordert LEVEL II, wenn eine MERGE-Anweisung in einem unabhängigen Segment ist, daß jede Ausgabeprozedur entweder im selben Segment oder ganz im residenten Segment ist. Wenn eine MERGE-Anweisung in einem residenten Segment ist, muß die Ausgabeprozedur entweder ganz im residenten Segment oder in einem einzelnen unabhängigen Segment sein.
6. In der 'GIVING'-Angabe erlaubt LEVEL II nur einen Dateinamen, COB1 verschiedene.

Die RERUN-Anweisung

In LEVEL II darf die 'INTO'-Angabe nicht benutzt werden, wenn die Inputdatei aus Datensätzen besteht, die mit verschiedenen Größen beschrieben sind.

Die SORT-Anweisung

1. Die COB1 'DUPLICATES'-Angabe gibt es in LEVEL II nicht.
2. Die folgenden Einschränkungen werden bei COB1 auf die Sortierschlüssel angewendet:
   a) Maximal 12 Sortierschlüssel pro Datei.
   b) Die gesamte Länge aller Schlüssel darf 256 nicht überschreiten.
   c) Schlüssel, die in einem gegebenen Sortierlauf sortiert werden,

      müssen vom Beginn des Datensatzes an in gleichen Abständen sein.
      Der Abstand darf 4092 Bytes nicht überschreiten.

   d) Ein gepackter dezimaler Sortierschlüssel sollte nicht mehr als 16 Ziffern haben.

3. In LEVEL II darf die SORT-Anweisung nicht in einer Ein-Ausgabeprozedur erscheinen, die zur MERGE-Anweisung gehört.
4. COB1, aber nicht LEVEL II, erlaubt eine MERGE-Anweisung in einer Ein-Ausgabeprozedur.
5. COB1, nicht LEVEL II, erlaubt ALTER-, GO TO- und PERFORM-Anweisungen in einer Ausgabeprozedur, um auf Prozedurnamen außerhalb dieser Prozedur hinzuweisen und erlaubt das Verzweigen von außen in diese Prozedur, vorausgesetzt die Verzweigung greift nicht in eine RETURN-Anweisung oder das Ende der Prozedur ein.
6. Ist in einem segmentierten LII-Programme eine SORT-Anweisung in einem unabhängigen Segment, muß ihre Ein-Ausgabeprozedur entweder im selben Segment oder ganz im residenten Segment sein. Wenn eine SORT-Anweisung in einem residenten Segment liegt, dann muß jede darauf bezogene Ein-Ausgabeprozedur entweder ganz im residenten Segment oder in einem einzelnen unabhängigen Segment sein.
7. LEVEL II läßt nur einen Dateinamen in der 'GIVING'-Angabe zu, während COB1 mehrere erlaubt.

Sonderregister

Die vier Sonderregister, die in COB1 zum Gebrauch mit der SORT-Anweisung zur Verfügung stehen, gibt es in LEVEL II nicht.

Kapitel 8 Segmentieren

1. LEVEL II fordert, daß alle Segmente, die als permanent angegeben werden, zusammenhängend im Quellprogramm stehen.
2. In LEVEL II müssen Kapitel in DECLARATIVES Segmentnummern unter 50 haben. In COB1 haben die Kapitel keine Segmentnummer und Null wird akzeptiert.
3. Die SEGMENT-LIMIT-Klausel wird in LEVEL II als Dokumentation behandelt.

Einschränkungen beim Programmüberlauf

1. Die LEVEL II-Einschränkungen der PERFORM-Anweisung sind:

   a) Anweisungen in einem residenten Segment sollen nur Bezug nehmen auf Kapitel und/oder Paragraphen, die ganz im residenten Segment beinhaltet sind, oder Kapitel und/oder Paragraphen, die ganz im einzelnen unabhängigen Segment beinhaltet sind.
   b) Anweisungen in einem unabhängigen Kapitel sollen nur Bezug nehmen auf Kapitel und/oder Paragraphen im gleichen Kapitel oder Kapitel und/oder Paragraphen, die in einem oder mehreren residenten Segmenten liegen.

   a) und b) gilt nicht für COB1.

2. Die LEVEL II Einschränkungen der SORT/MERGE-Anweisung:

   a) Für Anweisungen in einem residenten Segment muß jede Ein-Ausgabeprozedur ganz im residenten Segment oder ganz in einem einzelnen unabhängigen Segment enthalten sein.
   b) Für Anweisungen in einem unabhängigen Segment muß jede Ein-Ausgabeprozedur ganz in einem residenten Segment oder ganz im selben unabhängigen Segment liegen wie die SORT/MERGE-Anweisung.

   a) und b) gilt nicht für COB1.

3. Ein Programmaufruf in COB1 durch die CALL-Anweisung sollte seinen Eingangspunkt nur in einem Segment haben.

Zusätzliche indirekte Codedateien

LEVEL II erzeugt zusätzlich Dateien für segmentierte Programme wie folgt:

dateiname.Inn	eine für jedes unabhängige Segment
dateiname.ISR	Intersegment REFERECE Tabelle eins pro segmentiertem Progamm
dateiname.Dnn	Dictionary Dateien, eine für jedes unabhängige Segment außer dem letzten

dateiname ist der Name von der hauptsächlichen indirekten Codedatei, nn ist die Segmentnummer

Abgesehen von den datenname.Dnn Dateien, die nach der Übersetzung gelöscht werden können, müssen diese Dateien als Teil des Objektprogramms vorhanden sein.

Kapitel 9 Unterprogrammtechnik

Die Prozedur DIVISION HEADER

1. Bei COB1 muß die Zahl der Operanden in der PROCEDURE DIVISION USING-Klausel und in der CALL-Anweisung USING-Klausel gleich sein. LEVEL II erlaubt unterschiedliche Anzahl. In diesem Fall werden Operanden von links nach rechts verglichen. (Ungleiche Operanden in der CALL-Anweisung werden einfach ignoriert, aber die im PROCEDURE DIVISION HEADER bedeuten nicht vorhandene Datenfelder und bewirken einen Laufzeitfehler.)
2. Um ein aufgerufenes Programm zu erkennen, verlangt COB1 entweder die USING-Klausel im PROCEDURE DIVISION HEADER oder eine EXIT PROGRAM-Anweisung.

Die CALL-Anweisung

1. Format 2 der Anweisung, wie sie in LEVEL II benutzt wird z.B.

   CALL literal/bezeichner USING...

   wobei literal/bezeichner einen numerischen Wert hat. Ist in COB1 nicht vorhanden.

2. Die ON OVERFLOW-Klausel ist in LEVEL II, aber nicht in COB1 vorhanden.

3. In LEVEL II ist der Name des aufgerufenen Programms der Name der Datei, die das aufgerufene Programm beinhaltet. LEVEL II erlaubt einen Bezeichner, um ein Datenfeld zu zeigen, das anstelle eines Literals benutzt wird.

   Siehe LII COBOL Bedienungsanleitung!

   COB1 fordert von diesem Literal, das der Name im PROGRAM-ID-Paragraph (oder in der ENTRY-Anweisung) eines aufgerufenen Programms ist, daß es ein gültiger Programmname ist: Mehr als acht Zeichen lang, eindeutig in der Gruppe von Programmen, die zusammengebunden werden und ungleich den ersten acht Zeichen des Namens vom aufgerufenen Programm. Eine ENTRY-Anweisung wird benutzt, wenn der Eingangspunkt nicht nicht der Anfang des aufgerufenen Programms ist.

4. Bei LEVEL II muß in der Angabe 'USING datenname' datenname dann die Stufennummer 01 oder 77 haben, wenn der ANSI-Schalter gesetzt ist.

   In COB1 ist jede Stufennummer, außer 88 erlaubt.
   Anstelle vom Datennamen erlaubt COB1 genauso einen Bezeichner oder, wenn das aufgerufene Programm kein COBOL ist, einen Datei- oder Prozedurnamen. LEVEL II erlaubt die Qualifikation eines Datennamen, wenn er in der FILE SECTION definiert ist.

Die CANCEL-Anweisung

Diese Anweisung ist in COB1 nicht vorhanden.

Die ENTER-Anweisung

Diese Anweisung in der Form

'ENTER sprachenname [routinename]'

wird von LEVEL II erkannt, aber als Dokumentation behandelt. Dies ist syntaktisch mit dem COB1-Format der Anweisung vergleichbar z.B.

ENTER sprachenname eingangsname [USING bezeichner...],

z.B. 'ENTER LINKAGE.

verb.

wobei 'verb' für CALL, ENTRY oder RETURN steht.

In COB1 kann der Einsprungpunkt des Unterprogramms woanders liegen als der Programmbeginn.

Die ENTRY-Anweisung

Diese Anweisung, die in LEVEL II nicht existiert, wird in COB1 in einem Unterprogramm benutzt, um den Eingangspunkt eines Aufrufs anzugeben, wenn der Standardeingangspunkt, Beginn eines Programms, nicht benutzt wird.

Die EXIT PROGRAM-Anweisung

COB1 hat eine ANS-74 Standardeinschränkung, nämlich, daß die Anweisung in einem Satz erscheinen und dies der einzige Satz in diesem Paragraph sein muß. Für LEVEL II gelten diese Regeln nur, wenn der ANSI-Schalter gesetzt ist.

Kapitel 10 Bibliothek

Eine LII-Bibliothekquelldatei erstellt man mit einem Texteditor und erzeugt pro Bibliothek eine eigene Datei. Bei COB1 benutzt man das Dienstprogramm COBLUR und kann mehrere COPYElemente in einer Bibliothek abspeichern.

Die COPY-Anweisung

1. Das LEVEL II-Format

   COPY externes bibliotheksname-literal

   bei dem das alphanumerische Literal aus einem Dateinamen (in Apostrophen) besteht, gibt es bei COB1 nicht.

2. Die 'pseudo-text' Option ist in COB1 nicht vorhanden.

3. In COB1 muß der 'Textname' mit einem alphanumerischen Zeichen beginnen. Genauso müssen, da nur die ersten acht Zeichen vom System benutzt werden, diese innerhalb der Bibliothek eindeutig sein.

4. In COB1 beinhaltet die REPLACING-Angabe das Wort COPY nicht.

5. COB1 gibt den Standardverbindungsnamen COBLIB der Bibliothek an, wenn die 'OF/IN'-Angabe weggelassen wird.

Kapitel 11 Testhilfen und interaktive Testhilfen

Zusammenfassung

LEVEL II unterstützt das ANS-74 Standard-Testmodul. Ein interaktives Laufzeittestpaket ist vorhanden.
COB1 unterstützt die ANS-74 Standard-Testhilfezeilen und die WITH DEBUGGING MODE-Klausel. Es hat drei zusätzliche Reihentestanweisungen: EXHIBIT, ON und TRACE.

Die WITH DEBUGGING MODE-Klausel

In COB1 wird mit dieser Klausel das Übersetzen aller Testhilfezeilen bewirkt.
LEVEL II unterstützt USE FOR DEBUGGING, so daß diese Anweisungen ebenfalls übersetzt werden, wenn die WITH DEBUGGING MODE-Klausel angegeben ist.

Der COBOL DEBUG OBJECT TIME-Schalter

In LEVEL II muß dieser Schalter zusätzlich auf 'ON' gesetzt werden, damit die DEBUGGING MODE-Klausel auch wirkt. Der Schalter ist in COB1 nicht vorhanden.

Die USE FOR DEBUGGING-Anweisung

Diese Anweisung gibt es in COB1 nicht.

Das Sonderregister DEBUG-ITEM

Das Register DEBUG, das in LEVEL II mit Testhilfekapiteln benutzt wird, die in der USE FOR DEBUGGING-Anweisung angegeben sind, ist in COB1 nicht vorhanden.

Die EXHIBIT-Anweisung

Diese Anweisung, die in COB1 benutzt wird, um aktuelle Datenfelder zu zeigen, ist in LEVEL II nicht vorhanden.

Die ON-Anweisung

Diese COB1-Anweisung, die aus einer Anweisung besteht, die ausgeführt wird, wenn eine numerische Bedingung erfüllt ist, ist in LEVEL II nicht vorhanden.

Die TRACE-Anweisung

Die COB1 TRACE-Anweisung, die die TRACE-Funktion aufruft oder beendet, gibt es in LEVEL II nicht.

2 Aufbau der LEVEL II COBOL-Sprache

2.1 Programmaufbau

Ein LII COBOL-Programm besteht aus vier DIVISIONs:

1. IDENTIFICATION DIVISION
   Erkennungsteil des Programms
2. ENVIRONMENT DIVISION
   Beschreibung der Anlagenausstattung, die für die Übersetzung und Ausführung des Programms eingesetzt wird.
3. DATA DIVISION
   Beschreibung der Daten, die verarbeitet werden.
4. PROCEDURE DIVISION
   Eine Anzahl von Prozeduren, die die Operationen definieren, die mit den Daten durchgeführt werden.

Jede DIVISION ist in SECTIONs aufgeteilt, die in Paragraphen unterteilt sind, die wiederum aus Sätzen bestehen.

Die "ANSI-SCHALTER" Übersetzungsanweisung

Einige der standardmäßigen Anweisungen, die gemäß einer strengen ANSI-Interpretation erforderlich sind, müssen bei LII COBOL nicht angegeben werden.
Sie können Ihr Programm auf 2 Arten übersetzen:

1. Ohne FLAG-Direktive
   Da bei LII COBOL einige SECTIONS und Anweisungen nicht geschrieben werden müssen, können sie ohne Fehlermeldung weggelassen werden.

2. Mit FLAG-Direktive
   In diesem Fall besteht der Übersetzer auf einer strengen ANSI-Interpretation.
   Es werden alle LII COBOL-Erweiterungen mit Fehlerprotokoll dokumentiert.

Die FLAG-Direktive ist in der LII COBOL Bedienungsanleitung beschrieben.

Die bei LII COBOL optionalen SECTIONS und Anweisungen werden hier in diesem Handbuch mit [-]- dargestellt. An anderen Stellen wird auf den ANSI-Schalter hingewiesen.

2.2 IDENTIFICATION DIVISION

Allgemeine Beschreibung

Diese IDENTIFICATION DIVISION identifiziert sowohl das Quellprogramm als auch das daraus resultierende Ausgabe-Protokoll. Ferner kann der Anwender das Datum, an dem das Programm geschrieben wurde, das Datum, an dem die Übersetzung des Quellprogramms abgeschlossen war und ähnliche andere Informationen nach Wunsch unter den Paragraphen im unten gezeigten allgemeinen Format mitaufnehmen.

Organisation

Die Überschriften der Paragraphen identifizieren die Art der Information, die im Paragraphen enthalten ist. Der Name des Programms muß im ersten Paragraphen angegeben werden, der PROGRAM-ID Paragraph genannt wird. Die anderen Paragraphen können in dieser DIVISION nach Belieben des Anwenders in der Reihenfolge, wie im untenstehenden Format gezeigt, mitaufgenommen werden.

Aufbau

Im Folgenden wird das allgemeine Format der Paragraphen in der IDENTIFICATION DIVISION gezeigt. Dieses Format definiert die Reihenfolge der Darstellung in einem Quellprogramm.

IDENTIFICATION DIVISION

Allgemeines Format

```
┼ IDENTIFICATION DIVISION.]
┼ PROGRAM-ID.              programmname.┼
[ AUTHOR.                [ kommentar ┼  ... ]
[ INSTALLATION.          [ kommentar ]  ... ]
[ DATE-WRITTEN.          [ kommentar ]  ... ]
[ DATE-COMPILED.         [ kommentar ]  ... ]
[ SECURITY.              [ kommentar ]  ... ]
```

Syntaxregeln

1. Die IDENTIFICATION DIVISION muß mit den reservierten Wörtern IDENTIFICATION DIVISION beginnen, gefolgt von einem Punkt und einem Leerzeichen.
2. Die Kommentareintragung kann aus einer beliebigen Kombination von Zeichen aus dem Zeichenvorrat des Computers bestehen und kann in Bereich B auf einer oder mehreren Zeilen geschrieben werden. Die Fortsetzung der Kommentareintragung ist möglich, der Bindestrich im Indikatorbereich ist nicht zulässig.

Der PROGRAM-ID-Paragraph

Funktion

Im PROGRAM-ID-Paragraph steht der Name, durch den ein Programm identifiziert wird.

Allgemeines Format

```
PROGRAM-ID.    programmname.
```

Syntaxregel

programmname muß den Regeln für die Bildung eines Programmierer-wortes entsprechen.

Allgemeine Regeln

1. Der PROGRAM-ID-Paragraph muß den Namen des Programms enthalten und **muß** in jedem Programm, das mit Hilfe der **ANSI-Steueranweisung** übersetzt ist, enthalten sein. Wenn diese Steueranweisung nicht angegeben ist, so ist der Paragraph optional.
2. Der Programm-Name identifiziert das Quellprogramm und alle Protokolle dieses Programms.

Der DATE-COMPILED-Paragraph

Funktion

Der DATE-COMPILED Paragraph liefert das Übersetzungsdatum in der IDENTIFICATION DIVISION des Quellprogramm-Protokolls.

Allgemeines Format

```
DATE-COMPILED.      [kommentareintragung ...]
```

Syntaxregel

Die Kommentareintragung kann aus einer beliebigen Kombination von Zeichen aus dem Zeichenvorrat des Computers bestehen. Der Kommentar kann auf mehreren Zeilen stehen, der Bindestrich im Indikatorbereich ist aber dafür nicht zulässig.

Allgemeine Regel

Der Paragraph-Name DATE-COMPILED führt dazu, daß eine Datumsangabe während der Programmübersetzung eingefügt wird; es kann ein Kommentar dazu eingetragen werden. Wenn eine Übersetzer-Anweisung vorliegt, so wird der DATE-COMPILED Kommentareintragung völlig durch die in der Kommandozeile angegebene Datumszeichenfolge ersetzt.

In der LII COBOL Bedienungsanleitung finden Sie Einzelheiten über die Kommentareintragungs-Ersetzungszeichenfolge.

2.3 ENVIRONMENT DIVISION

Allgemeine Beschreibung

Die ENVIRONMENT DIVISION ermöglicht Angaben über die Konfiguration des SOURCE-COMPUTERS und des OBJECT-COMPUTERS. Ferner werden Informationen über die Eingabe/Ausgabe-Steuerung, spezielle Hardware-Eigenschaften und Steuerungstechniken gegeben.

Organisation

Die ENVIRONMENT DIVISION besteht aus zwei SECTIONs: Die CONFIGURATION-SECTION und die INPUT-OUTPUT-SECTION.

Die CONFIGURATION-SECTION behandelt die Eigenschaften des SOURCE-COMPUTERs und des OBJECT-COMPUTERs. Diese SECTION ist unterteilt in drei Paragraphen:

- dem Paragraphen SOURCE-COMPUTER, in dem die Computer-Konfiguration beschrieben wird, auf dem das Quellprogramm übersetzt wird;
- dem Paragraphen OBJECT-COMPUTER, in dem die Computer-Konfiguration beschrieben wird, auf der das Zielprogramm, das vom Übersetzer erzeugt wurde, ablaufen soll
- dem Paragraphen SPECIAL NAMES, der eine Beziehung zwischen den vom Übersetzer verwendeten Herstellerwörtern und den im Quellprogramm verwendeten Merknamen herstellt.

In der INPUT-OUTPUT-SECTION steht die Information, die für die Steuerung des Programmablaufs und die Bearbeitung der Daten zwischen physikalischen Geräten und dem Zielprogramm erforderlich ist. Diese SECTION ist in zwei Paragraphen aufgeteilt: den Paragraphen FILE-CONTROL, der Dateien mit peripheren Geräten verknüpft und diese benennt, sowie den Paragraphen I-O-CONTROL, der spezielle Steuerungstechniken definiert, die im Zielprogramm verwendet werden sollen.

Aufbau

Im Folgenden wird das allgemeine Format der SECTIONs und Paragraphen in der ENVIRONMENT DIVISION gezeigt, das die Reihenfolge der Darstellung im Quellprogramm definiert:

Allgemeines Format

```
┼   ENVIRONMENT DIVISION. ┼

┼   CONFIGURATION SECTION. ┼

┼   SOURCE-COMPUTER.    source-computer-eintrag  ┼

┼   OBJECT-COMPUTER.    object-computer-eintrag  ┼

[   SPECIAL-NAMES.      sonder-namen-eintrag     ]

[ ┼ INPUT-OUTPUT SECTION. ┼

    FILE-CONTROL. {datei-steuerungseintrag}  ...
   [I-O-CONTROL.  eingabe-/ausgabe-steuerungseintrag. ] ]
```

CONFIGURATION SECTION

Der SOURCE-COMPUTER-Paragraph

Funktion

Der SOURCE-COMPUTER Paragraph identifiziert den Computer, auf dem das Programm übersetzt werden soll.
Er dient nur der Dokumentation.

Allgemeines Format:

```
SOURCE-COMPUTER.    anlagenname.
```

Syntaxregel:

anlagenname muß ein COBOL Wort sein, das vom Anwender definiert wird.

Allgemeine Regeln

anlagenname ist ein Hilfsmittel, die Anlagenkonfiguration zu identifizieren, wobei der Computer-Name und seine implizierte Konfiguration vom Anwender angegeben werden.

Der OBJECT-COMPUTER-Paragraph

Funktion:

Der OBJECT-COMPUTER Paragraph identifiziert den Computer, auf dem das Programm ausgeführt werden soll. anlagenname und die MEMORY SIZE-Klausel dienen nur der Dokumentation.

Allgemeines Format:

```
OBJECT-COMPUTER. anlagenname [ , MEMORY SIZE bezeichner { WORDS      } ]
                                                        { CHARACTERS }
                                                        { MODULES    }
    [ ,PROGRAM COLLATING SEQUENCE IS alphabet-name].
```

Syntaxregel:

1. anlagenname muß ein COBOL Wort sein, das vom Anwender definiert wird.

Allgemeine Regeln

1. Der Anlagenname ist ein Hilfsmittel zur Identifizierung der Anlagenkonfiguration, wobei der Anlagenname und dessen implizierte Konfigurationen vom Anwender angegeben werden. Die Konfigurationsdefinition enthält besondere Information über die Speichergröße.
2. Wenn die PROGRAM COLLATING SEQUENCE-Klausel angegeben ist, wird die mit dem alphabetnamen verknüpfte Sortierfolge dazu verwendet, den Wahrheitswert von nichtnumerischen Vergleichen zu bestimmen:
   a) Ausdrücklich angegeben in Vergleichsbedingungen (vgl. Vergleichsbedingung in diesem Kapitel).
   b) Ausdrücklich angegeben in Bedingungsnamen-Bedingungen; vgl. Bedingungsnamen-Bedingungen (Bedingungsvariable).
3. Wenn die PROGRAM COLLATING SEQUENCE-Klausel nicht angegeben ist, so wird die maschineneigene Sortierfolge verwendet.

4. Wenn die PROGRAM COLLATING SEQUENCE-Klausel angegeben ist, ist die Programmsortierfolge, die mit dem in dieser Klausel angegebenen alphabetnamen verknüpfte Sortierfolge.
5. Die PROGRAM COLLATING SEQUENCE-Klausel wird auch bei nichtnumerischen Misch- oder Sortier-Schlüsseln angewandt, sofern nicht die COLLATING SEQUENCE-Angabe der entsprechenden SORT- oder MERGE-Anweisung angegeben ist.
6. Die PROGRAM COLLATING SEQUENCE-Klausel ist nur in dem Programm anwendbar, in dem sie angegeben ist.

Der SPECIAL-NAMES-Paragraph

Funktion

Der SPECIAL-NAMES Paragraph setzt Herstellerwörter und vom Anwender definierte Merknamen sowie Alphabetnamen mit Zeichensätzen und/oder Sortierfolgen zueinander in Beziehung.

Allgemeines Format:

```
SPECIAL-NAMES.

[ funktionsname-1 IS merkname-1 ] [ funktionsname-2 IS merkname-2 ] ...

 SWITCH { 0 } [IS merkname-3]   ,ON STATUS IS bedingungsname-1
        { 1 }
        { 2 }
        { 3 }
        { 4 }
        { 5 }
        { 6 }
        { 7 }
                                [ ,OFF STATUS IS bedingungsname-1]

 [,alphabetname IS
    { STANDARD-1                                                    }
    { NATIVE                                                        }
    { literal-1  [ {THROUGH} literal-2 ALSO literal-3               ] }
    {            [ {THRU   }                                        ] } ...
    {                       [,ALSO literal-4] ...                   }
    {  [ literal-5 [ {THROUGH} literal-6  ALSO literal-7 ]]         }
    {              [ {THRU   }            [,ALSO literal-8]... ]]...}

   [  , CURRENCY SIGN IS literal-9 ]

   [  , DECIMAL-POINT IS COMMA ]

   [  , CONSOLE IS CRT ]

   [  , CURSOR IS datenname-1 ] .
```

Syntaxregeln:

1. Merknamen können irgendwelche COBOL-Programmiererwörter sein; mindestens ein Zeichen davon muß ein alphabetisches Zeichen sein.
2. Die in der literal-Angabe in der alphabetname-Klausel genannten Literale müssen:
   a) wenn sie numerisch sind, Ganzzahlen (Integer-Größen) ohne Vorzeichen sein und einen Wert zwischen eins (1) und der maximalen Anzahl von Zeichen im maschineneigenen Zeichenvorrat annehmen,
   b) wenn sie nichtnumerisch und verknüpft mit einer THROUGH- oder ALSO-Angabe sind, jeweils ein Zeichen lang sein.
3. Wenn die literal-Angabe in der alphabetname-Klausel genannt ist, so darf ein vorgegebenes Zeichen nicht mehr als einmal in einer alphabetname-Klausel angegeben sein.
4. Die Worte THRU und THROUGH sind gleichbedeutend.

Allgemeine Regeln

1. funktionsname-1 spezifiert Systemgeräte oder Funktionen, die vom Übersetzer verwendet werden. Der Programmierer kann alle COBOL-Programmiererwörter mit einem Funktionsnamen verknüpfen. merkname-1, merkname-2 usw. können in ACCEPT, DISPLAY oder WRITE-Anweisungen verwendet werden. funktionsname-1, funktionsname-2 usw. kann einer der folgenden sein:

   SYSIN - logisches Systemeingabegerät: die Tastatur

   SYSOUT - logisches Systemausgabegerät: der Bildschirm

   FORMFEED - Normalerweise ein Seitenvorschub bis zum Anfang der nächsten Seite (WRITE ADVANCING)

   Es ist zu beachten, daß die FORMFEED Funktion dem ASCII Drukkersteuerzeichen 0C entspricht. Bei den meisten Druckern positioniert 0C das Papier an den Anfang einer Seite.

   Die genaue Zuordnung der Steuerzeichen zu den jeweiligen Druckern finden Sie in den Druckerbeschreibungen.

2. Mit der SWITCH-Klausel muß mindestens ein Bedingungsname verknüpft sein. Der Schalter wird vom Bediener zur Ausführungszeit gesetzt; der Schalter kann im Programm durch den Bedingungsnamen abgefragt werden. Die Schalter können während der Ausführung nicht geändert werden.

3. Mit der alphabetname-Klausel kann man einen Namen zu einem vorgegebenen Zeichenvorrat und/oder Sortierfolge in Beziehung setzen. Wenn alphabetname in der PROGRAM-COLLATING SEQUENCE-Klausel (vgl. der OBJECT COMPUTER-Paragraph in diesem Kapitel) oder aber der COLLATING SEQUENCE Angabe einer SORT- oder MERGE-Anweisung angegeben ist, so gibt die alphabetname-Klausel eine Sortierfolge an. Wenn alphabetname in einer CODE-SET-Klausel in einer Dateierklärung angegeben wurde, bezieht sie sich auf einen Zeichenvorrat.

   a) Wenn STANDARD-1 angegeben ist, so entspricht der identifizierte Zeichenvorrat oder die Sortierfolge dem American Standard Code for Information Interchange, X3.4-1968. In Abschnitt 1.4 wird die Beziehung zwischen den Zeichen des Standard-Zeichenvorrates und den Zeichen des maschineneigenen Zeichenvorrates definiert.

   b) Wenn NATIVE angegeben ist, so wird der maschineneigene Zeichenvorrat oder die maschineneigene Sortierfolge verwendet. Die maschineneigene Sortierfolge entspricht der ANSI Veröffentlichung X3.4-1968

4. Das Zeichen, das in der angegebenen Programm-Sortierfolge den höchsten Ordnungsplatz hat, wird mit der figurativen Konstante HIGH-VALUE verknüpft. Wenn mehr als ein Zeichen den höchsten Ordnungsplatz in der Programm-Sortierfolge hat, wird das letzte angegebene Zeichen mit der figurativen Konstante HIGH-VALUE verknüpft.

5. Das Zeichen, das den niedrigsten Ordnungsplatz in der angegebenen Programm-Sortierfolge hat, wird mit der figurativen Konstante LOW-VALUE verknüpft. Wenn mehr als ein Zeichen den niedrigsten Ordnungsplatz in der Programm-Sortierfolge hat, so wird das erste angegebene Zeichen mit der figurativen Konstante LOW-VALUE verknüpft.

6. Das Literal, das in der CURRENCY SIGN IS literal-Klausel erscheint, wird in der PICTURE-Klausel dazu benutzt, das Währungssymbol

darzustellen. Das Literal ist auf ein einziges Zeichen begrenzt und darf **nicht** eines der folgenden Zeichen sein:

- die Ziffern 0 bis 9;
- die Buchstaben A B C D L P R S V X Z oder das Leerzeichen;
- die Sonderzeichen * + - ' . ; ( ) " / =

Wenn diese Klausel nicht vorhanden ist, so wird nur das Währungszeichen in der PICTURE-Klausel verwendet.

7. Die Klausel DECIMAL-POINT IS COMMA bedeutet, daß die Funktionen von Komma und Punkt in der Zeichenfolge der PICTURE-Klausel und in numerischen Literalen gegeneinander ausgetauscht werden.

8. Die Klausel CONSOLE IS ändert den Standard in den ACCEPT- und DISPLAY-Anweisungen auf die LII COBOL interaktiven Erweiterungen. Dadurch ist es möglich, daß Daten an jedem beliebig angegebenen Punkt auf dem Bildschirm akzeptiert oder angezeigt werden können.

9. Die Klausel CURSOR IS gibt den Datennamen an, der die Cursor-Adresse, wie in der ACCEPT-Anweisung verwendet, enthält. Wenn CURSOR IS nicht angegeben ist, ist die Standard-Cursor-Position bei der Ausführung einer ACCEPT-Anweisung die 'Home' Position an der oberen linken Ecke des Bildschirmes. Die CURSOR IS-Klausel ermöglicht es, daß ein Programm die Position am Ende der Ausführung der letzten ACCEPT-Anweisung beibehält, oder die Anfangsposition zu Beginn irgendeiner ACCEPT-Anweisung angibt. Dies ist dann sinnvoll, wenn menü-orientiert programmiert wird. Der Bediener muß dann den Cursor nur an die gewünschte Stelle setzen und die RETURN-Taste drücken oder, bei der Standardoption, nur die RETURN-Taste drücken.

   Der Datenname enthält den Namen eines PICTURE 9999 Datenfeldes, in dem die beiden linken Stellen eine Zeilenangabe im Bereich eins bis 25 bedeutet und die beiden rechten Stellen eine Zeichenposition im Bereich eins bis 80 darstellt.
   Wenn der Datenname gleich Null ist, so hat dies die gleiche Wirkung, als ob die CURSOR IS-Klausel nicht verwendet wurde, d.h., die Cursor-Anfangsposition ist links oben auf dem Bildschirm.

Die Verwendung des Cursors ist ausführlich in der LII COBOL Bedienungsanleitung, Kapitel 4, beschrieben.

2.4 DATA DIVISION

Allgemeine Beschreibung

In der DATA DIVISION werden die Daten beschrieben, die das Zielprogramm als Eingabe akzeptieren, verarbeiten, erzeugen oder ausgeben sollen. Die zu verarbeitenden Daten werden in drei Kategorien aufgeteilt:

1. Daten, die in Dateien enthalten sind, und die in den internen Speicher des Computers von einem bestimmten Bereich oder bestimmten Bereichen gelesen bzw. zurückgeschrieben werden sollen.
2. Daten, die intern aufgebaut werden und sich im Zwischenspeicher oder Arbeitsspeicher befinden oder aber in ein bestimmtes Format für Ausgabezwecke gebracht werden.
3. Konstanten, die vom Anwender definiert werden.

Physikalische und logische Aspekte der Datenerklärung

DATA DIVISION Organisation

Die DATA DIVISION, die ohne den ANSI-Schalter wahlweise ist, wird in SECTIONs unterteilt, die FILE SECTION und die WORKING-STORAGE SECTION.

Die FILE SECTION definiert den Aufbau der Dateien. Jede Datei wird durch eine Dateierklärung und eine oder mehrere Datensatzerklärungen definiert. Datensatzerklärungen folgen unmittelbar nach der Dateibeschreibung.

Die WORKING-STORAGE SECTION beschreibt Datensätze und nicht benachbarte Datenfelder, die nicht Teil physikalischer Dateien sind, sondern intern entwickelt und verarbeitet werden. Hier werden auch die Datenfelder beschrieben, denen Werte im Quellprogramm zugewiesen werden, und die während der Ausführung des Zielprogramms nicht geändert werden.

Die LINKAGE SECTION erscheint in einem aufgerufenen Programm und beschreibt Datenfelder, die vom aufrufenden und vom aufgerufenen Programm angesprochen werden sollen. Der Aufbau ist der gleiche wie der der WORKING-STORAGE SECTION.

Bitte beachten Sie, daß die REPORT SECTION, SUB-SCHEMA SECTION und die COMMUNICATION SECTION bei LII COBOL nicht unterstützt werden!

Allgemeines Format

Nachfolgend wird das allgemeine Format der SECTIONs in der DATA DIVISION angegeben, das die Reihenfolge ihrer Darstellung im Quellprogramm definiert:

```
DATA DIVISION.

[ FILE SECTION.

  [dateibeschreibung [datensatzbeschreibung] ...                    ] ...
  [sort-merge-dateibeschreibung [datensatzbeschreibung] ...         ]

[ WORKING-STORAGE SECTION.

  [ 77-stufenbeschreibung ]
  [ datensatzbeschreibung ] ...

[ LINKAGE SECTION.

  [ 77-stufenbeschreibung ]
  [ datensatzbeschreibung ] ...
```

2.4.1 WORKING-STORAGE SECTION

Die WORKING-STORAGE SECTION besteht aus der Überschrift, gefolgt von Datenbeschreibungen für nicht benachbarte Datenfelder und/oder Datensatzbeschreibungen. Jeder Datensatz-Name in der WORKING-STORAGE SECTION und jeder nicht benachbarte Datenfeld-Name muß eindeutig sein, da er nicht qualifiziert werden kann. Untergeordnete Datennamen müssen nicht eindeutig sein, wenn sie durch Kennzeichnung eindeutig gemacht werden können.

Nicht benachbarte Felder in der WORKING-STORAGE SECTION

Datenfelder und Konstanten in der WORKING-STORAGE, die keine hierarchische Beziehung zueinander haben, müssen nicht in Datensätze zusammengefaßt werden, sofern sie nicht weiter unterteilt werden müssen. Sie werden stattdessen in Klassen eingeteilt und in einer separaten Datenbeschreibung definiert, die mit der Sonderstufennummer 77 beginnt.

Die folgenden Datenklauseln sind in jeder Datenbeschreibung erforderlich:

- Stufennummer 77
- datenname
- Die PICTURE-Klausel oder die USAGE IS INDEX-Klausel

Andere Datenerklärungsklauseln sind optional und dürfen verwendet werden, die Beschreibung des Datenfeldeszu vervollständigen, sofern dies notwendig ist.

WORKING-STORAGE-Datensätze

Datenelemente und Konstanten in der WORKING-STORAGE, die eine definierte hierachische Beziehung zueinander haben, müssen in Datensätze zusammengefaßt werden, entsprechend den Regeln für die Bildung von Datensatzerklärungen. Alle Klauseln, die in Datensatzbeschreibungen in der FILE SECTION verwendet werden, können auch in Datensatzerklärungen in der WORKING-STORAGE SECTION verwendet werden.

Anfangswerte

Der Anfangswert jedes Datenfeldes in der WORKING-STORAGE SECTION, mit Ausnahme des Index-Datenfeldes, wird unter Verwendung der VALUE-Klausel für das Datenfeld angegeben. Ist die VALUE-Klausel nicht angegeben, ist der Anfangswert unbestimmt. Der Anfangswert von allen Index-Datenfeldern kann erst in der PROCEDURE DIVISION gesetzt werden.

Die Datenbeschreibung - Vollständiges Eintragsschema

Funktion

Eine Datenbeschreibung gibt die Eigenschaften eines bestimmten Datenfeldes an.

Allgemeine Formate

Format 1

```
stufennummer { datenname-1 }
             { FILLER      }

   [ ; REDEFINES datenname-2]

   [ ; { PICTURE } IS picture-zeichenfolge ]
       { PIC     }

   [ ; [ USAGE IS ] { COMPUTATIONAL   } ]
                    { COMP            }
                    { COMPUTATIONAL-3 }
                    { COMP-3          }
                    { DISPLAY         }

     ; [ SIGN IS ] { LEADING  } [SEPARATE CHARACTER]
                   { TRAILING }

   [ ; { SYNCHRONIZED } [ LEFT  ] ]
       { SYNC         } [ RIGHT ]

   [ ; { JUSTIFIED } RIGHT ]
       { JUST      }

   [ ; BLANK WHEN ZERO ]

   [ ; VALUE IS literal ] .
```

Format 2:

```
66 datenname-1; RENAMES datenname-2 [ { [THROUGH] } datenname-3 ] .
                                     { [THRU]    }
```

Format 3:

```
88 bedingungsname; { [VALUE IS] } literal-1 [ { [THROUGH] } literal-2 ]
                   { [VALUES ARE] }           { [THRU]    }

   [ , literal-3 [ { [THROUGH] } literal-4 ] ] ... .
                 { [THRU]    }
```

Syntaxregeln

1. Die Stufennummer in Format 1 kann irgendeine Zahl zwischen 01 und 49 oder 77 sein.
2. Die Klauseln können mit zwei Ausnahmen in beliebiger Reihenfolge geschrieben werden: Unmittelbar nach der Stufennummer muß datenname-1 oder die FILLER-Klausel folgen; unmittelbar nach der datenname-1-Klausel muß, sofern verwendet, die REDEFINES-Klausel folgen.
3. Die PICTURE-Klausel muß für jedes Datenelement angegeben sein, außer dem Index-Datenfeld, bei dem die Verwendung dieser Klausel nicht zulässig ist.
4. Die Worte THRU und THROUGH sind gleichbedeutend.

Allgemeine Regeln

1. Die Klauseln SYNCHRONIZED, PICTURE, JUSTIFIED und BLANK WHEN ZERO dürfen nur für ein Datenelement angegeben werden.

2. Das Format 3 wird für alle Bedingungsnamen verwendet. Jeder Bedingungsname erfordert eine separate Eintragung mit Stufennummer 88. Das Format 3 enthält den Namen der Bedingung und den Wert, die Werte oder den Wertebereich, die mit dem Bedingungsnamen verknüpft sind. Die Bedingungsnameneintragung für eine bestimmte Bedingungsvaribale muß nach der Eintragungsbeschreibung stehen, die das Datenfeld beschreibt, das mit dem der Bedingungsnamen verknüpft ist. Ein Bedingungsname kann mit irgendeiner Datenerklärung verknüpft sein, die eine Stufennummer enthält, außer den folgenden:

   a) einem anderen Bedingungsnamen,

   b) einem Datenfeld der Stufe 66,

   c) einer Gruppe von Datenfeldern mit Beschreibungen, einschließlich JUSTIFIED, SYNCHRONIZED oder USAGE (jedoch nicht USAGE IS DISPLAY).

   d) einem Index-Datenfeld

Die BLANK WHEN ZERO-Klausel

Funktion

Die BLANK WHEN ZERO-Klausel druckt für ein Datenfeld Leerzeichen, wenn dessen Wert Null ist.

Allgemeines Format

```
BLANK WHEN ZERO
```

Syntaxregel

Die BLANK WHEN ZERO-Klausel kann nur für ein Datenelement verwendet werden, dessen PICTURE als numerisch angegeben ist (mit implizitem oder explizitem USAGE IS DISPLAY) oder das numerisch druckaufbereitet ist. (vgl. PICTURE-Klausel in diesem Kapitel)

Allgemeine Regeln

1. Wenn die BLANK WHEN ZERO-Klausel verwendet wird, enthält das Druckfeld nur Leerzeichen, wenn der Wert des Datenfelds Null ist.
2. Wenn die BLANK WHEN ZERO-Klausel für ein Datenfeld verwendet wird, dessen PICTURE numerisch ist, so wird die Kategorie des Datenfelds als numerisch druckaufbereitet angesehen.

Der Datenname oder die FILLER-Klausel

Funktion

Ein Datenname gibt den Namen von Daten an, die beschrieben werden. Das Wort FILLER gibt ein Datenelement oder Datengruppe des logischen Datensatzes an, auf das kein expliziter Bezug möglich ist.

Allgemeines Format

```
{datenname}
{FILLER   }
```

Syntaxregel

In den FILE, WORKING-STORAGE und LINKAGE SECTIONs muß ein Datenname oder das Schlüsselwort FILLER das erste Wort nach der Stufennummer in jeder Datenerklärung sein.

Allgemeine Regel

Das Schlüsselwort FILLER kann dazu verwendet werden, ein Datenelement oder eine Gruppe von Datenelementen in einem Datensatz zu benennen. Unter keinen Umständen kann ein FILLER-Datenfeld angesprochen werden. Das Schlüsselwort FILLER kann jedoch als eine Bedingungsvariable verwendet werden, da es sich in diesem Fall auf den Wert des Datenfelds und nicht auf das Datenfeld selbst bezieht.

Die JUSTIFIED-Klausel

Funktion

Die JUSTIFIED-Klausel gibt die nicht standardmäßige Positionierung von Daten innerhalb eines Empfangsdatenfeldes an.

Allgemeines Format

```
{JUSTIFIED}
{JUST     }   RIGHT
```

Syntaxregeln

1. Die JUSTIFIED-Klausel kann nur auf der Datenelementstufe angegeben werden.
2. JUST ist eine Abkürzung für JUSTIFIED.
3. Die JUSTIFIED-Klausel kann nicht für ein Datenfeld angegeben werden, das als numerisch beschrieben ist oder für die Druckaufbereitung angegeben ist.

Allgemeine Regeln

1. Wenn ein Empfangsdatenfeld mit der JUSTIFIED-Klausel beschrieben ist, und das Sendedatenfeld länger ist als das Empfangsdatenfeld, so werden die am weitesten links stehenden Zeichen abgetrennt.
   Wenn das Empfangsdatenfeld mit der JUSTIFIED-Klausel beschrieben ist und es länger als das Sendedatenfeld ist, so werden Daten an der am weitesten rechts stehenden Zeichenposition im Datenfeld ausgerichtet und die am weitesten links stehenden Zeichenpositionen werden mit Leerzeichen aufgefüllt.

   Es ist zu beachten, daß die Inhalte des Sendedatenfeldes nicht berücksichtigt werden, d.h., rechts stehende Leerzeichen werden nicht unterdrückt.

Beispiel

Wenn ein Datenfeld PIC X(4) dessen Wert A␣␣␣ (d.h. A gefolgt von drei Leerzeichen) in ein Datenfeld PIC X(6) JUSTIFIED gespeichert wird, ist das Ergebnis ␣␣A␣␣␣. Wenn das gleiche Datenfeld in eins mit PIC X(3) JUSTIFIED gespeichert wird, so ist das Ergebnis ␣␣␣, d.h., das am weitesten links stehende Zeichen wird abgetrennt.

2. Wenn die JUSTIFIED-Klausel weggelassen wird, so gelten die Standardregeln für die Ausrichtung von Daten innerhalb eines Datenelementes.

Stufennummer

Funktion

Die Stufennummer zeigt die Hierarchie der Daten innerhalb eines logischen Datensatzes. Ferner wird sie dazu verwendet, Eintragungen für WORKING-STORAGE-Datenfelder, LINKAGE-Datenfelder, Bedingungsnamen und für die RENAMES-Klausel zu identifizieren.

Allgemeines Format

```
stufennummer
```

Syntaxregeln

1. Als erstes Element in jeder Datenerklärung steht eine Stufennummer.
2. Datenerklärungen, die einer FD- oder SD-Eintragung untergeordnet sind, müssen Stufennummern mit den Werten 01 bis 49, 66 oder 88 besitzen.
3. Datenerklärungen in der WORKING-STORAGE SECTION und LINKAGE SECTION müssen Stufennummern mit den Werten 01 bis 49, 66, 77 oder 88 besitzen.
4. Eine Stufennummer kann eine einstellige oder zweistellige Zahl sein.

Stufennummer

Allgemeine Regeln

1. Die Stufennummer 01 identifiziert die erste Eintragung in jeder Datensatzerklärung.
2. Sonderstufen-Nummern sind besonderen Eintragungen zugeordnet, hierbei gibt es kein Stufenkonzept:
   a) Die Stufennummer 77 wird zugeordnet, um nicht benachbarte WORKING-STORAGE-Datenfelder oder nicht benachbarte LINKAGE Datenfelder zu identifizieren und kann nur im Format 1 der Datenerklärung verwendet werden.
   b) Die Stufennummer 66 wird zugeordnet, um RENAMES Eintragungen zu identifizieren und kann nur, im Format 2 verwendet werden.
   c) Die Stufennummer 88 wird Eintragungen zugeordnet, welche Bedingungsnamen, verbunden mit einer Bedingungsvariablen, definieren und kann nur im Format 3 verwendet werden.
3. Weitere Eintragungen auf Stufe 01, die irgendeiner gegebenen Stufenbezeichnung untergeordnet sind, stellen implizite Neudefinitionen des gleichen Bereiches dar.

Die PICTURE-Klausel

Funktion

Die PICTURE-Klausel beschreibt die allgemeinen Eigenschaften und Druckaufbereitungsangaben für ein Datenelement.

Allgemeines Format

```
{PICTURE}
{       } IS picture-zeichenfolge
{PIC    }
```

Syntaxregeln

1. Eine PICTURE-Klausel kann nur auf der Datenelementstufe angegeben werden.
2. Eine Zeichenfolge besteht aus bestimmten zulässigen Kombinationen von Zeichen aus dem COBOL Zeichenvorrat, die als Symbole verwendet werden. Die zulässigen Kombinationen bestimmen die Kategorie des Datenelementes.
3. Die maximal zulässige Anzahl von Zeichen in der Zeichenfolge ist 30.
4. Die PICTURE-Klausel muß für jedes Datenelement angegeben sein, außer einem Index-Datenfeld, bei dem die Verwendung dieser Klausel nicht zulässig ist.
5. PIC ist die Abkürzung für PICTURE.
6. Der Stern, wenn er als das Nullenunterdrückungssymbol verwendet wird, sowie die Klausel BLANK WHEN ZERO dürfen nicht in der gleichen Eintragung erscheinen.

Allgemeine Regeln

Es gibt fünf Kategorien von Daten, die mit einer PICTURE-Klausel beschrieben werden können: alphabetisch, numerisch, alphanumerisch, alphanumerisch druckaufbereitet und numerisch druckaufbereitet. Die allgemeinen Regeln innerhalb dieser Kategorien sind die folgenden:

Datenregeln bei alphabetischen Feldern

1. Die PICTURE Zeichenfolge kann nur die Symbole 'A' und 'B' enthalten.
2. Der Inhalt, wenn im Standard-Datenformat dargestellt, muß aus einer beliebigen Kombination der 26 Buchstaben des Alphabets und dem Leerzeichen aus dem COBOL Zeichenvorrat bestehen. Die Länge muß zwischen 1 und 8191 Zeichen liegen.

Datenregeln bei numerischen Feldern

1. Die PICTURE-Zeichenfolge kann nur die Symbole '9', 'P', 'S', und 'V' enthalten. Die Anzahl der Ziffernpositionen, die durch die PICTURE-Zeichenfolge beschrieben werden können, muß im Bereich von eins bis 18 einschließlich liegen.
2. Ohne Vorzeichen müssen die Daten im Standard-Datenformat eine Kombination der Ziffern '0' - '9' sein; mit Vorzeichen kann das Datenfeld auch ein '+', '-' oder irgendeine andere Darstellung eines Rechenvorzeichens enthalten (vgl. Die SIGN-Klausel in diesem Kapitel).

Datenregeln bei alphanumerischen Feldern

1. Die PICTURE-Zeichenfolge ist auf bestimmte Kombinationen der Symbole 'A', 'X' und '9' beschränkt, und das Datenfeld wird so behandelt, als ob die Zeichenfolge nur 'X' enthielte. Wenn nur A's oder nur 9en in einer PICTURE-Zeichenfolge enthalten sind, so wird damit kein alphanumerisches Datenfeld definiert.
2. Der Inhalt, wenn im Standard-Datenformat dargestellt, kann aus irgendwelchen Zeichen aus dem Zeichenvorrat des Computers bestehen. Die Länge muß zwischen 1 und 8191 Zeichen liegen.

Datenregeln bei alphanumerisch druckaufbereiteten Feldern

1. Die PICTURE-Zeichenfolge ist auf bestimmte Kombinationen der folgenden Symbole beschränkt: 'A', 'X', '9', 'B', '0' und '/', dabei muß die Zeichenfolge

   — mindestens ein 'B' und mindestens ein 'X' oder
   — mindestens eine '0' (Null) und mindestens ein 'X' oder
   — mindestens ein '/' (Schrägstrich) und mindestens ein 'X' oder
   — mindestens eine '0' (Null) und mindestens ein 'A' oder
   — mindestens ein '/' (Schrägstrich) und mindestens ein 'A'

   enthalten.

2. Der Inhalt, wenn im Standard-Datenformat dargestellt, besteht aus zulässigen Zeichen aus dem Zeichenvorrat des Computers. Die Länge muß zwischen 1 und 152 Zeichen liegen.

Numerisch druckaufbereitete Datenfelder

1. Die PICTURE-Zeichenfolge ist auf bestimmte Kombinationen der Symbole 'B', '/', 'P', 'V', 'Z', '0', '9', ',', '.', '*', '+', '-', 'DR', 'DB' und dem Währungssymbol festgelegt.

   Die zulässigen Kombinationen werden von der Präzedenzreihenfolge der Symbole und den Druckaufbereitungsregeln wie folgt bestimmt:

   a) Die Anzahl der Ziffernpositionen, die in der PICTURE-Zeichenfolge dargestellt werden kann, muß im Bereich zwischen eins und 18 einschließlich liegen.

   b) Die Zeichenfolge muß mindestens ein Zeichen aus der Menge: '0', 'B', '/', 'Z', '*', '+', ',', '.', '-', 'CR', 'DB' und das Währungszeichen enthalten.

2. Der Inhalt der Zeichenpositionen dieser Symbole, die zulässig sind, um eine Ziffer im Standard-Datenformat darzustellen, muß ein Numeral sein.

Datenelementsgröße

Die Größe eines Datenelementes, wobei Größe die Anzahl der Zeichenpositionen bedeutet, die das Datenelement im Standard-Datenformat belegt, wird durch die Anzahl der zulässigen Symbole und die Zeichenpositionen bestimmt. Eine Ganzzahl, die in Klammern nach 'A', 'X', '9', 'P', 'Z', '*', 'B', '/', '0', '+', '-' oder nach dem Währungssymbol steht, weist auf die Anzahl der aufeinanderfolgenden Symbole hin. Es ist zu beachten, daß die Symbole 'S', 'V', '.', 'CR' und 'DB' nur jeweils einmal in einem gegebenen PICTURE erscheinen dürfen.

Verwendete Symbole

Die Funktionen der Symbole, die zur Beschreibung eines Datenelementes verwendet werden, sind wie folgt erklärt:

A Jedes 'A' in der Zeichenfolge steht für eine Zeichenposition, die nur einen Buchstaben des Alphabets oder ein Leerzeichen enthalten kann.

B Jedes 'B' in der Zeichenfolge steht für eine Zeichenposition, in die das Leerzeichen eingefügt wird.

P Jedes 'P' steht für eine angenommene Dezimalpunktposition und wird dazu verwendet, die Position eines angenommenen Dezimalpunktes anzugeben, wenn dieser Punkt nicht in der Zahl, die im Datenfeld erscheint, enthalten ist.

Das Skalierungszeichen 'P' wird bei der Festlegung der Größe des Datenfeldes nicht gezählt. Skalierungszeichen werden jedoch bei der Festlegung der maximalen Anzahl von Ziffernpositionen (18) in numerisch druckaufbereiteten oder numerischen Datenfeldern gezählt. Das Skalierungszeichen 'P' kann nur links oder rechts als zusammenhängende Zeichenfolge von P's innerhalb einer PICTURE Skalierungsbeschreibung erscheinen; da das Skalierungszeichen 'P' einen angenommenen Dezimalpunkt impliziert (links der P's, wenn die P's die am weitesten links stehenden PICTURE-Zeichen sind, und rechts der P's, wenn die P's die am weitesten rechts stehenden PICTURE-Zeichen sind), ist das angenommene Dezimalpunktsymbol 'V' überflüssig.

Das Zeichen 'P' und das Einfügezeichen '.' (Punkt) können nicht gleichzeitig in der gleichen PICTURE-Zeichenfolge erscheinen. Wenn in irgendeiner Operation, die Konvertierung von einer Form interner Darstellung zu einer anderen nach sich zieht, und das

Datenfeld, das konvertiert wird, mit dem PICTURE-Zeichen 'P' beschrieben ist, so wird jede Ziffernposition, die durch ein 'P' beschrieben ist so angesehen, als ob sie den Wert Null enthält, und die Größe des Datenfeldes wird so betrachtet, als ob sie die so beschriebene Zeichenposition enthielte.

S Der Buchstabe 'S' wird in einer Zeichenfolge dazu verwendet, das Vorhandensein, aber nicht die Darstellung oder die Stellung eines Rechenvorzeichens anzuzeigen; er muß als das am weitesten links stehende Zeichen im PICTURE geschrieben werden. Das 'S' wird bei der Festlegung der Größe (in Standard-Datenformatzeichen) des Datenelementes gezählt, sofern die Eintragung nicht einer SIGN-Klausel unterliegt, die die optionale SEPARATE CHARACTER-Angabe angibt (vgl. Die SIGN Klausel in diesem Kapitel).

V Das 'V' wird in einer Zeichenfolge dazu verwendet, die Stellung des angenommenen Dezimalpunktes anzuzeigen und darf höchstens einmal in einer Zeichenfolge erscheinen. Das 'V' stellt keine Zeichenposition dar und zählt deshalb nicht bei Festlegung der Größe des Datenelementes. Wenn der angenommene Dezimalpunkt rechts des am weitesten rechts stehenden Symboles in der Zeichenfolge steht, so ist das 'V' überflüssig.

X Jedes 'X' in der Zeichenfolge wird dazu verwendet, eine Zeichenposition darzustellen, die irgendwelche zulässigen Zeichen aus dem Zeichenvorrat des Computers enthält.

Z Jedes 'Z' in einer Zeichenfolge darf nur dazu verwendet werden, um die am weitesten links stehenden numerischen Zeichenpositionen darzustellen. Diese Zeichenpositionen werden durch ein Leerzeichen ersetzt, wenn der Inhalt dieser Zeichenposition Null ist. Jedes 'Z' zählt bei der Festlegung der Größe des Datenfeldes.

9 Jede '9' in der Zeichenfolge steht für eine Zeichenposition, die ein Numeral enthält und zählt bei der Festlegung der Größe des Datenfeldes.

0 Jede '0' (Null) in der Zeichenfolge stellt eine Zeichenposition dar, in die das Numeral Null eingefügt wird. Die '0' zählt bei der Festlegung der Größe des Datenfeldes.

/ Jeder '/' (Schrägstrich) in der Zeichenfolge stellt eine Zeichenposition dar, in die das Zeichen '/' eingefügt wird. Der Schrägstrich zählt bei der Festlegung der Größe des Datenfeldes.

, Jedes ',' (Komma) in der Zeichenfolge stellt eine Zeichenposition dar, in die das Zeichen ',' eingefügt wird. Diese Zeichenposition zählt bei Festlegung der Größe dieses Datenfeldes. Das Einfügungszeichen ',' darf nicht das letzte Zeichen in der PICTURE-Zeichenfolge sein.

. Wenn das Zeichen '.' (Punkt) in der Zeichenfolge erscheint, so ist dies ein Druckaufbereitungssymbol, das den Dezimalpunkt für die Normierung und zudem eine Zeichenposition darstellt, in die das Zeichen '.' eingefügt wird. Das Zeichen '.' wird bei der Festlegung der Größe des Datenfeldes gezählt.
In einem vorgegebenen Programm werden die **Funktionen** des Punktes und des Kommas ausgetauscht, wenn die Klausel DECIMAL-POINT IS COMMA im SPECIAL-NAMES Paragraphen angegeben wird.
Das Einfügungszeichen '.' darf nicht das letzte Zeichen in der PICTURE-Zeichenfolge sein.

+ -, CR, DB
Diese Symbole werden als Druckaufbereitungsvorzeichen-Steuersymbole für die Druckaufbereitung verwendet. Wenn sie verwendet werden, stellen sie die Zeichenposition dar, in die das Drukkaufbereitungsvorzeichen-Steuerungssymbol eingesetzt wird. In einer Zeichenfolge darf nur maximal eines dieser Symbole verwendet werden. Jedes im Symbol verwendete Zeichen zählt bei der Festlegung der Größe des Datenfeldes.

* Jeder '*' (Stern) in der Zeichenfolge stellt eine führende numerische Zeichenposition dar, in die ein Stern gesetzt wird, wenn der Inhalt dieser Position Null ist. Jeder '*' zählt bei der Festlegung der Größe des Datenfeldes.

c- Das Währungssymbol in der Zeichenfolge stellt eine Zeichenposition dar, in die ein Währungssymbol gesetzt wird. Das Währungssymbol in einer Zeichenfolge ist durch das Währungszeichen oder durch ein einzelnes in der CURRENCY SIGN-Klausel im SPECIAL-NAMES-Paragraph angegebenes Zeichen dargestellt. Das Währungssymbol zählt bei der Festlegung der Größe des Datenfeldes.

Druckaufbereitungsregeln

Es gibt zwei allgemeine Methoden, die Druckaufbereitung in der PICTURE-Klausel durchzuführen, entweder durch Einfügung oder durch Unterdrückung und Ersetzung. Es stehen vier Arten von Einfügeoperationen zur Verfügung:

* Einfache Einfügung
* Sondereinfügung
* Feste Einfügung
* Gleitende Einfügung

Es gibt zwei Arten der Druckaufbereitung durch Unterdrückung und Ersetzung:

* Nullenunterdrückung und Ersetzung durch Leerzeichen
* Nullenunterdrückung und Ersetzung durch Sterne

Die Art der Druckaufbereitung, die bei einem Datenfeld durchgeführt werden kann, hängt von der Kategorie ab, zu der das Datenfeld gehört. Tabelle 2-1 gibt an, welche Art der Druckaufbereitung bei einer gegebenen Kategorie durchgeführt werden kann.

Kategorie	Druckaufbereitungsart
alphabetisch	einfache Einfügung nur 'B'
numerisch	keine
alphanumerisch	keine
alphanumerisch druckaufbereitet	einfache Einfügung '0', 'B' und '/'
numerisch druck-aufbereitet	ALL, siehe Anmerkung unten

Tabelle 2-1 Druckaufbereitungsarten für Datenkategorien

Anmerkung

Gleitende Einfügungsdruckaufbereitung und Druckaufbereitung durch Nullenunterdrückung und Ersetzung schließen sich in einer PICTURE-Klausel gegenseitig aus. Nur eine Art der Ersetzung kann bei der Nullenunterdrückung in einer PICTURE-Klausel verwendet werden.

Einfache Einfügungsdruckaufbereitung

',' (Komma), 'B' (Leerzeichen), '0' (Null) und '/' (Schrägstrich) werden als Einfügungszeichen verwendet. Die Einfügungszeichen zählen bei der Festlegung der Größe des Datenfeldes und stellen die Position im Datenfeld dar, in die das Zeichen eingefügt wird.

Sondereinfügungsdruckaufbereitung

Der '.' (Punkt) wird als Einfügungszeichen verwendet. Zusätzlich stellt er auch den Dezimalpunkt für Ausrichtungszwecke dar. Das Einfügungszeichen, das für den tatsächlichen Dezimalpunkt verwendet wird, zählt bei der Festlegung der Größe des Datenfeldes. Die Verwendung des angenommenen Dezimalpunktes, dargestellt durch das Symbol 'V', und des tatsächlichen Dezimalpunktes, dargestellt durch das Einfügungszeichen, in der gleichen PICTURE-Zeichenfolge ist nicht zulässig. Als Ergebnis der Sondereinfügungsdruckaufbereitung erscheint das Einfügungszeichen im Datenfeld an der gleichen Position wie in der Zeichenfolge angegeben.

Feste Einfügungsdruckaufbereitung

Das Währungssymbol und die Vorzeichen-Steuersymbole für die Druckaufbereitung '+', '-', 'CR' und 'DB' sind Einfügungszeichen. Nur ein Währungssymbol und nur eines der Druckaufbereitungsvorzeichen-Steuersymbole können in einer gegebenen PICTURE-Zeichenfolge verwendet werden. Wenn die Symbole 'CR', oder 'DB' verwendet werden, so stellen sie zwei Zeichenpositionen bei der Festlegung der Größe des Datenfeldes dar und müssen für die am weitesten rechts stehenden Zeichenpositionen stehen, die bei der Festlegung der Größe des Datenfeldes gezählt werden. Das Symbol '+' oder '-', sofern verwendet, muß entweder die am weitesten links stehende oder am weitesten rechts stehende Zeichenposition sein, die bei der Festlegung der Größe des Datenfeldes gezählt wird. Das Währungssymbol muß das am weitesten links stehende Zeichen sein.

Druckaufbereitung in der PICTURE-Zeichenfolge	Ergebnis	
	Datenfeld positiv oder null	Datenfeld negativ
+	+	–
–	Leerzeichen	–
CR	2 Leerzeichen	CR
DB	2 Leerzeichen	DB

Tabelle 2-2 Druckaufbereitungssymbole in PICTURE-Zeichenfolgen

Gleitende Einfügungsdruckaufbereitung

Das Währungssymbol und die Vorzeichen-Steuersymbole für die Druckaufbereitung '+' oder '-' sind gleitende Einfügungszeichen und schließen sich als solche in einer gegebenen PICTURE-Zeichenfolge gegenseitig aus.

Gleitende Einfügungsdruckaufbereitung wird in einer PICTURE-Zeichenfolge durch Verwendung einer Folge von mindestens zwei der gleitenden Einfügungszeichen angezeigt. Diese Folge von gleitenden Einfügungszeichen kann irgendwelche der festen Einfügungssymbole enthalten oder es können feste Einfügungssymbole unmittelbar rechts davon stehen. Diese einfachen Einfügungszeichen sind Teil der Zeichenfolge für gleitende Einfügungsdruckaufbereitung.

Das am weitesten links stehende Zeichen einer gleitenden Einfügungszeichenfolge stellt die linke Grenze des gleitenden Symbols im Datenfeld dar. Das am weitesten rechts stehende Zeichen der gleitenden Zeichenfolge stellt die rechte Grenze des gleitenden Symbols im Datenfeld dar.

Das zweite gleitende Zeichen von links stellt die linke Grenze der numerischen Daten dar, die im Datenfeld gespeichert werden können. Numerische Daten, die ungleich Null sind, können alle Zeichen ab dieser Grenze nach rechts ersetzen.

In einer PICTURE-Zeichenfolge gibt es nur zwei Möglichkeiten, gleitende Einfügungsdruckaufbereitung darzustellen. Eine Möglichkeit ist, irgendeine oder alle der führenden numerischen Zeichenpositionen links des Dezimalpunktes durch Einfügungszeichen darzustellen. Die andere Möglichkeit ist, alle numerischen Zeichenpositionen in der PICTURE-Zeichenfolge durch das Einfügungszeichen darzustellen.

Wenn die Einfügungszeichen nur links des Dezimalpunktes in der PICTURE-Zeichenfolge stehen, so hat dies zum Ergebnis, daß ein einziges gleitendes Einfügungszeichen in die Zeichenposition gesetzt wird, die unmittelbar vor dem Dezimalpunkt steht oder vor der ersten Ziffer, die ungleich Null ist, steht. Die Zeichenpositionen vor dem Einfügungszeichen werden durch Leerzeichen ersetzt.

Wenn alle numerischen Zeichenpositionen in der PICTURE-Zeichenfolge durch das Einfügungszeichen dargestellt werden, hängt das Ergebnis vom Wert der Daten ab. Wenn der Wert Null ist, so besteht das gesamte Datenfeld aus Leerzeichen. Wenn der Wert ungleich Null ist, so ist das Ergebnis das gleiche, als wenn das Einfügungszeichen nur links des Dezimalpunktes gestanden wäre.

Um Abschneiden zu vermeiden, muß die Mindestgröße der PICTURE-Zeichenfolge für das Empfangsdatenfeld der Anzahl von Zeichen im Sendedatenfeld entsprechen, plus der Anzahl der nicht gleitenden Einfügungszeichen, die in dem Empfangsdatenfeld aufbereitet werden, plus eins für das gleitende Einfügungszeichen.

Nullenunterdrückung-Druckaufbereitung

Die Unterdrückung von führenden Nullen in numerischen Zeichenpositionen wird durch die Verwendung des alphabetischen Zeichens 'Z' oder des Zeichens '*' (Stern) als Unterdrückungssymbol in einer PICTURE-Zeichenfolge angezeigt. Diese Symbole schließen sich in einer PICTURE-Zeichenfolge gegenseitig aus.
Jedes Unterdrückungssymbol wird bei der Festlegung der Größe des Datenfeldes gezählt. Wenn 'Z' verwendet wird, so ist das Ersatzzeichen das Leerzeichen, und wenn der Stern verwendet wird, so ist das Ersatzzeichen der '*'.

Nullen werden in einer PICTURE-Zeichenfolge durch Verwendung einer Folge von einem oder mehreren der zulässigen Symbole unterdrückt oder ersetzt, wenn die verknüpfte Zeichenposition in den Daten eine Null enthält. Jedes der einfachen Einfügezeichen in der Folge von Symbolen oder unmittelbar rechts dieser Folge sind Teil dieser Folge.

In einer PICTURE-Zeichenfolge gibt es nur zwei Möglichkeiten, die Nullenunterdrückung darzustellen. Eine Möglichkeit ist, eines oder alle der führenden numerischen Zeichenpositionen links des Dezimalpunktes durch Unterdrückungssymbole darzustellen. Die andere Möglichkeit ist, alle numerischen Zeichenpositionen in der PICTURE-Zeichenfolge durch Unterdrückungssymbole darzustellen.

Wenn das Unterdrückungssymbol nur links des Dezimalpunktes erscheint, wird jede führende Null in den Daten, die einem Symbol in der Folge entspricht, durch das Ersatzzeichen ersetzt. Unterdrückung endet bei der ersten Ziffer in den durch die Unterdrückungssymbolfolge dargestellten Daten, Null oder beim Dezimalpunkt, je nachdem was auch immer zuerst auftritt.

Wenn alle numerischen Zeichenpositionen in der PICTURE-Zeichenfolge durch Unterdrückungssymbole dargestellt werden und der Wert der Daten ungleich Null ist, so ist das Ergebnis das gleiche, als ob die Unterdrückungszeichen nur links des Dezimalpunktes stehen. Wenn der Wert Null ist, und das Unterdrückungssymbol wird durch 'Z' dargestellt, dann besteht das ganze Datenfeld aus Leerzeichen. Wenn der Wert Null ist und das Unterdrückungssymbol wird durch das Zeichen '*' dargestellt, dann besteht das ganze Datenfeld aus '*', außer dem tatsächlichen Dezimalpunkt.

Die Symbole '+', '-', '*', 'Z' und das Währungssymbol, sofern als gleitende Ersatzzeichen verwendet, schließen sich innerhalb einer gegebenen Zeichenfolge gegeneinander aus.

Rangfolgeregeln

Tabelle 2-3 zeigt die Reihenfolge, wie Zeichen als Symbole in einer Zeichenfolge verwendet werden. Ein 'X' bei einem Schnittpunkt zeigt an, daß die Symbole in der Spalte oben in einer vorgegebenen Zeichenfolge den Symbolen, die links in einer Zeile stehen, vorausgehen dürfen. Argumente, die in geschweiften Klammern stehen, zeigen an, daß die Symbole sich gegeneinander ausschließen. Das Währungssymbol wird durch das Symbol 'cs' angezeigt.

Mindestens eines der Symbole 'A', 'X', 'Z', '9' oder '*' oder mindestens zwei der Symbole '+', '-' oder 'cs' müssen in einer PICTURE-Zeichenfolge enthalten sein.

Erstes Symbol / Zweites Symbol		Nichtgleitende Einfügungszeichen									gleitende Einfügungszeichen					Andere Symbole					
		B	0	/	,	.	{+ −}	{CR DB}	c s		{Z *}	{+ −}	c s	c s		9	{A X}	S	V	P	P
Nicht	B	x	x	x	x	x	x		x		x x	x x	x	x		x	x		x		x
	0	x	x	x	x	x	x		x		x x	x x	x	x		x	x		x		x
	/	x	x	x	x	x	x		x		x x	x x	x	x		x	x		x		x
gleitendes	,	x	x	x	x	x	x		x		x x	x x	x	x		x			x		x
	.	x	x	x	x		x		x		x	x	x			x					
Einfügungs-	{+ −}	x	x	x	x	x			x		x x		x	x		x			x	x	x
	{CR DB}	x	x	x	x	x			x		x x		x	x		x			x	x	x
zeichen	cs						x														
Gleitende	{Z *}	x x	x x	x x	x x	 x	x x		x x		x x x								x		x
Einfügungs-	{+ −}	x x	x x	x x	x x				x x			x x x							x		x
zeichen	cs	x	x	x	x		x						x								
	cs	x	x	x	x	x	x						x	x					x		x
Andere	9	x	x	x	x	x	x		x		x	x	x			x	x	x	x		x
	{A X}	x	x	x												x	x				
	S																				
Symbole	V	x	x	x	x		x		x		x	x	x			x		x		x	
	P	x	x	x	x		x		x		x	x	x			x		x		x	
	P						x		x									x	x		x

Tabelle 2-3 Übersicht über die PICTURE-Zeichenfolge

In Tabelle 2-3 stehen die nichtgleitenden Einfügungssymbole '+' und '-', die gleitenden Einfügungssymbole 'Z', '*', '+', '<', '-' und 'cs' sowie das Symbol 'P' zweimal in der PICTURE Zeichenrangfolgetabelle. Die linke Spalte und die obere Zeile für jedes Symbol stellt dessen Verwendung links der Dezimalpunktposition dar. Das zweite Auftreten der Symbole in den Zeilen und Spalten stellt deren Verwendung rechts der Dezimalpunktposition dar.

Die REDEFINES-Klausel

Funktion

Die REDEFINES-Klausel ermöglicht es, den gleichen Computerspeicherbereich durch verschiedene Datenerklärungen zu beschreiben.

Allgemeines Format

```
stufennummer datenname-2; REDEFINES datenname-1
```

Anmerkung

stufennummer, datenname-2 werden im obigen Format nur zum besseren Verständnis angegeben. Stufennummer und datenname-2 sind nicht Teil der REDEFINES-Klausel.

Syntaxregeln

1. Die REDEFINES-Klausel, wenn angegeben, muß unmittelbar nach datenname-1 stehen, wenn der ANSI Schalter gesetzt ist; andernfalls kann sie nach der PICTURE-Klausel stehen.
2. Die Stufennummern von datenname-1 und datenname-2 müssen identisch sein; jedoch nicht 66 oder 88.
3. Diese Klausel darf nicht in Eintragungen auf der Stufe 01 in der FILE SECTION verwendet werden. (vgl. Allgemeine Regel 2 in Die DATA RECORDS-Klausel in Kapitel 5).
4. Diese Klausel darf nicht in Eintragungen auf der Stufe 01 in der COMMUNICATION SECTION verwendet werden.
5. datenname-2 kann einer Eintragung untergeordnet sein, die eine REDEFINES-Klausel enthält. datenname-2 kann jedoch einem Datenfeld untergeordnet sein, dessen Datenerklärung eine OCCURS-Klausel enthält. In diesem Fall darf die Referenz zu datenname-2 in der REDEFINES-Klausel nicht subskribiert oder indiziert sein. Die ursprüngliche Definition darf kein Datenfeld enthalten, dessen Größe variabel ist wie in der OCCURS-Klausel definiert. (vgl. Die OCCURS-Klausel in Kapitel 3).

6. Zwischen den Datenerklärungen von datenname-2 und datenname-1 darf keine Eintragung mit einer Stufennummer auftreten, die kleiner ist als die Stufennummer von datenname-2 und datenname-1.

Allgemeine Regeln

1. Die Umdefinition beginnt bei datenname-2 und endet, sobald eine Stufennummer kleiner oder gleich der von datenname-2 auftritt.
2. Wenn die Stufennummer von datenname-1 ungleich 01 ist, so muß sie die gleiche Anzahl von Zeichenpositionen angeben, die das durch datenname-2 referenzierte Datenfeld enthält (wenn der ANSI Schalter gesetzt ist). Es ist wichtig zu beachten, daß die REDEFINES-Klausel die Umdefinitionen eines Speicherbereiches definiert und nicht die Datenfelder, die diesen Bereich belegen.
3. Mehrfache Umdefinitionen der gleichen Zeichenpositionen sind erlaubt. Die Eintragungen der neuen Beschreibungen der Zeichenpositionen müssen nach den Eintragungen stehen, die den neudefinierten Bereich definieren, ohne zwischenliegende Eintragungen, die neue Zeichenpositionen definieren. Mehrfache Umdefinitionen der gleichen Zeichenpositionen müssen alle den Datennamen der Eintragung verwenden, der ursprünglich den Bereich definiert hat.
4. Die Eintragungen in der neuen Beschreibung von Zeichenpositionen dürfen keine VALUE-Klausel enthalten, außer bei Bedingungsnamen-Eintragungen.
5. Mehrfache Stufe 01-Eintragungen, die irgendeiner gegebenen Stufenbezeichnung untergeordnet sind, stellen implizite Neudefinitionen des gleichen Bereiches dar.

Die RENAMES-Klausel

Funktion

Die RENAMES-Klausel ermöglicht alternative evtl. überlappende Zusammenfassungen von Datenelementen.

Allgemeines Format

```
66 datenname-1 RENAMES datenname-2 [ { THROUGH } datenname-3 ] .
                                   [ { THRU    }             ]
```

Anmerkung

Stufennummer 66 und datenname-1 werden im obigen Format nur zum besseren Verständnis angegeben. Stufennummer und datenname-1 sind nicht Teil der RENAMES-Klausel.

Syntaxregeln

1. Alle RENAMES-Eintragungen, die sich auf Datenfelder innerhalb eines gegebenen logischen Datensatzes beziehen, müssen unmittelbar nach der letzten Datenerklärung der zugehörigen Datensatzerklärung stehen.

2. datenname-2 und datenname-3 müssen Namen von Datenelementen oder von Datengruppen sein, bestehend aus Datenelementen im gleichen logischen Datensatz, und dürfen nicht den gleichen Datennamen besitzen. Eine Eintragung auf Stufennummer 66 kann weder eine andere Eintragung auf Stufennummer 66 umbenennen noch eine Eintragung auf den Stufennummern 77, 88 oder 01.

3. datenname-1 kann nicht selbst als Kennzeichner verwendet werden und kann nur durch die Namen der zugehörigen Stufe 01, sowie der zugehörigen Eintragung FD oder SD gekennzeichnet werden. Weder datenname-2 noch datenname-3 können eine OCCURS-Klausel in ihren Datenerklärungen enthalten, noch einem Datenfeld untergeordnet sein, das eine OCCURS-Klausel in der Datenerklärung besitzt (vgl. Die OCCURS-Klausel in Kapitel 3).

4. Der Anfang des mit datenname-3 angegebenen Bereiches darf nicht links vom Anfang des durch datenname-2 beschriebenen Bereiches liegen. Das Ende des durch datenname-3 beschriebenen Bereiches muß rechts am Ende des durch datenname-2 beschriebenen Bereiches liegen. datenname-3 kann daher nicht datenname-2 untergeordnet sein.
5. datenname-2 und datenname-3 können gekennzeichnet werden.
6. Die Worte THRU und THROUGH sind äquivalent.
7. Die Größe der Datenfelder von datenname-2 und datenname-3 darf nicht variabel sein.

Allgemeine Regeln

1. Eine oder mehrere RENAMES-Eintragungen können für einen logischen Datensatz gemacht werden.
2. Wenn datenname-3 angegeben ist, so ist datenname-1 eine Datengruppe, die alle Datenelemente enthält, die mit datenname-2 (sofern datenname-2 ein Datenelement ist) oder die mit dem ersten Datenelement in datenname-2 (sofern datenname-2 ein Datengruppe ist) beginnen und die mit datenname-3 (sofern datenname-3 ein Datenelement ist), oder die mit dem letzten Datenelement in datenname-3 (sofern datenname-3 eine Datengruppe ist) enden.
3. Wenn datenname-3 nicht angegeben ist, kann datenname-2 entweder eine Gruppe oder ein Datenelement sein; wenn datenname-2 eine Datengruppe ist, so wird datenname-1 als Datengruppe behandelt, und wenn datenname-2 ein Datenelement ist, so wird datenname-1 als Datenelement behandelt.

Die SIGN-Klausel

Funktion

Die SIGN-Klausel gibt die Position und die Darstellungsart des Rechenvorzeichens an, wenn es notwendig ist, diese Eigenschaften explizit zu beschreiben.

Allgemeines Format

```
[SIGN IS] {LEADING  } [SEPARATE CHARACTER]
          {TRAILING }
```

Syntaxregeln

1. Die SIGN-Klausel kann nur für eine numerische Datenerklärung angegeben sein, deren PICTURE das Zeichen 'S' enthält, oder eine Datengruppe, die wenigstens eine solche numerische Datenerklärung enthält.
2. Die numerischen Datenerklärungen, für die die SIGN-Klausel angegeben ist, müssen als USAGE IS DISPLAY beschrieben werden.
3. Höchstens eine SIGN-Klausel kann für eine vorgegebene numerische Datenerklärung angegeben werden.
4. Wenn die CODE-SET-Klausel angegeben ist, müssen alle numerischen Datenerklärungen mit Vorzeichen, die mit dieser Dateierklärung verknüpft sind, mit der SIGN IS SEPARATE-Klausel beschrieben werden.

Allgemeine Regeln

1. Die optionale SIGN-Klausel, sofern vorhanden, gibt die Positionen und die Darstellungsart des Rechenvorzeichens für numerische Datenerklärung an, auf die sie zutrifft, oder für jede numerische Datenerklärung die der Gruppe, auf die sie zutrifft, untergeordnet ist. Die SIGN-Klausel trifft nur auf numerische Datenerklärungen zu, deren PICTURE das Zeichen 'S' enthält; 'S' zeigt das Vorhandensein, jedoch weder die Darstellung noch die Position des Rechenvorzeichens an.

2. Eine numerische Datenerklärung, deren PICTURE das Zeichen 'S' enthält, auf die jedoch keine optionale SIGN-Klausel zutrifft, hat ein Rechenvorzeichen; jedoch weder die Darstellung noch notwendigerweise die Position des Rechenvorzeichens wird durch das Zeichen 'S' angegeben. In diesem (Standard-) Fall gelten Punkt 3 bis 5 der Allgemeinen Regeln für solche numerischen Datenfelder mit Vorzeichen nicht. Die Darstellung des Standard-Rechenvorzeichens wird in Kapitel 2, Auswahl der Zeichendarstellung und des Zahlensystems, definiert.

3. Wenn die optionale SEPARATE CHARACTER-Angabe nicht vorhanden ist, dann:

   a) wird angenommen, daß das Rechenvorzeichen mit der führenden (oder entsprechend der letzten) Ziffernposition des elementaren numerischen Datenfeldes verknüpft ist;

   b) wird der Buchstabe 'S' in einer PICTURE-Zeichenfolge bei der Festlegung der Größe des Datenfeldes (in Standard-Datenformatzeichen) nicht gezählt.

4. Wenn die optionale SEPARATE CHARACTER-Angabe vorhanden ist, dann:

   a) wird angenommen, daß das Rechenzeichen die führende (oder entsprechend die letzte) Zeichenposition des elementaren numerischen Datenfeldes ist; diese Zeichenposition ist keine Ziffernposition;

   b) wird der Buchstabe 'S' in einer PICTURE-Zeichenfolge bei Festlegung der Größe des Datenfeldes (in Standard-Datenformatzeichen) gezählt;

   c) sind die Rechenvorzeichen für positiv und negativ die Standard-Datenformatzeichen '+' bzw. '-'.

5. Jede numerische Datenerklärung, deren PICTURE das Zeichen 'S' enthält, ist eine numerische Datenerklärung mit Vorzeichen. Wenn eine SIGN-Klausel auf eine solche Eintragung zutrifft und für Rechen- oder Vergleichszwecke eine Konvertierung notwendig ist, so findet diese Konvertierung automatisch statt.

Die SYNCHRONIZED-Klausel

Funktion

Die SYNCHRONIZED-Klausel gibt die Ausrichtung eines Datenelementes an den natürlichen Grenzen des Speichers an.

Allgemeines Format

```
{SYNCHRONIZED} [LEFT ]
{SYNC        } [RIGHT]
```

Syntaxregeln

1. Diese Klausel kann nur bei einem einzigen Datenelement erscheinen.
2. SYNC ist eine Abkürzung für SYNCHRONIZED.

Allgemeine Regeln

1. Diese Klausel gibt an, daß das betreffende Datenfeld so im Speicher ausgerichtet wird, daß kein anderes Datenfeld irgendwelche Zeichenpositionen zwischen den links bzw. rechts liegenden natürlichen Grenzen, die dieses Datenfeld begrenzen, belegt. Wenn die Anzahl der Zeichenpositionen, die für die Speicherung dieses Datenfeldes notwendig sind, kleiner ist als die Anzahl der Zeichenpositionen zwischen diesen natürlichen Grenzen, so dürfen die unbenutzten Zeichenpositionen (oder Teile davon) nicht für irgendwelche anderen Datenfelder verwendet werden. Solche unbenutzten Zeichenpositionen sind jedoch enthalten in:
   a) der Größe von irgendwelchen Datengruppen, zu denen das Datenelement gehört;
   b) der umdefinierten Zeichenposition, wenn dieses Datenfeld Gegenstand einer REDEFINES-Klausel ist.
2. SYNCHRONIZED ohne RIGHT oder LEFT gibt an, daß das Datenelement so zwischen die natürlichen Grenzen zu positionieren ist, daß es effizient benutzt werden kann.

3. SYNCHRONIZED LEFT gibt an, daß das Datenelement so positioniert werden soll, daß es bei der linken Zeichenposition der natürlichen Grenzen, in die es gesetzt wird, beginnt.

4. SYNCHRONIZED RIGHT gibt an, daß das Datenelement so positioniert werden soll, daß es bei der rechten Zeichenposition der natürlichen Grenzen endet, in die es gesetzt wird.

5. Wenn ein SYNCHRONIZED-Datenfeld im Quellprogramm angesprochen wird, hängen die Aktionen

   - auf Bündigkeit ausrichten
   - Abschneiden von Überhängen
   - Überlauf

   trotzdem von der ursprünglichen Länge - wie in der PICTURE-Klausel angegeben - ab.

6. Wenn die Datenerklärung eines Datenfeldes die SYNCHRONIZED-Klausel und ein Rechenvorzeichen enthält, so erscheint das Vorzeichen des Datenfeldes an der normalen Rechenvorzeichenposition, ohne Berücksichtigung, ob das Datenfeld SYNCHRONIZED LEFT oder SYNCHRONIZED RIGHT ist.

7. Wenn die SYNCHRONIZED-Klausel in einer Datenerklärung eines Datenfeldes angegeben ist, das auch eine OCCURS-Klausel enthält oder das einer Datenerklärung untergeordnet ist, die eine OCCURS-Klausel enthält, dann:

   a) ist jedes Element des Datenfeldes SYNCHRONIZED.

   b) werden alle impliziten FILLER, die für andere Datenfelder innerhalb der gleichen Tabelle generiert sind, für jedes Element dieser Datenfelder generiert.

8. Diese Klausel ist geräteabhängig.

Die USAGE-Klausel

Funktion

Die USAGE-Klausel gibt das Format eines Datenfeldes im Speicher an.

Allgemeines Format

```
[USAGE  IS]  { COMPUTATIONAL   }
             { COMP            }
             { DISPLAY         }
             { COMPUTATIONAL-3 }
             { COMP-3          }
```

Syntaxregeln

1. Die PICTURE-Zeichenfolge eines COMPUTATIONAL oder COMPUTATIONAL-3-Datenfeldes kann nur ein oder mehrere '9'en, das Rechenvorzeichen 'S', das implizite Dezimalpunktzeichen 'V' sowie ein oder mehrere 'P's enthalten (vgl. Die PICTURE-Klausel in diesem Kapitel).
2. COMP ist die Abkürzung für COMPUTATIONAL.

Allgemeine Regeln

1. Die USAGE-Klausel kann auf jeder Stufe geschrieben werden. Wenn die USAGE-Klausel auf der Gruppenstufe geschrieben wird, so gilt sie für jedes Datenelement in der Gruppe. Die USAGE-Klausel eines Datenelementes kann nicht im Gegensatz zu der USAGE-Klausel einer Gruppe, zu der das Datenfeld gehört, stehen.
2. Diese Klausel gibt die Art an, in der ein Datenfeld im Speicher eines Computers dargestellt ist. Sie hat keine Auswirkung auf die Verwendung des Datenfeldes, obwohl die Angaben für einige Anweisungen in der PROCEDURE DIVISION die USAGE-Klausel in bezug auf die Operanden beschränken kann. Die USAGE-Klausel kann das Zahlensystem oder die Art der Zeichendarstellung des Datenfeldes beeinflussen.

3. Ein COMPUTATIONAL oder COMPUTATIONAL-3-Datenfeld ist in der Lage, einen für Rechenoperationen verwendeten Wert darzustellen und muß numerisch sein. Wenn eine Datengruppe als COMPUTATIONAL(-3) beschrieben ist, so sind die Datenelemente in der Gruppe COMPUTATIONAL(-3); die Datengruppe selbst ist nicht COMPUTATIONAL(-3) und kann für Rechenoperationen nicht verwendet werden.
4. Die USAGE IS DISPLAY-Klausel zeigt an, daß das Format der Daten ein Standard-Datenformat ist.
5. Wenn die USAGE-Klausel für ein Datenelement oder für irgendeine Gruppe, zu der das Datenfeld gehört, nicht angegeben ist, so ist die Verwendung implizit DISPLAY.
6. Was für Leerzeichen für die verschiedenen USAGE-Speicheroptionen erforderlich ist, wird in Kapitel 2 "Auswahl der Zeichendarstellung und des Zahlensystems", aufgezeigt.

Die VALUE-Klausel

Funktion

Die VALUE-Klausel beschreibt den Wert von Konstanten, den Anfangswert von WORKING-STORAGE-Datenfeldern und die mit einem Bedingungsnamen verknüpften Werte.

Allgemeines Format

Format 1:

```
VALUE is Literal
```

Format 2:

```
{ VALUE IS  }          [ { THROUGH } literal-2 ]
{ VALUES ARE } literal-1 [ { THRU    }           ]

[ , literal-3 [ { THROUGH } literal-4 ] ] ...
              [ { THRU    }           ]
```

Syntaxregeln

1. Die VALUE-Klausel kann nicht für Datenfelder angegeben werden, deren Größe variabel ist. (vgl. Die OCCURS-Klausel in Kapitel 3).
2. Ein numerisches Literal mit Vorzeichen muß mit einer numerischen PICTURE-Zeichenfolge mit Vorzeichen verknüpft sein.

3. Alle numerischen Literale in einer VALUE-Klausel eines Datenfeldes müssen Werte haben, die im Bereich der in der PICTURE-Klausel angegebenen Werte liegen und dürfen keinen Wert haben, der Abschneidungen von Ziffern, die ungleich Null sind, erforderlich macht. Nichtnumerische Literale in einer VALUE-Klausel eines Datenfeldes dürfen die in der PICTURE-Klausel angegebene Größe nicht überschreiten.

4. Die Worte THRU und THROUGH sind gleichbedeutend.

Allgemeine Regeln

1. Die VALUE-Klausel darf nicht mit anderen Klauseln in der Datenerklärung eines Datenfeldes oder in der Datenbeschreibung innerhalb der Hierarchie des Datenfeldes in Widerspruch stehen. Die folgenden Regeln sind anzuwenden:

   a) Wenn die Kategorie des Datenfeldes numerisch ist, so müssen alle Literale in der VALUE-Klausel numerisch sein. Wenn das Literal den Wert eines WORKING-STORAGE-Datenfeldes definiert, wird das Literal im Datenfeld entsprechend den Standard-Ausrichtungsregeln ausgerichtet (Vgl. Standard-Ausrichtungsregeln in Kapitel 2).

   b) Wenn die Kategorie eines Datenfeldes alphabetisch, alphanumerisch, alphanumerisch druckaufbereitet oder numerisch druckaufbereitet ist, müssen alle Literale in der VALUE-Klausel nichtnumerische Literale sein. Das Literal wird im Datenfeld so ausgerichtet, als ob das Datenfeld als alphanumerisch beschrieben wäre. (vgl. Standard-Ausrichtungsregeln in Kapitel 2). Druckaufbereitungszeichen in der PICTURE-Klausel werden bei der Festlegung der Größe des Datenfeldes gezählt, sie haben jedoch keine Auswirkung auf die Initialisierung des Datenfeldes.

   c) Die Initialisierung findet unabhängig von irgendeiner evtl. vorgegebenen BLANK WHEN ZERO oder JUSTIFIED-Klausel statt.

2. Eine figurative Konstante kann sowohl in Format 1 als auch in Format 2 ersetzt werden, immer dort, wo ein Literal angegeben ist.

Regeln für Bedingungsnamen

1. Bei einer Bedingungsnamen-Eintragung ist die VALUE-Klausel erforderlich. Die VALUE-Klausel und der Bedingungsname selbst sind die beiden einzigen in dieser Eintragung zulässigen Klauseln. Die Eigenschaften eines Bedingungsnamens sind implizit die seiner Bedingungsvariablen.
2. Das Format 2 kann nur in Verbindung mit Bedingungsnamen verwendet werden. Wo immer die THRU-Angabe verwendet wird, muß das literal-1 kleiner als das literal-2 und das literal-3 kleiner als das literal-4 usw. sein.

Datenerklärungen anders als Bedingungsnamen

Die Regeln für die Verwendung der VALUE-Klausel unterscheiden sich in den einzelnen SECTIONs der DATA DIVISION:

1. In der FILE SECTION darf die VALUE-Klausel nur bei Bedingungsnamen-Eintragungen verwendet werden.
2. In der WORKING-STORAGE SECTION muß die VALUE-Klausel bei Bedingungsnamen-Eintragungen verwendet werden. Die VALUE-Klausel kann auch dazu verwendet werden, den Anfangswert eines Datenfeldes anzugeben; in diesem Fall nimmt das Datenfeld den anzugegebenden Wert zu Beginn der Abarbeitung des Zielprogrammes an. Wenn die VALUE-Klausel bei der Beschreibung eines Datenfeldes nicht verwendet wird, ist der Anfangswert undefiniert.
3. In der LINKAGE SECTION kann die VALUE-Klausel nur für Bedingungsnamen-Eintragungen verwendet werden.
4. Die VALUE-Klausel darf in einer Datenerklärung, die eine OCCURS-Klausel enthält oder in einer Eintragung, die einer anderen Eintragung untergeordnet ist, die eine OCCURS-Klausel enthält, nicht angegeben werden. Diese Regel trifft nicht für Bedingungsnamen-Eintragungen zu (vgl. Die OCCURS-Klausel in Kapitel 3).
5. Die VALUE-Klausel kann in einer Datenerklärung angegeben sein, die eine REDEFINES-Klausel enthält oder in einer Eintragung, die einer anderen Eintragung, die eine REDEFINES-Klausel enthält, untergeordnet ist. Diese Regel gilt nicht für Bedingungsnamen-Eintragungen.
6. Wenn die VALUE-Klausel in einer Eintragung auf der Gruppenstufe verwendet wird, muß das Literal eine figurative Konstante oder ein nichtnumerisches Literal sein und der Gruppenbereich wird initialisiert ohne Berücksichtigung einzelner Datenelemente oder Datengruppen, die in dieser Gruppe enthalten sind. Die VALUE-Klausel kann innerhalb dieser Gruppe, nicht auf untergeordneten Stufen angegeben werden.
7. Die VALUE-Klausel darf für eine Gruppe, die Datenfelder mit Beschreibungen, einschließlich JUSTIFIED, SYNCHRONIZED oder USAGE (nicht jedoch USAGE IS DISPLAY) enthält, nicht geschrieben werden.

2.5 PROCEDURE DIVISION

Allgemeine Beschreibung

Die PROCEDURE DIVISION muß in jedem COBOL-Quellprogramm vorhanden sein. Diese DIVISION darf vereinbarende Prozeduren enthalten.

Vereinbarungen

Vereinbarende SECTIONs müssen am Anfang der PROCEDURE DIVISION stehen und von den Schlüsselwörtern DECLARATIVES und END DECLARATIVES eingerahmt werden.

Prozeduren

Eine Prozedur besteht aus einem Paragraphen oder aus einer Gruppe von aufeinanderfolgenden Paragraphen (der erste Paragraph-Name ist optional), oder einer SECTION oder einer Gruppe von aufeinanderfolgenden SECTIONs innerhalb der PROCEDURE DIVISION. Wenn ein Paragraph in einer SECTION steht, so müssen alle Paragraphen in SECTIONS stehen. Ein Prozedur-Name ist ein Wort, das verwendet wird, um einen Paragraphen oder eine SECTION im selben Quellprogramm an anderer Stelle anzusprechen. Es besteht aus einem Paragraph-Namen (der gekennzeichnet sein kann) oder einem SECTION-Namen.

Das Ende der PROCEDURE DIVISION und das physikalische Ende des Programms ist jene physikalische Stelle in einem COBOL Quellprogramm, nach der keine weiteren Prozeduren erscheinen.

Eine SECTION besteht aus einer SECTION-Überschrift gefolgt von Null, einer oder mehreren aufeinanderfolgenden Paragraphen. Eine SECTION endet unmittelbar vor der nächsten SECTION oder am Ende der PROCEDURE DIVISION oder, wenn es sich um einen Vereinbarungsteil der PROCEDURE DIVISION handelt, bei den Schlüsselwörtern END DECLARATIVES.

Paragraphen

Ein Paragraph besteht aus einem Paragraph-Namen gefolgt von einem Punkt und einem Leerzeichen und von Null sowie einem oder mehreren aufeinanderfolgenden Sätzen. Ein Paragraph endet unmittelbar vor dem nächsten Paragraph-Namen oder SECTION-Namen oder am Ende der PROCEDURE DIVISION oder, wenn es sich um einen Vereinbarungsteil der PROCEDURE DIVISION handelt, bei den Schlüsselwörtern END DECLARATIVES.

Sätze

Ein Satz besteht aus einer oder mehreren Anweisungen und wird mit einem Punkt, gefolgt von einem Leerzeichen, abgeschlossen.

Anweisungen

Eine Anweisung ist eine syntakisch zulässige Kombination von Wörtern und Symbolen und beginnt mit einem COBOL Verb.

Bezeichner

Der Begriff 'Bezeichner' wird als das Wort oder die Wörter definiert, die notwendig sind, ein Datenfeld **eindeutig** zu identifizieren.

Ausführung

Die Ausführung beginnt bei der ersten Anweisung in der PROCEDURE DIVISION ausgenommen der Vereinbarungen. Die Anweisungen werden dann in der Reihenfolge, in der sie dem Übersetzer angeboten werden, ausgeführt, ausgenommen dort, wo durch die Regeln eine andere Reihenfolge festgelegt wird.

Allgemeines Format

PROCEDURE DIVISION Überschrift

Die PROCEDURE DIVISION muß mit der folgenden Überschrift beginnen:

```
PROCEDURE DIVISION [  USING datenname-1 [,datenname-2] ... ].
```

PROCEDURE DIVISION Rumpf

Der Rumpf der PROCEDURE DIVISION muß einem der nachfolgenden Formate entsprechen:

Format 1

```
DECLARATIVES.

{ sectionname  SECTION [ segmentnummer ] . vereinbarungssatz

  [paragraphname. [programmsatz] ... ] ... } ...

END DECLARATIVES.

{ sectionname  SECTION [ segmentnummer ].

  [paragraphname [programmsatz] ...] ... } ...
```

Format 2

```
{ sectionname  SECTION [ segmentnummer ].

   [paragraphname [programmsatz] ...] ... } ...
```

Anweisungen und Sätze

Es gibt drei Arten von Anweisungen:

1. Bedingte Anweisungen
2. Übersetzer-Steueranweisungen
3. Unbedingte Anweisungen

Es gibt drei Arten von Sätzen:

1. Bedingte Sätze
2. Übersetzer-Steuersätze
3. Unbedingte Sätze

Bedingte Anweisungen

Eine bedingte Anweisung schreibt vor, daß der Wahrheitswert einer Bedingung ausgewertet wird, und daß die nachfolgende Handlung des Zielprogramms von diesem Wahrheitswert abhängt.

Bedingte Anweisungen sind folgende:

* IF-, SEARCH- oder RETURN-Anweisungn
* READ-Anweisungen, die eine AT END- oder INVALID KEY-Angabe enthalten.
* WRITE-Anweisungen, die eine INVALID KEY oder END OF PAGE-Angabe enthalten.
* START-, REWRITE- oder DELETE-Anweisungen, die eine INVALID KEY-Angabe enthalten.
* Arithmetische Anweisungen (ADD, COMPUTE, DIVIDE, MULTIPLY, SUBTRACT), die eine SIZE ERROR-Angabe enthalten.
* RECEIVE-Anweisungen, die eine NO DATA-Angabe enthalten.
* STRING-, UNSTRING- oder CALL-Anweisungen, die eine ON OVERFLOW-Angabe enthalten.

Bedingte Sätze

Ein bedingter Satz ist eine bedingte Anweisung, vor der optional eine unbedingte Anweisung stehen kann, und die durch einen Punkt, gefolgt von einem Leerzeichen, abgeschlossen wird.

Übersetzersteueranweisung

Eine Übersetzersteueranweisung besteht aus einem Übersetzersteuerverb und seinen Operanden. Es gibt die Übersetzersteuerverben COPY, ENTER und USE. Eine Übersetzersteueranweisung bewirkt, daß der Übersetzer spezifische Aktionen während des Übersetzungsvorganges vornimmt.

Übersetzersteuersätze

Ein Übersetzersteuersatz ist eine einzelne Übersetzersteueranweisung, gefolgt von einem Punkt und einem Leerzeichen.

Unbedingte Anweisung

Eine unbedingte Anweisung gibt an, daß das Zielprogramm eine bestimmte unbedingte Handlung durchführen muß. Eine unbedingte Anweisung ist jede Anweisung, die weder eine bedingte Anweisung noch eine Übersetzersteueranweisung ist. Eine unbedingte Anweisung kann aus einer Folge von unbedingten Anweisungen bestehen, die alle gegeneinander durch Trennsymbole abgetrennt werden können.

Die unbedingten Verben sind:

ACCEPT	ENABLE	RELEASE
ADD 1)	EXIT	REWRITE 2)
ALTER	GO	SEND
CALL 3)	INSPECT	SET
CANCEL	MERGE	SORT
CLOSE	MOVE	START 2)
COMPUTE 1)	MULTIPLY 1)	STOP
DELETE 2)	OPEN	STRING 3)
DISABLE	PERFORM	SUBTRACT 1)
DISPLAY	READ 5)	UNSTRING 3)
DIVIDE 1)	RECEIVE 4)	WRITE 6)

1) ohne optionale SIZE ERROR-Angabe
2) ohne optionale INVALID KEY-Angabe
3) ohne optionale OVERFLOW-Angabe
4) ohne optionale NO DATA-Angabe
5) ohne optionale AT END oder INVALID KEY-Angabe
6) ohne optionale INVALID KEY oder END-OF-PAGE-Angabe

Wenn im allgemeinen Format für Anweisungen 'unbedingte Anweisung' erscheint, so benennt 'unbedingte Anweisung' die Folge von aufeinanderfolgenden unbedingten Anweisungen, die durch einen Punkt oder eine ELSE-Angabe, die zu einer davorstehenden IF-Anweisung gehört, beendet sein muß.

Unbedingter Satz

Ein unbedingter Satz ist eine unbedingte Anweisung, die durch einen Punkt, gefolgt von einem Leerzeichen abgeschlossen wird.

Kategorien der Anweisungen

Kategorie	Verben
Arithmetisch	ADD COMPUTE DIVIDE INSPECT (TALLYING) MULTIPLY SUBTRACT
Übersetzer-steuerung	COPY ENTER USE
Bedingt	ADD (SIZE ERROR) CALL (OVERFLOW) COMPUTE (SIZE ERROR) DELETE (INVALID KEY) DIVIDE (SIZE ERROR) IF MULTIPLY (SIZE ERROR) READ (END or INVALID KEY) RECEIVE (NO DATA) RETURN (END) REWRITE (INVALID KEY) SEARCH START (INVALID KEY) STRING (OVERFLOW) SUBTRACT (SIZE ERROR) UNSTRING (OVERFLOW) WRITE (INVALID KEY or END-OF-PAGE)
Zuweisungen	ACCEPT (DATE, DAY or TIME) ACCEPT MESSAGE COUNT INSPECT (REPLACING) MOVE STRING UNSTRING

Beendigung	STOP
Eingabe–Ausgabe	ACCEPT (bezeichner) CLOSE DELETE DISABLE DISPLAY ENABLE OPEN READ RECEIVE REWRITE SEND START STOP (Literal) WRITE
Interprogramm–kommunikation	CALL CANCEL
Sortieren	MERGE RELEASE RETURN SORT
Prozedur–verzweigung	ALTER CALL EXIT GO TO PERFORM
Tabellen–bearbeitung	SEARCH SET

'IF' ist ein Verb im Sinne von COBOL, obwohl es in der englischen Sprache kein Verb ist.

Arithmetische Ausdrücke

Definition eines arithmetischen Ausdrucks

Ein arithmetischer Ausdruck kann sein:

- ein Bezeichner für ein numerisches Datenelement
- ein numerisches Literal
- Bezeichner und Literale, die durch arithmetische Operatoren getrennt sind
- zwei durch einen arithmetischen Operator getrennte arithmetische Ausdrücke
- ein in Klammern gesetzter arithmetischer Ausdruck.

Vor allen arithmetischen Ausdrücken kann ein unärer Operator stehen. Die zulässigen Kombinationen aus Variablen, numerischen Literalen, arithmetischen Operatoren und Klammern werden in Tabelle 2-4 "Symbolkombinationen in arithmetischen Ausdrücken" aufgelistet.

Die in einem arithmetischen Ausdruck vorkommenden Bezeichner und Literale müssen entweder numerische Datenelemente oder numerische Literale darstellen, bei denen arithmetische Operationen durchgeführt werden können.

Arithmetische Operatoren

Es gibt fünf binäre arithmetische Operatoren und zwei unäre arithmetische Operatoren, die in arithmetischen Ausdrücken verwendet werden können. Sie werden durch ganz bestimmte Zeichen dargestellt, vor und nach denen ein Leerzeichen stehen kann.

Binäre Arithmetische Operatoren	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
**	Exponent

Unäre arithmetische Operatoren	Bedeutung
+	gleiche Wirkung wie Multiplikation mit numerischem Literal '+1'
-	gleiche Wirkung wie Multiplikation mit numerischem Literal '-1'

Bildungs- und Auswertungsregeln

1. In arithmetischen Ausdrücken können Klammern verwendet werden, um die Reihenfolge anzugeben, in der Elemente ausgewertet werden. Ausdrücke in Klammern werden zuerst ausgewertet; bei ineinander verschachtelten Klammern erfolgt die Auswertung von der am wenigsten umfassenden Menge bis zur umfassendsten Menge. Wenn keine Klammern verwendet werden oder Ausdrücke in Klammern auf der gleichen Schachtelungsstufe stehen, wird die folgende hierarchische Ausführungsordnung angewandt:

    1. Unäres Plus und Minus
    2. Exponentialfunktion
    3. Multiplikation und Division
    4. Addition und Subtraktion

2. Klammern werden entweder dazu benutzt, Mehrdeutigkeiten in der Logik zu beseitigen, wenn aufeinanderfolgende Operationen auf der gleichen hierarchischen Stufe erscheinen oder um die normale hierarchische Folge der Ausführung bei Ausdrücken dort zu modifizieren, wo es notwendig ist, Abweichungen von der normalen Priorität zu erzielen. Wenn die Ausführungsfolge nicht durch Klammern angegeben ist, so erfolgt die Ausführung aufeinanderfolgender Operationen der gleichen hierarchischen Stufe von links nach rechts.

3. Die Möglichkeiten, nach denen Operatoren, Variablen und Klammern in einem arithmetischen Ausdruck zusammengesetzt werden können, werden in Tabelle 2-4 zusammengefaßt, wobei

    a) der Buchstabe 'P' ein zulässiges Symbolpaar anzeigt,

    b) das Zeichen '-' ein unzulässiges Paar anzeigt,

    c) 'Variable' einen Bezeichner oder ein Literal anzeigt.

Erstes Symbol	Zweites Symbol				
	Variable	*/** + –	unäres + oder –	(	)
Variable	–	P	–	–	P
* / ** + –	P	–	P	P	–
unär + oder –	P	–	–	P	–
(	P	–	P	P	–
)	–	P	–	–	P

Tabelle 2-4 Symbolkombinationen in arithmetischen Ausdrücken

4. Ein arithmetischer Ausdruck kann nur mit dem Symbol '(', '+', '-' oder einer Variablen beginnen und kann nur mit einem ')' oder einer Variablen enden. Zwischen der linken und rechten Klammer eines arithmetischen Ausdruckes muß eine eindeutige Beziehung bestehen, so daß zu jeder linken Klammer die entsprechende rechte Klammer besteht.

5. Arithmetische Ausdrücke ermöglichen es dem Anwender, arithmetische Operationen zu kombinieren, ohne Berücksichtigung der Beschränkungen für das Format der Operanden und/ oder Empfangsdatenfelder (vgl. z.B. Syntaxregel 3 in "Die ADD-Anweisung" in diesem Kapitel).

Bedingte Ausdrücke

Bedingte Ausdrücke sind Bedingungen, deren Wahrheitswert untersucht wird und davon abhängig eine Verarbeitung eingeleitet wird. Bedingte Ausdrücke werden in IF, PERFORM und SEARCH-Anweisungen angegeben. Es gibt zwei Kategorien von Bedingungen, die mit bedingten Ausdrücken verknüpft sind: Einfache Bedingungen und zusammengesetzte Bedingungen. Jede kann in eine beliebige Anzahl von Klammerpaaren gesetzt sein, wobei deren Kategorie nicht geändert wird.

Einfache Bedingungen

Die einfachen Bedingungen sind der Vergleich, Klassenbedingung, Bedingungsnamen-Bedingung, Schalterstatus-Bedingung und Vorzeichenbedingung. Eine einfache Bedingung hat den Wahrheitswert 'true' oder 'false'. Wenn eine einfache Bedingung in Klammern gesetzt ist, ändert dies den einfachen Wahrheitswert nicht.

Vergleichsbedingung

Eine Vergleichbedingung wird zwischen zwei Operanden durchgeführt, die beide das Datenfeld sein können, das der Wert eines Bezeichners, eines Literals oder eines arithmetischen Ausdrucks sein kann. Eine Vergleichsbedingung hat einen Wahrheitswert 'true', wenn das Verhältnis zwischen Operanden existiert.
Der Vergleich von zwei numerischen Operanden ist ohne Berücksichtigung der in ihren entsprechenden USAGE-Klauseln angegebenen Formate zulässig. Bei allen anderen Vergleichen müssen die Operanden jedoch die gleiche USAGE-Klausel haben. Wenn einer der Operanden eine Datengruppe ist, so gelten die nichtnumerischen Vergleichsregeln.

Das allgemeine Format einer Vergleichsbedingung ist:

```
{ bezeichner-1                    }
{ literal-1                       }
{ arithmetischer-ausdruck-1       }

{ IS [ NOT ] GREATER THAN         }
{ IS [ NOT ] LESS THAN            }  { bezeichner-2                 }
{ IS [ NOT ] EQUAL TO             }  { literal-2                    }
{ IS [ NOT ] >                    }  { arithmetischer-ausdruck-2    }
{ IS [ NOT ] <                    }
{ IS [ NOT ] =                    }
```

Anmerkung

Die Vergleichszeichen '<', '>' und '=' sind nicht unterstrichen, um Verwechslungen mit anderen Symbolen wie z.B. '≥' (größer als oder gleich) zu vermeiden.

Der erste Operand (bezeichner-1, literal-1 oder arithmetischer-ausdruck-1) wird 'Subjekt' der Bedingung genannt, der zweite Operand (bezeichner-2 oder literal-2 oder arithmetischer ausdruck-2) wird 'Element' der Bedingung genannt. Die Vergleichsbedingung muß mindestens einen Bezug zu einer Variablen enthalten.

Der Vergleichsoperator gibt die Art des Vergleichs an. Vor und nach jedem reservierten Wort das ein Vergleichszeichen enthält, muß ein Leerzeichen stehen. Sofern verwendet, sind 'NOT' und das nächste Schlüsselwort oder Vergleichszeichen ein einziger Vergleichsoperator, der den Vergleich definiert, der für den Wahrheitswert ausgeführt wird; z.B. 'NOT EQUAL' ist ein Wahrheitstest für ein 'ungleich'.

Vergleich: 'NOT GREATER' ist ein Wahrheitstest für einen 'gleich' oder 'kleiner' Vergleich. Die Bedeutung der Vergleichsoperatoren wird in Tabelle 2-5 gezeigt.

Bedeutung	Vergleichsoperator
Größer als oder nicht größer als	IS NOT GREATER THAN IS NOT >
Kleiner als oder nicht kleiner als	IS NOT LESS THAN IS NOT <
Gleich oder nicht gleich	IS NOT EQUAL TO IS NOT =

Tabelle 2-5 Vergleichsoperatoren

Die Vergleichszeichen '>', '<' und '=' sind nicht unterstrichen, um Verwechslungen mit anderen Symbolen, wie '≥' (größer als oder gleich) zu vermeiden.

Vergleich von numerischen Operanden

Bei Operanden, deren Klasse numerisch ist, wird ein Vergleich in bezug auf den numerischen Wert der Operanden durchgeführt. Die Länge des Literals oder des arithmetischen Ausdrucks als Operanden gemessen, in der Anzahl der dargestellten Ziffern, ist bedeutungslos. Null wird als eindeutiger Wert ohne Berücksichtigung des Vorzeichens betrachtet.

Vergleich zwischen diesen Operanden ist zulässig ohne Berücksichtigung der Art, in der ihre Verwendung beschrieben ist. Numerische Operanden ohne Vorzeichen werden beim Vergleich als positiv betrachtet.

Vergleich von nichtnumerischen Operanden

Bei nichtnumerischen Operanden oder einem numerischen und einem nichtnumerischen Operanden wird ein Vergleich in bezug auf eine angegebene Sortierfolge von Zeichen betrachtet (vgl. Der OBJECT-COMPUTER-Paragraph in diesem Kapitel). Wenn einer der Operanden als numerisch angegeben ist, so muß er ein ganzzahliges Datenfeld oder ein ganzzahliges Literal sein und:

1. wenn der nichtnumerische Operand ein Datenelement oder ein nichtnumerisches Literal ist, so wird der numerische Operand behandelt, als ob er in ein alphanumerisches Datenelement der gleichen Größe übertragen worden wäre wie das numerische Datenfeld (in Standard-Datenformatzeichen), und der Inhalt dieses alphanumerischen Datenfeldes wird dann mit dem nichtnumerischen Operanden verglichen. (vgl. Die MOVE-Anweisung in diesem Kapitel und das PICTURE-Zeichen 'P' im Abschnitt 'Verwendete Symbole' in diesem Kapitel);

2. wenn der nichtnumerische Operand eine Datengruppe ist, so wird der numerische Operand behandelt, als ob er eine Datengruppe der gleichen Größe wäre wie das numerische Datenfeld (in Standard-Datenformatzeichen), und der Inhalt dieser Datengruppe wird dann mit dem nichtnumerischen Operanden verglichen (vgl. Die MOVE-Anweisung in diesem Kapitel und das PICTURE-Zeichen 'P' im Abschnitt 'Verwendete Symbole' in diesem Kapitel).

3. Ein nicht ganzzahliger numerischer Operand kann nicht mit einem nichtnumerischen Operanden verglichen werden.

   Die Größe eines Operanden entspricht der Gesamtzahl der Standard-Datenformatzeichen in diesem Operanden. Numerische und nichtnumerische Operanden können nur verglichen werden, wenn ihre Verwendung die gleiche ist und der ANSI-Schalter gesetzt ist. Wenn der ANSI-Schalter nicht gesetzt ist, können numerische und nichtnumerische Operanden ohne Rücksicht auf ihre Verwendung verglichen werden.

Es gibt zwei Fälle, die zu beachten sind:

1. Operanden gleicher Größe: Wenn die Operanden gleich groß sind, erfolgt der Vergleich effektiv durch Vergleichen der Zeichen in den entsprechenden Zeichenpositionen von der höchsten Stelle anfangend bis entweder ein Paar ungleicher Zeichen auftritt oder die niedrigste Stelle des Operanden erreicht ist. Die Operanden werden als gleich angesehen, wenn alle Zeichenpaare bis zum letzten Paar gleich sind und wenn das niederwertigste Zeichen erreicht ist.

   Das erste auftretende Paar ungleicher Zeichen wird verglichen, um ihre Position relativ zur Sortierfolge zu bestimmen. Der Operand, der das Zeichen enthält, das in der Sortierfolge höher eingeordnet ist, wird als der größere Operand angesehen.

2. Operanden ungleicher Größe: Wenn die Operanden nicht gleich groß sind, erfolgt der Vergleich, als ob der kürzere Operand rechts durch entsprechend viele Leerzeichen erweitert wird, so daß die Operanden gleich lang sind.

Klassenbedingung

Die Klassenbedingung bestimmt, ob der Operand numerisch ist, d.h. ausschließlich aus den Zeichen '0', '1', '2', '3', ... '9' mit oder ohne Rechenvorzeichen besteht oder alphabetisch ist, d.h., ausschließlich aus den Zeichen 'A', 'B', 'C', ..., 'Z' und/oder dem Leerzeichen besteht. Das allgemeine Format für die Klassenbedingung ist:

```
bezeichner IS    [ NOT ]  { NUMERIC      }
                          { ALPHABETIC    }
```

Die USAGE-Angabe des getesteten Operanden muß DISPLAY sein. Sofern verwendet, gibt 'NOT' und das nächste Schlüsselwort eine Klassenbedingung an, die den Klassentest definiert, der für den Wahrheitswert ausgeführt wird; z.B. 'NOT NUMERIC' ist ein Wahrheitstest für die Bestimmung, ob ein Operand nicht numerisch ist.

Der NUMERIC-Test kann nicht bei einem Datenfeld durchgeführt werden, dessen Datenerklärung das Datenfeld als alphabetisch oder als eine Datengruppe beschreibt, die aus Datenelementen besteht, auch wenn dessen Datenerklärung ein oder mehrere Rechenvorzeichen anzeigt. Wenn die Datenerklärung des zu testenden Datenfeldes kein Rechenvorzeichen anzeigt, so wird das zu testende Datenfeld nur dann als numerisch bestimmt, wenn der Inhalt numerisch und kein Rechenvorzeichen vorhanden ist. Wenn die Datenerklärung des Datenfeldes das Vorhandensein eines Rechenvorzeichens anzeigt, so wird das zu testende Datenfeld nur dann als numerisch angesehen, wenn der Inhalt numerisch und ein zulässiges Rechenvorzeichen vorhanden ist. Zulässige Rechenvorzeichen für mit der SIGN IS SEPARATE-Klausel beschriebene Datenfelder sind die Standard-Datenformatzeichen '+' und '-'.

Der ALPHABETIC-Test kann nicht bei einem Datenfeld durchgeführt werden, dessen Datenerklärung das Datenfeld als numerisch beschreibt. Das zu testende Datenfeld wird nur dann als alphabetisch bestimmt, wenn der Inhalt aus einer beliebigen Kombination der alphabetischen Zeichen 'A' bis 'Z' und/oder dem Leerzeichen besteht.

Bedingungsnamen-Bedingung

In einer Bedingungsnamen-Bedingung wird eine Bedingungsvariable getestet, um festzustellen, ob ihr Wert gleich einem der mit dem Bedingungsnamen verknüpften Wert ist oder nicht. Das allgemeine Format für die Bedingungsnamen-Bedingung ist:

```
bedingungsname
```

Schalterstatus-Bedingung

Wenn bedingungsname mit einem oder mehreren Wertebereichen verknüpft ist, so wird die bedingte Variable getestet, um zu bestimmen, ob ihr Wert in diesen Bereich, einschließlich der Randwerte, fällt oder nicht.

Die Regeln für den Vergleich von bedingten Variablen mit einem Bedingungsnamenwert sind die gleichen, wie für die Vergleichsbedingung.

Das Ergebnis des Tests ist wahr, wenn eines der Werte entsprechend des Bedingungsnamens gleich dem Wert seiner zugehörigen bedingten Variablen ist.

Schalterstatus Bedingung

Eine Schalterstatus Bedingung bestimmt den 'on' oder 'off' Status eines vom Hersteller definierten Schalters. Der Schalter und der 'on' oder 'off' Wert, der mit dem Test verknüpft ist, muß im SPECIAL-NAMES-Paragraph der ENVIRONMENT DIVISION angegeben sein. Das allgemeine Format für den Schalterstatus-Test ist wie folgt:

```
bedingungsname
```

Das Ergebnis des Tests ist wahr, wenn der Schalter auf die angegebene Position, die dem Bedingungsnamen entspricht, gesetzt ist.

Vorzeichenbedingung

Die Vorzeichenbedingung bestimmt, ob der Wert eines arithmetischen Ausdruckes kleiner als, größer als oder gleich Null ist oder nicht. Das allgemeine Format für eine Vorzeichenbedingung ist wie folgt:

```
arithmetischer-ausdruck IS [ NOT ] { POSITIVE }
                                   { NEGATIVE }
                                   { ZERO     }
```

Sofern verwendet, gibt 'NOT' und das nächste Schlüsselwort eine einzige Vorzeichenbedingung an, die den Test definiert, der für den Wahrheitswert ausgeführt wird; z.B. 'NOT ZERO' ist ein Wahrheitstest für einen (positiven oder negativen) Wert, der ungleich Null ist. Ein Operand ist positiv, wenn sein Wert größer als Null, negativ, wenn sein Wert kleiner als Null und Null, wenn sein Wert gleich Null ist. Der arithmetische Ausdruck muß mindestens eine Variable ansprechen.

Komplexe Bedingungen

Eine komplexe Bedingung wird durch Verknüpfen von einfachen Bedingungen, zusammengesetzten Bedingungen und/oder komplexen Bedingungen mit logischen Bindeworten (logische Operatoren 'AND' und 'OR') oder durch Negieren dieser Bedingungen mit logischer Negation (logischer Operator 'NOT') gebildet. Der Wahrheitswert einer komplexen Bedingung, ob in Klammern gesetzt oder nicht, ist der Wahrheitswert, der aus dem gegenseitigen Zusammenspiel aller logischer Operatoren, der Wahrheitswerte von einfachen Bedingungen, oder aber der Zwischenergebnisse von Bedingungen, die logisch verbunden oder logisch negiert sind, resultiert.

Logische Operatoren und ihre Bedeutung:

Logischer Operator	Bedeutung
AND	Logische Konjunktion; der Wahrheitswert ist 'true', wenn alle beiden mit AND verknüpten Bedingungen wahr sind; 'false', wenn eine der beiden verbundenen Bedingungen falsch ist.
OR	Logisch einschließliches oder; der Wahrheitswert ist 'true', wenn eine oder beide der eingeschlossenen Bedingungen wahr sind; 'false', wenn beide eingeschlossenen Bedingungen falsch sind.
NOT	Logische Negation oder Komplement des Wahrheitswertes; der Wahrheitswert ist 'true', wenn die Bedingung falsch ist; 'false', wenn die Bedingung wahr ist.

Vor und nach logischen Operatoren muß ein Leerzeichen stehen.

Negierte einfache Bedingungen

Eine einfache Bedingung ist negiert, wenn der logische Operator 'NOT' verwendet wird. Die negierte einfache Bedingung liefert den entgegengesetzten Wahrheitswert für eine einfache Bedingung. Daher ist der Wahrheitswert einer negierten einfachen Bedingung nur dann 'true', wenn der Wahrheitswert der einfachen Bedingung 'false' ist; der Wahrheitswert einer negierten einfachen Bedingung ist nur dann 'false', wenn der Wahrheitswert der einfachen Bedingung 'true' ist. Wenn eine negierte einfache Bedingung in Klammern gesetzt ist, wird der Wahrheitswert dadurch nicht geändert.

Das allgemeine Format für eine negierte einfache Bedingung ist:

```
NOT einfache-bedingung
```

Zusammengesetzte und negierte zusammengesetzte Bedingungen:

Eine zusammengesetzte Bedingung resultiert aus der Verbindung von Bedingungen mit einem der logischen Operatoren 'AND' oder 'OR'. Das allgemeine Format einer zusammengesetzten Bedingung ist:

```
bedingung { { { AND } } bedingung }
          { { { OR  } }           }
```

wobei bedingung sein kann:

a) eine einfache Bedingung oder

b) eine negierte einfache Bedingung oder

c) eine zusammengesetzte Bedingung oder

d) eine negierte zusammengesetzte Bedingung; d.h., der logische Operator 'NOT' gefolgt von einer zusammengesetzten Bedingung in Klammern oder

e) Zusammensetzungen der obigen, entsprechend den in Tabelle 2-6 zusammengefaßten Regeln.

Obwohl Klammern nie verwendet werden müssen, wenn entweder 'AND' oder 'OR' (jedoch nie beide gleichzeitig) in einer zusammengesetzten Bedingung allein verwendet werden, können Klammern verwendet werden, um einen Wahrheitswert zu berechnen, wenn eine Mischung aus 'AND', 'OR' und 'NOT' verwendet wird. (vgl. Tabelle 2-6 "Bedingungskombinationen, logische Operatoren und Klammern", sowie die Regeln für die Auswertung von Bedingungen in diesem Kapitel).

Tabelle 2-6 zeigt die Möglichkeiten, mit denen Bedingungen und logische Operatoren zusammengesetzt und in Klammern gesetzt werden können. Zwischen der linken und der rechten Klammer muß eine eineindeutige Beziehung bestehen, so daß jede linke Klammer eine entsprechende rechte Klammer als Gegenstück hat. In der Tabelle wird eine Folge von links nach rechts für die Elemente angenommen.

Element	Zulässiger Platz in Bedingungs-ausdruck	Vor dem Element kann nur stehen:	Nach dem Element kann nur stehen:
einfache Bedingung	beliebig	OR, NOT, AND, (	OR, AND,)
OR, oder AND	nicht als erstes oder letztes Element	einfache Bedingung,)	einfache Bedingung, NOT, (
NOT	nicht als letztes Element	OR, AND, (	einfache Bedingung, (
(	nicht als letztes Element	OR, NOT, AND, (	einfache Bedingung, NOT, (
)	nicht als erstes Element	einfache Bedingung,)	OR, AND,)

Tabelle 2-6 Bedingungskombinationen, logische Operatoren und Klammern

Daher ist das Elementpaar 'OR NOT' zulässig, während das Paar 'NOT OR' unzulässig ist; 'NOT (' ist zulässig, während 'NOT NOT' unzulässig ist.

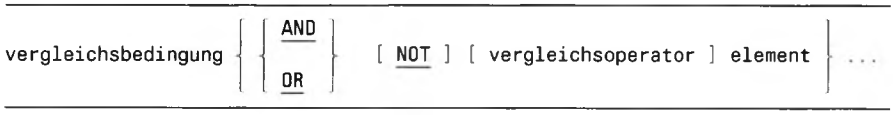
Abgekürzte zusammengesetzte Vergleichsbedingung

Jede Vergleichsbedingung kann abgekürzt werden, wenn einfache oder negiert einfache Vergleichsbedingungen mit logischen Bindeworten in einer unmittelbaren Folge zusammengesetzt werden, so daß ein nachfolgender Vergleich ein Subjekt oder ein Subjekt und einen Vergleichsoperator enthält, das mit der vorangehenden Vergleichsbedingung identisch ist und keine Klammern innerhalb einer solchen Aufeinanderfolge verwendet werden. Das erreicht man durch:

die Auslassung des Subjektes der Vergleichsbedingung oder

die Auslassung des Subjektes und des Vergleichsoperators der Vergleichsbedingung.

Das Format für eine abgekürzte zusammengesetzte Vergleichsbedingung ist:



Innerhalb einer Folge von Vergleichsbedingungen können beide obengenannten Abkürzungsformen verwendet werden. Die Auswirkung der Verwendung solcher Abkürzungen ist die gleiche, als ob das letzte vorangegangene Subjekt anstelle des ausgelassenen Subjekts eingefügt worden wäre und der letzte angegebene Vergleichsoperator anstelle des ausgelassenen Vergleichsoperators eingefügt worden wäre. Das Ergebnis einer solchen impliziten Einfügung muß mit den Regeln in Tabelle 2-6 "Bedingungskombinationen, logische Operatoren und Klammern" übereinstimmen. Diese Einfügung eines ausgelassenen Subjektes und/oder Vergleichsoperators ist beendet, sobald eine vollständige einfache Bedingung innerhalb einer komplexen Bedingung auftritt.

Die Interpretation, die für die Verwendung des Wortes 'NOT' in einer abgekürzten zusammengesetzten Vergleichsbedingung gilt, ist wie folgt:

1. Wenn das Wort, das unmittelbar nach 'NOT' folgt, 'GREATER', '>', 'LESS', '<', 'EQUAL', '=' ist, dann ist 'NOT' Teil des Vergleichsoperators; andernfalls
2. wird 'NOT' als ein logischer Operator interpretiert und die implizite Einfügung von Subjekt oder Vergleichsoperator resultiert daher in einer negierten Vergleichsbedingung.

Es folgen einige Beispiele für abgekürzte zusammengesetzte und negierte zusammengesetzte Vergleichsbedingungen und erweiterte Äquivalente:

Abgekürzte zusammengesetzte Vergleichsbedingung	Ausführliche gleichwertige Darstellung
`a > b AND NOT < c OR d`	`((a > b) AND (a NOT < c)) OR (a NOT > d)`
`a NOT EQUAL b OR c`	`(a NOT EQUAL b) OR (a NOT EQUAL c)`
`NOT a = b OR c`	`(NOT (a = b)) OR (a = c)`
`NOT (a GREATER b OR < c)`	`NOT ((a GREATER b) OR (a < c))`
`NOT (a NOT > b AND c AND NOT d)`	`NOT ((((a NOT > b) AND (a NOT > c)) AND (NOT (a NOT > d)))`

Regeln für die Auswertung von Bedingungen

Es können Klammern verwendet werden, um die Reihenfolge anzugeben, in der einzelne Bedingungen komplexer Bedingungen ausgewertet werden müssen, wenn es notwendig ist, von der impliziten Auswertungspriorität auszugehen. Bedingungen in Klammern werden zuerst ausgewertet, und innerhalb verschachtelter Klammern erfolgt die Auswertung von der am wenigsten umfassenden Bedingung aus bis zur umfassendsten Bedingung. Wenn keine Klammern verwendet werden oder Bedingungen in Klammern auf der gleichen Schachtelungsstufe stehen, wird die folgende hierarchische Ordnung logischer Bewertung eingehalten, bis der endgültige Wahrheitswert bestimmt wird:

1. Für arithmetische Ausdrücke werden Werte erstellt. (vgl. Bildung- und Auswertungsregeln in Arithmetische Ausdrücke in diesem Kapitel).

2. Wahrheitswerte für einfache Bedingungen werden in der folgenden Reihenfolge ermittelt:

   Vergleich (nach der Umwandlung irgendwelcher abgekürzter Vergleichsbedingungen in die ausführliche Darstellung)
   Klasse
   Bedingungsname
   Schalter-Status
   Vorzeichen

3. Wahrheitswerte für negierte Bedingungen werden ermittelt.
4. Wahrheitswerte für zusammengesetzte Bedingungen werden ermittelt:

   logische Operatoren 'AND',

   gefolgt von logischen Operatoren 'OR'
5. Wahrheitswerte für negierte zusammengesetzte Bedingungen werden ermittelt.
6. Wenn die Folge der Auswertung nicht vollständig durch Klammern angegeben ist, so erfolgt die Auswertung aufeinanderfolgender Operationen der gleichen Schachtelungsstufe von links nach rechts.

Gemeinsame Angaben und allgemeine Regeln für Anweisungsformate

In den folgenden Anweisungsbeschreibungen erscheinen einige Angaben häufig: die ROUNDED-Angabe, die SIZE ERROR-Angabe und die CORRESPONDING-Angabe.

Diese werden weiter unten beschrieben; ein Ergebnisbezeichner ist derjenige Bezeichner, der mit dem Ergebnis einer arithmetischen Operation verknüpft ist.

Die ROUNDED-Angabe

Wenn nach Dezimalpunktausrichtung die Anzahl der Dezimalstellen eines Ergebnisses einer arithmetischen Operation größer ist als die Anzahl der Stellen, die für den Ergebnisbezeichner vorhanden sind, erfolgt die Abschneidung relativ zur Größe, die für diesen Ergebnisbezeichner vorgesehen ist. Wenn Rundung erforderlich wird, so wird der absolute Wert des Ergebnisbezeichners um eins erhöht, wann immer die signifikanteste Ziffer des Überhanges größer oder gleich fünf ist.

Wenn die niederwertigen ganzzahligen Positionen in einem Ergebnisbezeichner durch das Zeichen 'P' für PICTURE für den Ergebnisbezeichner dargestellt sind, erfolgt Rundung oder Abschneidung relativ zu der am weitesten rechts stehenden Ganzzahlposition, für die Speicherplatz zur Verfügung steht.

Die SIZE ERROR-Angabe

Wenn nach Dezimalpunktausrichtung der absolute Wert eines Ergebnisses den größten Wert überschreitet, der im verknüpften Ergebnisbezeichner enthalten sein kann, besteht eine Überlaufbedingung. Eine Division durch Null führt immer zu einer Überlaufbedingung. Die Überlaufbedingung gilt nur für die Endergebnisse, außer in MULTIPLY und DIVIDE-Anweisungen, bei denen die Überlaufbedingung auch für die Zwischenergebnisse gilt. Wenn die ROUNDED-Angabe vorhanden ist, erfolgt die Rundung vor Überprüfung des Überlaufs. Wenn eine solche Überlaufbedingung auftritt, hängt die nachfolgende Operation davon ab, ob SIZE ERROR angegeben ist oder nicht:

SIZE ERROR ist nicht angegeben:

Wenn eine Überlaufbedingung auftritt, so ist der betroffene Wert dieses Ergebnisbezeichners bzw. dieser Ergebnisbezeichner undefiniert. Werte von Ergebnisbezeichnern, für die keine Überlaufbedingungen auftreten, werden nicht durch Überlauf betroffen, die für einen oder mehrere andere Ergebnisbezeichner bei Ausführung dieser Operation auftreten.

SIZE ERROR ist angegeben:

Wenn eine Überlaufbedingung auftritt, werden die durch den Überlauf betroffenen Werte des oder der Ergebnisbezeichner nicht geändert. Werte des oder der Ergebnisbezeichner, für die keine Überlaufbedingung auftritt, sind nicht durch Überlauf betroffen, die für einen oder mehrere andere Ergebnisbezeichner bei Ausführung dieser Operation auftreten. Nach Abschluß der Ausführung dieser Operation wird die unbedingte Anweisung in der SIZE ERROR-Angabe ausgeführt.

Bei der ADD-Anweisung mit der CORRESPONDING-Angabe und der SUBTRACT-Anweisung mit der CORRESPONDING-Angabe (sofern irgendeine der einzelnen Operationen eine Überlaufbedingung verursacht), wird die unbedingte Anweisung in der SIZE ERROR-Angabe nicht ausgeführt, bis alle einzelnen Additionen oder Subtraktionen durchgeführt sind.

Die CORRESPONDING-Angabe

Im folgenden Text müssen sowohl d1 als auch d2 Bezeichner sein, die sich auf Datengruppen beziehen. Ein Datenfeldpaar, ein Datenfeld von d1 und eines von d2, entsprechen einander, wenn die folgenden Bedingungen vorliegen:

1. Ein Datenfeld in d1 und ein Datenfeld in d2 sind nicht durch das Schlüsselwort FILLER gekennzeichnet und haben den gleichen Datennamen und die gleichen Bezeichner; bis hin zu, aber nicht einschließlich, d1 und d2.
2. Mindestens eines der Datenfelder ist ein Datenelement im Falle einer MOVE-Anweisung mit CORRESPONDING-Angabe; beide Datenfelder sind numerische Datenelemente im Falle einer ADD-Anweisung mit CORRESPONDING-Angabe oder der SUBTRACT-Anweisung mit CORRESPONDING-Angabe.
3. Die Beschreibung von d1 und d2 darf nicht die Stufennummern 66, 77 oder 88 oder die USAGE IS INDEX-Klausel enthalten.
4. Ein Datenfeld, das d1 oder d2 untergeordnet ist und eine REDEFINES-, RENAMES-, OCCURS- oder USAGE IS INDEX-Klausel enthält, wird ignoriert, wie auch die Datenfelder, die dem Datenfeld untergeordnet sind, das die REDEFINES-, OCCURS- oder USAGE IS INDEX-Klausel enthält. d1 und d2 können jedoch REDEFINES- oder OCCURS-Klauseln besitzen oder Datenfeldern mit REDEFINES- oder OCCURS-Klauseln untergeordnet sein.

Arithmetische Anweisungen

Arithmetische Anweisungen sind die ADD, COMPUTE, DIVIDE, MULITPLY und SUBTRACT-Anweisung. Gemeinsame Merkmale sind:

1. Die Datenerklärungen der Operanden müssen nicht gleich sein; alle notwendigen Umwandlungen und Dezimalpunktausrichtungen werden während der Berechnung durchgeführt.
2. Die maximale Größe jedes Operanden beträgt 18 Dezimalziffern. Auch wenn mehrere Operanden angegeben sind, darf die maximale Anzahl der Stellen eines Rechen-Zwischenfeldes 18 nicht übersteigen.

Überlappende Operanden

Wenn ein Sende- und ein Empfangsdatenfeld in einer arithmetischen Anweisung oder in einer INSPECT-, MOVE-, SET-, STRING- oder UNSTRING-Anweisung einen Teil ihrer Speicherbereiche miteinander gemeinsam haben, so ist das Ergebnis der Ausführung einer solchen Anweisung undefiniert.

Mehrere Ergebnisbezeichner in arithmetischen Anweisungen

ADD-, COMPUTE-, DIVIDE-, MULTIPLY- und SUBTRACT-Anweisungen können mehrere Ergebnisbezeichner haben. Solche Anweisungen verhalten sich, als ob sie auf folgende Weise geschrieben worden wären:

1. als eine Anweisung, die alle notwendigen arithmetischen Operationen ausführt, um das Ergebnis in den Empfangsdatenfeldern speichern zu können und die dieses Ergebnis in einem temporären Speicherplatz ablegt,

2. als eine Anweisungsfolge, die den Wert dieses temporären Speicherplatzes in ein einzelnes Ergebnisfeld überträgt.

   Das Ergebnis der Anweisung

   ```
   ADD a, b, c TO c, d (c), e
   ```

   ist äquivalent zu

   ```
   ADD a, b, c GIVING temp
   ADD temp TO c
   ADD temp TO d (c)
   ADD temp TO e
   ```

   wobei 'temp' ein vom Übersetzer gelieferter Zwischenergebnisbezeichner ist.

Inkompatible Daten

Ein Ergebnis ist nicht definiert, wenn mit Ausnahme der Klassenbedingung auf den Inhalt eines Datenfeldes in der PROCEDURE DIVISION Bezug genommen wird und der Inhalt dieses Datenfeldes nicht mit der für dieses Datenfeld in der entsprechenden PICTURE-Klausel angegebenen Klasse kompatibel ist.

Empfangsdatenfelder mit Vorzeichen

Wenn das Empfangsdatenfeld in einer arithmetischen Anweisung oder in einer MOVE-Anweisung ein mit einem Vorzeichen versehenes numerisches oder numerisch druckaufbereitetes Datenfeld ist, so wird das Vorzeichen im Empfangsdatenfeld unabhängig vom Abschneiden der absoluten numerischen Daten im Empfangsdatenfeld übertragen. Es ist daher möglich, daß der numerische Wert Null, das Vorzeichen jedoch negativ ist.

Bildschirmgeräte

Der Bildschirm wird direkt vom Laufzeitsystem über einen Puffer angesteuert. Man speichert Daten in und aus diesem Puffer mit Hilfe von ACCEPT- und DISPLAY-Anweisungen. Jede ACCEPT oder DISPLAY-Anweisung beginnt am Anfang des Bildschirmpuffers, sofern POSITION nicht angegeben ist. Die Syntax ist auf die Eingabe in bzw. die Ausgabe aus einem einzelnen Datennamen beschränkt. Der Datenname kann eine Datengruppe sein und mehrere dieser Datengruppen können den gleichen Speicherbereich redefinieren.

Die Verwendung von FILLER-Datenfeldern in Datensatzbeschreibungen, die für die Eingabe bzw. Ausgabe an einem Bildschirm verwendet werden, unterliegt besonderen Regeln. Bei der Ausgabe bewirkt jedes FILLER-Datenfeld in einem Datensatz, daß die Ausgabe der Zeichenpositionen, die es definiert, unterdrückt wird. Bei der Eingabe unterdrückt jedes FILLER-Datenfeld die Eingabe des Bedieners am Terminal in die Zeichenpositionen, die es definiert.

Die ACCEPT-Anweisung

Funktion

Durch die ACCEPT-Anweisung werden Daten, die am Terminal eingegeben werden, in einem angegebenen Datenfeld dem Programm zur Verfügung gestellt.

Allgemeine Formate

Format 1

```
ACCEPT bezeichner [ FROM { merkname  } ]
                         { CONSOLE }
```

Format 2

```
ACCEPT datenname-1 [ AT { datenname-2 } ] FROM CRT
                        { literal-1   }
```

Format 3

```
ACCEPT bezeichner [ FROM { DATE } ]
                         { DAY  }
                         { TIME }
```

Syntaxregeln

merkname im Format 1 muß auch im SPECIAL-NAMES Paragraph in der ENVIRONMENT DIVISION angegeben und mit dem Terminal verknüpft sein.

Allgemeine Regeln

1. Format 1 ist die Standard-ANSI ACCEPT-Anweisung

   Format 2 ist das erweiterte ACCEPT-Format.

   Die beiden Formate werden durch die zugehörigen FROM-Angaben unterschieden; standardmäßig wird FROM CONSOLE angenommen. Ein vom Programmierer definierter Merkname kann verwendet werden, sofern dieser mit einem Systemgerät im SPECIAL-NAMES Paragraph verknüpft ist (vgl. Der SPECIAL-NAMES-Paragraph in diesem Kapitel). Der Standard kann jedoch durch die Angabe CONSOLE IS CRT im SPECIAL-NAMES-Paragraphen geändert werden, so daß FROM CRT Standard wird. Genauere Angaben entnehmen Sie bitte der LII COBOL-Bedienungsanleitung.

Regeln für Format 1

2. Durch die ACCEPT-Anweisung werden Daten vom Bildschirm übertragen. Diese Eingabedaten ersetzen den Inhalt des Datenfeldes, das durch bezeichner benannt ist.

3. Die Daten werden als eine ganzzahlige Anzahl von Datensätzen (bis maximal 12) übertragen. Die Größe eines Datensatzes wird durch die Übersetzeranweisung CRTWIDTH definiert (vgl. LEVEL II COBOL Bedienungsanleitung). Da jeder Datensatz auf der Tastatur eingegeben wird, kann er entsprechend den Betriebssystemregeln für die Zeichenaufbereitung zeilenweise aufbereitet werden. Jeder Datensatz wird abgeschlossen, wenn die RETURN-Taste gedrückt wurde oder wenn der Datensatz exakt aufgefüllt ist. Nachdem jeder Datensatz übertragen worden ist, wird der Cursor an den Anfang der nächsten Zeile gesetzt.

4. Wenn die Größe des übertragenen Datensatzes der Größe des Emfangsdatenfeldes entspricht, werden die übertragenen Daten im Empfangsdatenfeld gespeichert.

5. Wenn die Eingabezeile ungleich der Größe des Empfangsdatenfeldes ist, dann:

   a) werden, wenn die Größe des Empfangsdatenfeldes (oder der Teil des Empfangsdatenfeldes, der noch nicht von übertragenen Daten belegt ist) die Größe des zu übertragenden Datensatzes überschreitet, die übertragenen Daten mit Ausrichtung links im Empfangsdatenfeld gespeichert (oder in dem Teil des Empfangsdatenfeldes, der noch nicht belegt ist), und ein weiterer Datensatz wird angefordert.

   b) werden, wenn die Größe des zu übertragenden Datensatzes die Größe des Empfangsdatenfeldes überschreitet (oder den Teil des Empfangsdatenfeldes, der noch nicht von übertragenen Daten belegt ist), nur die am weitesten links stehenden Zeichen der Eingabedaten im Empfangsdatenfeld (oder dem noch unbelegten Teil) gespeichert. Die restlichen Zeichen der Eingabedaten, die keinen Platz im Empfangsdatenfeld haben, werden nicht berücksichtigt.

Regeln für Format 2

1. Durch die ACCEPT-Anweisung werden Daten vom Bildschirm nach datenname-1 übertragen.

2. datenname-1 wird als Definition des Bildschirmbereiches angesehen, in dem Datenelemente Bereichen auf dem Bildschirm entsprechen, in die man Daten eingeben kann. FILLER-Felder entsprechen Bildschirmbereichen, auf die man nicht zugreifen kann. datenname-1 darf nicht subskribiert werden.

3. Die Datenfelder in datenname-1 können alphanumerisch, numerisch oder druckaufbereitet sein. Nicht ganzzahlige numerische Datenfelder werden wie zwei getrennte ganzzahlige numerische Felder behandelt, und druckaufbereitete Felder werden als alphanumerische Felder behandelt, mit Ausnahme der in Regel 11 beschriebenen Fälle.

4. AT datenname-2 oder literal-1 definieren die Bildschirmposition des am weitesten links stehenden Zeichens der Daten. Eines der beiden muß ein PIC 9999 Feld sein. Die rechten beiden Stellen werden als Zeilenadresse im Bereich von eins bis 25 angenommen. Die linken beiden Stellen werden als eine Zeichenposition im Bereich von eins bis 80, der Zeichenbreite des Bildschirms, angenommen.

5. datenname-1 kann auf einen Datensatz, eine Gruppe oder ein Datenelement Bezug nehmen; er darf jedoch nicht subskribiert sein. Für datenname-1 kann die REDEFINES-Klausel verwendet werden, wobei die erste Beschreibung der Daten verwendet wird und nachfolgende Beschreibungen nicht berücksichtigt werden. OCCURS und verschachtelte OCCURS können ebenfalls verwendet werden.

6. Bei der Ausführung der ACCEPT-Anweisung wird der Cursor an die Bildschirmposition gesetzt, die das erste beschreibbare Datenfeld bezeichnet. Alternativ wird der Cursor dahin positioniert, wie in datenname angegeben, wenn CURSOR im SPECIAL-NAMES-Paragraphen angegeben ist. Die CURSOR-Position wird in dem Datennamen im gleichen Format gespeichert, wie die Bildschirmposition, die durch datenname-2 gekennzeichnet ist. Wenn der CURSOR-datenname den Wert SPACE oder ZERO hat, wird er behandelt, als ob er nicht angegeben worden wäre und damit auch nicht aktualisiert. Wird eine zulässige Bildschirmposition angegeben, die auf ein FILLER-Datenfeld zeigt, so wird die Schreibmarke auf die danach folgende nicht-FILLER-Eingabeposition gesetzt. Wenn diese Position nach dem letzten Datenfeld liegt, wird der Cursor an den Anfang des ersten beschreibbaren Datenfeldes des akzeptierten Satzes positioniert. Der CURSOR-datenname hält die letzte Cursorposition nach Abschluß der Ausführung einer ACCEPT-Anweisung fest.

7. Wenn weder AT ... noch FROM CRT angegeben ist, so wird standardmäßig FROM CONSOLE angenommen (vgl. Regel 1 oben).

8. Sobald man ein Zeichen eingibt, bewegt sich der Cursor jeweils um eine Zeichenposition nach rechts oder auf die erste Position des nächsten Datenfeldes.

9. Wenn das Datenfeld ein ganzzahliges numerisches Zeichen ist, werden nur numerische Zeichen (0-9) in diesem Datenfeld akzeptiert. Durch die Eingabe des Dezimalpunktzeichens ('.' oder ',', wie in der DECIMAL POINT-Angabe angegeben) wird beim Übertragen eines numerischen Datenfeldes dieses Datenfeld rechtsbündig gespeichert und eventuell links mit Nullen aufgefüllt.

10. Bei der Eingabe springt der Cursor von einer Position zur nächsten Position eines Datenfeldes. Wenn keine solche Position mehr auf dem in datenname angegebenen Bildschirmteil vorhanden ist, verbleibt er auf der aktuellen Position. Alle weiteren danach eingegebenen Zeichen definieren das letzte Zeichen neu.

11. Wenn man die RETURN-Taste drückt, schließt man die Eingabe ab, wobei die Steuerung dann an die nächste Anweisung nach der ACCEPT-Anweisung übergeben wird. Bevor die Steuerung an die nächste Anweisung weitergegeben wird, geschieht folgendes:

    a) Der numerische Wert jedes numerisch druckaufbereiteten Datenfeldes wird intern von den eingegebenen Zeichen 0 - 9, +, -, . oder , gebildet und dann zum numerisch druckaufbereiteten Feld zurückgesetzt, wobei die ANSI PICTURE-Druckaufbereitung angewendet wird. Das Feld kann sich daher von dem unterscheiden, das gerade vor dem Betätigen der RETURN-Taste auf dem Bildschirm stand.

    b) Wenn im SPECIAL-NAMES Paragraph CURSOR IS datenname angegeben ist, wird der Inhalt dieser Position nachdem man <RETURN> gedrückt hat, in datenname geschrieben, außer, wenn der Wert bei der ACCEPT-Angabe einen undefinierten Wert hatte.

Regeln für Format 3

1. Durch die ACCEPT-Anweisung wird die angeforderte Information an bezeichner entsprechend den Regeln der MOVE-Anweisung übertragen. DATE, DAY und TIME sind vordefinierte Datenfelder und werden daher nicht im COBOL-Programm beschrieben.

2. TIME steht in der Form Stunde, Minute, Sekunde, Hundertstelsekunde (HHMMSSCC) zur Verfügung. TIME ist ein ganzzahliges, numerisches Datenelement ohne Vorzeichen in einer Länge von acht Ziffern.

Die ADD-Anweisung

Funktion

Durch die ADD-Anweisung werden zwei oder mehrere numerische Operanden zusammengezählt, und das Ergebnis wird gespeichert.

Allgemeines Format

Format 1

```
ADD {bezeichner-1} [,bezeichner-2] ... TO bezeichner-m [ ROUNDED ]
    {literal-1   } [,literal-2   ]

[ ,bezeichner-n [ROUNDED] ]...[;ON SIZE ERROR unbedingte-anweisung]
```

Format 2

```
ADD {bezeichner-1} [{bezeichner-2}] [,bezeichner-3] ...
    {literal-1   } [{literal-2   }] [,literal-3   ]

    GIVING bezeichner-m [ ROUNDED ] [,bezeichner-n [ROUNDED] ] ...

    [ ; ON SIZE ERROR unbedingte-anweisung]
```

Format 3

```
ADD {CORRESPONDING} bezeichner-1 TO bezeichner-2 [ ROUNDED ]
    {CORR          }

    [ ;ON SIZE ERROR unbedingte-anweisung]
```

Syntaxregeln

1. Im Format 1 muß bezeichner ein numerisches Datenelement sein. Im Format 2 müssen bezeichner-m und bezeichner-n numerische Datenelemente oder numerisch druckaufbereitete Datenelemente sein.

   Im Format 3 muß bezeichner eine Datengruppe sein.

2. Jedes Literal muß numerisch sein.

3. Die gemeinsame Stellenzahl der Operanden darf nicht mehr als 18 Ziffern betragen.

Allgemeine Regeln

1. Vgl. Die ROUNDED-Angabe, die SIZE ERROR-Angabe, die CORRESPONDING-Angabe, Arithmetische Anweisungen, Überlappende Operanden und Mehrfachergebnisse in arithmetischen Anweisungen in diesem Kapitel.

2. Im Format 1 werden die Werte der Operanden vor dem Wort TO addiert und zum aktuellen Wert von bezeichner-m [bezeichner n] addiert. Das Ergebnis steht dann in bezeichner-m [bezeichner-n].

3. Im Format 2 werden die Werte der Operanden vor dem Wort GIVING zusammengezählt und die Summe wird dann als der neue Wert von bezeichner-m, bezeichner-n gespeichert.

4. Im Format 3 werden die Datenfelder in bezeichner-1 zu den entsprechenden Datenfeldern in bezeichner-2 addiert und darin abgespeichert.

5. Der Übersetzer stellt sicher, daß genügend Stellen zur Verfügung stehen, so daß keine signifikanten Ziffern während der Ausführung verlorengehen.

Die ALTER-Anweisung

Funktion

Durch die ALTER-Anweisung wird eine vorgegebene Folge von Operationen geändert.

Allgemeines Format

```
ALTER prozedurname-1 TO [ PROCEED TO ] prozedurname-2

 [,prozedurname-3 TO [ PROCEED TO] prozedurname-4 ] ...
```

Syntaxregeln

1. prozedurname-1, prozedurname-3, ... ist der Name eines Paragraphen, der einen einzelnen Satz, bestehend aus einer GO TO-Anweisung ohne DEPENDING-Angabe, enthält.
2. prozedurname-2, prozedurname-4, ... ist der Name eines Paragraphen oder einer SECTION in der PROCEDURE DIVISION.

Allgemeine Regeln

1. Bei der Ausführung des Programms ändert die ALTER-Anweisung die mit prozedurname-1, prozedurname-3 in der GO TO Anweisung angegebenen Sprungadressen in prozedurname-2, prozedurname-4.
   Siehe dazu die GO TO-Anweisung.
2. Eine GO TO-Anweisung in einer SECTION, deren Segment-Nummer größer oder gleich 50 ist, darf nicht durch eine ALTER Anweisung in einer SECTION mit einer anderen Segment-Nummer angesprochen werden.

   Alle anderen Verwendungen der ALTER-Anweisung sind zulässig und werden durchgeführt, selbst wenn prozedurname-1 oder prozedurname-3 in einem überlagerbaren festen Segment liegen.

Die COMPUTE-Anweisung

Funktion

Durch die COMPUTE-Anweisung werden einem oder mehreren Datenfeldern der Wert eines arithmetischen Ausdrucks zugeordnet.

Allgemeines Format

```
COMPUTE bezeichner-1 [ ROUNDED ] [,bezeichner-2 [ ROUNDED ] ] ...

    = arithmetischer ausdruck    [ ;ON SIZE ERROR unbedingte-anweisung]
```

Syntaxregeln

bezeichner-1 und bezeichner-2 müssen sich entweder auf numerische Datenelemente oder numerisch druckaufbereitete Datenelemente beziehen.

Allgemeine Regeln

1. Vgl. Die ROUNDED-Angabe, die SIZE ERROR Angabe , Arithmetische Anweisungen, überlappende Operanden und Mehrfachergebnisse in arithmetischen Anweisungen.
2. Ist bezeichner-1, bezeichner-2 ein arithmetischer Ausdruck, so wird der Wert dieses arithmetischen Ausdrucks errechnet und dann dieser Wert als neuer Wert von bezeichner-1, bezeichner-2 abgespeichert.
3. Die COMPUTE-Anweisung ermöglicht es, arithmetische Operationen ohne die Beschränkungen für die Zusammenstellung von Operanden und/oder Empfangsdatenfeldern in den arithmetischen Anweisungen ADD, SUBTRACT, MULTIPLY und DIVIDE zusammenzusetzen.

Die DISPLAY-Anweisung

Funktion

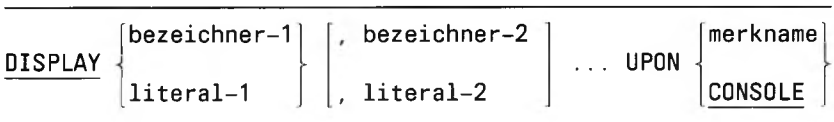
Durch die DISPLAY-Anweisung werden Daten aus angegebenen Datenfeldern auf den Bildschirm übertragen.

Allgemeine Formate

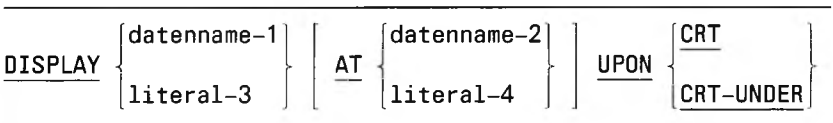
Format 1 zeigt die Standard ANSI DISPLAY-Anweisung.

Format 2 zeigt das erweiterte DISPLAY-Format.

Format 1



Format 2



Syntaxregeln

1. merkname im Format 1 muß mit der Datensichtstation im SPECIAL-NAMES-Paragraphen in der ENVIRONMENT DIVISION verbunden sein.
2. Jedes Literal kann außer ALL irgendeine figurative Konstante sein.
3. Wenn das Literal numerisch ist, muß es eine Ganzzahl ohne Vorzeichen sein.
4. literal-3 muß alphanumerisch, literal-4 muß numerisch sein.
5. datenname-1 kann auf einen Datensatz, eine Gruppe oder ein Datenelement Bezug nehmen; er darf jedoch nicht subskribiert sein.

Allgemeine Regeln

1. Die beiden Formate werden durch ihre UPON-Angaben unterschieden; standardmäßig wird UPON CONSOLE angenommen. Ein vom Benutzer definierter Merkname kann verwendet werden, sofern dieser mit einem Systemgerät im SPECIAL-NAMES-Paragraphen verknüpft ist (vgl. Der SPECIAL NAMES-Paragraph in diesem Kapitel). Der Standard kann jedoch durch die Angabe CONSOLE IS CRT im SPECIAL-NAMES-Paragraphen geändert werden, so daß UPON CRT Standard wird. Dieser geänderte Standard wird in der obigen Syntax nicht aufgeführt.

für Format 1

1. Durch die DISPLAY-Anweisung wird der Inhalt jedes Operanden in der angegebenen Reihenfolge an das Bildschirmgerät übertragen.
2. Die Daten werden als eine ganzzahlige Anzahl von Datensätzen (bis zu 12) übertragen. Die Größe eines Datensatzes wird durch die Übersetzer-Anweisung CRTWIDTH definiert (vgl. LII COBOL Bedienungsanleitung). Der Datensatz wird am Bildschirm an der aktuellen Cursorposition angezeigt, die letzten Leerzeichen werden abgeschnitten und der Cursor an den Anfang der nächsten Zeile (evtl. durch Rücksetzung der Bildschirmseite) gesetzt.

3. Wenn eine figurative Konstante am Bildschirm ausgegeben wird, wird sie nur einmal angezeigt.
4. Wenn ein Datenfeld vollständig auf dem Bildschirm als Datensatz übertragen werden kann, wird das Datenfeld 1:1 übertragen.
5. Wenn der Datensatz des Bildschirms nicht dem zu übertragenden Datenfeld entspricht:

   a) Das zu übertragende Datenfeld ist größer: Datenübertragung in den Datensatz des Bildschirms beginnt von links, rechte überschüssige Zeichen werden abgeschnitten.

   b) Der Datensatz des Bildschirms ist größer als das zu übertragende Datenfeld: Die Übertragungen in den Datensatz des Bildschirms beginnen von links, rechts wird mit Leerzeichen aufgefüllt.
6. Wenn Operanden in einer DISPLAY-Anweisung USAGE COMP oder USAGE COMP-3 sind, werden diese Operanden in USAGE DISPLAY umgewandelt. Die Länge des Sendedatenfelds entspricht der Summe der Längen aller Operanden (nach evtl. Umwandlung) und die Werte der Operanden werden in der Folge übertragen, in der sie auftreten.

für Format 2

1. Die DISPLAY-Anweisung wird dazu verwendet, Daten am Bildschirm in den angegebenen Bildschirmpositionen auszugeben.
2. datenname-1 definiert Bildschirmbereiche.
   In datenname-1 werden die Daten gespeichert, die auf dem Bildschirm ausgegeben werden. FILLER-Felder entsprechen Bereichen auf dem Bildschirm, in denen keine Daten gespeichert werden.
3. datenname-1 kann sein

   - ein Datenelement
   - eine Datengruppe, die Datengruppen und/oder Datenelemente enthält.

   Datenelemente müssen als USAGE DISPLAY definiert sein.

4. AT datenname-2 oder literal-4 definieren auf dem Bildschirm die Position des Zeichens der Daten, das am weitesten links steht. Eines der beiden muß ein PIC 9999 Feld sein. Die rechten beiden Stellen geben die Zeilenzahl wieder (im Bereich von eins bis 25), die beiden linken Stellen definieren die Zeichenposition (im Bereich von eins bis 80).

5. datenname-1 kann ein Datensatz, eine Datengruppe oder ein Datenelement sein; er kann jedoch nicht subskribiert werden.
   REDEFINES kann verwendet werden, wobei die erste Beschreibung der Daten verwendet wird und nachfolgende Beschreibungen ignoriert werden.
   Es können auch mehrdimensionale Tabellendefinitionen verwendet werden. datenname-1 entspricht dann in der Länge der gesamten Tabelle.

6. DISPLAY SPACE bewirkt ein Löschen des Bildschirminhalts zur Ausführungszeit (d.h. Auffüllen des gesamten Bildschirmes mit Leerzeichen). Auf DISPLAY " " (ein Leerzeichen) wird jedoch nur ein einziges Leerzeichen angezeigt.

7. Durch die CRT-UNDER-Angabe werden die Datenelemente invers am Bildschirm angezeigt.

Die DIVIDE-Anweisung

Funktion

Die DIVIDE-Anweisung dividiert ein numerisches Datenfeld durch andere und setzt die Werte der Datenfelder gleich dem Quotienten.

Allgemeine Formate

Format 1

```
DIVIDE {bezeichner-1} INTO bezeichner-2 [ROUNDED]
       {literal-1   }

[ , bezeichner-3 [ROUNDED] ]...
    [ ; ON SIZE ERROR unbedingte-anweisung ]
```

Format 2

```
DIVIDE {bezeichner-1} INTO {bezeichner-2}
       {literal-1   }     {literal-2   }

GIVING bezeichner-3[ROUNDED][,bezeichner-4[ROUNDED]]...
                [ ; ON SIZE ERROR unbedingte-anweisung]
```

Format 3

```
DIVIDE {bezeichner-1} BY {bezeichner-2}
       {literal-1   }    {literal-2   }

GIVING bezeichner-3[ROUNDED][,bezeichner-4[ROUNDED]]...
                [ ; ON SIZE ERROR unbedingte-anweisung]
```

Format 4

```
DIVIDE {bezeichner-1} INTO {bezeichner-2}
       {literal-1   }      {literal-2   }

GIVING bezeichner-3 [ROUNDED]

REMAINDER bezeichner-4
                [ ; ON SIZE ERROR unbedingte-anweisung]
```

Format 5

```
DIVIDE {bezeichner-1} BY {bezeichner-2}
       {literal-1   }    {literal-2   }

GIVING bezeichner-3 [ROUNDED]

REMAINDER bezeichner-4
                [ ; ON SIZE ERROR unbedingte-anweisung]
```

Syntaxregeln

1. Jeder Bezeichner muß sich auf ein numerisches Datenelement beziehen. Ist ein Bezeichner mit der GIVING- oder REMAINDER-Angabe verknüpft, kann er auch ein numerisch druckaufbereitetes Datenelement sein.
2. Jedes Literal muß numerisch sein.
3. Das Ergebnis aller Operanden-Aktionen darf, in einem Datenfeld gerechnet, nach der Dezimalpunktausrichtung 18 Stellen nicht übersteigen.

Allgemeine Regeln

1. Zur Beschreibung dieser Funktionen siehe Die ROUNDED-Angabe, Die SIZE ERROR-Angabe, Arithmetische Anweisungen, Überlappende Operanden und Mehrfachresultate in arithmetischen Anweisungen in diesem Kapitel.

2. Im Format 1 wird der Wert bezeichner-2 durch den Wert von bezeichner-1 oder literal-1 dividiert. Der Wert des Dividenden (bezeichner-2) wird durch diesen Quotienten ersetzt; auf gleiche Weise wird für bezeichner-1 oder literal-1 und bezeichner-3 usw. verfahren.

3. Im Format 2 wird der Wert von bezeichner-2 oder literal-2 durch den von bezeichner-1 oder Literal-1 dividiert, und das Ergebnis wird in bezeichner-3, bezeichner-4 usw. gespeichert.

4. Im Format 3 wird der Wert von bezeichner-1 oder literal-1 durch den Wert von bezeichner-2 oder literal-2 dividiert, und das Ergebnis wird in bezeichner-3, bezeichner-4 usw. gespeichert.

5. Die Formate 4 und 5 werden verwendet, wenn ein Rest aus einem Divisionsvorgang gewünscht wird, abgespeichert in bezeichner-4. Der Rest wird unter COBOL definiert als das Ergebnis der Subtraktion des Produktes vom Quotienten (bezeichner-3) und des Divisors vom Dividenden. Wenn bezeichner-3 als numerisch druckaufbereitetes Datenfeld definiert ist, so wird der zur Berechnung des Rests verwendete Quotient als Hilfsspeicherplatz angesehen, der den nicht aufbereiteten Quotienten enthält.
   Wenn die ROUNDED-Angabe verwendet wird, ist der zur Berechnung des Restes verwendete Quotient ein Hilfsspeicherplatz, der den Quotienten der DIVIDE-Anweisung (abgeschnitten, aber nicht gerundet) enthält.

6. In den Formaten 4 und 5 wird die Genauigkeit des REMAINDER-Datenfeldes (bezeichner-4) durch die oben beschriebene Berechnung definiert. Für den Inhalt des durch bezeichner-4 angesprochenen Datenfeldes wird eine geeignete Dezimalausrichtung mit Abschneiden (nicht Rundung) durchgeführt.

7. In den Formaten 4 und 5 gelten bei der ON SIZE ERROR-Angabe die folgenden Regeln:

   a) Wenn Überlauf beim Quotienten auftritt, ist eine Restberechnung nicht sinnvoll. Daher bleibt der Inhalt des durch bezeichner-3 und bezeichner-4 benannten Datenfeldes unverändert.

   b) Wenn der Überlauf beim Rest auftritt, bleibt der Inhalt des durch bezeichner-4 angesprochenen Datenfeldes unverändert. Wie bei anderen Mehrfachergebnissen in arithmetischen Anweisungen, wird der Anwender jedoch eine Analyse durchführen, um feststellen zu können, welche Situation tatsächlich aufgetreten ist.

Die ENTER-Anweisung

Funktion

Durch die ENTER-Anweisung kann ein Unterprogramm in einer anderen Programmiersprache als COBOL aufgerufen werden.

Allgemeines Format

```
ENTER sprachenname  [routinenname] .
```

Syntaxregeln

sprachenname und routinenname können irgendein vom Programmierer definiertes Wort oder ein alphanumerisches Literal sein.

Allgemeine Regel

Diese Anweisung wird nur zur Dokumentation verwendet.

Der Zugriff auf andere Sprachen kann mit Hilfe von CALL erzielt werden.

Die EXIT-Anweisung

Funktion

Durch die EXIT-Anweisung wird ein gemeinsamer Ausgangspunkt am Ende von Prozeduren gesetzt.

Allgemeines Format

```
EXIT.
```

Syntaxregeln

1. Die EXIT-Anweisung muß in einem einzelnen Satz erscheinen.
2. Der EXIT Satz muß der einzige Satz im Paragraphen sein.

Allgemeine Regel

Eine EXIT-Anweisung dient nur dazu, es dem Anwender zu ermöglichen, einem Prozedurnamen einen vorgegebenen Punkt in einem Programm zuzuordnen. Eine solche EXIT-Anweisung hat keine weitere Auswirkung auf die Übersetzung oder Ausführung des Programms.

Die GO TO-Anweisung

Funktion

Durch die GO TO-Anweisung wird der Programmablauf an dem nach GO TO angegebenen Sprungziel fortgesetzt.

Allgemeine Formate

Format 1

```
GO TO [prozedurname-1]
```

Format 2

```
GO TO prozedurname-1 [ , prozedurname-2] ...
                [,prozedurname-n] DEPENDING ON bezeichner
```

Syntaxregeln

1. bezeichner ist der Name eines numerischen Datenelements, das ohne Positionen rechts des angenommenen Dezimalpunktes beschrieben wird.
2. Wurde ein Paragraph über eine ALTER-Anweisung angesprochen, so darf dieser Paragraph nur aus einer Paragraphenüberschrift, gefolgt von einer GO TO-Anweisung im Format 1 bestehen.
3. Eine GO TO-Anweisung im Format 1 (ohne prozedurname-1) kann nur in einem Paragraphen mit einer einzigen Anweisung erscheinen.
4. Wenn eine im Format 1 dargestellte GO TO-Anweisung in einer Folge unbedingter Anweisungen innerhalb eines Satzes erscheint, so muß sie die letzte Anweisung in dieser Folge sein.

Allgemeine Regeln

1. Wenn eine im Format 1 dargestellte GO TO-Anweisung ausgeführt wird, ist das Sprungziel im Ablauf prozedurname-1.
2. Wenn prozedurname-1 im Format 1 nicht angegeben ist, muß eine ALTER-Anweisung, die sich auf diese GO TO-Anweisung bezieht, vor der Ausführung dieser GO TO-Anweisung ausgeführt werden.
3. Wenn eine GO TO-Anweisung im Format 2 ausgeführt wird, wird die Steuerung an prozedurname-1, prozedurname-2 usw. übertragen, in Abhängigkeit vom bezeichner, der die Werte 1, 2, ..., n annehmen kann. Wenn der Wert vom bezeichner ungleich der positiven bzw. vorzeichenlosen Ganzzahlen 1, 2, ..., n ist, erfolgt keine Übertragung und der Ablauf wird mit der nächsten Anweisung fortgesetzt.

Die IF-Anweisung

Funktion

Durch eine IF-Anweisung wird eine Bedingung ausgewertet (vgl. Bedingte Ausdrücke in diesem Kapitel). Die nachfolgende Operation des Zielprogramms hängt davon ab, ob der Wert der Bedingung wahr oder falsch ist.

Allgemeines Format

```
IF Bedingung; THEN { anweisung-1       } [ ;ELSE anweisung-2       ]
                   { NEXT SENTENCE     } [ ;ELSE NEXT SENTENCE     ]
```

Syntaxregeln

1. anweisung-1 oder anweisung-2 sind entweder unbedingte oder bedingte Anweisungen, wobei nach beiden eine bedingte Anweisung stehen kann.
2. Die ELSE NEXT SENTENCE-Angabe kann ausgelassen werden, sofern sie direkt vor dem Schlußpunkt des Satzes steht.

Allgemeine Regeln

1. Wenn eine IF-Anweisung ausgeführt wird, ist der Ablauf folgendermaßen:

   a) Wenn die Bedingung wahr ist, wird anweisung-1 ausgeführt, sofern angegeben. Wenn anweisung-1 eine Prozedurverzweigung oder eine bedingte Anweisung enthält, wird die Anweisung ausgeführt. Wenn anweisung-1 keine Prozedurverzweigung oder bedingte Anweisung enthält, wird auch bei ELSE-Angabe an den nächsten ausführbaren Satz gesprungen. Das ist der Satz nach dem Punkt.

   b) Wenn die Bedingung wahr ist und die NEXT SENTENCE-Angabe anstelle der anweisung-1 steht, wird auch bei ELSE-Angabe an den ersten Satz nach dem Punkt gesprungen.

c) Wenn die Bedingung falsch ist, wird anweisung-1 oder die Ersatzangabe NEXT SENTENCE ignoriert und anweisung-2, sofern angegeben, wird ausgeführt. Wenn anweisung-2 eine Prozedurverzweigung oder eine bedingte Anweisung enthält, wird diese Anweisung ausgeführt. Wenn anweisung-2 keine Prozedurverzweigung oder bedingte Anweisung enthält, geht die Steuerung an den nächsten ausführbaren Satz über. Wenn die ELSE-Angabe in anweisung-2 nicht angegeben ist, wird anweisung-1 ignoriert, und der nächste ausführbare Satz wird angesprungen.

d) Wenn die Bedingung falsch ist und die ELSE NEXT SENTENCE-Angabe vorhanden ist, wird anweisung-1, sofern angegeben, ignoriert, und an den nächsten ausführbaren Satz gesprungen.

2. anweisung-1 und/oder anweisung-2 können eine IF-Anweisung enthalten. Diesen Fall nennt man verschachtelte IF-Anweisung.

   IF-Anweisungen innerhalb von IF-Anweisungen können als IF- und ELSE-Kombinationspaare angesehen werden, die von links nach rechts ausgeführt werden. Daher bezieht sich jedes ELSE auf das unmittelbar vorhergehende IF, das noch nicht mit einem ELSE verbunden ist.

   Beispiele

   IF A = B anweisung-1.
   anweisung-2.
   Wenn A = B wahr ist, werden anweisung-1 und anweisung-2 ausgeführt.
   Wenn A = B falsch ist, wird nur anweisung-2 ausgeführt.

   IF A = B anweisung-1 ELSE anweisung-2.
   anweisung-3.
   Wenn A = B wahr ist, werden anweisung-1 und anweisung-3 ausgeführt.
   Wenn A = B falsch ist, werden anweisung-2 und anweisung-3 ausgeführt.

Die INSPECT-Anweisung

Funktion

Mit der INSPECT-Anweisung ist es möglich, einzelne Zeichen oder Zeichengruppen in einem Datenfeld zu zählen (Format 1), zu ersetzen (Format 2) oder zu zählen und zu ersetzen (Format 3).

Allgemeine Formate

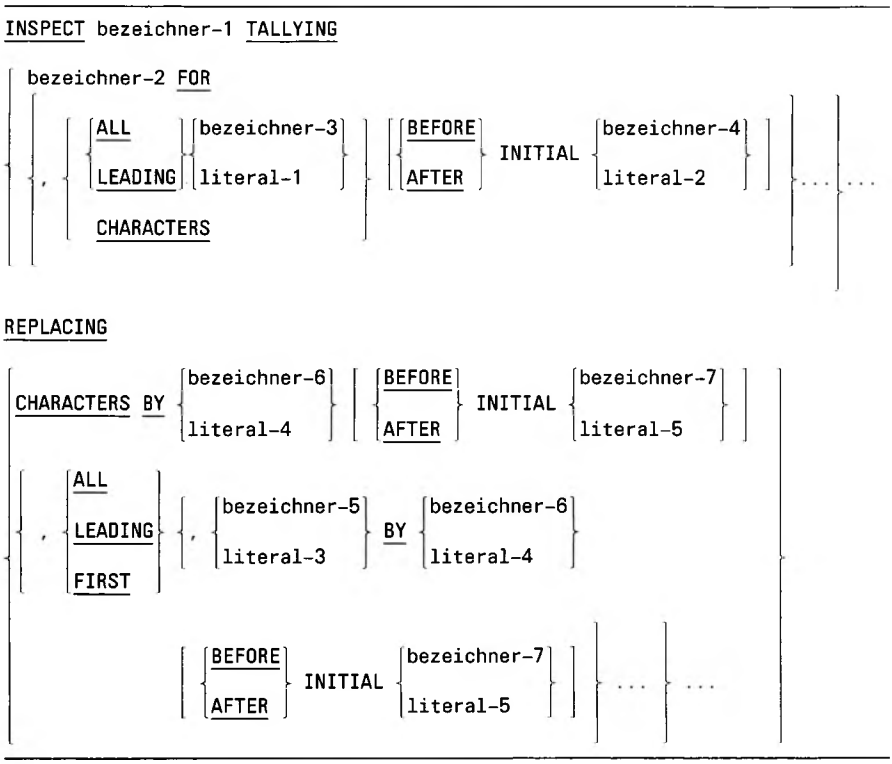
Format 1

```
INSPECT bezeichner-1 TALLYING

    { , bezeichner-2 FOR { , { [ALL       ] { bezeichner-3 } } }
                         {   { [LEADING   ] { literal-1    } } }
                         {   [CHARACTERS                     ] }

      [ { [BEFORE] } INITIAL { bezeichner-4 } ] } ... } ...
      [ { [AFTER ] }         { literal-2    } ] }     }
```

Format 2

```
INSPECT bezeichner-1 REPLACING

{ CHARACTERS BY { bezeichner-6 } [ { [BEFORE] } INITIAL { bezeichner-7 } ] }
                { literal-4    }   { [AFTER ] }         { literal-5    }   }

{ , [ALL      ] { , { bezeichner-5 } } BY { bezeichner-6 } }
    [LEADING  ]     { literal-3    }      { literal-4    }
    [FIRST    ]

      [ { [BEFORE] } INITIAL { bezeichner-7 } ] } ... } ...
      [ { [AFTER ] }         { literal-5    } ] }     }
```

Format 3


INSPECT bezeichner-1 TALLYING
bezeichner-2 FOR
ALL
LEADING
CHARACTERS
bezeichner-3
literal-1
BEFORE
AFTER
INITIAL
bezeichner-4
literal-2
REPLACING
CHARACTERS BY
bezeichner-6
literal-4
BEFORE
AFTER
INITIAL
bezeichner-7
literal-5
ALL
LEADING
FIRST
bezeichner-5
literal-3
BY
bezeichner-6
literal-4
BEFORE
AFTER
INITIAL
bezeichner-7
literal-5

Syntaxregeln

Für alle Formate

1. bezeichner-1 muß entweder eine Datengruppe oder Datenelemente irgendeiner Kategorie bezeichnen, die (entweder direkt oder indirekt) als USAGE IS DISPLAY beschrieben sind.
2. bezeichner-3 ... bezeichner-n muß entweder ein alphabetisches, alphanumerisches oder numerisches Datenelement bezeichnen, das (entweder direkt oder indirekt) als USAGE IS DISPLAY beschrieben ist.
3. Jedes Literal muß nicht numerisch und kann irgendeine figurative Konstante sein, ausgenommen von ALL.
4. literal-1, literal-2, literal-3, literal-4 und literal-5, sowie die Datenfelder, die durch bezeichner-3, bezeichner-4, bezeichner-5, bezeichner-6 und bezeichner-7 benannt werden, können beliebig viele Zeichen lang sein bis zu der für Literale bzw. Datenfelder zulässigen Grenze.

Für Formate 1 und 3:

1. bezeichner-2 muß ein numerisches Datenelement sein.
2. Wenn literal-1 oder literal-2 eine figurative Konstante ist, so bezieht sich diese auf ein implizites Datenfeld, das ein Zeichen lang ist.

Für Formate 2 und 3:

1. literal-4 oder bezeichner-6 müssen die gleiche Länge haben wie literal-3 oder bezeichner-5.
   Wenn literal-4 eine figurative Konstante ist, entspricht literal-4 in der Länge der Datenfelder literal-3 oder bezeichner-5.
2. Ist CHARACTERS angegeben, dürfen literal-4, literal-5, bezeichner-6 oder bezeichner-7 nur ein Zeichen lang sein.
3. Ist literal-3 eine figurative Konstante, dürfen literal-4 oder bezeichner-6 nur ein Zeichen lang sein.

Allgemeine Regeln

Für alle Formate

1. Bei Ausführung einer INSPECT-Anweisung wird bezeichner-1 unabhängig von seiner Datenklasse zeichenweise von links nach rechts abgearbeitet.

2. Bei der INSPECT-Anweisung wird der Inhalt der Datenfelder bezeichner-1, bezeichner-3, bezeichner-4, bezeichner-5 bezeichner-6 oder bezeichner-7 wie folgt behandelt:

   a) Wenn einer der Bezeichner alphanumerisch beschrieben ist, so behandelt die INSPECT-Anweisung den Inhalt jeder dieser Beschreibungen als eine Zeichenfolge.

   b) Wenn einer der Bezeichner alphanumerisch druckaufbereitet, numerisch druckaufbereitet oder vorzeichenlos numerisch beschrieben ist, so wird das Datenfeld so geprüft, als ob es als alphanumerisch beschrieben wäre.

   c) Wenn einer der Bezeichner numerisch mit Vorzeichen beschrieben ist, wird das Datenfeld so geprüft, als ob es ein vorzeichenloses numerisches Datenfeld der gleichen Länge wäre.

3. In den Allgemeinen Regeln gelten alle Bezugsnamen auf literal-1, literal-2, literal-3, literal-4 bzw. literal-5 auch für den Inhalt von bezeichner-3, bezeichner-4, bezeichner-5, bezeichner-6 bzw. bezeichner-7.

4. Bei der Prüfung des Inhaltes von bezeichner-1 wird jedes Auftreten von literal-1 gezählt (Formate 1 und 3) und/oder jedes übereinstimmende Auftreten von literal-3 wird durch literal-4 ersetzt (Formate 2 und 3).
   Die durch das Verb INSPECT benannten Datenfelder sollten so gesetzt werden, daß sie innerhalb der ersten 10000 Bytes des Zwischencodes liegen.

5. Das Auftreten von literal-1 oder Ersetzen von literal-3 wird wie folgt geprüft und durchgeführt:

   a) Die Operanden der TALLYING- und REPLACING-Angabe werden in der Reihenfolge abgearbeitet, wie sie in der INSPECT-Anweisung angegeben sind. Das erste literal-1, literal-3 wird mit der gleichen Anzahl von aufeinanderfolgenden Zeichen von bezeichner-1 verglichen, beginnend mit dem am weitesten links stehenden Zeichen (siehe BEFORE- und AFTER-Angabe). Übereinstimmung ist nur dann erreicht, wenn literal-1, literal-3 mit dem entsprechenden Teil von bezeichner-1 Zeichen gleich ist.

   b) Wird keine übereinstimmung beim Vergleich des ersten literal-1, literal-3 erzielt, wird der Vergleich mit jedem nachfolgenden literal-1, literal-3 solange wiederholt, bis entweder eine Übereinstimmung besteht oder kein literal-1, literal-3 mehr nachfolgt. Folgt kein literal-1, literal-3 mehr nach, wird die Zeichenposition innerhalb von bezeichner-1 um ein Zeichen nach rechts verschoben und der Vergleichszyklus beginnt wieder mit dem ersten literal-1, literal-3.

   c) Liegt eine Übereinstimmung vor, wird bezeichner-2 um 1 hochgezählt (TALLYING) beziehungsweise ersetzt (REPLACING). Der neue Vergleichszyklus beginnt mit dem Zeichen von bezeichner-1, das unmittelbar hinter dem letzten Zeichen der übereinstimmenden Zeichenfolge steht. Der Vergleichszyklus beginnt wieder mit dem ersten literal-1 bzw. bzw. literal-3.

   d) Der Vergleich wird solange fortgesetzt, bis das letzte Zeichen von bezeichner-1 abgearbeitet worden ist bzw. bis das Ende von bezeichner-1 erreicht ist (siehe BEFORE- und AFTER-Angabe).

   e) Wird CHARACTERS angegeben, findet kein Vergleich mit bezeichner-1 statt, sondern alle Zeichen von bezeichner-1 werden gezählt.

6. Das bisher beschriebene Verfahren, um die Übereinstimmung von literal-1 bzw. literal-3 mit bezeichner-1 zu ermitteln, wird durch die BEFORE- und AFTER-Angabe wie folgt beeinflußt:

a) Ist weder BEFORE noch AFTER angegeben, so läuft die Vergleichsoperation wie oben beschrieben ab.

b) Ist BEFORE angegeben, wird der Vergleich von bezeichner-1 mit literal-1 bzw. literal-3 mit dem ersten Zeichen von bezeichner-1 begonnen und nur solange fortgesetzt, bis die Zeichenfolge von literal-2 bzw. literal-5 in bezeichner-1 auftritt. Das erste Zeichen von literal-2 bzw. literal-5 stellt für die Vergleichsoperation implizit das Ende von bezeichner-1 dar.
Wurde die Zeichenfolge von literal-2 bzw. literal-5 innerhalb von bezeichner-1 nicht gefunden, wird der gesamte Inhalt von bezeichner-1 verglichen.

c) Ist AFTER angegeben, beginnt der Vergleich von bezeichner-1 mit literal-1 bzw. literal-3 mit dem Zeichen, das unmittelbar auf die erste gefundene Zeichenfolge von literal-2 bzw. literal-5 folgt und endet mit dem letzten Zeichen von bezeichner-1.
Wurde die Zeichenfolge von literal-2 bzw. literal-5 innerhalb von bezeichner-1 nicht gefunden, wird kein Zeichen von bezeichner-1 verglichen.

Regeln für Format 1

1. Der Inhalt von bezeichner-2 wird bei der Ausführung der INSPECT-Anweisung nicht initialisiert.

2. Die Regeln für TALLYING sind wie folgt:

   a) Wenn die ALL-Angabe auftritt, wird der Inhalt von bezeichner-2 bei jedem Auftreten von literal-1 innerhalb von bezeichner-1 um eins erhöht (siehe BEFORE- und AFTER-Angabe).

   b) Ist LEADING angegeben, wird bezeichner-2 bei jedem direkt aufeinanderfolgenden Auftreten von literal-1 in bezeichner-1 um 1 erhöht. Wichtig ist, daß das erste Auftreten von literal-1 unmittelbar an der Stelle ist, an der die Vergleichsoperation mit literal-1 beginnt (siehe BEFORE- und AFTER-Angabe).

   c) Ist CHARACTERS angegeben, wird der Inhalt von bezeichner-2 für jedes verglichene Zeichen innerhalb von bezeichner-1 um eins erhöht (siehe BEFORE- und AFTER-Angabe).

Regeln für Format 2

1. Die erforderlichen Worte ALL, LEADING und FIRST sind Adjektive, die für jede nachfolgende BY-Angabe gelten, bis das nächste Adjektiv auftritt.

2. Es gelten die folgenden Ersetzungsregeln:

   a) Ist CHARACTERS angegeben, wird jedes geprüfte Zeichen innerhalb von bezeichner-1 durch literal-4 ersetzt. (siehe BEFORE- und AFTER-Angabe).

   b) Ist ALL angegeben, wird jedes literal-3, das innerhalb von bezeichner-1 auftritt, durch literal-4 ersetzt (siehe BEFORE- und AFTER-Angabe).

   c) Ist LEADING angegeben, wird jede direkt aufeinanderfolgende Übereinstimmung von literal-3 innerhalb von bezeichner-1 durch literal-4 ersetzt. Wichtig ist, daß das erste Auftreten von literal-3 unmittelbar an der Stelle ist, an der die Vergleichsoperation mit literal-3 beginnt (siehe BEFORE- und AFTER-Angabe).

   d) Ist FIRST angegeben, wird nur das erste literal-3 das im bezeichner-1 auftritt, durch literal-4 ersetzt.

Regeln für Format 3

1. Die INSPECT-Anweisung im Format 3 wird interpretiert und ausgeführt, als ob zwei nacheinanderfolgende INSPECT-Anweisungen mit dem gleichen bezeichner-1 geschrieben worden wären. Die erste im Format 1 mit TALLYING-Angabe und den Anweisungen im Format 3, die zweite im Format 2 mit REPLACING-Angabe und den Angaben, wie sie im Format 3 beschrieben sind.

Beispiele

Nachfolgend sechs Beispiele für die Verwendung der INSPECT-Anweisung.

INSPECT Wort TALLYING Zähler FOR LEADING "L" BEFORE INITIAL "A", Zähler-1 FOR LEADING "A" BEFORE INITIAL "L".

Wort	Zähler	Zähler-1
LARGE	1	0
ANALYST	0	1

INSPECT Wort TALLYING Zähler FOR ALL "L", REPLACING LEADING "A" BY "E" AFTER INITIAL "L".

Wort	Zähler	Wort
CALLAR	2	CALLAR
SALAMI	1	SALEMI
LATTER	1	LETTER

INSPECT Wort REPLACING ALL "A" BY "G" BEFORE INITIAL "X".

Wort	Wort
ARXAX	GRXAX
HANDAX	HGNDGX

INSPECT Wort TALLYING Zähler FOR CHARACTERS AFTER INITIAL "J"
REPLACING ALL "A" BY "B"

Wort	Zähler	Wort
ADJECTIVE	6	BDJECTIVE
JACK	3	JBCK
JUJMAB	5	JUJMBB

INSPECT Wort REPLACING ALL "X" BY "Y ", "B" BY "Z", "W" BY "Q"
AFTER INITIAL "R".

Wort	Wort
RXXBQWY	RYYZQQY
RAWRXEB	RAQRYEZ

INSPECT Wort REPLACING CHARACTERS BY "B" BEFORE INITIAL "A".

Das Wort vor der Ersetzung = 1 2 ␣ X 2 A B C D
Das Wort nach der Ersetzung = B B B B B A B C D

Die MOVE-Anweisung

Funktion

Durch die MOVE-Anweisung werden Daten entsprechend den Druckaufbereitungsregeln in einen oder mehrere Datenbereiche übertragen.

Allgemeine Formate

Format 1

```
MOVE {bezeichner-1} TO bezeichner-2 [,bezeichner-3]...
     {literal     }
```

Format 2

```
MOVE {CORRESPONDING} bezeichner-1 TO bezeichner-2
     {CORR         }
```

Syntaxregeln

1. bezeichner-1 und literal stellen den Sendebereich dar; bezeichner-2, bezeichner-3, ... stellen den Empfangsbereich dar.
2. CORR ist eine Abkürzung für CORRESPONDING.
3. Wenn die CORRESPONDING-Angabe verwendet wird, müssen beide Bezeichner Datengruppen sein.
4. Ein Index-Datenfeld kann nicht als Operand einer MOVE-Anweisung erscheinen. (vgl. Die USAGE-Klausel in diesem Kapitel)

Allgemeine Regeln

1. Bei der CORRESPONDING-Angabe werden entsprechende Datenfelder von einer Gruppe zu einer anderen übertragen. Die Gruppen sind bezeichner-1 und bezeichner-2 im Format 2.

2. Die durch das Literal oder bezeichner-1 benannten Daten werden zuerst nach bezeichner-2, dann nach bezeichner-3, ... übertragen. Alle Regeln für bezeichner-2 gelten auch für die anderen Empfangsbereiche. Alle mit bezeichner-2 verknüpften Subskribierungen oder Indizierungen werden vor der Übertragung aufgelöst.

   Beispiel

   MOVE a (b) TO b, c (b)

   ist gleich:

   MOVE a (b) TO temp
   MOVE temp TO b
   MOVE temp TO c (b)

   wobei temp ein durch den Übersetzer erzeugtes Ergebnisdatenfeld darstellt.

3. Jede MOVE-Anweisung bei der die Sendedatenfelder und die Empfangsdatenfelder Datenelemente sind, ist eine elementare Übertragung. Jedes Datenelement gehört zu einer der folgenden Kategorien:

   - numerisch (auch numerische Literale und die figurative Konstante ZERO)
   - alphabetisch (auch die figurative Konstante SPACE)
   - alphanumerisch (auch nicht numerische Literale und alle anderen figurativen Konstanten)
   - numerisch druckaufbereitet
   - alphanumerisch druckaufbereitet

   Die folgenden Regeln gelten für eine elementare Übertragung zwischen diesen Kategorien:

   a) Die figurative Konstante SPACE oder ein alphanumerisch druckaufbereitetes oder ein alphabetisches Datenfeld darf nicht in ein numerisches oder numerisch druckaufbereitetes Datenfeld übertragen werden.

      Ein numerisch druckaufbereitetes Datenfeld darf nicht in ein numerisch druckaufbereitetes Datenfeld übertragen werden.

b) Ein numerisches Literal, die figurative Konstante ZERO, ein numerisches Datenfeld oder ein numerisch druckaufbereitetes Datenfeld darf nicht in ein alphabetisches Datenfeld übertragen werden.

c) Ein nicht ganzzahliges numerisches Literal oder ein nicht ganzzahliges numerisches Datenfeld darf nicht in ein alphanumerisches oder in ein alphanumerisch druckaufbereitetes Datenfeld übertragen werden.

d) Alle anderen elementaren Übertragungen sind zulässig und werden entsprechend den in der Allgemeinen Regel 4 vorgegebenen Bestimmungen durchgeführt.

4. Jede erforderliche Umwandlung von Daten von einem Format der internen Darstellung in ein anderes erfolgt durch zulässige elementare Übertragsvorgänge zusammen mit der für das Empfangsdatenfeld beschriebenen Druckaufbereitung nach den Standard-Ausrichtungsregeln:

a) Das Empfangsdatenfeld ist alphanumerisch oder alphanumerisch druckaufbereitet

- Wenn das Sendedatenfeld größer ist als das Empfangsdatenfeld, werden die überflüssigen rechten Zeichen des Sendedatenfelds abgeschnitten.

Das Empfangsdatenfeld ist numerisch mit Vorzeichen beschrieben

- Das Rechenvorzeichen wird nicht übertragen
- Wenn das Rechenvorzeichen eine eigene Zeichenposition belegt, wird das Zeichen nicht übertragen und die Länge des Sendedatenfeldes wird um diese eine Zeichenposition verkürzt
- Wenn das Sendedatenfeld kleiner ist als das Empfangsdatenfeld, wird das Feld linksbündig übertragen und rechts mit Leerzeichen aufgefüllt.

b) Das Empfangsdatenfeld ist numerisch oder numerisch druckaufbereitet

- Ausrichtung nach Standard-Ausrichtungsregeln evtl. bei Druckaufbereitung Ersetzen der Nullen

Wenn das Empfangsdatenfeld ein numerisches Datenfeld mit Vorzeichen ist, wird das Vorzeichen des Sendedatenfeldes in das Empfangsdatenfeld übertragen. Wenn das Sendedatenfeld kein Vorzeichen hat, wird für das Empfangsdatenfeld ein positives Vorzeichen generiert.

Wenn das Empfangsdatenfeld ein vorzeichenloses numerisches Datenfeld ist, wird der absolute Wert des Sendedatenfelds übertragen, und es wird kein Rechenvorzeichen für das Empfangsdatenfeld generiert.

Wenn das Sendedatenfeld ein alphanumerisches Datenfeld ist, werden die Daten so übertragen, als ob das Sendedatenfeld als eine vorzeichenlose numerische Ganzzahl beschrieben worden wäre.

c) Wenn das Empfangsfeld alphabetisch beschrieben ist, erfolgt die Ausrichtung und evtl. erforderliche Auffüllung mit Leerzeichen wie in den Standard-Ausrichtungsregeln in Kapitel 2 definiert. Wenn die Größe des Sendedatenfelds größer als die Größe des Empfangsdatenfelds ist, werden die nach Auffüllung des Empfangsdatenfelds nach rechts überhängenden Zeichen abgeschnitten.

5. Jede Übertragung, die keine elementare Übertragung ist, wird so behandelt, als ob es sich um eine alphanumerische oder alphanumerische elementare Übertragung handeln würde, mit der keine Umwandlung der Daten von einer Art der internen Darstellung in eine andere erfolgt. Bei einer solchen Übertragung wird der Empfangsbereich ohne Berücksichtigung der einzelnen Datenelemente oder Datengruppen aufgefüllt, die im Sende- oder Empfangsbereich enthalten sind, ausgenommen wie in Allgemeine Regel 4 der OCCURS-Klausel angegeben.

6. In Tabelle 2-7 wird die Zulässigkeit der verschiedenen Typen der MOVE-Anweisungen aufgezeigt. Die Referenzangaben weisen auf die Regeln hin, die die Übertragung verbieten oder das Verhalten einer zulässigen Übertragung angeben.

Kategorie des Sendefeldes		Kategorie des Empfangsdatenfedes *1)*		
		alphabetisch	alphanumerisch druck-aufbereitet alphanumerisch	numerisch ganzzahlig numerisch nicht ganzzahlig numerisch druckauf-bereitet
alphabetisch		Ja (4c)	Ja (4a)	Nein (3a)
alphanumerisch		Ja (4c)	Ja (4a)	Ja (4b)
alphanumerisch druck-aufbereitet		Ja (4c)	Ja (4a)	Nein (3a)
numerisch	ganzzahlig	Nein (3b)	Ja (4a)	Ja (4b)
	nicht ganzzahlig	Nein (3b)	Nein (3c)	Ja (4b)
numerisch druck-aufbereitet		Nein (3b)	Ja (4a)	Nein (3a)

1) Hinweis

Die Nummern beziehen sich auf die obigen allgemeinen Regeln.

Tabelle 2-7 Zulässige Datenkategorien für MOVE-Anweisungen

Die MULITPLY-Anweisung

Funktion

Durch die MULTIPY-Anweisung werden numerische Datenfelder multipliziert und in Ergebnisdatenfelder übertragen.

Allgemeine Formate

Format 1

```
MULTIPLY {bezeichner-1} BY  bezeichner-2 [ROUNDED]
         {literal-1   }

      [ ,bezeichner-3 [ROUNDED] ]... [;ON SIZE ERROR unbedingte-anweisung]
```

Format 2

```
MULTIPLY {bezeichner-1} BY {bezeichner-2} GIVING  bezeichner-3 [ROUNDED]
         {literal-1   }    {literal-2   }

         [, bezeichner-4 [ ROUNDED ]]...

         [, ON SIZE ERROR unbedingte-anweisung]
```

Syntaxregeln

1. Jeder Bezeichner muß ein numerisches Datenelement sein, mit der Ausnahme, daß im Format 2 jeder Bezeichner nach dem Wort GIVING auch ein numerisch druckaufbereitetes Datenfeld sein kann.
2. Jedes Literal muß numerisch sein.
3. Die Maximalgröße des Produkts beträgt nach der Dezimalpunktausrichtung 18 Stellen.

Allgemeine Regeln

1. Vgl. Die ROUNDED-Angabe, die SIZE ERROR-Angabe, Arithmetische Anweisungen, Überlappende Operanden und Mehrfachergebnisse in arithmetischen Anweisungen in diesem Kapitel.
2. Im Format 1 wird der Wert von bezeichner-1 oder literal-1 mit dem Wert von bezeichner-2 multipliziert. Das Ergebnis wird in bezeichner-2 abgespeichert; entsprechend wird für bezeichner-1 oder literal-1 und bezeichner-3 usw. verfahren.
3. Wenn Format 2 verwendet wird, wird der Wert von bezeichner-1 bzw. literal-1 mit bezeichner-2 bzw. literal-2 multipliziert, und das Ergebnis wird in bezeichner-3, bezeichner-4 usw. gespeichert.

Die PERFORM-Anweisung

Funktion

Die PERFORM-Anweisung dient dazu, während des Programmablaufs eine Prozedur anzuspringen. Am Ende der Prozedur wird mit der nächsten Anweisung nach dem PERFORM fortgesetzt.

Allgemeine Formate

Format 1

```
PERFORM prozedurname-1 [ [ { THROUGH } ] prozedurname-2 ]
                       [ [ { THRU    } ]               ]
```

Format 2

```
PERFORM prozedurname-1 [ [ { THROUGH } ] prozedurname-2 ] [ { bezeichner-1 } ] TIMES
                       [ [ { THRU    } ]               ] [ { ganzzahl-1  } ]
```

Format 3

```
PERFORM prozedurname-1 [ [ { THROUGH } ] prozedurname-2 ] UNTIL bedingung-1
                       [ [ { THRU    } ]               ]
```

Format 4

```
PERFORM prozedurname-1 [ { [ THROUGH ] } prozedurname-2 ]
                       [ { [ THRU    ] }               ]

    VARYING { bezeichner-2 } FROM { bezeichner-3 }
            { indexname-1  }      { indexname-2  }
                                  { literal-1    }

        BY { bezeichner-4 } UNTIL bedingung-1
           { literal-2    }

          [ AFTER { bezeichner-5 } FROM { bezeichner-6 } ]
                  { indexname-3  }      { indexname-4  }
                                        { literal-3    }

              BY { bezeichner-7 } UNTIL bedingung-2
                 { literal-4    }

          [ AFTER { bezeichner-8 } FROM { bezeichner-9 } ]
                  { indexname-5  }      { indexname-6  }
                                        { literal-5    }

              BY { bezeichner-10 } UNTIL bedingung-3 ] ]
                 { literal-6     }
```

Syntaxregeln

1. Jeder Bezeichner stellt ein in der DATA DIVISION beschriebenes numerisches Datenelement dar. Im Format 2 muß der bezeichner-1 als eine numerische Ganzzahl beschrieben sein.
2. Jedes Literal stellt ein numerisches Literal dar.
3. Die Worte THRU und THROUGH sind gleichbedeutend.
4. Wenn indexname in der VARYING- oder AFTER-Angabe angesprochen ist, dann gilt:
   a) Der Bezeichner in den zugehörigen FROM- und BY-Angaben muß ein ganzzahliges Datenfeld sein.
   b) Das Literal in der zugehörigen FROM-Angabe muß eine positive Ganzzahl sein.
   c) Das Literal in der zugehörigen BY-Angabe muß eine ganze Zahl sein, die ungleich Null ist.
5. Wenn indexname in der FROM-Angabe steht, dann gilt:
   a) Der Bezeichner in der zugehörigen VARYING- oder AFTER-Angabe muß ein ganzzahliges Datenfeld sein.
   b) Der Bezeichner in der zugehörigen BY-Angabe muß ein ganzzahliges Datenfeld sein.
   c) Das Literal in der zugehörigen BY-Angabe muß eine ganze Zahl sein.
6. Das Literal in der BY-Angabe darf nicht Null sein.
7. bedingung-1, bedingung-2 und bedingung-3 können irgendwelche bedingten Ausdrücke sein wie in "Bedingte Ausdrücke" in diesem Kapitel beschrieben.
8. Wenn prozedurname-1 und prozedurname-2 angegeben sind und einer der beiden ist der Name einer Prozedur in der Prozedurvereinbarung des Programms, dann müssen beide Prozedurnamen in der gleichen Prozedurvereinbarung sein.

Allgemeine Regeln

1. Die durch bezeichner-4, bezeichner-7 und bezeichner-10 benannten Datenfelder dürfen nicht Null sein.

2. Wenn indexname in der VARYING- oder AFTER-Angabe und bezeichner in der zugehörigen FROM-Angabe steht, dann muß das durch den Bezeichner benannte Datenfeld einen positiven Wert besitzen.

3. Wenn die PERFORM-Anweisung ausgeführt wird, wird die erste Anweisung der Prozedur prozedurname-1 ausgeführt. Nach dem Ende der aufgerufenen Prozedur wird an die nächste ausführbare Anweisung nach der PERFORM-Anweisung zurückgesprungen.

   a) Wenn prozedurname-1 ein Paragraphenname und prozedurname-2 nicht angegeben sind, dann erfolgt der Rücksprung nach der letzten Anweisung von prozedurname-1.

   b) Wenn prozedurname-1 ein SECTION-Name und prozedurname-2 nicht angegeben sind, dann erfolgt der Rücksprung nach der letzten Anweisung des letzten Paragraphen von prozedurname-1.

   c) Wenn prozedurname-2 angegeben ist und es sich dabei um einen Paragraphennamen handelt, dann erfolgt der Rücksprung nach der letzten Anweisung des Paragraphen.

   d) Wenn prozedurname-2 angegeben ist und es sich dabei um einen SECTION-Namen handelt, dann erfolgt der Rücksprung nach der letzten Anweisung des letzten Paragraphen in der SECTION.

4. prozedurname-1 und prozedurname-2 gibt man dann an, wenn die Anweisungsfolgen zweier oder mehrerer Prozeduren durchlaufen werden sollen. Am Ende von prozedurname-2 erfolgt der Rücksprung. prozedurname-2 kann ein Paragraphenname sein, der aus einem EXIT besteht und mit GO TO von mehreren Anweisungen angesprungen wird.
   Es ist jederzeit möglich, innerhalb von prozedurname-1 und prozedurname-2 GO TO oder PERFORM anzugeben.

5. Die PERFORM-Anweisung läuft folgendermaßen ab:

   a) Format 1 zeigt die grundlegende PERFORM-Anweisung. Eine Prozedur, die durch diesen Typ von PERFORM-Anweisung angesprungen wird, wird einmal ausgeführt, und dann geht die Steuerung an die nächste ausführbare Anweisung über, die nach der PERFORM-Anweisung steht.

   b) Format 2 zeigt die PERFORM ... TIMES-Anweisung. Die Prozeduren werden so oft durchgeführt wie durch ganzzahl-1 oder durch den Anfangswert des durch bezeichner-1 benannten Datenfeldes für diese Ausführung angegeben. Wenn zum Zeitpunkt der Ausführung einer PERFORM-Anweisung der Wert des durch bezeichner-1 benannten Datenfeldes Null oder negativ ist, geht die Steuerung an die nächste ausführbare Anweisung über, die nach der PERFORM-Anweisung steht. Nachdem die Prozeduren so oft wie angegeben ausgeführt wurden, wird die Steuerung an die nächste ausführbare Anweisung übergeben, die nach der PERFORM-Anweisung steht.

      Wird bezeichner-1 während des Durchlaufs verändert, ändert dies nicht die Anzahl der Durchläufe der Prozeduren. Entscheidend ist der Anfangswert beim ersten Durchlauf.

   c) Format 3 zeigt die PERFORM ... UNTIL-Anweisung. Die angegebenen Prozeduren werden ausgeführt, bis die durch die UNTIL-Angabe angegebene Bedingung wahr ist. Wenn die Bedingung wahr ist, wird die nächste Anweisung nach der PERFORM-Anweisung durchgeführt. Dieser Fall tritt auch ein, wenn schon beim ersten Durchlauf die Bedingung erfüllt ist, d.h. prozedurname-1 wird gar nicht angesprungen.

d) Format 4 zeigt die PERFORM ... VARYING-Anweisung. Diese PERFORM-Anweisung wird dazu verwendet, die durch einen oder mehrere Bezeichner oder Indexnamen benannten Werte zu erhöhen. Wenn Indexname in einer VARYING- und/oder AFTER-Angabe auftritt, wird er initialisiert und danach erhöht (wie unten beschrieben) entsprechend den Regeln der SET-Anweisung.

Wenn Indexname in der FROM-Angabe und der Bezeichner einer zugehörigen VARYING- oder AFTER-Angabe erscheint, so wird der Bezeichner entsprechend den Regeln der SET-Anweisung initialisiert; die nachfolgende Erhöhung erfolgt wie unten beschrieben.

Wenn ein Bezeichner im Format 4 geändert wird, wird bezeichner-2 auf den Wert von literal-1 oder bezeichner-3 gesetzt; dann wird die Folge von Prozeduren (prozedurname-1 bis prozedurname-2), einmal ausgeführt, wenn die Bedingung der UNTIL-Angabe falsch ist. Der Wert von bezeichner-2 wird durch den angegebenen Wert (der Wert von bezeichner-4 oder literal-2) erhöht oder verringert. bedingung-1 wird erneut ausgewertet. Der Zyklus wird durchgeführt, bis diese Bedingung wahr ist. Danach wird an der nächsten ausführbaren Anweisung nach PERFORM fortgesetzt.

Wenn bedingung-1 zu Beginn der Ausführung der PERFORM-Anweisung wahr ist, wird die nächste ausführbare Anweisung nach der PERFORM-Anweisung durchgeführt.

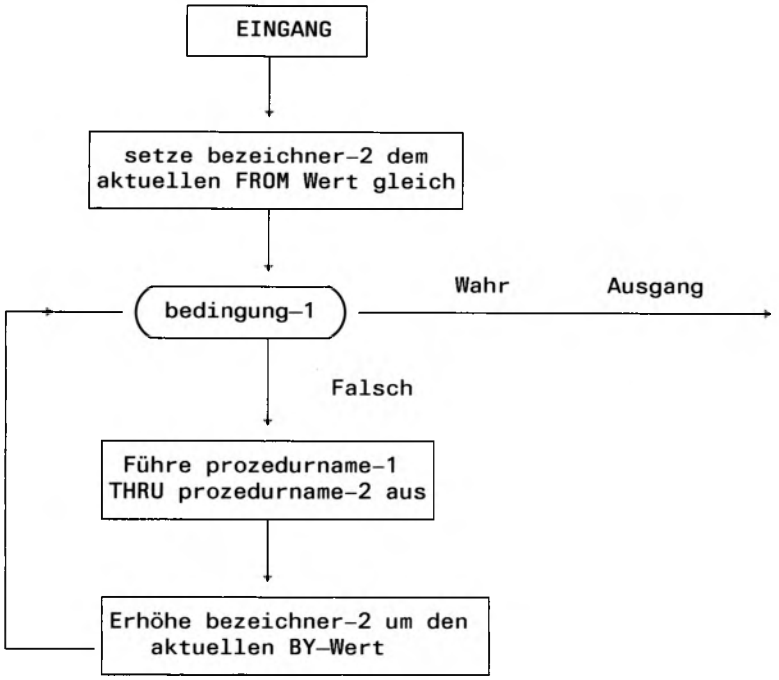


Bild 2-1 Flußdiagramm einer VARYING-Angabe einer PERFORM-Anweisung mit einer Bedingung.

Wenn zwei Bezeichner im Format 4 geändert werden, werden bezeichner-2 bzw. bezeichner-5 auf den aktuellen Wert von bezeichner-3 und bezeichner-6 gesetzt. Dann wird bedingung-1 geprüft. Ist die Bedingung wahr, wird die nächste Anweisung nach PERFORM ausgeführt. Ist die Bedingung falsch, wird bedingung-2 geprüft. Ist bedingung-2 falsch, wird prozedurname-1 bis prozedurname-2 einmal durchgeführt und bezeichner-5 wird um bezeichner-7 bzw. literal-4 erhöht. Danach wird bedingung-2 erneut geprüft. Dieser Zyklus von Bewertung und Erhöhung erfolgt, bis diese Bedingung wahr ist. Wenn bedingung-2 wahr ist, wird bezeichner-5 auf den Wert von literal-3 oder bezeichner-6 gesetzt und bezeichner-2 wird durch bezeichner-4 erhöht, und bedingung-1 wird neu ausgewertet.
Die PERFORM-Anweisung ist abgeschlossen, wenn bedingung-1 wahr ist; wenn bedingung-1 nicht wahr ist, werden die Zyklen so lange durchgeführt, bis sie wahr ist.

Folgende Variablen beeinflussen die Ausführung der PERFORM-Anweisung:

- beim VARYING bezeichner-2, indexname-1
- bei BY bezeichner-4
- bei AFTER bezeichner-5, indexname-3
- bei FROM bezeichner-3, indexname-3

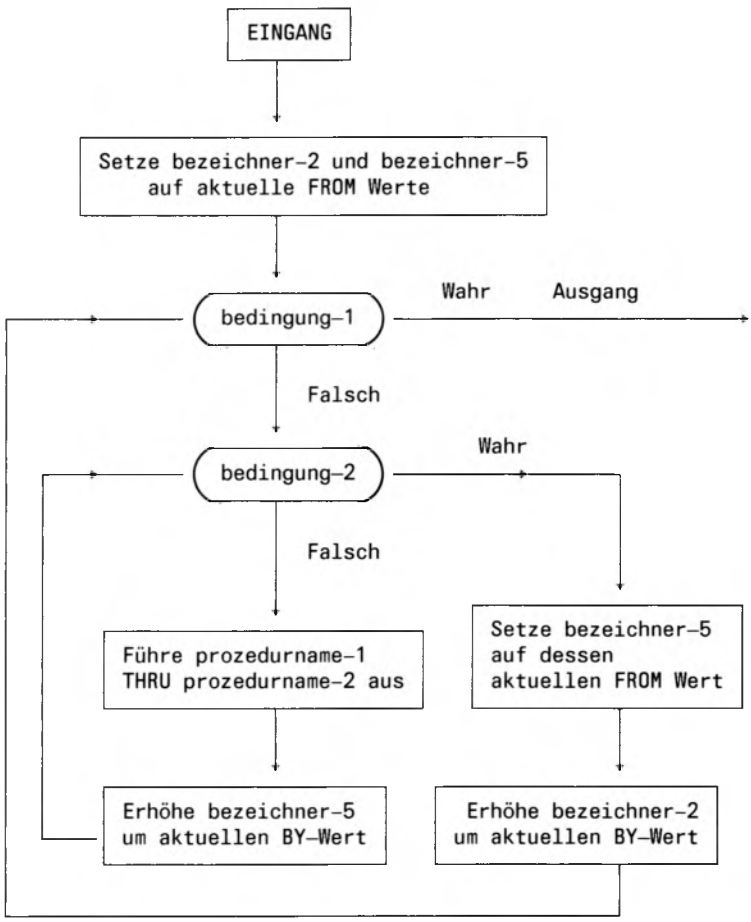


Bild 2-2 Flußdiagramm einer VARYING-Angabe einer PERFORM-Anweisung mit zwei Bedingungen.

Nach Abschluß der PERFORM-Anweisung enthält bezeichner-5 den aktuellen Wert von bezeichner-6. bezeichner-2 erhält den aktuellen Wert von bezeichner-3, falls bedingung-1 falsch war.

Wenn zwei Bezeichner verändert werden, durchläuft bezeichner-5 jedes Mal, wenn bezeichner-2 verändert wird, einen kompletten Zyklus (FROM, BY, UNTIL).

Bei drei Bezeichnern wird verfahren, wie für zwei Bezeichner, mit der Ausnahme, daß bezeichner-8 jedes Mal einen kompletten Zyklus durchläuft, wenn bezeichner-5 um bezeichner-7 oder literal-4 erhöht wird. bezeichner-5 durchläuft wiederum einen kompletten Zyklus, wenn bezeichner-2 verändert wird.

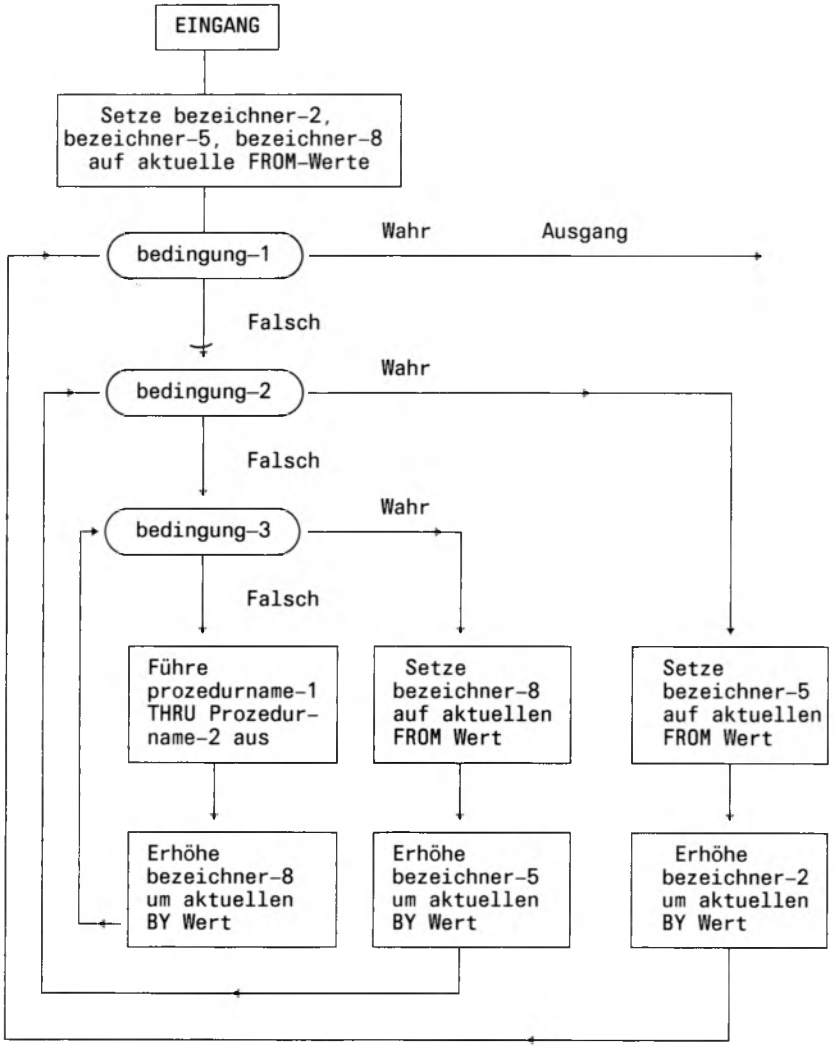


Bild 2-3 Flußdiagramm einer VARYING-Angabe in einer PERFORM-Anweisung mit drei Bedingungen.

Nach Abschluß einer PERFORM-Anweisung im Format 4, enthalten bezeichner-5 bzw. bezeichner-6 den aktuellen Wert von bezeichner-6 bzw. bezeichner-9. bezeichner-2 erhält den aktuellen Wert von bezeichner-3, falls bedingung-1 falsch war.

6. Wenn eine Prozedur im Bereich einer PERFORM-Anweisung eine weitere PERFORM-Anweisung enthält, muß nun der Bereich der zweiten PERFORM-Anweisung völlig innerhalb oder außerhalb des ursprünglichen PERFORM-Bereichs liegen.
   Überlagern sich die PERFORM-Bereiche, darf der PERFORM-Aufruf des zweiten Bereichs nicht innerhalb des ersten PERFORM-Bereichs liegen.

Beispiel

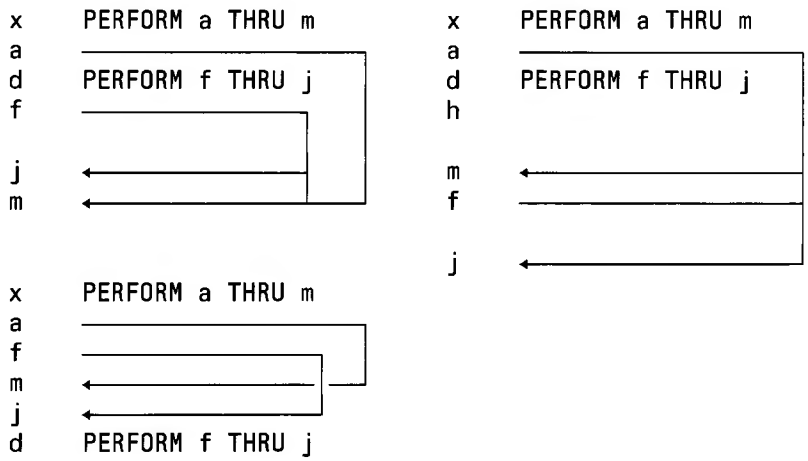


Bild 2-4 PERFORM-Anweisung als Folge

7. Tritt eine PERFORM-Anweisung in einem Kapitel auf, dessen Segmentnummer kleiner als 50 ist, so darf sie in ihrem Bereich nur folgende Prozeduren enthalten:
   - entweder Kapitel, die alle eine Segmentnummer kleiner 50 haben,
   - oder Kapitel, die als Ganzes in einem einzigen Segment sind, dessen Segmentnummer größer als 49 ist.
8. Tritt eine PERFORM-Anweisung in einem Kapitel auf, dessen Segmentnummer größer als 49 ist, so darf sie in ihrem Bereich nur folgende Prozeduren enthalten:
   - entweder Kapitel, die alle die gleiche Segmentnummer haben wie das Kapitel, das die PERFORM-Anweisung enthält
   - oder Kapitel, die alle eine Segmentnummer kleiner 50 haben.

Die STOP-Anweisung

Funktion

Durch die STOP-Anweisung wird ein Zielprogramm zeitweise unterbrochen oder beendet.

Allgemeines Format

```
STOP { RUN     }
     { literal }
```

Syntaxregeln

1. literal kann numerisch, nichtnumerisch oder eine der figurativen Konstanten sein, ausgenommen ALL.
2. Wenn literal numerisch ist, dann muß es eine vorzeichenlose Ganzzahl sein.
3. Wenn eine STOP RUN-Anweisung in einer aufeinanderfolgenden Folge von unbedingten Anweisungen innerhalb eines Satzes erscheint, so muß sie als letzte Anweisung in dieser Folge stehen.

Allgemeine Regeln

1. Die STOP RUN-Anweisung beendet die Ausführung des Programms und gibt die Steuerung an das Betriebssystem zurück.
2. Wenn das Literal STOP angegeben ist, wird es dem Operator mitgeteilt. Das Zielprogramm wird mit Ausführung der nächsten ausführbaren Anweisung in der Folge fortgesetzt.

Die STRING-Anweisung

Funktion

Durch die STRING-Anweisung wird der vollständige Inhalt von zwei oder mehreren Datenfeldern oder Teilen davon in einem einzelnen Datenfeld hintereinander gestellt.

Allgemeines Format

```
STRING {bezeichner-1} [ ,bezeichner-2 ] ... DELIMITED BY {bezeichner-3}
       {literal-1   }   {             }                  {literal-3   }
                        [ ,literal-2  ]                  {SIZE        }

     [ ,{bezeichner-4} ] [ , bezeichner-5 ] ...
        {literal-4   }   [ , literal-5    ]

     DELIMITED BY {bezeichner-6} [ ]
                  {literal-6   }   ...
                  {SIZE        }

     INTO bezeichner-7 [WITH POINTER bezeichner-8]

     [ ; ON OVERFLOW unbedingte-anweisung]
```

Syntaxregeln

1. Jedes Literal kann eine figurative Konstante außer ALL literal sein.
2. Alle Literale müssen nichtnumerische Literale sein und alle Bezeichner, ausgenommen bezeichner-8, müssen implizit oder explizit als USAGE IS DISPLAY beschrieben sein.
3. bezeichner-7 muß ein alphanumerisches Datenelement ohne Drukkaufbereitungssymbole oder der JUSTIFIED-Klausel darstellen.
4. bezeichner-8 muß ein numerisches ganzzahliges Datenelement sein, das eine Stelle größer ist als bezeichner-7. Das Symbol 'P' darf in PICTURE-Zeichenfolgen von bezeichner-8 nicht verwendet werden.
5. Wenn bezeichner-1, bezeichner-2, oder bezeichner-3 numerische Datenelemente sind, müssen sie als ganze Zahl ohne das Symbol 'P' in deren PICTURE-Zeichenfolgen beschrieben sein.

Allgemeine Regeln

1. Alle Bezugnahmen auf bezeichner-1, bezeichner-2, bezeichner-3, literal-1, literal-2 oder literal-3 gelten sinngemäß für bezeichner-4, bezeichner-5, bezeichner-6, literal-4, literal-5 und literal-6 und für alle Wiederholungen.

2. bezeichner-1, literal-1, bezeichner-2 und literal-2 stellen Sendedatenfelder dar. bezeichner-7 stellt das Empfangsdatenfeld dar.

3. literal-3 und bezeichner-3 geben die Zeichen an, die die Übertragung begrenzen. Wenn die SIZE-Angabe verwendet wird, wird das vollständige durch bezeichner-1, literal-1, bezeichner-2 bzw. literal-2 definierte Datenfeld übertragen. Wenn eine figurative Konstante als Begrenzer verwendet wird, so steht diese für ein einzelnes nichtnumerisches Literal.

4. Wenn eine figurative Konstante als literal-1, literal-2 oder literal-3 angegeben ist, so bezieht es sich auf ein implizites Datenfeld bestehend aus einem Zeichen, mit USAGE DISPLAY.

5. Beim Ausführen einer STRING-Anweisung läuft die Übertragung von Zeichen nach folgenden Regeln ab:

   a) literal-1, literal-2 bzw. der Inhalt der als bezeichner-1, bezeichner-2 definierten Datenfelder werden aufeinanderfolgend in das Datenfeld von bezeichner-7 übertragen. Hierbei gelten die Regeln für die MOVE-Anweisung von alphanumerischen Feldern zu alphanumerischen Feldern; es findet aber kein Auffüllen mit Leerzeichen statt.

   b) Ist DELIMITED BY ohne SIZE angegeben, wird der Inhalt von bezeichner-1, bezeichner-2 bzw. der Wert von literal-1, literal-2 zeichenweise von links nach rechts in das Empfangsfeld übertragen. Dabei wird mit dem äußersten linken Zeichen begonnen und die Übertragung wird solange fortgesetzt, bis entweder das Ende des jeweiligen Sendefeldes erreicht ist oder die Zeichenfolge von literal-3 bzw. der Inhalt von bezeichner-3 erkannt wird. Die Zeichen von literal-3 oder bezeichner-3 werden nicht übertragen.

   c) Ist DELIMITED BY SIZE angegeben, wird der gesamte Inhalt von von literal-1, literal-2, bzw. von bezeichner-1, bezeichner-2 nach bezeichner-7 übertragen, bis alle Sendefelder übertragen

sind oder das Ende von bezeichner-7 erreicht ist. Die Reihenfolge der Übertragung entspricht der Reihenfolge der Sendefelder in der STRING-Anweisung.

6. Wird POINTER angegeben, ist bezeichner-8 für den Benutzer verfügbar, d.h. der Anfangswert muß vom Benutzer gesetzt werden. Er darf nicht kleiner als 1 sein.

7. Wird POINTER weggelassen, werden die nachfolgenden Regeln so angewendet, als ob der Benutzer bezeichner-8 mit dem Anfangswert 1 versehen hätte.

8. Bei Übertragung nach bezeichner-7 wird jedes Zeichen einzeln aus dem Sendefeld an die Zeichenstelle von bezeichner-7, die durch den Wert von bezeichner-8 bestimmt wird, übertragen. Vor Übertragen des nächsten Zeichens wird bezeichner-8 um eins erhöht. Der Wert von bezeichner-8 wird während der Ausführung der STRING-Anweisung nur auf diese Weise geändert.

9. Nach Ausführen der STRING-Anweisung hat nur der Teil von bezeichner-7, in den Zeichen übertragen wurden, einen geänderten Inhalt; der Rest von bezeichner-7 bleibt unverändert.

10. Wenn zu irgendeinem Zeitpunkt während oder nach der Initialisierung der STRING-Anweisung, jedoch vor ihrer vollständigen Abarbeitung, der Wert von bezeichner-8 entweder kleiner als 1 oder größer als die Anzahl der Zeichenstellen im Datenfeld von bezeichner-7 ist, werden keine weiteren Daten nach bezeichner-7 übertragen, sondern es wird die unbedingte Anweisung in der ON OVERFLOW-Angabe (sofern angegeben) ausgeführt.

11. Wenn die ON OVERFLOW-Angabe nicht gemacht ist, und wenn die in Allgemeine Regel 10 oben beschriebenen Bedingungen auftreten, so wird die Steuerung an die nächste ausführbare Anweisung übertragen.

Die SUBTRACT-Anweisung

Funktion

Die SUBTRACT-Anweisung wird dazu verwendet, den Wert eines Datenfeldes oder die Summe der Werte von zwei oder mehreren numerischen Datenfeldern von einem oder mehreren Datenfeldern zu subtrahieren und die Werte in ein oder mehrere Datenfelder abzulegen.

Allgemeine Formate

Format 1

```
SUBTRACT {bezeichner-1} [ , {bezeichner-2} ] ... FROM
         {literal-1   }     {literal-2   }

         bezeichner-m [ROUNDED] [, bezeichner-n [ROUNDED] ] ...

         [ ; ON SIZE ERROR unbedingte-anweisung]
```

Format 2

```
SUBTRACT {bezeichner-1} [ , {bezeichner-2} ] ... FROM {bezeichner-m}
         {literal-1   }     {literal-2   }          {literal-m   }

         GIVING bezeichner-n [ROUNDED][,bezeichner-o [ROUNDED]] ...

         [ ; ON SIZE ERROR unbedingte-anweisung]
```

Format 3

```
SUBTRACT {CORRESPONDING} bezeichner-1 FROM bezeichner-2 [ROUNDED]
         {CORR         }

         [ ; ON SIZE ERROR unbedingte-anweisung]
```

Syntaxregeln

1. Jeder Bezeichner muß sich auf ein numerisches Datenelement beziehen, im Format 2 nach dem Wort GIVING auch auf ein numerisches druckaufbereitetes Datenfeld. Im Format 3 muß sich jeder Bezeichner auf eine Datengruppe beziehen.
2. Jedes Literal muß numerisch sein.
3. Die gemeinsame Stellenzahl der Operanden darf nicht mehr als 18 Ziffern sein (vgl. Arithmetische Anweisungen in diesem Kapitel).
4. Im Format 1 nehmen zwei oder mehrere Operanden an der Substraktion teil und werden in einen oder mehreren Ergebnisfeldern abgespeichert. Format 2 benutzt auch nach GIVING ein anderes Ergebnisfeld. Im Format 3 werden Paare von entsprechenden Datenfeldern bestimmt und subtrahiert.
5. CORR ist eine Abkürzung für CORRESPONDING.

Allgemeine Regeln

1. Vgl. Die ROUNDED-Angabe, Die SIZE ERROR-Angabe, Arithmetische Anweisungen, Überlappende Operanden und Mehrfachergebnisse in arithmetischen Anweisungen in diesem Kapitel.
2. Im Format 1 werden alle Literale oder Bezeichner vor dem Wort FROM addiert, diese Summe wird von dem aktuellen Wert von bezeichner-m subtrahiert, und dieses Ergebnis wird in bezeichner-m gespeichert. Dieser Vorgang wird entsprechend für jeden Operanden nach dem Wort FROM wiederholt.
3. Im Format 2 werden alle Literale oder Bezeichner vor dem Wort FROM addiert, die Summe wird von literal-m bzw. bezeichner-m subtrahiert und das Ergebnis der Subtraktion wird als der neue Wert von bezeichner-n, bezeichner-o usw. gespeichert.
4. Wenn Format 3 verwendet wird, werden die Datenfelder in bezeichner-1 von den entsprechenden Datenfeldern in bezeichner-2 subtrahiert und dort gespeichert.

Die UNSTRING-Anweisung

Funktion

Durch die UNSTRING-Anweisung werden benachbarte Daten in einem Sendedatenfeld getrennt und in mehrere Empfangsdatenfelder übertragen.

Allgemeines Format

```
UNSTRING bezeichner-1

[ DELIMITED BY [ALL] { bezeichner-2 } [ , OR [ALL] { bezeichner-3 } ] ... ]
                     { literal-1    }               { literal-2    }

 INTO bezeichner-4  [ , DELIMITER IN bezeichner-5]

 [ , COUNT IN bezeichner-6]

 [ , bezeichner-7  [, DELIMITER IN bezeichner-8]

 [ , COUNT IN bezeichner-9] ] ...

 [ WITH POINTER bezeichner-10] [TALLYING IN bezeichner-11]

 [ ; ON OVERFLOW unbedingte-anweisung]
```

Syntaxregeln

1. Jedes Literal muß nichtnumerisch sein. Zusätzlich kann jedes Literal eine figurative Konstante sein, außer ALL literal.
2. bezeichner-1, bezeichner-2, bezeichner-3, bezeichner-5 und bezeichner-8 müssen implizit oder explizit als ein alphanumerisches Datenfeld beschrieben sein.
3. bezeichner-4 und bezeichner-7 müssen USAGE IS DISPLAY beschrieben sein und:
   - alphabetisch (aber kein B in der PICTURE-Zeichenfolge)
   - alphanumerisch oder numerisch (aber kein P in der PICTURE-Zeichenfolge).
4. bezeichner-6, bezeichner-9, bezeichner-10 und bezeichner-11 müssen als numerische ganzzahlige Datenelemente beschrieben sein. (mit der Ausnahme, daß das Symbol 'P' in der PICTURE-Zeichenfolge nicht verwendet werden darf).
5. Kein Bezeichner darf mit der Stufennummer 88 definiert sein.
6. Die DELIMITER IN-Angabe und die COUNT IN-Angabe kann nur gemacht werden, wenn DELIMITED BY angegeben ist.

Allgemeine Regeln

1. Alle Bezugnahmen auf bezeichner-2, literal-1, bezeichner-4, bezeichner-5 bzw. bezeichner-6 gelten gleichermaßen auch für bezeichner-3, literal-2, bezeichner-7, bezeichner-8 bzw. bezeichner-9 und für alle Wiederholungen davon.
2. bezeichner-1 stellt das Sendedatenfeld dar.
3. bezeichner-4 stellt das Empfangsdatenfeld dar. bezeichner-5 stellt das Empfangsdatenfeld für Begrenzer dar.
4. literal-1 und bezeichner-2 sind Begrenzer.
5. bezeichner-6 enthält die Anzahl von Zeichen, die innerhalb von bezeichner-1 bis zum Begrenzer gefunden wurden, d.h. die Anzahl von Zeichen von bezeichner-1, die nach bezeichner-4 übertragen werden sollen. Darin ist die Anzahl der Begrenzungszeichen nicht enthalten.
6. Das durch bezeichner-10 benannte Datenfeld enthält einen Wert, der eine relative Zeichenposition innerhalb des durch bezeichner-1 definierten Datenfeldbereichs anzeigt.
7. Das durch bezeichner-11 benannte Datenfeld ist ein Zähler, der die Anzahl der Datenfelder die während der Ausführung einer UNSTRING-Anweisung angesprochen werden, zählt.
8. Wenn als Begrenzer eine figurative Konstante verwendet wird, so steht dieser für ein einzelnes nichtnumerisches Literal.

   Ist ALL angegeben, werden aufeinanderfolgende Wiederholungen von literal-1 bzw. bezeichner-2 so interpretiert, als wären sie nur einmal aufgetreten. Die Übertragung erfolgt nach Regel 13d.
9. Wenn für zwei aufeinanderfolgende Begrenzer eine Überprüfung auftritt, so wird das aktuelle Empfangsfeld entweder mit Leerzeichen oder Nullen entsprechend der Beschreibung des Empfangsfeldes aufgefüllt.
10. literal-1 oder der Inhalt des durch bezeichner-2 benannten Datenfeldes kann irgendein Zeichen aus dem Zeichenvorrat des Computers enthalten.

11. literal-1 bzw. bezeichner-2 stellen Begrenzer dar. Wenn ein Begrenzer ein oder mehrere Zeichen enthält, so müssen alle Zeichen in zusammenhängenden Positionen des Sendefeldes und in der vorgegebenen Reihenfolge vorhanden sein, damit sie als Begrenzer erkannt werden.

12. Wenn zwei oder mehrere Begrenzer in der DELIMITED BY-Angabe stehen, werden sie mit OR verknüpft. Jeder Begrenzer wird mit dem Sendefeld verglichen. Wenn eine Übereinstimmung vorhanden ist, werden die Zeichen im Sendefeld als ein einzelner Begrenzer betrachtet. Kein Zeichen im Sendefeld kann als Teil von mehr als einem Begrenzer betrachtet werden.

    Die Begrenzer werden mit dem Sendefeld in der Reihenfolge verglichen, in der sie in der UNSTRING-Anweisung angegeben sind.

13. Nach Initialisierung der UNSTRING-Anweisung ist bezeichner-4 der aktuelle Empfangsbereich. Die Daten werden von bezeichner-1 nach bezeichner-4 nach folgenden Regeln übertragen:

    a) Ist POINTER angegeben, beginnt die Prüfung mit der Zeichenfolge von bezeichner-1 mit der relativen Zeichenposition, die durch bezeichner-10 angezeigt wird.
    Ist POINTER nicht angegeben, beginnt die Prüfung mit dem äußersten linken Zeichen von bezeichner-1.

    b) Ist DELIMITED BY angegeben, wird die Prüfung von links nach rechts fortgesetzt, bis ein Begrenzer, entweder dargestellt durch den Wert von literal-1 oder das Datenfeld von bezeichner-2, angetroffen wird.

        Ist DELIMITED BY nicht angegeben, ist die Anzahl der geprüften Zeichen genauso groß wie die Größe des aktuellen Empfangsbereichs.

        Ist jedoch das Vorzeichen des Empfangsfeldes so spezifiziert, daß es eine eigene Zeichenposition belegt, ist die Anzahl der Zeichen um eins geringer als die Größe des aktuellen Empfangsbereichs.

        Ist das Ende von bezeichner-1 erreicht, ehe ein Begrenzer gefunden wurde, endet die Prüfung mit dem zuletzt geprüften Zeichen.

c) Die Zeichenfolge, die bis zum Begrenzer ermittelt wurde, wird als alphanumerisches Datenelement behandelt und entsprechend den Regeln der MOVE-Anweisung in den aktuellen Empfangsbereich übertragen (siehe MOVE-Anweisung).

d) Ist DELIMITER IN angegeben, so werden die begrenzenden Zeichen als alphanumerische Datenelemente behandelt und werden in das durch bezeichner-5 benannte Datenfeld nach den Regeln für die MOVE-Anweisung übertragen (vgl. Die MOVE-Anweisung). Wenn die begrenzende Bedingung das Ende des durch bezeichner-1 benannten Datenfeldes darstellt, dann wird bezeichner-5 mit Leerzeichen aufgefüllt.

e) Ist COUNT IN angegeben, wird die Anzahl der geprüften Zeichen (ausgenommen evtl. vorhandene Begrenzungszeichen) in den Bereich von bezeichner-1 nach den Regeln der Elementübertragung übertragen.

f) Ist DELIMITED BY angegeben, wird die Überprüfung von bezeichner-1 mit dem ersten Zeichen rechts vom Begrenzer fortgesetzt.
Ist DELIMITED BY nicht angegeben, wird die Überprüfung mit dem nächsten Zeichen nach dem letzten übertragenen Zeichen fortgesetzt.

g) Nachdem die Daten nach bezeichner-4 übertragen worden sind, ist bezeichner-7 der aktuelle Empfangsbereich. Die beschriebene Prozedur wird solange wiederholt, bis alle Zeichen im bezeichner-1 verarbeitet oder keine Empfangsbereiche mehr vorhanden sind.

14. Die Anfangswerte für die Datenfelder mit der POINTER- oder TALLYING-Angabe müssen vom Benutzer gesetzt werden.

15. Der Inhalt von bezeichner-10 wird bei jedem im Datenfeld von bezeichner-1 geprüften Zeichen um eins erhöht.

16. Wenn die UNSTRING-Anweisung mit POINTER-Angabe ausgeführt ist, enthält bezeichner-10 einen Wert, der dem Anfangswert, zuzüglich der Anzahl der im Datenfeld von bezeichner-1 geprüften Zeichen, entspricht.

    Wenn eine UNSTRING-Anweisung mit TALLYING-Angabe ausgeführt ist, enthält bezeichner-11 einen Wert, der dem Anfangswert entspricht,zuzüglich der Anzahl der Empfangsfelder, die angesprochen wurden.

17. Durch jede der folgenden Situationen wird eine OVERFLOW-Bedingung verursacht:

    a) Bei der Ausführung von UNSTRING ist der Wert im Datenfeld von bezeichner-10 kleiner als 1 oder größer als die Länge von bezeichner-1.

    b) Während der Ausführung der UNSTRING-Anweisung sind alle Datenempfangsbereiche angesprochen worden, das Datenfeld von bezeichner-1 enthält jedoch noch Zeichen, die nicht geprüft worden sind.

18. Wenn eine OVERFLOW-Bedingung vorliegt, wird die UNSTRING-Anweisung beendet.

    Ist ON OVERFLOW angegeben, wird die in dieser Angabe enthaltene unbedingte Anweisung ausgeführt. Wurde die ON OVERFLOW-Angabe weggelassen, geht die Steuerung zur nächsten auszuführenden Anweisung über.

    Die Auswertung der Subskribierung und Indizierung für die Bezeichner geschieht wie folgt:

    a) Sind die Felder bezeichner-1, bezeichner-10 und bezeichner-11 subskribiert der indiziert, wird der Indexwert für diese Felder nur einmal berechnet und zwar unmittelbar vor der Ausführung der UNSTRING-Anweisung.

    b) Jede Subskribierung oder Indizierung im Zusammenhang mit bezeichner-2, bezeichner-3, bezeichner-4, bezeichner-5 oder bezeichner-6 wird ausgewertet, bevor die Datenübertragung in das jeweilige Datenfeld erfolgt.

3 Tabellenbearbeitung

3.1 Einführung in das Tabellenbearbeitungsmodul

Das Tabellenbearbeitungsmodul bietet die Möglichkeit, Tabellen aus aufeinanderfolgenden Datenfeldern zu definieren und auf ein Datenelement relativ zu dessen Position in der Tabelle zuzugreifen. Durch eine Klausel wird definiert, wie viele Datenelemente eine Tabelle hat. Jedes Datenfeld kann durch Verwendung eines Subskripts oder eines Index' angesprochen werden (vgl. Kapitel 2).

Es ist auch möglich, auf Datenfelder mit variabler Länge in mehrdimensionalen Tabellen zuzugreifen. Die maximale Dimensionszahl, sofern der ANSI-Schalter gesetzt ist (vgl. Kapitel 2), ist auf drei begrenzt, ansonsten auf 49. In der Praxis wird die maximale Dimensionszahl jedoch durch die maximale Größe der DATA DIVISION begrenzt. Zusätzlich besteht die Möglichkeit, aufsteigende oder absteigende Schlüssel anzugeben und eine Dimension einer Tabelle nach einem Datenfeld zu durchsuchen, das einer angegebenen Bedingung entspricht.

3.2 DATA DIVISION im Tabellenbearbeitungsmodul

OCCURS

Die OCCURS-Klausel

Funktion

Durch die OCCURS-Klausel braucht man für Datenfelder, die sich wiederholen, keine getrennten Eintragungen zu machen. Ferner liefert die OCCURS-Klausel Information, die für die Anwendung von Subskripten oder Indizes erforderlich ist.

Allgemeine Formate

Format 1

```
OCCURS ganzzahl-2 TIMES

    [ [ {ASCENDING  } KEY IS datenname-2 [,datenname-3]... ] ... ]
        {DESCENDING }

    [ INDEXED BY indexname-1 [, indexname-2] ... ]
```

Format 2

```
OCCURS ganzzahl-1 TO ganzzahl-2 TIMES DEPENDING ON datenname-1

  [ [ {ASCENDING  } KEY IS datenname-2  [,datenname-3]... ] ... ]
      {DESCENDING }

  [ INDEXED BY indexname-1 [, indexname-2] ... ]
```

Syntaxregeln

1. Wenn ganzzahl-1 und ganzzahl-2 verwendet werden, muß der Wert von ganzzahl-1 kleiner als der Wert von ganzzahl-2 sein.

2. datenname-1 muß als positives ganzzahliges, numerisches Datenfeld beschrieben sein.

3. datenname-1, datenname-2, datenname-3 ... können gekennzeichnet sein.

4. datenname-1 muß entweder der Name der Eintragung sein, die die OCCURS-Klausel enthält, oder der Name einer Eintragung, die der Eintragung untergeordnet ist, die die OCCURS-Klausel enthält.

5. datenname-3 usw. muß der Name einer Eintragung sein, der einer solchen Datengruppe untergeordnet ist, die Subjekt dieser Eintragung ist.

6. Eine INDEXED BY-Angabe ist erforderlich, wenn auf das Subjekt dieser Eintragung oder einer anderen Eintragung, die dieser Eintragung untergeordnet ist, durch Indizierung angesprochen werden soll. Der durch diese Klausel identifizierte Indexname darf nicht an anderer Stelle definiert sein und, da er keine Daten enthält, kann er nicht mit irgendeiner anderen Datenhierarchie verknüpft sein. Der Übersetzer generiert ein implizites Datenfeld mit USAGE INDEX.

7. Nach einer Datenbeschreibung, die eine OCCURS-Klausel im Format 2 enthält, dürfen in dieser Datensatzbeschreibung nur Datenbeschreibungen stehen, die dieser untergeordnet sind.

8. Die OCCURS-Klausel kann nicht in einer Datenbeschreibung angegeben werden, die

   - mit einer der Stufennummern 01, 66, 77 oder 88 verknüpft ist (sofern der ANSI-Schalter gesetzt ist);
   - ein Datenfeld beschreibt, dessen Größe variabel ist. Ein Datenfeld ist variabel, wenn die Datenbeschreibung irgendwelcher untergeordneter Datenfelder die OCCURS-Klausel in Format 2 enthält.

9. Im Format 2 darf das durch datenname-1 benannte Datenfeld keine Zeichenposition belegen, die zwischen der ersten und der letzten Zeichenposition liegt, die durch eine OCCURS-Klausel enthaltende Datensatzbeschreibung definiert ist.

10. Wenn datenname-2 nicht Subjekt dieser Eintragung ist, dann gilt:

    - Alle durch datenname in der KEY IS-Angabe identifizierten Datenfelder müssen in einer Datengruppe liegen, die Subjekt dieser Eintragung ist.
    - Datenfelder, die durch den Datennamen in der KEY IS-Angabe identifiziert werden, dürfen keine OCCURS-Klausel enthalten.
    - Es darf keine Eintragung, die eine OCCURS-Klausel enthält, zwischen den Datenfeldern stehen, die durch die Datennamen in der KEY IS-Angabe und das Subjekt dieser Eintragung identifiziert werden.

11. indexname-1, indexname-2, ... müssen eindeutige Wörter innerhalb des Programms darstellen.

Allgemeine Regeln

1. Die OCCURS-Klausel wird dazu verwendet, Tabellen zu definieren. Wenn die OCCURS-Klausel verwendet wird, muß der Datenname, der Subjekt dieser Eintragung ist, entweder subskribiert oder indiziert werden, wenn auf ihn in einer anderen Anweisung als einer SEARCH- oder USE FOR DEBUGGING-Anweisung Bezug genommen wird. Ist der Name einer Datengruppe Subjekt dieser Eintragung, müssen alle Datennamen, die zu dieser Gruppe gehören, subskribiert oder indiziert werden, wenn sie als Operanden verwendet werden, ausgenommen als Objekt einer REDEFINES-Klausel.

2. Mit Ausnahme der OCCURS-Klausel selbst gelten alle Datenbeschreibungsklauseln, die mit einem Datenfeld verknüpft sind, und dessen Beschreibung eine OCCURS-Klausel enthält, für jedes Tabellenelement des beschriebenen Datenfeldes.

3. Die Anzahl der Tabellenelemente der Subjekteintragung wird wie folgt definiert:

   a) Im Format 1 stellt der Wert von ganzzahl-2 die genaue Anzahl der Tabellenelemente dar.

   b) Im Format 2 stellt der aktuelle Wert des durch datenname-1 benannten Datenfeldes die Anzahl der Tabellenelemente dar.

   Dieses Format gibt an, daß das Subjekt dieser Eintragung eine variable Anzahl von Tabellenelementen hat. Der Wert von ganzzahl-2 stellt die maximale Anzahl und der Wert von ganzzahl-1 stellt die Mindestanzahl von Tabellenelementen dar. Dies bedeutet nicht, daß die Länge des Eintragungssubjektes variabel ist, sondern nur, daß die Anzahl der Tabellenelemente variabel ist.

   Der Wert des durch datenname-1 benannten Datenfeldes muß im Bereich von ganzzahl-1 bis ganzzahl-2 liegen. Wenn der Wert des durch datenname-1 benannten Datenfeldes vermindert wird, wird der Inhalt der Datenfelder unbestimmt, deren Tabellenelementnummern jetzt den Wert des durch datenname-1 benannten Datenfeldes überschreiten.

4. Wenn eine Datengruppe angesprochen wird, der eine im Format 2 der OCCURS-Klausel geschriebene Eintragung untergeordnet ist, wird in der Operation nur der Teil des Tabellenbereiches verwendet, der durch den Wert in datenname-1 spezifiziert ist.

5. Die KEY IS-Angabe wird verwendet, um anzuzeigen, daß die wiederholten Daten in aufsteigender oder absteigender Folge entsprechend den in datenname-2, datenname-3 usw. enthaltenen Werten angeordnet sind. Die aufsteigende oder absteigende Reihenfolge wird entsprechend den Regeln für den Vergleich von Operanden bestimmt. Die Datennamen müssen in absteigender Reihenfolge ihrer Wertigkeit aufgelistet sein.

Beispiel

für eine einstufige Tabelle

```
01  TABELLE.
    02   TAB-1          OCCURS  10.
         03   IRMIN     PIC X(5).
         03   MEIER     PIC X(5).
         03   FRANKE    PIC X(6).
```

Beispiel

für eine zweistufige Tabelle

```
01 TABELLE.
   02    REINHARD              OCCURS 5.
         03   MARGOT           OCCURS 3.
              05  JUDITH       PIC X(5).
              05  PIA          PIC X(3).
              05  DINO         PIC X(7).
```

Beispiel

für eine dreistufige Tabelle

```
01  TABELLE.
    02  KUNO                    OCCURS 5.
        05   KARIN              OCCURS 3.
             10  INGO         PIC X(3).
             10  STUFE-3       OCCURS 4.
                 15  JENS    PIC X(4).
                 15  IRMIN   PIC X(5).
```

In diesem Fall muß man JENS und IRMIN mit drei Subskripten ansprechen:

MOVE JENS (4 2 3) TO IRMIN (1 2 1).

Die USAGE-Klausel

Funktion

Die USAGE-Klausel gibt das Format eines Datenfeldes im Speicher an.

Allgemeines Format

```
[ USAGE IS] INDEX
```

Syntaxregeln

1. Ein Index-Datenfeld kann angesprochen werden:
   - in einer SEARCH- oder SET-Anweisung
   - in einer Vergleichsbedingung
   - in der USING-Angabe einer PROCEDURE DIVISION Überschrift
   - in der USING-Angabe einer CALL-Anweisung
2. Die SYNCHRONIZED, JUSTIFIED, PICTURE, VALUE und BLANK WHEN ZERO-Klauseln können nicht dazu verwendet werden, Datengruppen oder Datenelemente zu beschreiben, die mit der USAGE IS INDEX-Klausel beschrieben sind.

Allgemeine Regeln

1. Die USAGE-Klausel kann auf jeder Stufe geschrieben werden. Wenn die USAGE-Klausel auf einer Datengruppenstufe geschrieben ist, so gilt sie für jedes Datenelement in der Gruppe. Die USAGE-Klausel eines Datenelementes darf nicht im Widerspruch stehen zu der USAGE-Klausel einer Gruppe, zu der das Datenelement gehört.

2. Ein Datenelement, das mit der USAGE IS INDEX-Klausel beschrieben ist, wird Index-Datenfeld genannt und enthält einen Wert, der der Nummer eines Tabellenelementes entspricht. Das Datenelement darf keine Bedingungs-Variable sein. Der Übersetzer weist ein binäres Feld von zwei Bytes mit PICTURE 9(4) COMP zu.
   Wenn eine Datengruppe mit der USAGE IS INDEX-Klausel beschrieben ist, sind alle Datenelemente in der Gruppe Index-Datenfelder. Die Gruppe selbst ist kein Index-Datenfeld und kann nicht in einer SEARCH oder SET-Anweisung oder in einer Vergleichsbedingung verwendet werden.

3. Ein Index-Datenfeld kann einen Teil einer Gruppe darstellen, auf die in einer MOVE, Eingabe- oder Ausgabe-Anweisung Bezug genommen wird, wobei dann keine Konvertierung stattfindet.

3.3 PROCEDURE DIVISION in der Tabellenbearbeitung

3.3.1 Vergleichsbeschreibung

Vergleiche zwischen Indexnamen und/oder Index-Datenfeldern

Vergleichstests können zwischen folgenden Datenfeldern durchgeführt werden:

— Zwei Indexnamen. Das Ergebnis ist das gleiche, als ob die entsprechenden Tabellenelementnummern verglichen würden.

— Ein Indexname und ein Datenfeld (aber kein Index-Datenfeld) oder ein Literal. Die Tabellenelementnummer, die dem Wert des Indexnamens entspricht, wird mit dem Datenfeld oder dem Literal verglichen.

— Ein Index-Datenfeld und ein Indexname oder ein anderes Index-Datenfeld. Die tatsächlichen Werte werden ohne Konvertierung verglichen.

— Das Ergebnis des Vergleichs eines Index-Datenfeldes mit irgendeinem Datenfeld oder Literal, das oben nicht angegeben ist, ist undefiniert.

3.3.2 Überlappende Operanden

Wenn sich Sende- und Empfangsdatenfeld einer SET-Anweisung im Speicher überlappen, so ist das Ergebnis einer Vergleichs-Anweisung undefiniert.

3.3.3 Die Anweisungen

Die SEARCH-Anweisung

Funktion

Die SEARCH-Anweisung wird dazu verwendet, ein Tabellenelement in einer Tabelle zu suchen, das einer angegebenen Bedingung entspricht, und den zugehörigen Indexnamen so zu setzen, daß er auf dieses Tabellenelement zeigt.

Allgemeine Formate

Format 1

```
SEARCH bezeichner-1   [ VARYING { bezeichner-2 } ]
                                { indexname-1  }

    [ ; AT END unbedingte-anweisung-1]

     ; WHEN bedingung-1 { unbedingte-anweisung-2 }
                        { NEXT SENTENCE          }

    [ ; WHEN bedingung-2 { unbedingte-anweisung-2 } ] ...
                         { NEXT SENTENCE          }
```

Format 2

```
SEARCH ALL bezeichner-1  [ ; AT END unbedingte-anweisung-1]

                                            ⎧bezeichner-3       ⎫
                             ⎧IS EQUAL TO⎫  ⎪                   ⎪
          ⎧  datenname-1     ⎨           ⎬  ⎨literal-1          ⎬
 ; WHEN   ⎨                  ⎩IS =       ⎭  ⎪                   ⎪
          ⎪                                 ⎩arithm. ausdruck-1 ⎭
          ⎩  bedingungsname-1

                                            ⎧bezeichner-4       ⎫
    ⎡                        ⎧IS EQUAL TO⎫  ⎪                   ⎪       ⎤
    ⎢ AND ⎧  datenname-2     ⎨           ⎬  ⎨literal-2          ⎬       ⎥ ...
    ⎢     ⎨                  ⎩IS =       ⎭  ⎪                   ⎪       ⎥
    ⎣     ⎩  bedingungsname-2               ⎩arithm. ausdruck-2 ⎭       ⎦

    ⎧unbedingte-anweisung-2⎫
    ⎩NEXT SENTENCE         ⎭
```

Anmerkung

Das erforderliche Vergleichszeichen '=' ist nicht unterstrichen, um Verwechslungen mit anderen Symbolen zu vermeiden.

Syntaxregeln

1. Sowohl im Format 1 als auch im Format 2 darf bezeichner-1 nicht subskribiert oder indiziert sein, seine Beschreibung muß jedoch eine OCCURS-Klausel und eine INDEXED BY-Klausel enthalten. Die Beschreibung von bezeichner-1 im Format 2 muß außerdem eine KEY IS-Angabe in ihrer OCCURS-Klausel enthalten.

2. bezeichner-2 muß, sofern angegeben, als USAGE IS INDEX oder als numerisches Datenelement ohne Dezimalstellen rechts vom angenommenen Dezimalpunkt beschrieben sein.

3. Im Format-1 kann bedingung-1, bedingung-2 usw. irgendeine Bedingung sein, wie in "Bedingungsausdrücke" in Kapitel 2 beschrieben.

4. Im Format 2 müssen alle angesprochenen Bedingungsnamen nur einem Wert zugeordnet sein.
   Der mit einem Bedingungsnamen verknüpfte Datenname muß in der KEY-Klausel von bezeichner-1 erscheinen. Jeder datenname-1 bzw. datenname-2 kann gekennzeichnet werden. Jeder datenname-1 bzw. datenname-2 muß durch den ersten Indexnamen indiziert werden, der mit bezeichner-1 verknüpft ist, zusammen mit anderen Indizes oder Literalen, falls erforderlich, und muß in der KEY-Klausel von bezeichner-1 angesprochen werden.
   bezeichner-3, bezeichner-4 oder Bezeichner, die in arithmetischer-ausdruck-1, bzw. arithmetischer-ausdruck-2 angegeben sind, dürfen nicht in der KEY-Klausel von bezeichner-1 angesprochen oder durch den ersten mit bezeichner-1 verknüpften Indexnamen indiziert werden.

   Im Format 2 müssen, wenn ein Datenname in der KEY-Klausel von bezeichner-1 angesprochen wird, oder wenn ein mit einem Datennamen verknüpfter Bedingungsname in der KEY-Klausel von bezeichner-1 angesprochen wird, alle vorhergehenden Datennamen in der KEY-Klausel von bezeichner-1 oder deren zugehörigen Bedingungsnamen ebenfalls angesprochen werden.

Allgemeine Regeln

1. Wenn Format 1 der SEARCH-Anweisung verwendet wird, findet eine serielle Suche statt, wobei mit dem Wert des aktuellen Index begonnen wird.

   a) Wenn zu Beginn der Ausführung der SEARCH-Anweisung der mit bezeichner-1 verknüpfte Indexname einen Wert enthält, der einer Tabellenelementnummer entspricht, die größer ist als die höchste zulässige Tabellenelementnummer für bezeichner-1, wird SEARCH sofort beendet. Die Anzahl von Tabellenelementen von bezeichner-1, deren letzte die höchste zulässige Tabellenelementnummer darstellt, wird in der OCCURS-Klausel behandelt.
   Dann wird, wenn AT END angegeben ist, unbedingte-anweisung-1 ausgeführt; wenn AT END nicht angegeben ist, geht die Steuerung an den nächsten ausführbaren Satz über.

   b) Wenn zu Beginn der Ausführung der SEARCH-Anweisung der mit bezeichner-1 verknüpfte Indexname einen Wert kleiner gleich der höchsten zulässigen Tabellenelementnummer für bezeichner-1 enthält, testet die SEARCH-Anweisung das aktuelle Tabellenelement durch Auswertung der Bedingungen in der angegebenen Reihenfolge. Wenn keine der Bedingungen erfüllt ist, wird der Indexname für bezeichner-1 um eins erhöht.

   Der Vorgang wird dann mit dem neuen Index so lange wiederholt, bis der neue Wert für bezeichner-1 einem Tabellenelement außerhalb des zulässigen Bereiches der Werte von Tabellenelementen entspricht; in diesem Fall wird die Suche wie in 1a) oben beschrieben beendet. Wenn bei der Auswertung eine der Bedingungen erfüllt ist, wird die Suche abgebrochen, und die mit dieser Bedingung verknüpfte unbedingte Anweisung wird ausgeführt; der Indexname bleibt auf dem Tabellenelement stehen, bei dem die Bedingung erfüllt worden ist.

2. Bei einer SEARCH-Anweisung im Format 2 sind die Ergebnisse einer SEARCH ALL-Operation nur voraussagbar, wenn

   a) die Daten in der Tabelle in der gleichen Weise geordnet sind, wie in der ASCENDING/DESCENDING KEY-Klausel beschrieben und die mit der Beschreibung von bezeichner-1 verknüpft ist und

b) der Inhalt der in der WHEN-Klausel angesprochenen Schlüssel ausreicht, um ein Tabellenelement eindeutig zu identifizieren.

3. Wenn für die SEARCH-Anweisung Format 2 verwendet wird, findet eine nicht-serielle Suche statt; der Anfangswert des Indexnamens für bezeichner-1 wird ignoriert. Der Index wird bei der Suche jeweils erhöht, er überschreitet jedoch niemals die Grenzen der Tabelle. Wenn irgendeine der in der WHEN-Klausel angegebenen Bedingungen innerhalb des zulässigen Bereiches nicht erfüllt wird, geht die Steuerung an die unbedingte-anweisung-1 der AT END-Angabe über oder aber an den nächsten ausführbaren Satz, sofern AT END nicht angegeben ist; in beiden Fällen ist der Endwert des Index nicht voraussagbar. Wenn alle Bedingungen erfüllt werden können, zeigt der Index auf das entsprechende Tabellenelement und die Steuerung geht an die unbedingte-anweisung-2 über.

4. Nach Ausführung von unbedingte-anweisung-1, unbedingte-anweisung-2 oder unbedingte-anweisung-3, die nicht mit einer GO TO-Anweisung enden, geht die Steuerung an den nächsten ausführbaren Satz über.

5. Im Format 2 ist der Indexname, der für die Suche verwendet wird, der erste (oder einzige) Indexname, der in der INDEXED BY-Angabe von bezeichner-1 erscheint. Alle anderen Indexnamen von bezeichner-1 bleiben unverändert.

6. Im Format 1 ist, sofern die VARYING-Angabe nicht verwendet wird, der Indexname, der für die Suche verwendet wird, der erste (oder einzige) Indexname, der in der INDEXED BY-Angabe von bezeichner-1 erscheint. Alle anderen Indexnamen von bezeichner-1 bleiben unverändert.

7. Im Format 1 wird, wenn VARYING indexname-1 verwendet wird, und wenn indexname-1 in der INDEXED BY-Angabe von bezeichner-1 erscheint, dieser Indexname für die Suche verwendet. Wenn dies nicht der Fall ist, oder wenn die VARYING-Angabe bezeichner-2 angegeben ist, wird der erste (oder einzige) in der INDEXED BY-Angabe von bezeichner-1 vorgegebene Indexname für die Suche verwendet. Zusätzlich finden die folgenden Operationen statt:

   a) Wenn VARYING indexname-1 verwendet wird, und wenn indexname-1 in der INDEXED BY-Angabe einer anderen Tabelleneintragung erscheint, wird die durch indexname-1 dargestellte Tabellenelementnummer um den gleichen Betrag und zum selben Zeitpunkt erhöht, wie der mit bezeichner-1 verknüpfte Indexname erhöht wird.

   b) Wenn VARYING bezeichner-2 angegeben ist und bezeichner-2 ein Index-Datenfeld darstellt, dann wird das durch bezeichner-2 benannte Datenfeld um den gleichen Betrag und zum gleichen Zeitpunkt erhöht, wie der mit bezeichner-1 verknüpfte Index. Wenn bezeichner-2 kein Index-Datenfeld ist, wird das durch bezeichner-2 benannte Datenfeld um den Wert eins zum gleichen Zeitpunkt erhöht wie der Index von bezeichner-1.

8. Wenn bezeichner-1 ein Datenfeld ist, das einem Datenfeld untergeordnet ist, das eine OCCURS-Klausel enthält (um eine zwei- oder dreidimensionalen Tabelle zu ermöglichen), muß ein Indexname mit jeder Dimension der Tabelle durch die INDEXED BY-Angabe der OCCURS-Klausel verknüpft sein. Nur wenn man den Indexnamen von bezeichner-1 und, sofern vorhanden, bezeichner-2 oder indexname-1 setzt, wird die Ausführung der SEARCH-Anweisung modifiziert. Um eine ganze zwei- oder dreidimensionale Tabelle durchsuchen zu können, ist es nicht notwendig, eine SEARCH-Anweisung mehrmals auszuführen. Vor jeder Durchführung einer SEARCH-Anweisung müssen SET-Anweisungen ausgeführt werden, und zwar immer, wenn Indexnamen verändert werden.

Bild 3-1 zeigt ein Flußdiagramm der SEARCH Operation im **Format 1** mit zwei WHEN-Angaben.

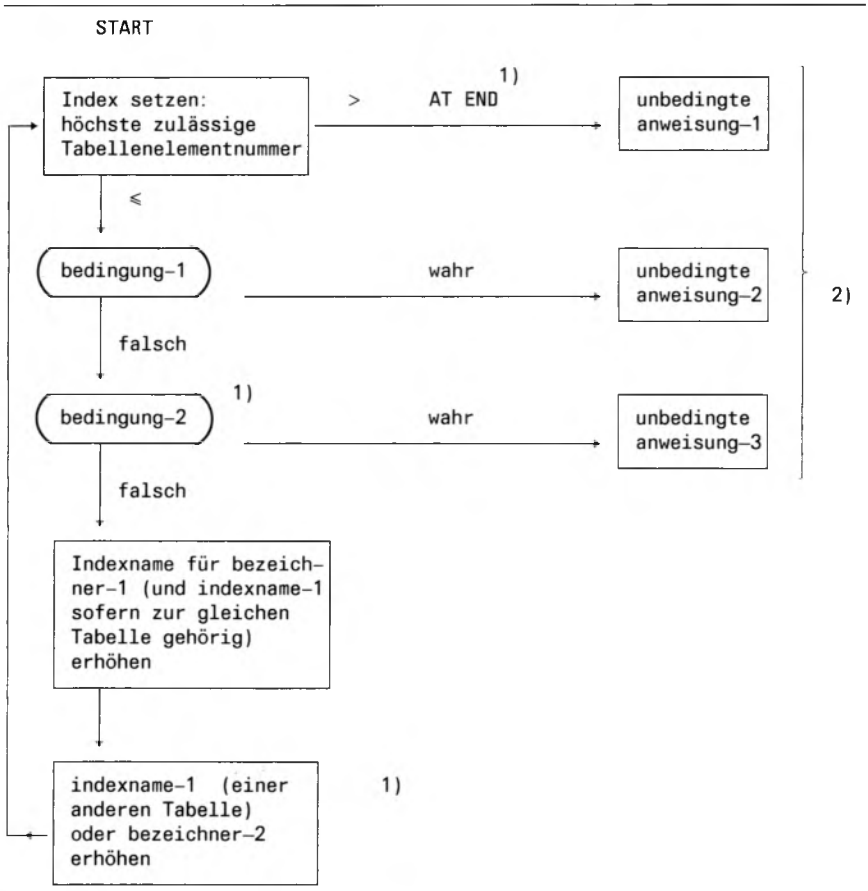


Bild 3-1 Flußdiagramm einer SEARCH-Operation mit zwei WHEN-Angaben

Hinweise

1) Diese Operationen sind optional und werden nur durchgeführt, wenn in der SEARCH-Anweisung angegeben.

2) Jeder dieser Übergänge des Programmablaufes erfolgt zum nächsten ausführbaren Satz, sofern die unbedingte Anweisung nicht mit einer GO TO Anweisung endet.

Die SET-Anweisung

Funktion

Die SET-Anweisung setzt Anfangswerte für Tabellenbearbeitungsoperationen, indem Indexnamen gesetzt werden, die mit den Tabellenelementen verknüpft sind.

Allgemeine Formate

Format 1

```
SET { bezeichner-1 [, bezeichner-2] ... } TO { bezeichner-3 }
    { indexname-1  [, indexname-2] ...  }    { ganzzahl-1  }
                                             { indexname-3 }
```

Format 2

```
SET { indexname-4 [,indexname-5] ... } { UP BY   } { bezeichner-4 }
    { bezeichner-5 [,bezeichner-6]...} { DOWN BY } { ganzzahl-2   }
                                                   { indexname-6  }
```

Syntaxregeln

1. Alle Bezüge auf indexname-1, bezeichner-1, indexname-4 bzw. bezeichner-5 gelten gleichermaßen für indexname-2, bezeichner-2, indexname-5 bzw. bezeichner-6.
2. bezeichner-1, bezeichner-3 und bezeichner-5 müssen entweder Index-Datenfelder oder ganzzahlige Datenelemente beschreiben.
3. bezeichner-4 muß als ein ganzzahliges numerisches Datenelement beschrieben werden.
4. ganzzahl-1 und ganzzahl-2 können Vorzeichen haben. ganzzahl-1 muß positiv sein.

Allgemeine Regeln

1. Indexnamen werden bezogen auf eine bestimmte Tabelle betrachtet, wenn der ANSI-Schalter gesetzt ist, und werden dadurch definiert, daß sie in der INDEXED BY-Klausel angegeben werden.

2. Wenn indexname-3 angegeben ist, muß der Wert des Index vor Ausführung der SET-Anweisung einer Tabellenelementnummer eines Elements in der zugehörigen Tabelle entsprechen.

   Wenn indexname-4, bzw. indexname-5 angegeben ist, muß der Wert des Index sowohl vor als auch nach der Ausführung der SET-Anweisung einer Tabellenelementnummer eines Datenfeldes in der zugehörigen Tabelle entsprechen. Wenn indexname-1 bzw. indexname-2 angegeben ist, muß der Wert des Index nach der Ausführung der SET-Anweisung einer Tabellenelementnummer eines Datenfeldes in der zugehörigen Tabelle entsprechen. Der Wert des mit einem Indexnamen verknüpften Index kann nach der Ausführung einer SEARCH- oder PERFORM-Anweisung undefiniert sein.

3. Im Format 1 finden die folgenden Aktionen statt:

   a) indexname-1 wird auf einen Wert gesetzt, der dann auf das entsprechende Tabellenelement verweist, das mit der Tabellenelementnummer eines durch indexname-3, bezeichner-3 oder ganzzahl-1 benannten Tabellenelementes übereinstimmt. Wenn bezeichner-3 ein Index-Datenfeld ist oder wenn indexname-3 sich auf die gleiche Tabelle wie indexname-1 bezieht, findet keine Konvertierung statt.

   b) Wenn bezeichner-1 ein Index-Datenfeld ist, wird es entweder dem Inhalt von indexname-3 oder dem von bezeichner-3 gleichgesetzt, wobei bezeichner-3 ebenfalls ein Index-Datenfeld ist; in beiden Fällen findet keine Konvertierung statt.

   c) Wenn bezeichner-1 kein Index-Datenfeld ist, kann es nur auf die Tabellenelementnummer gesetzt werden, die dem Wert von indexname-3 entspricht. Weder bezeichner-3 noch ganzzahl-1 können in diesem Fall verwendet werden.

d) Der Vorgang wird für indexname-2, bezeichner-2 usw. wiederholt, sofern angegeben. Jedes Mal wird der Wert von indexname-3 oder bezeichner-3 so verwendet, wie er zu Beginn der Ausführung der Anweisung war. Jede mit bezeichner-1 usw. verknüpfte Subskribierung oder Indizierung wird ausgewertet, unmittelbar bevor der Wert des entsprechenden Datenfeldes geändert wird.

4. Im Format 2 wird der Inhalt von indexname-4 durch einen Wert erhöht (UP BY) oder verringert (DOWN BY), der der Anzahl von Tabellenelementen entspricht, dargestellt durch den Wert von ganzzahl-2 oder bezeichner-4 ; danach wird der Vorgang für indexname-5 usw. wiederholt. Jedes Mal wird der Wert von bezeichner-4 so verwendet, wie er zu Beginn der Ausführung der Anweisung war.

5. Die Daten in Tabelle 3-1 stellen die Zulässigkeit der verschiedenen Kombinationen von Operanden in der SET-Anweisung dar. Die Referenznummern geben die anzuwendende Allgemeine Regel an.

Sendefeld	Empfangsfeld 1)		
	Ganzzahl-Datenfeld	Indexname	Index-Datenfeld
Ganzzahl	Nein/3c	zulässig/3a	Nein/3b
Literal	Nein/3c	zulässig/3a	Nein/3b
Ganzzahl-Datenfeld	Nein/3c	zulässig/3a	Nein/3b
Indexname	zulässig/3c	zulässig/3a	zulässig/3b 2)
Index-Datenfeld	Nein/3c	zulässig/3a 2)	zulässig/3b 2)

1) = Nummern beziehen sich auf die Allgemeinen Regeln
2) = Es findet keine Konvertierung statt.

Tabelle 3-1 Zulässige Operandenkombinationen für die SET-Anweisung

4 Organisation

4.1 Einführung

Das Ein-Ausgabemodul "Sequentielle Datei" ermöglicht es, auf Datensätze einer Datei in festgelegter Folge zuzugreifen, d.h. die Datensätze werden in der durch die Datei vorgegebenen Reihenfolge gelesen oder geschrieben. Es ist auch möglich, Wiederanlaufpunkte anzugeben und Speicherbereiche zwischen Dateien aufzuteilen.
Bitte beachten Sie, daß Bandverarbeitung bei LII COBOL nicht möglich ist.

4.2 Sprachkonzepte

Organisation

Sequentielle Dateien sind so organisiert, daß jeder Datensatz in der Datei einen eindeutigen vorhergehenden Datensatz hat. Dies trifft nicht auf den ersten Datensatz zu. Ebenso hat jeder Datensatz einen eindeutigen nachfolgenden Datensatz. Dies trifft nicht auf den letzten Datensatz zu. Diese Vorgänger-Nachfolgebeziehungen werden festgelegt durch die Reihenfolge der WRITE-Anweisungen beim Erstellen der Datei. Einmal festgelegt, ändern sich diese Beziehungen nicht mehr. Ausnahme: der letzte Datensatz, wenn Datensätze an das Ende der Datei angefügt werden.

Zugriffsmodus

Im sequentiellen Zugriffsmodus wird auf die Datensätze zugegriffen, in der Reihenfolge, in der die Datensätze ursprünglich geschrieben wurden.

Aktueller Satzzeiger

Ein Zeiger, der benutzt wird, um den nächsten Datensatz auszuwählen. Das Konzept des aktuellen Satzzeigers hat keine Bedeutung für eine im Ausgabemodus eröffnete Datei. Der aktuelle Satzzeiger wird nur durch OPEN- und READ-Anweisungen beeinflußt.

Ein-Ausgabezustand

Wenn in einem Dateisteuerungseintrag die FILE STATUS-Klausel angegeben ist, wird in das aus zwei Zeichen bestehende angegebene Datenfeld ein Wert gesetzt, und zwar während der Ausführung einer OPEN, CLOSE, READ, WRITE oder REWRITE-Anweisung und bevor eine USE-Prozedur ausgeführt wird, um dem COBOL-Programm den Zustand dieser Ein-Ausgabe-Operation anzuzeigen.

Zustandsanzeiger 1

Das linke Zeichen des in der FILE STATUS-Klausel angegebenen Datenfeldes wird als Zustandsanzeiger 1 bezeichnet und wird gesetzt, um die Beendigungsbedingungen der Ein-Ausgabeoperation anzuzeigen.

'0' = Erfolgreiche Beendigung
'1' = Endebedingung
'3' = Permanenter Fehler
'9' = Systemfehler

Die oben angegebenen Werte haben folgende Bedeutung:

0 = Erfolgreiche Beendigung.

Die Ein-Ausgabeanweisung wurde erfolgreich ausgeführt.

1 = Endebedingung.

Eine READ-Anweisung wurde erfolglos ausgeführt, weil

- versucht wurde, einen Datensatz zu lesen, obwohl kein nächster logischer Datensatz in der Datei vorhanden war, oder
- die erste READ-Anweisung für eine Datei mit OPTIONAL-Angabe ausgeführt wurde und diese Daten dem Programm zur Ausführungszeit der zugehörigen OPEN-Anweisung nicht zur Verfügung standen.

3 = Permanenter Fehler.

Die Ein-Ausgabeanweisung wurde erfolglos ausgeführt, weil

- versucht wurde, außerhalb der festgelegten Bereichsgrenzen einer sequentiellen Datei zu schreiben oder
- aufgrund eines Ein-Ausgabefehlers, wie z.B. Übertragungsfehler.

9 = Fehlermeldung des Betriebssystems.

Die Ein-Ausgabeanweisung wurde erfolglos ausgeführt, weil eine Bedingung aufgetreten ist, die in der Fehlermeldung des Betriebssystems beschrieben ist. Dieser Wert wird nur dazu benutzt, eine Bedingung anzuzeigen, die nicht durch andere definierte Werte des Zustandsanzeigers 1 oder durch beschriebene Kombinationen der Werte der Zustandsanzeiger 1 und 2 angezeigt werden.

Zustandsanzeiger 2

Die rechtsbündige Zeichenposition, d.h das zweite Zeichen des in der FILE STATUS-Klausel angegebenen Datenfeldes wird als Zustandsanzeiger 2 bezeichnet und wird zur näheren Beschreibung der Ergebnisse der Ein-Ausgabeoperation verwendet. Es enthält einen der folgenden Werte:

- Ist bezüglich der Ein-Ausgabeoperation keine weitere Information verfügbar, enthält der Zustandsanzeiger 2 den Wert "0".
- Enthält der Zustandsanzeiger 1 den Wert "3", ist ein nicht behebbarer Fehler aufgetreten. Dieser Fehler wird vom Betriebssystem als schwerwiegender Fehler behandelt.
- Enthält der Zustandsanzeiger 1 den Wert '9' ist der Wert des Zustandsanzeigers 2 die Nummer der Fehlermeldung des Betriebssystems. In der LII COBOL Bedienungsanleitung werden die Einzelheiten dieser Darstellung vom Zustandsanzeiger 2 behandelt.

Wenn Sie auf den Zustandsanzeiger 2 zugreifen wollen, müssen Sie das zweistellige FILE STATUS-Feld unterdefinieren.

Zulässige Kombinationen von Zustandsanzeiger 1 und 2:

In der nachfolgenden Tabelle sind die zulässigen Kombinationen der Werte der beiden Zustandsanzeiger aufgelistet. Ein 'X' im Schnittpunkt bezeichnet eine zulässige Kombination.

Zustandsanzeiger 1	Zustandsanzeiger 2 keine weitere Information (0)
erfolgreiche Beendigung (0)	X
Endebedingung (1)	X
Permanenter Fehler (3)	X
Systemfehler (9)	Fehlernummer des Betriebssystems

AT END-Bedingung

Die AT END-Bedingung kann aufgrund der Ausführung einer READ Anweisung auftreten. Einzelheiten über die Ursachen dieser Bedingung siehe READ-Anweisung.

LINAGE-COUNTER

Das reservierte Wort LINAGE-COUNTER ist der Name für ein Sonderregister. Dieses Sonderregister wird generiert, wenn die LINAGE-Klausel in einer Dateibeschreibung angegeben wurde. Der LINAGE-Counter ist ein 4 Byte langes Datenfeld, das eine vorzeichenlose Ganzzahl enthält, die maximal den Wert von ganzzahl-1 oder datenname-1 in der LINAGE-Klausel annehmen kann.

4.3 ENVIRONMENT DIVISION im Ein-Ausgabemodul

INPUT-OUTPUT SECTION

FILE-CONTROL-Paragraph

Funktion

Der FILE-CONTROL-Paragraph ordnet jeder Datei einen Namen zu und erlaubt die Beschreibung anderer dateibezogener Informationen.

Allgemeines Format:

```
FILE-CONTROL. {dateisteuerungseintrag} ...
```

Dateisteuerungseintrag

Funktion

Der Dateisteuerungseintrag benennt eine Datei und kann andere dateibezogene Informationen vorgeben.

Allgemeines Format

```
SELECT [OPTIONAL] dateiname

    ASSIGN TO {externes-dateiname-literal  }[ ,{externes dateiname-Literal  }] ...
              {dateibezeichner             }  {dateibezeichner             }

[ ; RESERVE ganzzahl-1 [AREA ]]
                       [AREAS]

[ ; ORGANIZATION IS {SEQUENTIAL     }  ]
                    {LINE SEQUENTIAL}  ]

  [; ACCESS MODE IS SEQUENTIAL]

  [; FILE STATUS IS datenname-1].
```

Syntaxregeln

1. Die SELECT-Klausel muß der erste Eintrag im Dateisteuerungseintrag sein. Alle Klauseln, die der SELECT-Klausel folgen, können in beliebiger Reihenfolge geschrieben werden.

2. Jede in der DATA DIVISION beschriebene Datei darf nur einmal als Dateiname in der SELECT-Klausel angegeben werden.
   Jede im Dateisteuerungseintrag angegebene Datei muß einen Dateibeschreibungseintrag (z.B. FD) in der DATA DIVISION des Quellprogramms haben.

3. Ist die ACCESS MODE-Klausel nicht angegeben, wird ACCESS IS SEQUENTIAL angenommen.

4. datenname-1 muß in der DATA DIVISION als ein zwei Zeichen langes alphanumerisches Datenfeld definiert sein; datenname darf nicht in der FILE SECTION (oder in der COMMUNICATION SECTION) angegeben werden.

5. datenname-1 kann gekennzeichnet sein.

6. Ist die ORGANIZATION-Klausel nicht angegeben, wird ORGANIZATION IS SEQUENTIAL angenommen.

7. OPTIONAL kann nur für Eingabedateien angegeben werden.
   OPTIONAL ist für die Eingabedateien erforderlich, die nicht unbedingt bei jeder Ausführung des Zielprogramms vorhanden sind.

8. dateibezeichner ist ein Programmiererwort; es darf jedoch nicht dasselbe wie dateiname sein.

Allgemeine Regeln

1. Die ASSIGN-Klausel beschreibt die Verknüpfung der durch datei-name benannten Datei mit einem Speichermedium (vgl. LII COBOL Bedienungsanleitung). Die Erstzuweisung ist maßgebend. Nachfolgene Zuweisungen in einer ASSIGN-Klausel werden nur zur Dokumentation angegeben.

2. Mit der RESERVE-Klausel kann der Benutzer die Anzahl der Ein-Ausgabebereiche bestimmen, die dem Zielprogramm vom Übersetzer zugeordnet werden sollen. Die RESERVE-Klausel wird nur zur Dokumentation benutzt.

3. Die ORGANIZATION-Klausel gibt den logischen Aufbau einer Datei an. Die Dateiorganisation wird bei der Erstellung der Datei festgelegt und kann danach nicht mehr geändert werden.

4. Ist LINE SEQUENTIAL ORGANIZATION angegeben, wird die Datei so behandelt, als ob sie aus Datensätzen mit variabler Länge bestünde, wobei jeder Datensatz eine Datenzeile enthält. Datenzeile heißt bei SINIX der ausgeschriebene Bereich bis zu der Stelle, an der die Taste ↵ gedrückt wurde.

5. Auf die Datensätze in der Datei wird in der Reihenfolge zugegriffen, die durch die Dateiorganisation vorgeschrieben ist. Diese Reihenfolge wird durch Vorgänger-Nachfolger-Beziehungen während der Ausführung von WRITE-Anweisungen festgelegt, wenn die Datei erstellt oder erweitert wird.

6. Ist die FILE STATUS-Klausel angegeben, wird durch das Betriebssystem ein Wert in das durch datenname-1 angegebene Datenfeld übertragen und zwar nach Ausführung jeder Anweisung, die auf diese Datei entweder explizit oder implizit Bezug nimmt. Dieser Wert gibt den Zustand der Ausführung der Anweisung an (vgl. Ein-Ausgabezustand in diesem Kapitel).

I-O-CONTROL-Paragraph

Funktion

Der I-O CONTROL-Paragraph legt Wiederanlaufpunkte fest, wenn bestimmte Ereignisse eintreten, und legt den Speicherbereich fest, der von verschiedenen Dateien gleichzeitig benutzt werden soll. Ferner gibt er die Lage von Dateien auf Mehrdateienbändern (mehrere Dateien auf einem Band) an (nur zur Dokumentation, Bandverarbeitung nicht möglich!).

Allgemeines Format

```
I-O-CONTROL.

                                              |            |REEL|  |               ||
                                              | [END OF] {        }|               ||
                                              |            |UNIT|  } OF dateiname-2||
                                              |ganzzahl-1 RECORDS |                ||...
[;RERUN [ON {dateiname-1    }]  EVERY        {ganzzahl-2 CLOCK-UNITS               ||
            {herstellername }]                |bedingungsname                      ||

[;SAME [RECORD] AREA FOR dateiname-3 {, dateiname-4}...]...

[;MULTIPLE FILE TAPE CONTAINS dateiname-5 [POSITION ganzzahl-3]

       [,dateiname-6 [POSITION ganzzahl-4]]...]... .
```

Syntaxregeln

1. Der I-O-CONTROL-Paragraph ist wahlweise.
2. dateiname-1 muß eine sequentiell organisierte Datei sein.
3. Die END OF REEL/UNIT-Angabe darf nur verwendet werden, wenn dateiname-2 eine sequentiell organisierte Datei bezeichnet.
4. herstellername muß in der RERUN-Klausel verwendet werden, wenn ganzzahl-1 RECORDS oder ganzzahl-2 CLOCK-UNITS angegeben ist.

5. Unter Beachtung folgender Einschränkungen können für einen gegebenen dateiname-2 mehrere RERUN-Klauseln angegeben werden:

   a) Sind mehrere ganzzahl-1 RECORDS-Zusätze angegeben, darf der gleiche dateiname-2 nur einmal darin vorkommen.

   b) Sind mehrere END OF REEL- oder END OF UNIT-Zusätze angegeben, darf der gleiche dateiname-2 nur einmal darin vorkommen.

6. Die beiden Formate der SAME-Klausel (SAME AREA, SAME RECORD AREA) werden nachfolgend getrennt behandelt.

   Es kann mehr als eine SAME-Klausel in einem Programm verwendet werden, dann gilt jedoch:

   a) Ein Dateiname darf nicht in mehreren SAME AREA-Klauseln vorkommen.

   b) Ein Dateiname darf nicht in mehreren SAME RECORD AREA-Klauseln vorkommen.

   c) Ein Dateiname darf gleichzeitig in einer SAME AREA- und in einer SAME RECORD AREA-Klausel vorkommen. In diesem Fall müssen alle Dateinamen, die in der SAME AREA-Klausel aufgeführt sind, auch in der SAME RECORD AREA-Klausel stehen. Die SAME RECORD AREA-Klausel darf darüber hinaus weitere Dateinamen enthalten, die nicht in der SAME AREA-Klausel stehen.

   Die Regel, daß nur eine der angegebenen Dateien in der SAME AREA-Klausel zu einem gegebenen Zeitpunkt eröffnet sein kann, hat Vorrang vor der Regel, daß alle der angegebenen Dateien in der SAME RECORD AREA-Klausel zu einem gegebenen Zeitpunkt eröffnet sein können.

7. Die in der SAME AREA- oder SAME RECORD AREA-Klausel angegebenen Dateien müssen nicht alle die gleiche Organisation bzw. den gleichen Zugriff haben.

Allgemeine Regeln

1. Die RERUN-Klausel wird nur zur Dokumentation angegeben.
2. Die SAME AREA-Klausel gibt an, daß mehrere Dateien, jedoch keine Sortier- oder Mischdateien während der Programmausführung einen Ein-Ausgabebereich gemeinsam benutzen sollen. Dieser Ein-Ausgabebereich umfaßt den gesamten Speicherbereich, der den angegebenen Dateien zugewiesen wird. Daher ist es nicht zulässig, daß mehr als eine Datei zum selben Zeitpunkt eröffnet ist (vgl. Syntaxregel 6c).
3. Die SAME RECORD AREA-Klausel gibt an, daß mehrere Dateien denselben Speicherbereich für die Verarbeitung des aktuellen logischen Datensatzes verwenden müssen. Alle Dateien können zur selben Zeit eröffnet sein. Ein logischer Datensatz in der SAME RECORD AREA gilt als ein logischer Datensatz für alle zur Ausgabe eröffneten Dateien, deren Dateinamen in dieser SAME RECORD AREA-Klausel aufgeführt sind. Er ist ebenfalls logischer Datensatz derjenigen Datei aus dieser Klausel, aus der die letzte Eingabe erfolgte. Dies entspricht einer impliziten Überlagerung aller Datensatzbereiche, wobei die Datensätze auf die am weitesten links stehende Zeichenposition ausgerichtet sind.
4. Die MULTIPLE FILE-Klausel wird nur zur Dokumentation angegeben.

4.4 DATA DIVISION im Ein-Ausgabemodul

FILE SECTION

In einem LEVEL II COBOL-Programm stellt die Dateierklärung (FD) die höchste Organisationsstufe in der FILE SECTION dar. Der Kapitelüberschrift FILE SECTION folgt eine Dateibeschreibung, die aus einer Stufenbezeichnung (FD), einem Dateinamen und einer Anzahl von unabhängigen Klauseln besteht.

Die FD-Klausel gibt an:

- wie groß die logischen und physischen Datensätze sind,
- ob Kennsätze vorhanden sind,
- welchen Wert die Kennsatzdatenfelder haben,
- welche Namen die Datensätze haben, die die Datei betreffen.

Die Eintragung wird mit einem Punkt abgeschlossen.

Aufbau der Datensatzbeschreibung

Eine Datensatzbeschreibung besteht aus einer Anzahl von Datenbeschreibungen, die die Merkmale eines bestimmten Datensatzes beschreiben. Jede Datenbeschreibung besteht aus einer Stufennummer, gegebenenfalls einem Datennamen und einer Folge von unabhängigen Klauseln. Eine Datensatzbeschreibung hat einen hierarchischen Aufbau. Daher können die bei einer Eintragung verwendeten Klauseln sehr unterschiedlich sein. Das hängt davon ab, ob untergeordnete Eintragungen nachfolgen oder nicht. Der Aufbau einer Datensatzbeschreibung wird im Abschnitt "Stufenkonzept" im Kapitel 2 definiert. Die in einer Datensatzbeschreibung zulässigen Elemente werden im Kapitel 2 "Dateibeschreibung - vollständiges Eintragungsschema" beschrieben.

Dateibeschreibung - vollständiges Eintragungsschema

Funktion

Die Dateibeschreibung bestimmt den physikalischen Aufbau, die Bezeichnung und die Datensatznamen einer Datei.

Allgemeines Format

```
FD dateiname

  [; BLOCK CONTAINS [ganzzahl-1 TO] ganzzahl-2 {RECORDS    }]
                                               {CHARACTERS}

   [; RECORD CONTAINS [ganzzahl-3 TO] ganzzahl-4 CHARACTERS]

   ; LABEL {RECORD IS  } {STANDARD}
           {RECORDS ARE} {OMITTED }

  [; VALUE OF datenname-1 IS {datenname-2}
                             {literal-1  }

    [ , datenname-3 IS {datenname-4}]...]
                       {literal-2  }

  [; DATA {RECORD IS  } datenname-5 [, datenname-6] ... ]
          {RECORDS ARE}

  [; LINAGE IS {datenname-7} LINES [, WITH FOOTING AT {datenname-8}]
               {ganzzahl-5 }                          {ganzzahl-6 }

    [, LINES AT TOP {datenname-9} ] [, LINES AT BOTTOM {datenname-10} ] ]
                    {ganzzahl-7 }                      {ganzzahl-8  }

  [; CODE-SET IS alphabetname. ]
```

Syntaxregeln

1. Die Stufenbezeichnung FD legt den Anfang einer Dateibeschreibung fest und muß vor dem Dateinamen stehen.
2. Die dem Dateinamen folgenden Klauseln sind in den meisten Fällen wahlweise. Die Reihenfolge ihres Auftretens ist beliebig.
3. Eine Dateibeschreibung muß von einer Datensatzerklärung oder von mehreren Datensatzbeschreibungen gefolgt sein.

BLOCK CONTAINS-Klausel

Funktion

Die BLOCK CONTAINS-Klausel gibt die Größe eines physischen Datensatzes an.

Allgemeines Format

```
BLOCK CONTAINS [ganzzahl-1 TO] ganzzahl-2 {RECORDS    }
                                          {CHARACTERS }
```

Allgemeine Regel

Diese Klausel ist nur zur Dokumentation erforderlich.

CODE-SET-Klausel

Funktion

Die CODE-SET-Klausel definiert die Zeichencodevereinbarungen, in denen Daten auf externen Geräten dargestellt werden.

Allgemeines Format

```
CODE-SET IS alphabetname
```

Syntaxregeln

1. Ist die CODE-SET-Klausel für eine Datei angegeben, müssen in dieser Datei alle Daten mit USAGE IS DISPLAY und alle mit Vorzeichen versehenen numerischen Daten mit SIGN IS SEPARATE beschrieben sein.
2. Die alphabetname zugeordnete ALPHABET-Klausel darf nicht die Literal-Angabe enthalten.
3. Die CODE-SET-Klausel darf nicht für Plattenspeicherdateien angegeben werden.

Allgemeine Regel

Die CODE-SET-Klausel wird nur zur Dokumentation angegeben.

DATA RECORDS-Klausel

Funktion

Die DATA RECORDS-Klausel dient nur zur Dokumentation der Namen von Datensätzen und der zugehörigen Dateien.

Allgemeines Format

```
DATA { RECORD IS  } datenname-1 [, datenname-2] ...
     { RECORDS ARE}
```

Syntaxregel

datenname-1, datenname-2,... bezeichnen Datensätze. Jeder dieser Datensätze muß eine zugehörige Datensatzbeschreibung auf der Stufennummer 01 mit dem gleichen Namen haben.

Allgemeine Regeln

1. Das Vorhandensein von mehr als einem Datennamen zeigt an, daß die Datei mehr als eine Art von Datensätzen enthält. Diese Datensätze können von verschiedener Größe, verschiedenem Format usw. sein. Die Reihenfolge ihrer Auflistung ist ohne Bedeutung.
2. Konzeptionell teilen sich alle Datensätze in einer Datei den gleichen Bereich. Dies wird auch dadurch nicht geändert, wenn mehr als eine Art von Datensätzen in der Datei vorhanden ist.

LABEL RECORDS-Klausel

Funktion

Die LABEL RECORDS-Klausel gibt an, ob Kennsätze vorhanden sind.

Allgemeines Format

```
LABEL { RECORD IS  } { STANDARD }
      { RECORDS ARE} { OMITTED  }
```

Syntaxregel

Diese Klausel ist dann für eine Dateibeschreibung erforderlich, wenn der ANSI-Schalter gesetzt ist.

Allgemeine Regel

Diese Klausel wird ansonsten nur zur Dokumentation verwendet.

LINAGE-Klausel

Funktion

Die LINAGE-Klausel bietet ein Hilfsmittel, die Größe einer logischen Seite in Anzahl von Zeilen anzugeben. Außerdem kann der Abstand vom oberen zum unteren Rand der logischen Seite festgelegt sowie die Zeile innerhalb des Seitenrumpfes angegeben werden, in der der Fußteil beginnen soll.

Allgemeines Format

```
LINAGE IS {datenname-1  } LINES [,WITH FOOTING AT {datenname-2  }]
          {ganzzahl-1   }                         {ganzzahl-2   }

[,LINES AT TOP {datenname-3  }][,LINES AT BOTTOM {datenname-4  }]
               {ganzzahl-3   }                   {ganzzahl-4   }
```

Syntaxregeln

1. datenname-1, datenname-2, datenname-3 bzw. datenname-4 müssen ganzzahlige numerische Datenelemente ohne Vorzeichen sein.
2. Der Wert von ganzzahl-1 muß größer als Null sein.
3. Der Wert von ganzzahl-2 darf nicht größer als der von ganzzahl-1 sein.
4. Der Wert von ganzzahl-3 bzw. ganzzahl-4 kann Null sein.

Allgemeine Regeln

1. Mit der LINAGE-Klausel kann man die Größe einer logischen Seite in Form der Zeilenanzahl bestimmen. Die logische Seitengröße ergibt sich aus der Summe aller Angaben aus der LINAGE-Klausel mit Ausnahme der FOOTING-Angabe wenn LINES AT TOP oder LINES AT BOTTOM nicht angegeben ist, sind die Werte dieser Funktionen Null. Ist FOOTING nicht angegeben, wird hierfür der Wert von ganzzahl-1 bzw. datenname-1 angenommen.

   Die Größe einer logischen Seite muß nicht der Größe einer physischen Seite entsprechen.

2. Der Wert von ganzzahl-1 oder des Datenfeldes, auf das sich datenname-1 bezieht, gibt die Anzahl der Zeilen an, die auf einer logischen Seite beschrieben werden, bzw. die Leerzeichen enthalten können. Der Wert muß größer als Null sein. Der Teil der logischen Seite, in dem diese Zeilen beschrieben und/oder mit Leerzeichen aufgefüllt werden können, wird Seitenrumpf genannt.

3. Der Wert von ganzzahl-3 oder des Datenfeldes, auf das sich datenname-3 bezieht, bestimmt die Anzahl der Zeilen, die den oberen Rand einer logischen Seite darstellen. Dieser Bereich wird nicht beschrieben. Der Wert kann Null sein.

4. Der Wert von ganzzahl-4 oder des Datenfeldes, auf das sich datenname-4 bezieht, bestimmt die Anzahl der Zeilen, die den unteren Rand einer logischen Seite darstellen. Dieser Bereich wird nicht beschrieben.

5. Der Wert von ganzzahl-2 oder des Datenfeldes, auf das sich datenname-2 bezieht, bestimmt die Zeilennummer innerhalb des Seitenrumpfes, an der der Seitenfuß (s.u.) beginnt. Der Wert muß größer als Null, aber nicht größer als der Wert von ganzzahl-1 oder des durch datenname-1 benannten Datenfeldes sein.

   Als Seitenfuß wird derjenige Teil der logischen Seite bezeichnet, der zwischen der in ganzzahl-2 bzw. datenname-2 angegebenen Zeilennummer liegt und der Zeile einschließlich, die in ganzzahl-1 bzw. datenname-1 angegeben ist.

6. Die Werte von ganzzahl-1, ganzzahl-3, ganzzahl-4, falls angegeben, werden zur Ausführungszeit einer OPEN-Anweisung mit OUTPUT-Angabe dazu verwendet, die jeweilige Zeilenzahl für jeden der genannten Bereiche innerhalb einer logischen Seite festzulegen. Gleichzeitig wird anhand des Wertes von ganzzahl-2, falls angegeben, der Seitenfuß bestimmt. Diese Werte werden während der Ausführung des Programms für alle logischen Seiten verwendet, die für die Datei beschrieben werden sollen.

7. Die Werte der Datenfelder, falls angegeben, auf die sich datenname-1, datenname-3 und datenname-4, beziehen, werden wie folgt verwendet:

   a) Sie werden zur Ausführungszeit einer OPEN-Anweisung mit OUTPUT-Angabe dazu verwendet, die jeweilige Zeilenanzahl für jeden der genannten Bereiche innerhalb der ersten logischen Seite festzulegen.

   b) Sie werden zur Ausführungszeit einer WRITE-Anweisung mit einer ADVANCING PAGE-Angabe oder bei einer Seitenüberlaufbedingung (vgl. WRITE-Anweisung) dazu verwendet, die jeweilige Zeilenanzahl für jeden der genannten Bereiche in der nächsten logischen Seite festzulegen.

8. Der Wert des Datenfeldes, auf das sich datenname-2, sofern angegeben, bezieht, wird zur Ausführungszeit einer OPEN-Anweisung mit der OUTPUT-Angabe für eine Datei dazu verwendet, den Seitenfuß der ersten logischen Seite zu definieren. Während der Ausführung einer WRITE-Anweisung mit ADVANCING PAGE-Angabe oder wenn eine Seitenüberlaufbedingung auftritt, wird dieser Wert dazu verwendet, den Seitenfuß für die nächste logische Seite zu definieren.

9. Der LINAGE-COUNTER enthält als Wert die Zeilennummer, auf die das Ausgabegerät innerhalb des aktuellen Seitenrumpfes positioniert ist. Die Regeln für den LINAGE-COUNTER sind:

   a) Für jede Datei, die in der FILE SECTION beschrieben ist und deren Dateibeschreibung eine LINAGE-Klausel enthält, wird ein separater LINAGE-COUNTER generiert.

   b) Der LINAGE-COUNTER kann durch Anweisungen der PROCEDURE DIVISION angesprochen, aber nicht verändert werden. In einem Programm können mehrere LINAGE-COUNTER vorhanden sein. Ist dies der Fall, muß der Anwender die LINAGE-COUNTER durch Dateinamen kennzeichnen.

   c) Der LINAGE-COUNTER wird entsprechend den folgenden Regeln automatisch während der Ausführung einer WRITE-Anweisung für die zugehörige Datei verändert:

      - bei Angabe ADVANCING PAGE in einer WRITE-Anweisung wird der LINAGE-COUNTER auf den Wert 1 gesetzt,
      - bei Angabe ADVANCING ganzzahl oder ADVANCING bezeichner-2 in einer WRITE-Anweisung wird der LINAGE-COUNTER jeweils um den Wert erhöht, den ganzzahl bzw. das Datenfeld enthält, auf das sich bezeichner-2 bezieht,
      - fehlt ADVANCING in einer WRITE-Anweisung, wird der LINAGE-COUNTER um 1 erhöht (vgl. WRITE-Anweisung),
      - der Wert des LINAGE-COUNTER wird auf 1 gesetzt, wenn das Ausgabegerät auf die erste Zeile einer nachfolgenden logischen Seite, auf die geschrieben werden kann, positioniert wird (vgl. WRITE-Anweisung).

   d) Bei Ausführung einer OPEN-Anweisung für eine Datei, wird der zugehörige LINAGE-COUNTER automatisch auf den Wert 1 gesetzt.

10. Jede logische Seite grenzt ohne Zwischenräume direkt an die nächste.

RECORD CONTAINS-Klausel

Funktion

Die RECORD CONTAINS-Klausel gibt die Größe von Datensätzen an.

Allgemeines Format

```
RECORD CONTAINS [ganzzahl-1 TO] ganzzahl-2 CHARACTERS
```

Allgemeine Regel

Die Größe eines jeden Datensatzes wird vollständig in der Datensatzerklärung definiert, so daß diese Klausel niemals erforderlich wird. Die RECORD CONTAINS-Klausel wird nur zur Dokumentation angegeben.

VALUE OF-Klausel

Funktion

Die VALUE OF-Klausel dokumentiert den Inhalt eines Datenfeldes mit den jeweiligen Datei-Kennsätzen.

Allgemeines Format

```
VALUE OF datenname-1 IS { datenname-2 }
                        { literal-1   }

        [ , datenname-3 IS { datenname-4 } ] ...
                           { literal-2   }
```

Syntaxregeln

1. datenname-2, datenname-4 usw. können gekennzeichnet, dürfen jedoch nicht subskribiert oder indiziert sein. Sie dürfen keine Datenfelder sein, die mit USAGE IS INDEXED beschrieben sind.
2. datenname-2, datenname-4 usw. müssen in der WORKING-STORAGE SECTION enthalten sein.

Allgemeine Regeln

1. Diese Klausel wird nur zur Dokumentation verwendet.
2. Bei Eingabe wird datenname-1 mit datenname-2 bzw. literal-1 verglichen. Ebenso wird datenname-3 mit datenname-4 bzw. literal-2 verglichen usw.

   Bei Ausgabe wird datenname-1 durch datenname-2 bzw. literal-1 ersetzt. Ebenso wird datenname-3 durch datenname-4 bzw. literal-2 ersetzt usw.
3. Im obigen Format kann jedes genannte Literal durch eine figurative Konstante ersetzt werden.

4.5 PROCEDURE DIVISION im Ein-Ausgabemodul

CLOSE-Anweisung

Funktion

Die CLOSE-Anweisung beendet die Verarbeitung von Bändern (REEL), Platten (UNIT) und Dateien, wobei wahlweise Rückspul- und/oder Sperrfunktionen durchgeführt werden.
Beachten Sie bitte, daß Bandverarbeitung bei LII COBOL nicht unterstützt wird.

Allgemeines Format

```
CLOSE dateiname-1 [ [ {REEL  } ] [WITH NO REWIND ] ]
                  [ [ {UNIT  } ] [FOR REMOVAL    ] ]
                  [                                  ]
                  [   WITH  {NO REWIND}             ]
                  [         {LOCK     }             ]

    [ ,dateiname-2 [ [ {REEL  } ] [WITH NO REWIND ] ] ]
                   [ [ {UNIT  } ] [FOR REMOVAL    ] ]
                   [                                  ]
                   [   WITH  {NO REWIND}             ]
                   [         {LOCK     }             ]  ...
```

Syntaxregeln

1. Die Angaben REEL, UNIT, WITH NO REWIND, FOR REMOVAL werden syntaktisch überprüft.
   Sie werden nur zur Dokumentation angegeben.
2. Die in der CLOSE-Anweisung angegebenen Dateien brauchen nicht alle die gleiche Organisation oder die gleiche Zugriffsart zu haben.

Allgemeine Regeln

Falls nichts anderes angegeben, sind die Ausdücke REEL (Band) und UNIT (Platte) gleichbedeutend und innerhalb der CLOSE-Anweisung austauschbar. Die Behandlung von sequentiellen Plattenspeicherdateien ist logisch gleichzusetzen mit der Behandlung von Dateien auf Band oder ähnlichen sequentiellen Medien.

1. Eine CLOSE-Anweisung kann nur für eine Datei im OPEN-Modus ausgeführt werden.

2. Um die Wirkung von verschiedenen CLOSE-Anweisungen angewandt auf verschiedene Speichermedien zu zeigen, werden alle Ein-Ausgabedateien in die folgenden Kategorien eingeteilt:

   a) Einheitsdatensatzdatei: Eine dem Ein- oder Ausgabegerät zugewiesene Datei, für die die Begriffe REWIND, REEL und UNIT keine Bedeutung haben.

   b) Sequentielle Eindatenträgerdatei: Eine sequentielle Datei, die vollkommen auf einem Datenträger enthalten ist. Auf diesem Datenträger können sich mehrere Dateien befinden.

   c) Sequentielle Mehrdatenträgerdatei: Eine sequentielle Datei, deren Daten auf mehr als einem Datenträger stehen.

3. Falls eine Datei während der Ausführung einer STOP RUN-Anweisung noch eröffnet ist, wird sie geschlossen!

   Ist eine Datei in einem aufgerufenen Programm eröffnet, wird sie während der CANCEL-Ausführung geschlossen!

4. Ist im FILE-CONTROL-Paragraph der ENVIRONMENT DIVISION für eine Datei OPTIONAL angegeben, und die Datei ist nicht vorhanden, werden die Standardabschlußroutinen für diese Datei nicht durchlaufen.

5. Falls eine CLOSE-Anweisung ohne die Angabe REEL oder UNIT für eine Datei ausgeführt wurde, muß als nächste Ein-Ausgabeanweisung eine OPEN-Anweisung durchlaufen werden. Ausnahme: SORT- oder MERGE-Anweisungen mit USING- oder GIVING-Angabe.

6. Die Angaben WITH NO REWIND und FOR REMOVAL haben zur Programmausführungszeit keine Auswirkung, sofern sie nicht für die Geräte gelten, auf denen die Datei gespeichert ist.
7. Nach erfolgreicher Ausführung einer CLOSE-Anweisung steht der der Datei zugeordnete Datensatzbereich nicht mehr zur Verfügung. Durch die erfolglose Ausführung einer solchen CLOSE-Anweisung ist der Zustand des Datensatzbereiches undefiniert.
8. Wenn WITH LOCK angegeben ist, kann die Datei während der aktuellen Programmausführung nicht wieder geöffnet werden.

OPEN-Anweisung

Funktion

Die OPEN-Anweisung eröffnet Dateien für die Verarbeitung. Sie führt Ein-Ausgabeoperationen durch.

Allgemeines Format

```
      ⎡                          ⎡REVERSED      ⎤         ⎤
      ⎢ INPUT dateiname-1        ⎢              ⎥         ⎥
      ⎢                          ⎣WITH NO REWIND⎦         ⎥
      ⎢                          ⎡REVERSED      ⎤         ⎥
      ⎢     [,dateiname-2        ⎢              ⎥ ...     ⎥
      ⎢                          ⎣WITH NO REWIND⎦         ⎥
OPEN  ⎨                                                   ⎬ ...
      ⎢ OUTPUT dateiname-3       [WITH NO REWIND]         ⎥
      ⎢     [,dateiname-4        [WITH NO REWIND]] ...    ⎥
      ⎢                                                   ⎥
      ⎢ I-O dateiname-5          [,dateiname-6] ...       ⎥
      ⎣ EXTEND dateiname-7       [,datei-name-8] ...      ⎦
```

Syntaxregeln

1. Die REVERSED- und NO REWIND-Angaben dienen der Dokumentation.

2. Die I-O-Angabe kann nur für Plattenspeicherdateien verwendet werden, jedoch nicht für Dateien mit der Angabe ORGANIZATION LINE SEQUENTIAL.

3. Die EXTEND-Angabe kann nur für sequentielle und für Dateien mit der Angabe ORGANIZATION LINE SEQUENTIAL (zeilensequentielle Dateien) angegeben werden.

4. Die EXTEND-Angabe darf nicht für ein Mehrdateienband gemacht werden (wird von LII COBOL nicht unterstützt).

5. Die in der OPEN-Anweisung angesprochenen Dateien müssen nicht alle die gleiche Organisation bzw. den gleichen Zugriff haben.

Allgemeine Regeln

1. Die erfolgreiche Ausführung einer OPEN-Anweisung bewirkt, daß die angegebene Datei verfügbar ist und sich in eröffnetem Zustand befindet.

2. Die erfolgreiche Ausführung einer OPEN-Anweisung macht den zugehörigen Datensatzbereich für das Programm verfügbar.

3. Vor der erfolgreichen Ausführung einer OPEN-Anweisung für eine vorgegebene Datei kann keine Anweisung (mit Ausnahme einer SORT- oder MERGE-Anweisung mit USING oder GIVING Angaben) ausgeführt werden, die diese Datei entweder explizit oder implizit anspricht.

4. Eine OPEN-Anweisung muß vor der Ausführung irgendeiner der zulässigen Ein-Ausgabeanweisungen erfolgreich ausgeführt werden. In Tabelle 4-1 zeigt ein 'X' am Schnittpunkt, daß die angegebene Anweisung, die im sequentiellen Zugriffsmodus verwendet wird, mit der sequentiellen Dateiorganisation und dem OPEN-Modus verwendet werden kann, der im Kopf der Spalten vorgegeben ist.

Anweisung	Eröffnungsmodus			
	INPUT	OUTPUT	I-O *	EXTEND
READ	X		X	
WRITE		X		X
REWRITE			X	

Tabelle 4-1 Zulässige Kombinationen von Ein-Ausgabeanweisungen für jeden OPEN-Modus.

* Dieser OPEN-Modus wird nicht unterstützt, wenn ORGANIZATION LINE SEQUENTIAL angegeben ist.

5. Eine Datei kann mit den Angaben INPUT, OUTPUT, EXTEND und I-O im selben Programm eröffnet werden.
   Nach der logisch ersten Ausführung einer OPEN-Anweisung für eine Datei muß vor jeder Ausführung einer nachfolgenden OPEN-Anweisung für dieselbe Datei eine CLOSE-Anweisung durchlaufen worden sein.

6. Die Ausführung der OPEN-Anweisung **bewirkt nicht**, daß der erste Datensatz zur Verfügung steht oder übergeben wird.

7. Der durch die ASSIGN-Klausel in der SELECT-Klausel angegebene Name für eine Datei wird wie folgt verarbeitet:

   a) INPUT bedeutet, daß die Datei als Eingabedatei verarbeitet werden soll. Ist INPUT angegeben, verursacht die Ausführung einer OPEN-Anweisung, daß der in der ASSIGN-Klausel angegebene Name entsprechend den Betriebssystemkonventionen für das Eröffnen von Eingabedateien überprüft wird.

   b) OUTPUT bedeutet, daß die Datei als Ausgabedatei verarbeitet werden soll. Ist OUTPUT angegeben, verursacht die OPEN-Anweisung, daß der in der ASSIGN-Klausel angegebene Name entsprechend den Betriebssystemkonventionen für das Eröffnen von Ausgabedateien geschrieben wird.

8. Die Dateibeschreibung für dateiname-1, dateiname-2, dateiname-5, dateiname-6, dateiname-7 bzw. dateiname-8 muß gleich zu der sein, die bei Erstellung der Datei verwendet wurde.

9. Falls für eine Eingabedatei die Angabe OPTIONAL in der SELECT-Klausel gemacht wurde, wird beim Ablauf des Zielprogramms geprüft, ob die Datei vorhanden ist. Existiert die Datei nicht, bewirkt die Ausführung der ersten READ-Anweisung, daß die AT END-Bedingung durchlaufen wird.

10. Wenn das Speichermedium für die Datei Rückspulen zuläßt, gelten die folgenden Regeln:

    a) Die Datei wird bei Ausführung einer OPEN-Anweisung auf den Dateianfang positioniert.

    b) Ist REVERSED angegeben, wird bei Ausführung einer OPEN-Anweisung an das Dateiende positioniert.

11. Ist REVERSED angegeben, stellen die nachfolgenden READ-Anweisungen für die Datei die Daten in umgekehrter Reihenfolge zur Verfügung, d.h., beginnend mit dem letzten Datensatz.

12. Bei Dateien, die mit INPUT oder I-O eröffnet wurden, verursacht die OPEN-Anweisung, daß der aktuelle Satzzeiger auf den ersten Datensatz der Datei zeigt. Wenn keine Datensätze in der Datei vorhanden sind, wird der aktuelle Satzzeiger so gesetzt, daß die nächste ausgeführte READ-Anweisung für die Datei eine AT END-Bedingung zur Folge hat. Wenn die Datei nicht vorhanden ist, verursacht OPEN INPUT eine Fehleranzeige.

13. Ist EXTEND angegeben, positioniert die OPEN-Anweisung die Datei unmittelbar nach dem letzten logischen Datensatz dieser Datei. Nachfolgende WRITE-Anweisungen, die diese Datei ansprechen, führen dazu, daß Datensätze an diese Datei so angefügt werden, als ob die Datei mit OUTPUT eröffnet worden wäre. Wenn die Datei nicht vorhanden ist, wird sie erstellt.

14. Ist EXTEND angegeben und zeigt die LABEL RECORDS-Klausel an, daß Kennsätze vorhanden sind, werden bei der Ausführung der OPEN-Anweisung folgende Schritte durchlaufen:

    a) Die Dateianfangskennsätze werden nur im Falle einer Eindatenträgerdatei verarbeitet.

    b) Die Dateianfangskennsätze auf der letzten vorhandenen Platte werden so verarbeitet, als ob die Datei mit der INPUT-Angabe eröffnet worden wäre.

    c) Die vorhandenen Dateiendekennsätze werden so verarbeitet, als ob die Datei mit der INPUT-Angabe eröffnet worden wäre. Diese Kennsätze werden dann gelöscht.

    d) Die Verarbeitung wird dann fortgesetzt, als ob die Datei mit der OUTPUT-Angabe eröffnet worden wäre.

15. Die I-O-Angabe erlaubt das Eröffnen einer Plattenspeicherdatei für Eingabe- und Ausgabeoperationen. Ausnahme: Es handelt sich um eine Datei mit ORGANIZATION LINE SEQUENTIAL.
    Ist die Datei nicht vorhanden, wird sie erstellt und als leere Datei für Eingabeoperationen verwendet. Der Versuch eines Schreibzugriffs führt zu einem Fehler.

16. Ist I-O angegeben und zeigt die LABEL RECORDS-Klausel an, daß Kenndatensätze vorhanden sind, werden bei der Ausführung der OPEN-Anweisung folgende Schritte durchlaufen:

    a) Die Kennsätze werden entsprechend den Betriebssystemkonventionen für die Ein-Ausgabe von Kennsätzen überprüft.

    b) Die neuen Kennsätze werden entsprechend den Betriebssystemkonventionen für das Schreiben von Ein-Ausgabe-Kennsätzen beschrieben.

17. Bei erfolgreicher Ausführung einer OPEN-Anweisung mit der OUTPUT-Angabe wird eine Datei erstellt. Zu diesem Zeitpunkt enthält die zugehörige Datei keine Datensätze. Existiert schon eine Datei mit dem gleichen Namen, wird sie gelöscht. Ist diese schreibgeschützt, tritt ein Fehler auf.

READ-Anweisung

Funktion

Die READ-Anweisung stellt den nächsten logischen Datensatz einer Datei zur Verfügung.

Allgemeines Format

```
READ dateiname RECORD [INTO bezeichner]
        [;AT END unbedingte-anweisung]
```

Syntaxregeln

1. Die INTO-Angabe darf nicht verwendet werden, wenn die Datensatzbeschreibung für die Eingabedatei variable logische Datensätze enthält. Der durch bezeichner angegebene Speicherbereich darf nicht derselbe sein, wie der durch dateiname angegebene Datensatzbereich.
2. Ist für dateiname keine USE-Prozedur angegeben, muß die AT END-Bedingung angegeben werden.

Allgemeine Regeln

1. Bevor eine READ-Anweisung für eine Datei durchgeführt werden kann, muß diese Datei mit INPUT oder I-O eröffnet worden sein.
2. Der durch die READ-Anweisung zur Verfügung gestellte Datensatz wird wie folgt bestimmt:
   a) nach der OPEN-Anweisung wird der Datensatz zur Verfügung gestellt, auf den der aktuelle Satzzeiger zeigt.
   b) Wurde der aktuelle Satzzeiger bei der Ausführung einer READ-Anweisung positioniert, wird er so aktualisiert, daß er auf den nächsten vorhandenen Datensatz zeigt. Dieser Datensatz wird dann zur Verfügung gestellt.
3. Die Ausführung einer READ-Anweisung bewirkt, daß der Inhalt des Datenfeldes fortgeschrieben wird, das in der FILE STATUS-Klausel der zugehörigen Dateibeschreibung angegeben wurde.

4. Ein Datensatz ist solange für das Zielprogramm verfügbar, bis eine andere der READ-Anweisung folgende Anweisung ausgeführt wird.

5. Wenn eine Datei mehr als einen logischen Datensatztyp enthält, teilen sich diese Datensätze automatisch den gleichen Speicherbereich. Dies entspricht einer impliziten Redefinition des Speicherbereiches. Der Inhalt von Datenfeldern, die außerhalb des Bereiches des aktuellen Datensatzes liegen, sind bei Abschluß der READ-Anweisung undefiniert.

6. Ist INTO angegeben, wird der gelesene Datensatz aus dem Datensatzbereich in den durch bezeichner angegebenen Bereich übertragen (impliziter MOVE). Die Übertragung findet entsprechend den Regeln für die MOVE-Anweisung ohne CORRESPONDING-Angabe statt. Der implizite MOVE wird nicht ausgeführt, wenn die Ausführung der READ-Anweisung erfolglos war. Jede mit bezeichner zusammenhängende Subskribierung oder Indizierung wird ausgewertet, nachdem der Datensatz gelesen wurde und unmittelbar bevor er in das Datenfeld übertragen wird.

7. Ist INTO angegeben, steht der gelesene Datensatz sowohl im Eingabedatenbereich als auch in dem durch bezeichner festgelegten Bereich zur Verfügung.

8. Wenn bei Ausführung einer READ-Anweisung die Position des aktuellen Satzzeigers für diese Datei undefiniert ist, so ist die Ausführung dieser READ-Anweisung erfolglos.

9. Wird das Ende einer Platte während der Ausführung einer READ-Anweisung erkannt, tritt die Dateiendebedingung auf.

   Es folgt:

   a) Die Standarddatenträger-Endekennsatzprozedur

   b) Ein Datenträgerwechsel

   c) Die Standarddatenträger-Anfangskennsatzprozedur

   d) Der erste Datensatz des neuen Datenträgers wird zur Verfügung gestellt.

10. Ist eine Datei, die mit der OPTIONAL-Angabe in der SELECT-Klausel beschrieben ist, zum Eröffnungszeitpunkt nicht vorhanden, tritt bei der Ausführung der ersten READ-Anweisung für die Datei die AT END-Bedingung auf. Die Ausführung der READ-Anweisung ist erfolglos. Die Standard-Dateiendeprozeduren werden nicht durchlaufen Die Ausführung des Programms wird dann nach den unter Punkt 12. beschriebenen Regeln durchgeführt.

11. Ist bei Ausführung einer READ-Anweisung kein nächster logischer Datensatz in der Datei vorhanden, tritt die AT END-Bedingung auf. Die Ausführung der READ-Anweisung wird als erfolglos betrachtet.

12. Wird die AT END-Bedingung erkannt, laufen die folgenden Schritte in der angegebenen Reihenfolge ab.

    a) Der Inhalt des Datenfeldes aus der FILE STATUS-Klausel, falls für diese Datei angegeben, wird mit einem Wert versorgt, der die AT END-Bedingung anzeigt.

    b) Falls in der READ-Anweisung AT END angegeben ist, wird auf die unbedingte Anweisung nach der AT END-Angabe verzweigt. Eine für dieses Datei vorhandene USE-Prozedur wird nicht durchlaufen.

    c) Ist AT END in der READ-Anweisung nicht angegeben, muß entweder explizit oder implizit eine USE-Prozedur für diese Datei vorliegen. Diese USE-Prozedur wird dann durchlaufen, wenn eine Endebedingung auftritt.

    Bei Auftreten der AT END-Bedingung ist die entsprechende Ein-Ausgabeanweisung erfolglos.

13. Nach der erfolglosen Ausführung einer READ-Anweisung sind der Inhalt des zur Datei gehörigen Datensatzbereiches und die Position des aktuellen Satzzeigers undefiniert.

14. Nach Auftreten der AT END-Bedingung darf für diese Datei keine READ-Anweisung gegeben werden, bevor nicht eine erfolgreiche CLOSE-Anweisung, gefolgt von einer erfolgreichen OPEN-Anweisung für diese Datei durchgeführt wurde.

REWRITE-ANWEISUNG

Funktion

Die REWRITE-Anweisung ersetzt einen Datensatz logisch, der in einer Plattenspeicherdatei steht.

Allgemeines Format

```
REWRITE datensatzname [FROM bezeichner]
```

Syntaxregeln

1. datensatzname und bezeichner dürfen sich nicht auf den gleichen Speicherbereich beziehen.
2. datensatzname ist der Name eines logischen Datensatzes in der FILE SECTION der DATA DIVISION und kann gekennzeichnet werden.

Allgemeine Regeln

1. Die mit datensatzname verknüpfte Datei muß eine Plattenspeicherdatei sein und muß bei Ausführung der REWRITE-Anweisung im I-O-Modus eröffnet worden sein (vgl. OPEN-Anweisung).
2. Die letzte Ein-Ausgabe-Anweisung, die vor der Ausführung der REWRITE-Anweisung für die zugehörige Datei ausgeführt wurde, muß eine erfolgreich ausgeführte READ-Anweisung sein. Das Betriebssystem ersetzt den Datensatz logisch, auf den durch die READ-Anweisung zugegriffen wurde.
3. Die Anzahl der Zeichenpositionen in dem durch datensatzname benannten Datensatz muß gleich der Anzahl der Zeichenpositionen des zu ersetzenden Datensatzes sein.

4. Der logische Datensatz, der durch eine erfolgreich abgelaufene REWRITE-Anweisung zurückgeschrieben wurde, steht im Datensatzbereich nicht mehr zur Verfügung. Eine Ausnahme stellt die Verwendung der SAME RECORD AREA-Klausel dar. In diesem Fall steht der Datensatz sowohl allen anderen Dateien, die in der SAME RECORD AREA-Klausel aufgeführt sind, als auch der gerade bearbeiteten Datei als Datensatz zur Verfügung.

5. Die Ausführung einer REWRITE-Anweisung mit der FROM-Angabe entspricht den folgenden Anweisungen:

   MOVE bezeichner TO datensatzname
   REWRITE datensatzname.

   Der Inhalt des Datensatzbereiches vor Ausführung der impliziten MOVE-Anweisung hat keine Auswirkung auf die Ausführung REWRITE-Anweisung.

6. Der aktuelle Satzzeiger wird durch die Ausführung einer REWRITE-Anweisung nicht beeinflußt.

7. Die Ausführung der REWRITE-Anweisung bewirkt, daß der Inhalt des Datenfeldes aktualisiert wird, das in der FILE STATUS-Klausel der zugehörigen Dateibeschreibung angegeben wurden.

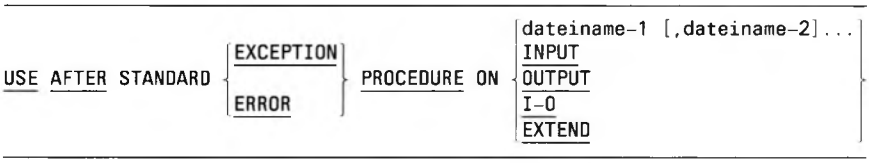
8. Die REWRITE-Anweisung kann nicht für Dateien verwendet werden, die mit ORGANIZATION LINE SEQUENTIAL beschrieben wurde.

USE-Anweisung

Funktion

Die USE-Anweisung gibt zusätzlich zu den Standardprozeduren der Ein-Ausgabe-Prozeduren für die Bearbeitung von Ein-Ausgabefehlern an.

Allgemeines Format



Syntaxregeln

1. Ist die USE-Anweisung angegeben, muß sie innerhalb der Prozedurvereinbarungen DECLARATIVES) stehen. Der erste Eintrag in den Prozedurvereinbarungen ist die Kapitelüberschrift und zwar: Kapitelname, gefolgt von dem Wort SECTION und einem Punkt. Unmittelbar anschließend an die Kapitelüberschrift muß die USE-Anweisung geschrieben werden. Danach können Paragraphen folgen, die die Prozedur definieren.
2. Die USE-Anweisung selbst wird nicht ausgeführt. Sie vereinbart Prozeduren, die durchlaufen werden sollen, wenn ein Ein-Ausgabefehler für eine Datei auftritt.
3. Da das Format variiert werden kann, kann in verschiedenen USE-Anweisungen der gleiche Dateiname auftreten. Dies darf jedoch nicht den gleichzeitigen Aufruf mehrerer Prozedurvereinbarungen bewirken.
4. Die Wörter ERROR und EXCEPTION sind gleichwertig und können alternativ verwendet werden.
5. Die Dateien, die implizit angesprochen werden (INPUT, OUTPUT, I-O, EXTEND) oder explizit (dateiname-1, dateiname-2,...) brauchen nicht dieselbe Organisation und denselben Zugriff zu haben.

Allgemeine Regeln

1. Die USE-Prozeduren werden durchlaufen:

   a) Bei Auftreten einer AT END-Bedingung, wenn die Ein-Ausgabe-Anweisung, bei der die AT END-Bedingung auftrat, keine AT END-Angabe enthält.

   b) Bei Auftreten eines schwerwiegenden Fehlers.

   Treffen a) oder b) zu und fehlt eine entsprechende USE-Prozedur für die Datei, führt das zu Programmabbruch.

2. Nach Ausführung einer USE-Prozedur wird die Steuerung an die aufrufende Routine zurückgegeben.

3. Innerhalb einer USE-Prozedur darf eine Bezugnahme auf Prozeduren außerhalb der Prozedurvereinbarungen (DECLARATIVES) nur mit der PERFORM-Anweisung enthalten sein.
   Auf Prozedurnamen, die einer USE-Anweisung untergeordnet sind, darf aus einer anderen Prozedur oder außerhalb der Prozedurvereinbarungen (DECLARATIVES) nur mit einer PERFORM-Anweisung Bezug genommen werden.

4. Innerhalb einer USE-Prozedur darf keine Anweisung ausgeführt werden, die eine USE-Prozedur aufrufen würde, die ihrerseits die Steuerung noch nicht an die aufrufende Routine zurückgegeben hatte.

WRITE-Anweisung

Funktion

Die WRITE-Anweisung stellt einen logischen Datensatz für eine Ausgabedatei zur Verfügung. Sie kann ebenso für den Zeilenvorschub innerhalb einer logischen Seite verwendet werden.

Allgemeines Format

```
WRITE datensatzname [FROM bezeichner-1]

[ { BEFORE } ADVANCING { ganzzahl [ LINE ]  } ]
[ { AFTER  }           { merkname  [ LINES ] } ]
                       { PAGE                }

[ ; AT { END-OF-PAGE } unbedingte-anweisung ]
       { EOP         }
```

Syntaxregeln

1. datensatzname und bezeichner-1 dürfen nicht den gleichen Speicherbereich belegen.
2. datensatzname ist der Name eines logischen Datensatzes in der FILE SECTION der DATA DIVISION und kann gekennzeichnet werden.
3. Wird bezeichner-2 in der ADVANCING-Angabe verwendet, muß er der Name eines ganzzahligen Datenelementes sein.
4. Der Wert von ganzzahl oder des durch bezeichner-2 angesprochenen Datenfeldes kann Null sein.
5. Ist END OF PAGE angegeben, muß innerhalb der Dateibeschreibung der entsprechenden Datei eine LINAGE-Klausel vorhanden sein.

6. Die Angaben END OF PAGE und EOP sind gleichbedeutend.

Allgemeine Regeln

1. Bei Ausführung einer WRITE-Anweisung muß die Datei mit den Angaben OUTPUT oder EXTEND in der OPEN-Anweisung eröffnet worden sein. (vgl. OPEN-Anweisung).
2. Der durch eine WRITE-Anweisung ausgegebene Datensatz steht im Datensatzbereich nur dann zur Verfügung, wenn:
   - die zum Datensatz gehörende Datei in einer SAME RECORD AREA-Klausel angegeben ist, oder wenn
   - die Ausführung der WRITE-Anweisung aufgrund einer Verletzung der Dateigrenzen erfolglos war.

3. Die Ausführung einer WRITE-Anweisung mit der FROM-Angabe ist gleichbedeutend mit den folgenden Anweisungen:

   MOVE bezeichner-1 TO datensatzname
   WRITE datensatzname

   Die Übertragung findet entsprechend den Regeln für die MOVE-Anweisung ohne CORRESPONDING-Angabe statt.

   Der Inhalt des Datensatzbereichs vor Ausführung der impliziten MOVE-Anweisung hat keinen Einfluß auf den Ablauf der WRITE-Anweisung.

   Nach erfolgreicher Ausführung der WRITE-Anweisung steht die Information in bezeichner-1, auch wenn die Information in dem durch datensatzname angesprochenen Bereich nicht zur Verfügung steht.

4. Der aktuelle Satzzeiger wird durch die Ausführung einer WRITE-Anweisung nicht beeinflußt.

5. Nach Ausführung einer WRITE-Anweisung wird der Wert des Datenfeldes einer für diese Datei vorhandenen FILE STATUS-Klausel fortgeschrieben.

6. Die maximale Datensatzgröße einer Datei wird beim Erstellen der Datei festgelegt. Sie darf danach nicht mehr verändert werden.

7. Die auf einer Platte erforderliche Anzahl von Zeichenpositionen zum Speichern eines logischen Datensatzes in einer Datei muß nicht mit der für diesen Datensatz im Programm definierten Anzahl übereinstimmen.

8. Durch die Ausführung der WRITE-Anweisung wird dem Betriebssystem ein logischer Datensatz zur Verfügung gestellt.

9. Sowohl die ADVANCING- als auch die END OF PAGE-Angabe bieten die Möglichkeit, auf jede einzelne Zeile innerhalb einer zu druckenden Seite zu positionieren.

   a) Ist ORGANIZATION SEQUENTIAL angegeben und fehlt die ADVANCING-Angabe, wird ein automatischer Vorschub entsprechend der Angabe AFTER ADVANCING 1 LINE ausgeführt, wenn die Ausgabe auf einem Drucker erfolgt.
   Ist ADVANCING angegeben, erfolgt der Vorschub nach folgenden Regeln:

      I. Ist bezeichner-2 angegeben, wird um die durch den Wert von bezeichner-2 angegebene Zeilenanzahl vorgeschoben.

      II. Ist ganzzahl angegeben, wird um die durch ganzzahl angegebene Zeilenzahl vorgeschoben.

      III. Ist merkname angegeben, wird um die Zeilenanzahl vorgeschoben, die für das damit bezeichnete Gerät vorgeschrieben ist.

      IV. Ist BEFORE angegeben, wird der entsprechende Datensatz gedruckt, bevor die zu druckende Seite entsprechend den Regeln I-III vorgeschoben wird.

      V. Ist AFTER angegeben, wird der entsprechende Datensatz gedruckt, nachdem die zu druckende Seite entsprechend den Regeln I-III vorgeschoben worden ist.

      VI. Ist PAGE angegeben, wird der entsprechende Datensatz gedruckt, bevor oder nachdem das Gerät auf die nächste logische Seite positioniert wird.
      Handelt es sich um den Datensatz einer Datei, die mit LINAGE beschrieben ist, wird auf den Anfang der ersten Zeile einer neuen logischen Seite positioniert.

   b) Ist ORGANIZATION LINE SEQUENTIAL angegeben und fehlt die ADVANCING-Angabe, wird um eine Zeile vorgeschoben. Normalerweise erfolgt der Vorschub so, als wäre BEFORE ADVANCING 1 LINE angegeben worden.

      Ist ADVANCING angegeben, erfolgt der Vorschub wie unter a) I-VI beschrieben.

10. Wird bei Ausführung einer WRITE-Anweisung mit END OF PAGE-Angabe das Ende einer logischen Seite erreicht, wird der Programmlauf mit der unbedingten Anweisung, die der END-OF-PAGE-Anweisung folgt, fortgesetzt. Das logische Ende einer Seite ist durch die LINAGE-Klausel in der Dateibeschreibung der Datei definiert.

11. Eine END-OF-PAGE-Bedingung tritt auf, wenn während der Ausführung einer WRITE-Anweisung mit END-OF-PAGE-Angabe innerhalb des Seitenfußes gedruckt oder vorgeschoben wird. Dies ist dann der Fall, wenn während der Ausführung einer solchen WRITE-Anweisung der Wert des USAGE-COUNTER-Sonderregisters gleich dem oder größer als der Wert von ganzzahl-2 oder des durch bezeichner-2 angesprochenen Datenfeldes einer LINAGE-Klausel wird. In diesem Fall wird die WRITE-Anweisung ausgeführt und danach der Programmablauf bei der in der END-OF-PAGE-Angabe genannten unbedingten Anweisung fortgesetzt.

    Eine automatische Seitenüberlaufbedingung tritt immer dann auf, wenn eine WRITE-Anweisung - mit oder ohne AT-END-OF-PAGE-Angabe - nicht mehr innerhalb der aktuellen Seitengrenzen ausgeführt werden kann.

    Dies ist dann der Fall, wenn während der Ausführung einer solchen WRITE-Anweisung der Wert der LINAGE-COUNTER-Sonderregisters größer als der Wert von ganzzahl-1 oder des durch bezeichner-1 angesprochenen Datenfeldes einer LINAGE-Klausel wird. In diesem Fall wird der Datensatz gedruckt, bevor oder nachdem - abhängig von der Angabe BEFORE/AFTER- auf die erste Zeile der nächsten logischen Seite positioniert worden ist. Dies ist durch die Angabe der LINAGE-Klausel festgelegt. Die in der END-OF-PAGE-Angabe genannte unbedingte Anweisung wird durchlaufen, nachdem der Datensatz geschrieben und das Gerät neu positioniert wurde.

    Sind die Angaben ganzzahl-2 bzw. datenname-2 in der LINAGE-Klausel nicht angegeben, treten die END-OF-PAGE-Bedingung und die Seitenüberlauf-Bedingung gleichzeitig auf. Die END-OF-PAGE-Bedingung unterscheidet sich in diesem Falle nicht von der Seitenüberlaufbedingung.

Wenn ganzzahl-2 oder datenname-2 in der LINAGE-Klausel angegeben sind, die Ausführung einer WRITE-Anweisung jedoch dazu führen würde, daß der Wert des LINAGE-COUNTER-Sonderregisters sowohl

- gleich dem oder größer als der Wert von ganzzahl-2 oder des durch datenname-2 angesprochenen Datenfeldes, als auch
- gleich dem oder größer als der Wert von ganzzahl-1 oder des durch datenname-1 angesprochenen Datenfeldes

ist, läuft die Verarbeitung so ab, als ob ganzzahl-2 oder datenname-2 nicht vorhanden wären.

12. Wird bei Ausführung einer WRITE-Anweisung versucht, außerhalb der physikalischen Grenzen einer Datei zu schreiben, tritt eine Ausnahmebedingung (EXCEPTION-Bedingung) auf und die folgenden Schritte laufen ab:

    a) Ist FILE STATUS angegeben, wird der Wert des FILE STATUS-Datenfeldes der entsprechenden Datei gesetzt, um eine Verletzung der Dateigrenzen anzuzeigen (vgl. Ein-Ausgabezustand)

    b) Eine für diese Datei explizit oder implizit vorhandene USE AFTER STANDARD EXCEPTION-Prozedur wird durchlaufen.

    c) Ist für diese Datei weder explizit noch implizit eine USE AFTER STANDARD EXCEPTION-Prozedur vorhanden, ist das Ergebnis undefiniert.

13. Wird bei einer Mehrdatenträger-Ausgabedatei für Plattenspeicher das Ende des Datenträgers erkannt, werden bei Ausführung der WRITE-Anweisung folgende Aktionen ausgeführt:

    a) Es werden die Standard-Datenträger-Endekennsatzprozeduren durchlaufen.

    b) Es wird ein Datenträgerwechsel durchgeführt.

    c) Es werden die Standard-Datenträger-Anfangskennsatzprozeduren durchlaufen.

5 Ein-Ausgabe bei relativen Dateien

5.1 Einführung

Das Ein-Ausgabemodul "Relative Datei" ermöglicht es, auf Datensätze in einer Plattenspeicherdatei entweder wahlfrei oder sequentiell zugreifen. Jeder Datensatz in einer relativen Datei wird eindeutig bestimmt durch einen ganzzahligen Wert größer Null, der die Lage des Datensatzes innerhalb der logischen Reihenfolge der Datei angibt.

5.2 Sprachkonzepte

Organisation

Die relative Dateiorganisation ist nur für Plattenspeichergeräte zulässig. Eine relative Datei besteht aus Datensätzen, die durch relative Satznummern identifiziert werden. Die Datei kann man sich vorstellen als eine serielle Folge von Bereichen, von denen jeder einen logischen Datensatz enthält. Jeder dieser Bereiche wird durch eine relative Satznummer bezeichnet. Anhand dieser Nummer werden Datensätze gespeichert und aufgefunden. Zum Beispiel wird der zehnte Datensatz über die relative Satznummer 10 adressiert und liegt im zehnten Satzbereich. Dies ist unabhängig davon, ob Datensätze in den ersten bis neunten Satzbereich geschrieben wurden.

Zugriffsmodus

Beim sequentiellen Zugriffsmodus wird auf die in der Datei vorhandenen Datensätze in aufsteigender Reihenfolge der relativen Satznummer zugegriffen.

Beim wahlfreien Zugriffsmodus wird die Reihenfolge, in der auf Datensätze zugegriffen wird, durch den Programmierer gesteuert. Auf den gewünschten Datensatz kann zugegriffen werden, nachdem das in der RELATIVE KEY-Klausel angegebene Datenfeld mit dem Wert der relativen Satznummer versorgt worden ist.

Beim dynamischen Zugriffsmodus kann der Programmierer nach Wunsch vom sequentiellen Zugriffsmodus in den wahlfreien Zugriffsmodus wechseln, indem er entsprechende Formen der Ein-Ausgabe-Anweisungen verwendet.

Aktueller Satzzeiger

Der aktuelle Satzzeiger ist ein Zeiger, der benutzt wird, um den nächsten Datensatz auszuwählen. Er hat keine Bedeutung für eine im Ausgabemodus eröffnete Datei. Er kann nur durch OPEN-, START- und READ-Anweisungen neu gesetzt werden.

Ein-Ausgabezustand

Wenn in einem Dateisteuerungseintrag die FILE STATUS-Klausel angegeben ist, wird in das aus zwei Zeichen bestehende angegebene Datenfeld, während der Ausführung einer OPEN-, CLOSE-, READ-, WRITE-, REWRITE-, DELETE- oder START-Anweisung und bevor eine USE-Prozedur ausgeführt wird, ein Wert gesetzt, um dem COBOL-Programm den Zustand dieser Ein-Ausgabe-Operation anzuzeigen

Zustandsanzeiger 1

Das linke Zeichen des in der FILE STATUS-Klausel angegebenen Datenfeldes wird als Zustandsanzeiger 1 bezeichnet. Der Zustandsanzeiger 1 wird gesetzt, um die Beendigungsbedingungen anzuzeigen:

0 = Erfolgreiche Beendigung
1 = Endebedingung
2 = Schlüsselfehlerbedingung
3 = Permanenter Fehler
9 = Systemfehler

Die oben angegebenen Werte haben folgende Bedeutung:

0 = Erfolgreiche Beendignung.
Die Ein-Ausgabeanweisung wurde erfolgreich ausgeführt.

1 = Endebedingung.
Eine READ-Anweisung wurde erfolglos ausgeführt, weil versucht wurde, einen Datensatz zu lesen, obwohl kein nächster logischer Datensatz in der Datei vorhanden war.

2 = Schlüsselfehlerbedingung.
Die Ein-Ausgabeanweisung wurde aus einem der folgenden Gründe erfolglos ausgeführt:

- doppelter Schlüssel
- Datensatz nicht gefunden
- Überschreitung der Bereichsgrenzen

3 = Permanenter Fehler.
Die Ein-Ausgabeanweisung wurde aufgrund eines Ein-Ausgabefehlers erfolglos ausgeführt, z.B. Datenüberprüfung, Paritätsfehler oder Übertragungsfehler.

9 = Systemfehler.
Die Ein-Ausgabeanweisung wurde erfolglos ausgeführt, weil ein Fehler aufgetreten ist, den das Betriebssystem gesondert ausgibt. Dieser Wert wird nur dazu benutzt, eine Bedingung anzuzeigen, die nicht durch andere definierte Werte des Zustandsanzeigers 1 oder durch beschriebene Kombinationen der Werte der Zustandsanzeiger 1 und 2 angezeigt werden.

Zustandsanzeiger 2

Die rechtsbündige Zeichenposition, d.h. das zweite Zeichen des in der FILE STATUS-Klausel angegebenen Datenfeldes wird als Zustandsanzeiger 2 bezeichnet und wird zur näheren Beschreibung der Ergebnisse der Ein-Ausgabe-Operation verwendet. Es enthält einen der folgenden Werte:

- Ist bezüglich der Ein-Ausgabe-Operation keine weitere Information verfügbar, enthält der Zustandsanzeiger 2 den Wert "0".
- Enthält Zustandsanzeiger 1 den Wert 2, wird eine INVALID KEY-Bedingung angezeigt. Der Zustandsanzeiger 2 wird dann dazu verwendet, die Ursache dieser Bedingung durch folgende Werte anzugeben:

  2 = Doppelter Schlüssel.

  Es wurde versucht, einen Datensatz zu schreiben, der einen doppelten Schlüssel innerhalb der Datei erzeugt hätte.

  3 = Datensatz nicht gefunden.

  Es wurde versucht, anhand eines Schlüssels auf einen Datensatz zuzugreifen, der in der Datei nicht vorhanden war.

  4 = Überschreitung der Bereichsgrenzen.

  Es wurde versucht, außerhalb der festgelegten Bereichsgrenzen einer relativen Datei zu schreiben.

- Enthält der Zustandsanzeiger 1 den Wert 9, ist der Wert des Zustandsanzeigers 2 die Nummer der Fehlermeldung des Betriebssystems.

Zulässige Kombinationen der Zustandsanzeiger 1 und 2

In der nachfolgenden Tabelle sind die zulässigen Kombinationen der Werte der beiden Zustandsanzeiger aufgelistet. Ein X im Schnittpunkt bezeichnet eine zulässige Kombination.

Zustands-anzeiger 1	Zustandsanzeiger 2			
	keine weitere Information (0)	doppelter Schlüssel (2)	Daten-satz nicht gefunden (3)	Überschrei-tung der Bereichs-grenzen (4)
erfolgreiche Beendigung (0)	X			
Endebedingung (1)	X			
Schlüsselfehler-beendigung (2)		X	X	X
Permanenter Fehler (3)	X			
Systemfehler (9)	Nummer der Fehlermeldung des Betriebssystems			

INVALID KEY-Bedingung

Die INVALID KEY-Bedingung kann nach der Ausführung einer START-, READ-, WRITE-, REWRITE- oder DELETE-Anweisung auftreten. Einzelheiten über das Auftreten dieser Bedingung sind bei den angeführten Anweisungen angegeben.

Falls eine INVALID KEY-Bedingung erkannt wurde, werden vom Betriebssystem die nachfolgenden Maßnahmen in der angegebenen Reihenfolge durchgeführt.

1. Ist für die Datei FILE STATUS angegeben, wird ein Wert in das durch FILE STATUS bezeichnete Datenfeld übertragen, der die INVALID KEY-Bedingung anzeigt.
2. Ist INVALID KEY in der Anweisung angegeben, die die Schlüsselfehlerbedingung verursacht hat, wird die INVALID KEY folgende unbedingte Anweisung ausgeführt. Eine für diese Datei angegebene USE-Prozedur wird nicht durchlaufen.
3. Ist INVALID KEY nicht angegeben, jedoch explizit oder implizit eine USE-Prozedur für diese Datei vorhanden, wird diese durchlaufen.

Falls eine Schlüsselfehlerbedingung auftritt, war die entsprechende Ein-Ausgabeanweisung erfolglos und die Datei befindet sich im gleichen Zustand wie vor dieser Ein-Ausgabe-Anweisung.

AT END-Bedingung

Die AT END Bedingung kann nach der Ausführung einer READ-Anweisung auftreten. Einzelheiten über die Ursachen dieser Bedingung werden in der READ-Anweisung in diesem Kapitel behandelt.

5.3 ENVIRONMENT DIVISION im Ein- Ausgabemodul

INPUT-OUTPUT SECTION

FILE-CONTROL-Paragraph

Funktion

Der FILE-CONTROL Paragraph ordnet jeder Datei einen Namen zu und erlaubt die Beschreibung anderer dateibezogener Informationen (vgl. LEVEL II COBOL Bedienungsanleitung).

Allgemeines Format

```
FILE-CONTROL {dateisteuerungseintrag} ...
```

Dateisteuerungseintrag

Funktion

Der Dateisteuerungseintrag bezeichnet jede Datei mit Namen und definiert andere dateibezogene Informationen.

Allgemeines Format

```
SELECT dateiname

  ASSIGN TO { externes dateiname-literal }
            { dateibezeichner           }

            [ , { externes dateiname-literal } ] ...
                { dateibezeichner           }

  [; RESERVE ganzzahl-1 { AREA  } ]
                        { AREAS }

   ; ORGANIZATION IS RELATIVE

  [                      { SEQUENTIAL [,RELATIVE KEY IS datenname-1]  }  ]
  [; ACCESS MODE IS      { RANDOM                                     }  ]
  [                      { DYNAMIC    [,RELATIVE KEY IS datenname-1]  }  ]

   [; FILE STATUS IS datenname-2].
```

Syntaxregeln

1. Die SELECT-Klausel muß der erste Eintrag im Dateisteuerungseintrag sein. Alle Klauseln, die in der SELECT-Klausel folgen, können in beliebiger Reihenfolge geschrieben werden.

2. Jede in der DATA DIVISION beschriebene Datei darf nur einmal als Dateiname in der SELECT-Klausel angegeben werden.
   Jede im Dateisteuerungseintrag angegebene Datei muß einen Dateibeschreibungseintrag (z.B. FD) in der DATA DIVISION des Quellprogramms haben.

3. Ist die ACCESS MODE-Klausel nicht angegeben, wird ACCESS IS SEQUENTIAL angenommen.

4. datenname-2 muß in der DATA DIVISION als ein zwei Zeichen langes alphanumerisches Datenfeld definiert sein; er darf nicht in der FILE SECTION (oder in der COMMUNICATION SECTION) angegeben werden.

5. datenname-1 und datenname-2 können gekennzeichnet sein.

6. Soll auf eine relative Datei mit einer START-Anweisung Bezug genommen werden, muß für diese Datei RELATIVE KEY angegeben sein.

7. datenname-1 darf nicht in einer Datensatzbeschreibung angegeben sein, die mit dateiname verknüpft ist.

8. Das durch datenname-1 bezeichnete Datenfeld muß als Ganzzahl ohne Vorzeichen definiert sein.

9. dateibezeichner ist ein Programmiererwort, darf jedoch nicht den gleichen Namen haben wie dateiname.

Allgemeine Regeln

1. Die ASSIGN-Klausel beschreibt die Verknüpfung der durch datei-name benannten Datei mit einem Speichermedium (vgl. LII COBOL Bedienungsanleitung). Die Erstzuweisung ist maßgebend. Nachfolgende Zuweisungen in einer ASSIGN-Klausel werden nur zur Dokumentation angegeben.

2. Mit RESERVE-Klausel kann der Benutzer die Anzahl der Ein-Ausga-bebereiche bestimmen, die dem Zielprogramm vom Übersetzer zugeordnet werden sollen. Die RESERVE-Klausel wird nur zur Dokumentation benutzt.

3. Die ORGANIZATION-Klausel gibt den logischen Aufbau einer Datei an. Die Dateiorganisation wird bei der Erstellung der Datei festgelegt und kann danach nicht mehr geändert werden.

4. Ist ACCESS MODE IS SEQUENTIAL angegeben, wird auf die Datensätze in der Reihenfolge zugegriffen, die durch die Dateior-ganisation vorgeschrieben ist. Diese Reihenfolge entspricht den absteigenden relativen Datensatznummern der in der Datei vor-handenen Datensätze.

5. Ist die FILE STATUS-Klausel angegeben, wird ein Wert in das durch datenname-2 angegebene Datenfeld übertragen und zwar nach Ausführung jeder Anweisung, die auf diese Datei entweder explizit oder implizit Bezug nimmt. Dieser Wert gibt den Zustand der Ausführung der Anweisung an.

6. Ist ACCESS MODE IS RANDOM angegeben, zeigt der Wert des RELATIVE KEY-Datenfeldes den Datensatz an, auf den zugegriffen werden soll.

7. Ist ACCESS MODE IS DYNAMIC angegeben, kann auf die in der Datei vorhandenen Datensätze sequentiell oder wahlfrei zugegrif-fen werden.

8. Alle in einer Datei vorhandenen Datensätze sind eindeutig durch ihre relativen Satznummern bezeichnet. Die relative Satznummer eines Datensatzes gibt die logische Position des Datensatzes innerhalb der Satzfolge der Datei an. Der erste logische Datensatz hat die relative Satznummer 1 und die folgenden logischen Daten-sätze haben die relativen Satznummern 2, 3, 4... .

9. Das mit datenname-1 bezeichnete Datenfeld dient zur Weitergabe einer relativen Satznummer zwischen dem Benutzer und dem Betriebssystem.

10. Ist dateibezeichner nicht explizit angegeben, wird er implizit definiert.

I-O-CONTROL-Paragraph

Funktion

Der I-O-CONTROL-Paragraph definiert die Ereignisse, bei deren Eintreten Wiederanlaufpunkte festzulegen sind und den Speicherbereich, der von verschiedenen Dateien gleichzeitig benutzt werden soll.

Allgemeines Format

```
I-O-CONTROL.

[ ; RERUN ON { dateiname-1    } EVERY { ganzzahl-1 RECORDS OF dateiname-2 } ... ]
             { herstellername }       { ganzzahl-2 CLOCK-UNITS           }
                                      { bedingungsname                   }

[ ; SAME [RECORD] AREA FOR dateiname-3 {, dateiname-4} ... ] ... .
```

Syntaxregeln

1. Der I-O-CONTROL-Paragraph ist wahlweise.
2. dateiname-1 muß eine sequentiell organisierte Datei sein.
3. herstellername muß in der RERUN-Klausel angegeben sein, wenn ganzzahl-1 RECORDS oder ganzzahl-2 CLOCK-UNITS definiert sind.
4. Sind mehrere ganzzahl-1 RECORD angegeben, muß dateiname-2 eindeutige Namen haben.
5. Es darf nur eine RERUN-Klausel angegeben werden, die CLOCK-UNITS enthält.
6. Die beiden Formate der SAME-Klausel (SAME AREA, SAME RECORD werden nachfolgend getrennt behandelt.

   Es kann mehr als eine SAME-Klausel in einem Programm verwendet werden, dann gilt jedoch:

   a) Ein Dateiname darf nicht in mehreren SAME AREA-Klauseln vorkommen.

   b) Ein Dateiname darf nicht in mehreren SAME RECORD-AREA-Klauseln vorkommen.

c) Ein Dateiname darf gleichzeitig in einer SAME AREA- und in einer SAME RECORD AREA-Klausel vorkommen. In diesem Fall müssen alle Dateinamen, die in der SAME AREA-Klausel aufgeführt sind, auch in der SAME RECORD AREA-Klausel stehen. Die SAME RECORD AREA-Klausel darf darüber hinaus weitere Dateinamen enthalten, die nicht in der SAME AREA-Klausel stehen. Die Regel, daß nur eine der angegebenen Dateien in der SAME AREA-Klausel zu einem gegebenen Zeitpunkt eröffnet sein kann, hat Vorrang vor der Regel, daß alle der angegebenen Dateien in der SAME RECORD AREA-Klausel zu einem gegebenen Zeitpunkt eröffnet sein können.

7. Die in der SAME AREA- oder SAME RECORD AREA-Klausel angegebenen Dateien müssen nicht alle die gleiche Organisation bzw. den gleichen Zugriff haben.

Allgemeine Regeln

1. Die RERUN-Klausel wird nur zur Dokumentation benutzt.

2. Die SAME AREA-Klausel gibt an, daß mehrere Dateien, jedoch keine Sortier- oder Mischdateien, während der Programmausführung einen Ein-Ausgabebereich gemeinsam benutzen sollen. Dieser Ein-Ausgabebereich umfaßt den gesamten Speicherbereich (einschließlich Wechselbereiche) der den angegebenen Dateien zugewiesen wird. Es ist daher nicht zulässig, daß mehr als eine Datei zum selben Zeitpunkt eröffnet ist.

3. Die SAME RECORD AREA-Klausel gibt an, daß mehrere Dateien denselben Speicherbereich für die Verarbeitung des aktuellen logischen Datensatzes verwenden müssen. Alle Dateien können zur selben Zeit, eröffnet sein. Ein logischer Datensatz gilt als logischer Datensatz aller zur Ausgabe eröffneten Dateien, deren Dateiname in dieser SAME RECORD AREA-Klausel aufgeführt sind. Er ist ebenfalls logischer Datensatz derjenigen Datei dieser Klausel, aus der die letzte Eingabe erfolgte. Dies entspricht einer impliziten Überlagerung aller Datensatzbereiche, wobei die Datensätze auf die am weitesten links stehende Zeichenposition ausgerichtet sind.

5.4 DATA DIVISION im Ein-Ausgabemodul

FILE SECTION

In einem COBOL-Programm stellt die Dateibeschreibung (FD) die höchste Organisationsstufe in der FILE SECTION dar. Der Kapitelüberschrift FILE SECTION folgt eine Dateibeschreibung, die aus einer Stufenbezeichnung (FD), einem Dateinamen und einer Anzahl unabhängigen Klauseln besteht.
Die FD-Klausel gibt an:

- wie groß die logischen und physischen Datensätze sind,
- ob Kennsätze vorhanden sind,
- welchen Wert die Kennsatzdatenfelder haben, die die Datei betreffen.

Die Eintragung wird mit einem Punkt abgeschlossen.

Aufbau der Datensatzbeschreibung

Eine Datensatzbeschreibung besteht aus einer Anzahl von Datenbeschreibungen, die die Merkmale eines bestimmten Datensatzes beschreiben. Jede Datenbeschreibung besteht aus einer Stufennummer, gegebenenfalls einem Datennamen und einer Folge von unabhängigen Klauseln. Eine Datensatzbeschreibung hat einen hierarchischen Aufbau. Daher können die bei einer Eintragung verwendeten Klauseln sehr unterschiedlich sein. Das hängt davon ab, ob untergeordnete Eintragungen nachfolgen oder nicht.

Der Aufbau einer Datensatzbeschreibung wird im Abschnitt "Stufenkonzepte" im Kapitel 2 beschrieben. Die in einer Datensatzbeschreibung zulässigen Elemente werden unten beschrieben.

Dateibeschreibung - vollständiges Eintragungsschema

Funktion

Die Dateibeschreibung bestimmt den physischen Aufbau, die Bezeichnung und die Datensatznamen einer Datei.

Allgemeines Format

```
FD dateiname

    [; BLOCK CONTAINS [ganzzahl-1 TO] ganzzahl-2 {RECORDS   }]
                                                 {CHARACTER }

     [; RECORD CONTAINS [ganzzahl-3 TO] ganzzahl-4 CHARACTERS]

     ; LABEL {RECORD IS   } {STANDARD}
             {RECORDS ARE} {OMITTED }

    [; VALUE OF datenname-1 IS {datenname-2}
                               {literal-1  }

          [, datenname-3 IS {datenname-4}] ... ]
                            {literal-2  }

    [; DATA {RECORD IS   } datenname-5  [, datenname-6] ... ] .
            {RECORDS ARE}
```

Syntaxregeln

1. Die Stufenbezeichnung FD legt den Anfang einer Dateibeschreibung fest und muß vor dem Dateinamen stehen.
2. Die dem Dateinamen folgenden Klauseln sind in den meisten Fällen wahlweise. Die Reihenfolge ihres Auftretens ist beliebig.
3. Eine Dateibeschreibung muß von einer Datensatzerklärung oder von mehreren Datensatzbeschreibungen gefolgt sein.

BLOCK CONTAINS-Klausel

Funktion

Die BLOCK CONTAINS-Klausel gibt die Größe eines physischen Datensatzes an.

Allgemeines Format

```
BLOCK CONTAINS [ganzzahl-1 TO] ganzzahl-2 {RECORDS    }
                                          {CHARACTERS }
```

Allgemeine Regel

Diese Klausel ist nur zur Dokumentation erforderlich.

DATA RECORDS-Klausel

Funktion

Die DATA RECORDS-Klausel dient nur zur Dokumentation der Namen von Datensätzen und der zugehörigen Dateien.

Allgemeines Format

```
DATA { RECORD IS  } datenname-1 [, datenname-2] ...
     { RECORDS ARE}
```

Syntaxregel

datenname-1, datenname-2,... bezeichnen Datensätze. Jeder dieser Datensätze muß eine zugehörige Datensatzbeschreibung auf der Stufennummer 01 mit dem gleichen Namen haben.

Allgemeine Regeln

1. Wenn mehr als ein Datenname vorhanden ist, enthält die Datei mehr als eine Art von Datensätzen. Diese Datensätze können von verschiedener Größe, verschiedenem Format usw. sein. Die Reihenfolge ihrer Auflistung ist ohne Bedeutung.
2. Konzeptionell teilen sich alle Datensätze in einer Datei den gleichen Bereich. Dies wird auch dadurch nicht geändert, wenn mehr als eine Art von Datensätzen in der Datei vorhanden ist.

LABEL RECORDS-Klausel

Funktion

Die LABEL RECORDS-Klausel gibt an, ob Kennsätze vorhanden sind.

Allgemeines Format

```
LABEL { RECORD IS  } { STANDARD }
      { RECORDS ARE} { OMITTED  }
```

Syntaxregel

Diese Klausel ist für jede Dateibeschreibung erforderlich, wenn der ANSI-Schalter gesetzt ist.

Allgemeine Regel

Diese Klausel wird nur zur Dokumentation verwendet.

RECORD CONTAINS-Klausel

Funktion

Die RECORD CONTAINS-Klausel gibt die Größe von Datensätzen an.

Format

```
RECORD CONTAINS [ganzzahl-1 TO] ganzzahl-2 CHARACTERS
```

Allgemeine Regel

Die Größe eines jeden Datensatzes wird vollständig in der Datensatzerklärung definiert, so daß diese Klausel niemals erforderlich wird. Die RECORD CONTAINS-Klausel wird nur zur Dokumentation angegeben.

VALUE OF-Klausel

Funktion

Die VALUE OF-Klausel dokumentiert den Inhalt eines Datenfeldes in den mit einer Datei zusammenhängenden Kennsätzen.

Allgemeines Format

```
VALUE OF datenname-1 IS { datenname-2 }
                        { literal-1   }

      [ , datenname-3 IS { datenname-4 } ] ...
      [                  { literal-2   } ]
```

Syntaxregeln

1. Datennamen können gekennzeichnet, dürfen jedoch nicht subskribiert oder indiziert sein. Sie dürfen keine Datenfelder sein, die mit USAGE IS INDEX beschrieben sind.
2. datenname-2, datenname-4 usw. müssen in der WORKING-STORAGE SECTION enthalten sein.

Allgemeine Regeln

1. Diese Klausel wird nur zur Dokumentation verwendet.

   Der Compiler überprüft, ob für Eingabedateien:

   - datenname-1 dem Wert von datenname-2 bwz. literal-1 entspricht,
   - datenname-3 dem Wert von datenname-4 bzw. literal-2 entspricht usw.;

   für Ausgabedateien:

   - der Wert von datenname-1 durch den Wert von datenname-2 bzw. literal-1 ersetzt ist,
   - der Wert von datenname-3 durch den Wert von datenname-4 bzw. literal-2 ersetzt ist.

2. Im obigen Format kann jedes genannte Literal durch eine figurative Konstante ersetzt werden.

5.5 PROCEDURE DIVISION im Ein- Ausgabemodul

CLOSE-Anweisung

Funktion

Durch die CLOSE-Anweisung wird die Verarbeitung von Dateien beendet.

Allgemeines Format

```
CLOSE dateiname-1 [WITH LOCK] [,dateiname-2 [WITH LOCK]] ...
```

Syntaxregel

Die in der CLOSE-Anweisung angegebenen Dateien brauchen nicht alle die gleiche Organisation bzw. die gleiche Zugriffsart zu haben.

Allgemeine Regeln

1. Eine CLOSE-Anweisung darf nur für eine Datei im OPEN-Modus ausgeführt werden.
2. Relative Dateien werden klassifiziert, als ob sie der Kategorie der nicht sequentiellen Eindatenträgerdatei bzw. Mehrdatenträgerdatei angehören würden. Die Ergebnisse der Ausführung der verschiedenen CLOSE-Anweisungen für die Dateitypen werden in nachfolgender Tabelle zusammengefaßt.

CLOSE-Anweisungsformat	Dateityp = nicht sequentielle Eindatenträgerdatei/ Mehrdatenträgerdatei
CLOSE	A
CLOSE WITH LOCK	A , B

A - Dateiabschluß

Eingabe und Ein-Ausgabedateien (sequentieller Zugriffsmodus):

Ist das Ende der Datei erreicht und sind Kennsätze für die Datei angegeben, werden die Kennsätze entsprechend den Standardkennsatzkonventionen des Betriebssystems verarbeitet. Das Verhalten der CLOSE-Anweisung ist undefiniert, wenn Kennsätze angegeben, aber nicht vorhanden sind, oder wenn Kennsätze nicht angegeben, aber vorhanden sind.

Ist das Ende der Datei erreicht und sind keine Kennsätze für die Datei angegeben, findet keine Kennsatzverarbeitung statt. Es werden Abschlußoperationen in Abhängigkeit vom Laufzeitsystem ausgeführt.
Wird das Ende der Datei nicht erreicht, werden die vom Laufzeitsystem abhängigen Abschlußoperationen ausgeführt; es erfolgt jedoch keine Abschlußkennsatzverarbeitung.

Eingabe und Ein-Ausgabedateien (wahlfreier oder dynamischer Zugriffsmodus);
Ausgabedateien (wahlfreie, dynamische oder sequentielle Zugriffsmethode):

Ist LABEL RECORDS für die Datei angegeben, werden die Kennsätze entsprechend den Kennsatzkonventionen des Betriebssystems verarbeitet. Das Verhalten der CLOSE-Anweisung ist undefiniert, wenn LABEL RECORDS angegeben ist, aber keine Kennsätze vorhanden sind oder wenn LABEL RECORDS nicht angegeben ist, aber Kennsätze vorhanden sind. Ist LABEL RECORDS für die Datei nicht angegeben, findet keine Kennsatzverarbeitung statt.

B - Dateisperrung

Die Datei kann während des aktuellen Programmlaufs nicht wieder eröffnet werden.

3. Eine eröffnete Datei muß geschlossen werden, bevor eine STOP RUN-Anweisung ausgeführt werden soll. Eine eröffnete Datei in einem aufgerufenen Programm muß geschlossen werden, bevor eine CANCEL-Anweisung für dieses Programm ausgeführt werden soll.

4. Wurde eine CLOSE-Anweisung für eine Datei ausgeführt, kann keine andere Anweisung, die explizit oder implizit auf diese Datei Bezug nimmt, ausgeführt werden, bevor nicht eine erneute OPEN-Anweisung ausgeführt worden ist.
5. Nach erfolgreicher Ausführung einer CLOSE-Anweisung steht der der Datei zugeordnete Satzbereich nicht mehr zur Verfügung. Nach erfolgloser Ausführung einer CLOSE-Anweisung ist die Verfügbarkeit des Satzbereichs undefiniert.
6. Ist WITH LOCK angegeben, kann die Datei während des aktuellen Ablaufs der Ausführungseinheit nicht wieder eröffnet werden.

DELETE-Anweisung

Funktion

Durch die DELETE-Anweisung wird ein Datensatz einer Plattenspeicherdatei logisch gelöscht.

Allgemeines Format

```
DELETE dateiname RECORD [;INVALID KEY unbedingte-anweisung]
```

Syntaxregeln

1. Für eine DELETE-Anweisung, die sich auf eine Datei mit sequentiellen Zugriffsmodus bezieht, darf INVALID KEY nicht angegeben sein.
2. Für eine DELETE-Anweisung, die sich auf eine Datei mit nicht sequentiellem Zugriffsmodus bezieht, und für die keine anwendbare USE-Prozedur angegeben ist, muß INVALID KEY angegeben sein.

Allgemeine Regeln

1. Die in der DELETE-Anweisung angesprochene Datei muß sich während der Ausführung dieser Anweisung im Eröffnungsmodus I-O befinden (vgl. OPEN-Anweisung in diesem Kapitel).
2. Bei einer Datei, die sich im sequentiellen Zugriffsmodus befindet, muß die der DELETE-Anweisung vorausgegangene Ein-Ausgabe-Anweisung eine erfolgreich ausgeführte READ-Anweisung gewesen sein. Das Betriebssystem löscht logisch den durch die READ-Anweisung gelesenen Datensatz aus der Datei.
3. Bei einer Datei mit wahlfreiem oder dynamischen Zugriffsmodus löscht das Betriebssystem logisch den Datensatz, der in der RELATIVE KEY-Klausel der Datei angegeben wurde. Falls der durch den Schlüssel angegebene Datensatz nicht in der Datei ist, besteht eine INVALID KEY-Bedingung (vgl. INVALID KEY-Bedingung in diesem Kapitel).
4. Nach erfolgreicher Ausführung einer DELETE-Anweisung ist der in Frage kommende Datensatz logisch aus der Datei gelöscht; es kann nicht mehr auf ihn zugegriffen werden.
5. Die Ausführung einer DELETE-Anweisung beeinflußt nicht den Inhalt des zur Datei gehörigen Datensatzbereiches.
6. Der aktuelle Satzzeiger wird durch die Ausführung einer DELETE-Anweisung nicht betroffen.
7. Nach Ausführung der DELETE-Anweisung wird der Inhalt des Datenfeldes fortgeschrieben, das für die Datei in der FILE STATUS-Klausel angegeben wurde.

OPEN-Anweisung

Funktion

Die OPEN-Anweisung eröffnet Dateien für die Verarbeitung. Sie führt die Prüfung, das Schreiben von Kennsätzen und andere Ein-Ausgabe-Operationen durch.

Allgemeines Format

```
        ⎧INPUT  dateiname-1 [,dateiname-2 ... ]⎫
OPEN    ⎨OUTPUT dateiname-3 [,dateiname-4 ... ]⎬ ...
        ⎩I-O    dateiname-5 [,dateiname-6 ... ]⎭
```

Syntaxregeln

Die in der OPEN-Anweisung benannten Dateien müssen nicht alle die gleiche Organisation oder den gleichen Zugriff haben.

Allgemeine Regeln

1. Die erfolgreiche Ausführung einer OPEN-Anweisung bestimmt die Verfügbarkeit der Datei und führt dazu, daß sich die Datei im OPEN-Modus befindet.
2. Nach erfolgreicher Ausführung einer OPEN-Anweisung steht der zugehörige Datensatzbereich dem Programm zur Verfügung.
3. Vor der erfolgreichen Ausführung einer OPEN-Anweisung für eine Datei darf keine Anweisung ausgeführt werden, die explizit oder implizit auf die Datei Bezug nimmt.

4. Vor Ausführung einer der zulässigen Ein-Ausgabe-Anweisungen muß eine OPEN-Anweisung erfolgreich ausgeführt worden sein. In nachfolgender Tabelle sind die für einen OPEN-Modus zulässigen Anweisungen aufgeführt. Ein X im Schnittpunkt bedeutet erlaubt.

Dateizugriffsmodus	Anweisung	OPEN-Modus		
		INPUT	OUTPUT	I-O
sequentiell	READ	X		X
	WRITE		X	
	REWRITE			X
	START	X		X
	DELETE			X
direkt	READ	X		X
	WRITE		X	X
	REWRITE			X
	START			
	DELETE			X
dynamisch	READ	X		X
	WRITE		X	X
	REWRITE			X
	START	X		X
	DELETE			X

Tabelle 5-1 Zulässige Kombinationen von Anweisungen und OPEN-Modus für relative I-O.

5. Alle Angaben INPUT, OUTPUT oder I-O sind bei verschiedenen OPEN-Anweisungen für eine Datei im selben Programm möglich. Bevor eine weitere OPEN-Anweisung nach der logisch ersten für dieselbe Datei in einem Programm durchlaufen werden kann, muß eine CLOSE-Anweisung durchlaufen worden sein.

6. Die Ausführung der OPEN-Anweisung bewirkt nicht, daß der erste Datensatz erhalten oder daß er übergeben wird.

7. Der mit SELECT ASSIGN angegebene Dateiname wird wie folgt verarbeitet:

   a) INPUT bedeutet, daß die Datei als Eingabedatei verarbeitet werden soll. Ist INPUT angegeben, verursacht die Ausführung einer OPEN-Anweisung, daß der in der ASSIGN-Klausel angegebene Name entsprechend den Betriebssystemkonventionen für das Eröffnen von Eingabedateien überprüft wird.

   b) OUTPUT bedeutet, daß die Datei als Ausgabedatei verarbeitet werden soll. Ist OUTPUT angegeben, verursacht die OPEN-Anweisung, daß der in der ASSIGN-Klausel angegebene Name entsprechend den Betriebssystemkonventionen für das Eröffnen von Ausgabedateien geschrieben wird.

8. Die Dateibeschreibung für dateiname-1, dateiname-2, dateiname-5 dateiname-6 muß gleich der sein, die bei der Erstellung der Datei verwendet wurde.

9. Bei Dateien, die mit INPUT oder I-O eröffnet wurden, verursacht die OPEN-Anweisung, daß der aktuelle Satzzeiger auf den ersten in der Datei momentan existierenden Datensatz zeigt. Sind keine Datensätze in der Datei vorhanden, wird der aktuelle Satzzeiger so gesetzt, daß die nächste ausgeführte READ-Anweisung vom Format 1 für die Datei eine AT END-Bedingung zur Folge hat. Ist die Datei nicht vorhanden, verursacht OPEN INPUT eine Fehleranzeige.

10. I-O bedeutet, daß die Datei für Eingabe- und Ausgabe-Operationen eröffnet werden soll. Ist die Datei nicht vorhanden, wird sie erstellt. Beim sequentiellen Zugriffsmodus wird sie dann für Eingabeoperationen verwendet. Jeder Versuch eines Schreibzugriffs führt zu einem Fehler.

11. Ist I-O angegeben und zeigt die LABEL RECORDS-Klausel an, daß Kennsätze vorhanden sind, werden bei der Ausführung der OPEN-Anweisung folgende Schritte durchlaufen:

    a) Die Kennsätze werden entsprechend den Betriebssystemkonventionen für die Ein-Ausgabe von Kennsätzen überprüft.

    b) Die neuen Kennsätze werden entsprechend den Betriebssystemkonventionen für das Schreiben von Ein-Ausgabe-Kennsätzen beschrieben.

12. Bei erfolgreicher Ausführung einer OPEN-Anweisung mit OUTPUT-Angabe wird eine Datei erstellt. Zu diesem Zeitpunkt enthält die zugehörige Datei keine Datensätze. Existiert schon eine Datei mit dem gleichen Namen, wird sie gelöscht. Ist diese aber schreibgeschützt, tritt ein Fehler auf.

READ-Anweisung

Funktion

Bei sequentiellem Zugriff wird durch die READ-Anweisung der nächste logische Datensatz aus einer Datei zur Verfügung gestellt. Bei wahlfreiem Zugriff wird durch die READ-Anweisung ein angegebener Datensatz aus der Plattenspeicherdatei zur Verfügung gestellt.

Allgemeine Formate

Format 1

```
READ dateiname [NEXT] RECORD [INTO bezeichner]
     [; AT END unbedingte-anweisung]
```

Format 2

```
READ dateiname RECORD [INTO bezeichner] [; INVALID KEY unbedingte-anweisung]
```

Syntaxregeln

1. Die INTO-Angabe darf nicht verwendet werden, wenn die Datensatzbeschreibung für die Eingabedatei variable logische Datensätze enthält. Der durch bezeichner angegebene Speicherbereich darf nicht derselbe sein, wie der durch dateiname angegebene Datensatzbereich.
2. Format 1 ohne NEXT-Angabe muß für alle Dateien im sequentiellen Zugriffsmodus verwendet werden.
3. Format 1 mit NEXT-Angabe muß für alle Dateien im dynamischen Zugriffsmodus verwendet werden, wenn Datensätze sequentiell wieder aufgefunden werden sollen.

4. Format 2 muß für alle Dateien im wahlfreien oder dynamischen Zugriffsmodus verwendet werden, wenn Datensätze wahlfrei wieder aufgefunden werden sollen.

5. Ist für dateiname keine anwendbare USE-Prozedur spezifiziert, muß INVALID KEY oder AT END angegeben werden.

Allgemeine Regeln

1. Bevor eine READ-Anweisung für eine Datei durchgeführt werden kann, muß diese Datei mit INPUT oder I-O eröffnet worden sein (vgl. OPEN-Anweisung).

2. Der durch die READ-Anweisung im Format 1 zur Verfügung gestellte Datensatz wird wie folgt bestimmt:

   a) Der Datensatz, auf den der aktuelle Satzzeiger zeigt, wird zur Verfügung gestellt, wenn der aktuelle Satzzeiger durch eine START- oder OPEN-Anweisung positioniert worden ist. Kann auf den Datensatz nicht mehr zugegriffen werden, weil er z.B. gelöscht ist, wird der aktuelle Satzzeiger so fortgeschrieben, daß er auf den nächsten existierenden Datensatz in der Datei zeigt. Dieser Datensatz wird dann zur Verfügung gestellt.

   b) War der aktuelle Satzzeiger bei Ausführung einer vorhergehenden READ-Anweisung gesetzt, wird er so fortgeschrieben, daß er auf den nächsten vorhandenen Datensatz in der Datei zeigt. Dieser Datensatz wird dann zur Verfügung gestellt.

3. Die Ausführung der READ-Anweisung bewirkt, daß der Inhalt des Datenfeldes fortgeschrieben wird, das in der FILE STATUS-Klausel der zugehörigen Dateibeschreibung angegeben wurde (vgl. Ein-Ausgabezustand).

4. Ein Datensatz bleibt für das Zielprogramm solange verfügbar, bis eine andere der READ-Anweisung folgende Anweisung ausgeführt wird.

5. Wenn eine Datei mehr als einen logischen Datensatztyp enthält, teilen sich diese Datensätze automatisch den gleichen Speicherbereich. Dies entspricht einer impliziten Redefinition des Speicherbereiches. Der Inhalt von Datenfeldern, die außerhalb des Bereiches des aktuellen Datensatzes liegen, sind bei Abschluß der READ-Anweisung undefiniert.

6. Ist INTO angegeben, wird der gelesene Datensatz aus dem Datensatzbereich in den durch bezeichner angegebenen Bereich übertragen (impliziter MOVE). Die Übertragung findet entsprechend den Regeln für die MOVE-Anweisung ohne CORRESPONDING-Angabe statt. Der implizite MOVE wird nicht ausgeführt, wenn die Ausführung der READ-Anweisung erfolglos war. Jede mit bezeichner zusammenhängende Subskribierung oder Indizierung wird ausgewertet, nachdem der Datensatz gelesen wurde und unmittelbar bevor er in das Datenfeld übertragen wird.

7. Ist INTO angegeben, steht der gelesene Datensatz sowohl im Eingabebereich als auch in dem durch bezeichner festgelegten Bereich zur Verfügung.

8. Ist zur Ausführungszeit einer READ-Anweisung vom Format 1 die Position des aktuellen Satzzeigers für diese Datei undefiniert, ist die Ausführung dieser READ-Anweisung erfolglos.

9. Ist bei Ausführung einer READ-Anweisung kein nächster logischer Datensatz in der Datei vorhanden, tritt die AT END-Bedingung auf. Die Ausführung der READ-Anweisung wird als erfolglos betrachtet (vgl. Ein-Ausgabezustand).

10. Wird die AT END-Bedingung erkannt, laufen die folgenden Schritte in der angegebenen Reihenfolge ab:

    a) Der Inhalt des Datenfeldes aus der FILE STATUS-Klausel, falls für diese Datei angegeben, wird mit einem Wert versorgt, der die AT END-Bedingung anzeigt.

    b) Ist in der READ-Anweisung AT END angegeben, wird auf die unbedingte Anweisung nach der AT END-Angabe verzweigt. Eine für die Datei vorhandene USE-Prozedur wird nicht durchlaufen.

    c) Ist AT END in der READ-Anweisung nicht angegeben, muß entweder explizit oder implizit eine USE-Prozedur für diese Datei vorliegen. Diese Prozedur wird dann durchlaufen.

    Bei Auftreten der AT END-Bedingung ist jede Ein-Ausgabeanweisung erfolglos.

11. Nach der erfolglosen Ausführung einer READ-Anweisung sind der Inhalt des zugehörigen Datensatzbereiches und die Position des aktuellen Satzzeigers undefiniert.

12. Nach Auftreten der AT END-Bedingung darf für diese Datei keine READ-Anweisung vom Format 1 gegeben werden, bevor nicht eine der folgenden Aktionen ausgeführt worden ist:

    a) Eine erfolgreiche CLOSE-Anweisung, gefolgt von einer erfolgreichen OPEN-Anweisung für diese Datei.

    b) Eine erfolgreiche START-Anweisung für diese Datei.

    c) Eine erfolgreiche READ-Anweisung vom Format 2 für diese Datei.

13. Bei einer Datei im dynamischen Zugriffsmodus wird durch eine READ-Anweisung vom Format 1 mit NEXT-Angabe der nächste logische Datensatz in der Datei wieder aufgefunden (vgl. Allgemeine Regel 2).

14. Ist RELATIVE KEY nicht angegeben, wird durch die Ausführung einer READ-Anweisung vom Format 1 das durch die RELATIVE KEY-Klausel bezeichnete Datenfeld so fortgeschrieben, daß es die relative Satznummer des zur Verfügung gestellten Datensatzes enthält.

15. Durch die Ausführung einer READ-Anweisung vom Format 2 zeigt der aktuelle Satzzeiger auf den Datensatz, dessen relative Satznummer in dem durch RELATIVE KEY bezeichneten Datenfeld für diese Datei enthalten ist. Dieser Datensatz wird dann zur Verfügung gestellt. Enthält die Datei keinen solchen Datensatz, tritt die INVALID-KEY-Bedingung ein. Die Ausführung der READ-Anweisung ist erfolglos (vgl. INVALID KEY-Bedingung).

REWRITE-Anweisung

Funktion

Die REWRITE-Anweisung ersetzt einen Datensatz logisch, der in einer Plattenspeicherdatei vorhanden ist.

Allgemeines Format

```
REWRITE datensatzname [FROM bezeichner] [; INVALID KEY unbedingte-anweisung]
```

Syntaxregeln

1. datensatzname und bezeichner dürfen sich nicht auf den gleichen Speicherbereich beziehen.
2. datensatzname ist der Name eines logischen Datensatzes in der FILE SECTION der DATA DIVISION und kann gekennzeichnet sein.
3. INVALID KEY darf nicht für eine REWRITE-Anweisung angegeben werden, die sich auf eine Datei im sequentiellen Zugriffsmodus bezieht.
4. Ist keine USE-Prozedur für Dateien im wahlfreien oder dynamischen Zugriffsmodus vorhanden, muß INVALID KEY in der REWRITE-Anweisung angegeben sein.

Allgemeine Regeln

1. Die mit datensatzname verknüpfte Datei muß zur Ausführungszeit dieser Anweisung mit OPEN I-O eröffnet worden sein (vgl. OPEN-Anweisung in diesem Kapitel).
2. Für Dateien im sequentiellem Zugriffsmodus gilt: Einer REWRITE-Anweisung muß eine erfolgreich abgelaufene READ-Anweisung vorangegangen sein und zwar als letzte Ein-Ausgabeanweisung für die zugehörige Datei.

   Der Datensatz, auf den durch die READ-Anweisung zugegriffen worden ist, wird vom Betriebssystem logisch ersetzt.

3. Die Anzahl der Zeichenpositionen in dem durch datensatzname bezeichneten Datensatz muß mit der Anzahl der Zeichenpositionen in dem zu ersetzenden Datensatz übereinstimmen.

4. Der logische Datensatz, der durch eine erfolgreich abgelaufende REWRITE-Anweisung zurückgeschrieben wurde, steht im Satzbereich nicht mehr zur Verfügung; eine Ausnahme stellt die Verwendung der SAME RECORD AREA-Klausel dar. In diesem Fall steht der logische Datensatz dem Programm zur Verfügung, sowohl als Datensatz aller anderen Dateien, die in der selben SAME RECORD AREA-Klausel aufgeführt wurden, als auch der mit datensatzname verknüpften Datei.

5. Die Ausführung einer REWRITE-Anweisung mit FROM-Angabe entspricht den folgenden Anweisungen:

   MOVE bezeichner TO datensatzname.
   REWRITE datensatzname.

   Der Inhalt des Speicherbereichs, beschrieben durch datensatzname, hat vor Ausführung der impliziten MOVE-Anweisung keine Bedeutung für die Ausführung der REWRITE-Anweisung.

6. Durch die Ausführung einer REWRITE-Anweisung wird der aktuelle Satzzeiger nicht beeinflußt.

7. Die Ausführung der REWRITE-Anweisung bewirkt, daß der Inhalt des in der FILE STATUS-Klausel angegebenen Datenfeldes fortgeschrieben wird.

8. Für Dateien im wahlfreien oder dynamischen Zugriffsmodus ersetzt das Betriebssystem den Datensatz, der mit dem verknüpften RELATIVE KEY-Datenfeld angegeben wurde. Enthält die Datei den durch den Schlüssel angegebenen Datensatz nicht, tritt eine INVALID KEY-Bedingung ein (vgl. INVALID KEY-Bedingung in diesem Kapitel). Es findet keine Fortschreibung statt. Die Daten im Datensatzbereich bleiben unberührt.

START-Anweisung

Funktion

Die START-Anweisung bietet eine Basis für logisches Positionieren in einer relativen Datei und für das aufeinanderfolgende sequentielle Lesen von Datensätzen.

Allgemeines Format

```
START dateiname [KEY { IS   EQUAL TO          } datenname]
                     { IS =                   }
                     { IS   GREATER THAN      }
                     { IS >                   }
                     { IS   NOT LESS THAN     }
                     { IS   NOT <             }

    [; INVALID KEY unbedingte-anweisung ]
```

Anmerkung

Die erforderlichen Vergleichszeichen '>' und '<' und '=' sind nicht unterstrichen, um Verwechslungen mit anderen Symbolen zu vermeiden.

Syntaxregeln

1. dateiname muß eine Datei im dynamischen oder sequentiellen Zugriffsmodus bezeichnen.
2. datenname kann gekennzeichnet sein.
3. Die INVALID KEY-Bedingung muß vorhanden sein, falls keine entsprechende USE-Prozedur angegeben ist.
4. datenname muß der Name des für diese Datei erklärten RELATIVE KEY-Datenfeldes sein.

Allgemeine Regeln

1. Die durch dateiname angegebene Datei muß zum Zeitpunkt der Ausführung der START-Anweisung durch INPUT oder I-O eröffnet sein.

2. Ist eine KEY-Angabe nicht vorhanden, wird 'IS EQUAL TO' angenommen.

3. Die Art des Vergleichs wird durch den Vergleichsoperator in der KEY-Angabe festgelegt. Der logische Schlüssel eines jedes Satzes in der durch dateiname bezeichneten Datei wird mit dem Inhalt des durch datenname bezeichneten Datenfeldes (RELATIVE KEY) verglichen.

   a) Innerhalb der Datei wird auf den ersten existierenden Datensatz positioniert, der die Vergleichsoperation erfüllt.

   b) Falls für keinen Datensatz der Datei die Vergleichsbedingung erfüllt ist, tritt eine INVALID KEY-Bedingung auf. Die Ausführung der START-Anweisung ist erfolglos und die Position des aktuellen Satzzeigers undefiniert (vgl. INVALID KEY-Bedingung in diesem Kapitel).

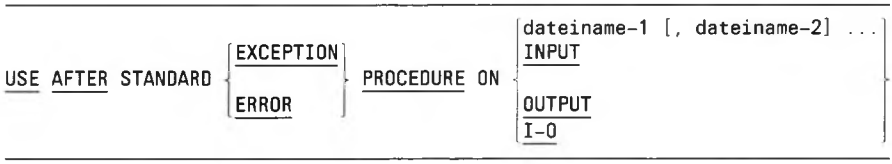
4. Ist FILE STATUS angegeben, wird nach Ausführung einer START-Anweisung der Inhalt des FILE STATUS-Datenfeldes fortgeschrieben.

USE-Anweisung

Funktion

Die USE-Anweisung gibt zusätzlich zu den Standardprozeduren der Ein-Ausgabesteuerung Prozeduren für die Behandlung von Ein-Ausgabefehlern an.

Allgemeines Format



Syntaxregeln

1. Ist die USE-Anweisung angegeben, muß sie innerhalb der Prozedurvereinbarungen (DECLARATIVES) stehen. Der erste Eintrag in den Prozedurvereinbarungen ist die Kapitelüberschrift und zwar: Kapitelname, gefolgt von dem Wort SECTION und einem Punkt. Unmittelbar anschließend an die Kapitelüberschrift muß die USE-Anweisung geschrieben werden. Danach können Paragraphen folgen, die die Prozedur definieren.

2. Die USE-Anweisung selbst wird nicht ausgeführt. Sie vereinbart Prozeduren, die durchlaufen werden sollen, wenn ein Ein-Ausgabefehler für eine Datei auftritt.

3. Da das Format variiert werden kann, kann in verschiedenen USE-Anweisungen der gleiche Dateiname auftreten. Dies darf jedoch nicht den gleichzeitigen Aufruf mehrerer Prozedurvereinbarungen bewirken.

4. Die Wörter ERROR und EXCEPTION sind gleichwertig und können alternativ verwendet werden.

5. Die Dateien, die implizit angesprochen werden (INPUT, OUTPUT, I-O, EXTEND) oder explizit (dateiname-1, dateiname-2,...) brauchen nicht dieselbe Organisation und denselben Zugriff zu haben.

Allgemeine Regeln

1. Die USE-Prozeduren werden durchlaufen bei Auftreten einer AT END- oder INVALID KEY-Bedingung, wenn die Ein-Ausgabe-Anweisung bei der die AT END- oder INVALID KEY-Bedingung auftrat, keine AT END- oder INVALID KEY-Angabe enthält.

2. Nach Ausführung einer USE-Prozedur wird die Steuerung an die aufrufende Routine zurückgegeben.

3. Innerhalb einer USE-Prozedur darf eine Bezugnahme auf Prozeduren außerhalb der Prozedurvereinbarungen (DECLARATIVES) nur durch die PERFORM-Anweisung vorgenommen werden.
   Auf Prozedurnamen, die einer USE-Anweisung untergeordnet sind, darf aus einer anderen Prozedur oder außerhalb der Prozedurvereinbarungen (DECLARATIVES) nur mit einer PERFORM-Anweisung Bezug genommen werden.

4. Innerhalb einer USE-Prozedur darf keine Anweisung ausgeführt werden, die eine USE-Prozedur anspringt, die die Steuerung noch nicht an die aufrufende Routine zurückgegeben hatte.

WRITE-Anweisung

Funktion

Die WRITE-Anweisung stellt einen logischen Datensatz für eine Ausgabe- oder Ein-Ausgabedatei zur Verfügung.

Allgemeines Format

```
WRITE datensatzname [FROM bezeichner] [; INVALID KEY unbedingte-anweisung]
```

Syntaxregeln

1. datensatzname und bezeichner dürfen nicht denselben Speicherbereich belegen.
2. datensatzname ist der Name eines logischen Datensatzes in der FILE SECTION der DATA DIVISION und kann gekennzeichnet werden.
3. INVALID KEY muß angegeben werden, wenn keine entsprechende USE-Prozedur für die Datei vereinbart wurde.

Allgemeine Regeln

1. Bei Ausführung einer WRITE-Anweisung muß die Datei mit OUTPUT oder I-O eröffnet sein (vgl. OPEN-Anweisung in diesem Kapitel).
2. Der durch eine WRITE-Anweisung ausgegebene Datensatz steht im Datensatzbereich nur dann zur Verfügung, wenn:
   - die zum Datensatz gehörende Datei in einer SAME AREA-Klausel angegeben ist oder wenn
   - die Ausführung der WRITE-Anweisung aufgrund einer INVALID KEY-Bedingung erfolglos war.

   Der logische Datensatz ist für das Programm als Datensatz anderer Dateien ebenfalls verfügbar, die in der gleichen SAME RECORD AREA-Klausel wie die zugehörigen Ausgabedateien angegeben sind sowie für die mit datensatzname verknüpften Dateien.

3. Die Ausführung einer WRITE-Anweisung mit FROM-Angabe ist gleichbedeutend mit folgenden Anweisungen:

   MOVE bezeichner TO datensatzname
   WRITE datensatzname

   Die Übertragung findet entsprechend den Regeln für die MOVE-Anweisung ohne CORRESPONDING-Angabe statt.

   Der Inhalt des Datensatzbereiches vor Ausführung der impliziten MOVE-Anweisung hat keinen Einfluß auf die Ausführung der WRITE-Anweisung.

   Nach erfolgreicher Ausführung der WRITE-Anweisung steht die Information in dem durch bezeichner angesprochenen Bereich zur Verfügung, auch wenn die Information in dem durch datensatzname angesprochenen Bereich nicht zur Verfügung steht (vgl. Allgemeine Regel 2).

4. Der aktuelle Satzzeiger wird durch die Ausführung einer WRITE-Anweisung nicht beeinflußt.

5. Nach Ausführung einer WRITE-Anweisung wird der Wert des Datenfeldes einer für diese Datei vorhandenen FILE STATUS-Klausel fortgeschrieben (vgl. Ein-Ausgabezustand).

6. Die maximale Datensatzgröße wird beim Erstellen der Datei festgelegt. Sie darf danach nicht mehr verändert werden.

7. Die auf einer Platte erforderliche Anzahl von Zeichenpositionen zum Speichern eines logischen Datensatzes in einer Datei muß nicht mit der für diesen Datensatz im Programm definierten Anzahl übereinstimmen.

8. Durch die Ausführung der WRITE-Anweisung wird dem Betriebssystem ein logischer Datensatz zur Verfügung gestellt.

9. Befindet sich eine Datei im Eröffnungsmodus OUTPUT, können Datensätze wie folgt in die Datei eingetragen werden:

   a) Im sequentiellen Zugriffsmodus veranlaßt die WRITE-Anweisung daß dem Betriebssystem ein Datensatz zum Erstellen einer neuen Datei zur Verfügung gestellt wird. Der erste Datensatz bekommt die relative Datensatznummer 1, während die danach folgenden Datensätze die Nummern 2, 3, 4 usw. erhalten. Wurde die RELATIVE KEY-Klausel angegeben, wird während der Ausführung der WRITE-Anweisung vom Betriebssystem die relative Satznummer des aktuellen Datensatzes in das RELATIVE-KEY-Datenfeld übertragen.

   b) Im wahlfreien oder dynamischen Zugriffsmodus muß der Wert des RELATIVE KEY-Datenfeldes auf den Wert der relativen Datensatznummer gesetzt werden, die der im Satzbereich befindliche Datensatz erhalten soll. Dieser Datensatz wird dann dem Betriebssystem durch Ausführung der WRITE-Anweisung zur Verfügung gestellt.

10. Wurde eine Datei mit I-O eröffnet und ist der Zugriffsmodus wahlfrei oder dynamisch, werden Datensätze in die bereits bestehende Datei eingefügt. Der Wert des RELATIVE KEY-Datenfeldes muß vom Programm auf den Wert der relativen Satznummer gesetzt werden, die der im Satzbereich befindliche Datensatz erhalten soll. Durch die Ausführung der WRITE-Anweisung wird dem Betriebssystem der Inhalt des Datensatzbereichs zur Verfügung gestellt.

11. Eine INVALID KEY-Bedingung tritt ein:

    a) wenn der Zugriffsmodus wahlfrei oder dynamisch ist und das RELATIVE KEY-Datenfeld einen Datensatz bezeichnet, der in der Datei bereits vorhanden ist

    b) wenn ein Versuch unternommen wurde, über die extern definierten Grenzen der Datei hinaus zu schreiben.

12. Tritt eine INVALID KEY-Bedingung auf, ist die Ausführung der WRITE-Anweisung erfolglos. Der Inhalt des Datensatzbereiches steht weiterhin zur Verfügung. Ist FILE STATUS angegeben, wird das FILE STATUS-Datenfeld auf einen Wert gesetzt, der die Ursache der INVALID KEY-Bedingung anzeigt. Die Fortsetzung des Programms hängt von den Regeln der INVALID KEY-Bedingung ab (vgl. Ein-Ausgabezustand in diesem Kapitel).

6 Ein-Ausgabe bei indizierten Dateien

6.1 Einführung

Das Ein-Ausgabemodul "Indizierte Datei" bietet eine Möglichkeit, auf Datensätze in einer Plattenspeicherdatei entweder wahlfrei oder sequentiell zuzugreifen. Jeder Datensatz in einer indizierten Datei wird eindeutig durch den Wert eines oder mehrerer Schlüssel innerhalb dieses Datensatzes identifiziert.

6.2 Sprachkonzepte

Organisation

Eine Datei mit indizierter Organisation ist eine Plattenspeicherdatei, in der auf Datensätze durch den Wert eines Schlüssels zugegriffen werden kann. Eine Datensatzbeschreibung kann mehrere Schlüsseldatenfelder enthalten, wobei jedes mit einem Index verknüpft ist. Jeder Index bietet einen logischen Weg zu den Datensätzen, entsprechend dem Inhalt eines Datenfeldes in jedem Datensatz, der den Datensatzschlüssel für diesen Index darstellt.

Das in der RECORD KEY-Klausel eines Dateisteuerungseintrags für eine Datei angegebene Datenfeld ist der primäre Datensatzschlüssel für diese Datei. Für das Einfügen, Fortschreiben und Löschen von Datensätzen in einer Datei, ist jeder Datensatz allein durch den Wert seines primären Datensatzschlüssels identifiziert. Daher muß dieser Wert eindeutig sein. Er kann beim Fortschreiben des Datensatzes nicht geändert werden. Die Länge des Schlüssels darf 127 Bytes nicht überschreiten.

Ein in der ALTERNATE RECORD KEY-Klausel des Dateisteuerungseintrags angegebenes Datenfeld ist ein alternativer Datensatzschlüssel für diese Datei. Der alternative Datensatzschlüssel kann einen nicht eindeutigen Wert haben, wenn DUPLICATES für ihn definiert ist. Diese Schlüssel bieten alternative Zugriffswege für das Wiederauffinden von Datensätzen in der Datei. Es können maximal 80 alternative Schlüssel angegeben werden.

Zugriffsmodus

Bei sequentiellem Zugriffsmodus wird in aufsteigender Reihenfolge der Datensatzschlüsselwerte auf die Datensätze zugegriffen. Innerhalb einer Folge von Datensätzen, die doppelte Schlüsselwerte haben, ist die Reihenfolge für das Wiederauffinden von Datensätzen dieselbe, in der die Datensätze geschrieben wurden.

Bei wahlfreiem Zugriffsmodus wird die Reihenfolge, in der auf Datensätze zugegriffen wird, vom Programmierer festgelegt. Auf den gewünschten Datensatz wird zugegriffen, indem der Wert seines Datensatzschlüssels in das entsprechende Datenfeld eingetragen wird.

Bei dynamischem Zugriffsmodus darf der Programmierer beliebig von sequentiellem Zugriff auf wahlfreien Zugriff wechseln, unter Verwendung entsprechender Formen der Ein-Ausgabe-Anweisungen.

Sperren von Dateien

Bei indizierten Dateien können entweder einzelne Sätze, Satzbereiche oder die gesamte Datei für andere Benutzer gesperrt werden.

Wie so eine Sperrung funktioniert, ist in der LII COBOL Bedienungsanleitung beschrieben.

Verschiedene Angaben wie READ WITH LOCK, COMMIT, ROLLBACK etc. sind in diesem Kapitel aufgeführt.

Aktueller Satzzeiger

Der aktuelle Satzzeiger ist ein Zeiger, der benutzt wird, um den nächsten Datensatz auszuwählen. Er hat keine Bedeutung für eine im Ausgabemodus eröffnete Datei. Er kann nur durch OPEN-, START- und READ-Anweisungen neu gesetzt werden.

Ein-Ausgabezustand

Wenn in einem Dateisteuerungseintrag die FILE STATUS-Klausel angegeben ist, wird in das aus zwei Zeichen bestehende angegebene Datenfeld, während der Ausführung einer OPEN-, CLOSE-, READ-, WRITE-, REWRITE-, DELETE- oder START-Anweisung und bevor eine USE-Prozedur ausgeführt wird, ein Wert gesetzt, um dem COBOL-Programm den Zustand dieser Ein-Ausgabe-Operation anzuzeigen.

Zustandsanzeiger 1

Das am weitesten links stehende Zeichen des in der FILE STATUS-Klausel angegebenen Datenfeldes wird als Zustandsanzeiger 1 bezeichnet. Der Zustandsanzeiger 1 wird gesetzt, um die Beendigungsbedingungen anzuzeigen:

0 = Erfolgreiche Beendigung
1 = Endebedingung
2 = Schlüsselfehlerbedingung
  22 = Reihenfolgefehler, doppelter Schlüssel
  23 = keinen Datensatz gefunden
  24 = Überschreitung der Bereichsgrenzen

3 = Permanenter Fehler
9 = Systemfehler oder detaillierte Zustandsbeschreibung

Die oben angegebenen Werte haben folgende Bedeutung:

0 = Erfolgreiche Beendigung.
Die Ein-Ausgabeanweisung wurde erfolgreich ausgeführt.

1 = Endebedingung.
Eine READ-Anweisung wurde erfolglos ausgeführt, weil versucht wurde, einen Datensatz zu lesen, obwohl kein nächster logischer Datensatz in der Datei vorhanden war.

2 = Schlüsselfehlerbedingung.
Die Ein-Ausgabe-Anweisung wurde aus einem der folgenden Gründe erfolglos ausgeführt:

- Reihenfolgefehler
- Doppelter Schlüssel
- Datensatz nicht gefunden
- Überschreitung der Bereichsgrenzen

3 = Permanenter Fehler.
Die Ein-Ausgabe-Anweisung wurde aufgrund eines Ein-Ausgabefehlers erfolglos ausgeführt, z.B. Datenüberprüfung, Paritätsfehler oder Übertragungsfehler.

4 = Systemfehler.
Die Ein-Ausgabeanweisung wurde erfolglos ausgeführt, weil eine Bedingung aufgetreten ist, die in der Fehlermeldung des Betriebssystems beschrieben ist. Dieser Wert wird nur dazu

benutzt, eine Bedingung anzuzeigen, die nicht durch andere definierte Werte des Zustandsanzeigers 1 oder durch beschriebene Kombinationen der Werte der Zustandsanzeiger 1 und 2 angezeigt werden.

Zustandsanzeiger 2

Die rechtsbündige Zeichenposition, d.h. das zweite Zeichen des in der FILE STATUS-Klausel angegebenen Datenfeldes wird als Zustandsanzeiger 2 bezeichnet und wird zur näheren Beschreibung der Ergebnisse der Ein-Ausgabeoperation verwendet. Es enthält einen der folgenden Werte:

- Ist bezüglich der Ein-Ausgabeoperation keine weitere Information verfügbar, enthält der Zustandsanzeiger 2 den Wert 0.
- Enthält Zustandsanzeiger 1 den Wert '0', der eine erfolgreiche Ausführung anzeigt, kann Zustandsanzeiger 2 den Wert "2" enthalten, der einen doppelten Schlüssel anzeigt. Diese Bedingung zeigt eine von zwei Möglichkeiten an:

  1. Bei einer READ-Anweisung ist der Wert für den aktuellen Bezugsschlüssel derselbe wie der Wert desselben Schlüssels im nächsten Datensatz innerhalb des aktuellen Bezugsschlüssels.
  2. Bei einer WRITE- oder REWRITE-Anweisung wird durch den gerade geschriebenen Datensatz ein doppelter Schlüsselwert erzeugt, und zwar für mindestens einen alternativen Datensatzschlüssel, für den das Verdoppeln erlaubt ist.

3. Enthält der Zustandsanzeiger 1 den Wert 2, der eine INVALID KEY-Bedingung anzeigt, enthält Zustandsanzeiger 2 einen der folgenden Werte, der die Ursache für diese Bedingung angibt:

   1 = Reihenfolgefehler für eine indizierte Datei, auf die sequentiell zugegriffen wird.
   Die aufsteigende Reihenfolge von aufeinanderfolgenden Satzschlüsselwerten wurde verletzt (vgl. WRITE-Anweisung in diesem Kapitel) oder der Wert des primären Datensatzschlüssels wurde vom COBOL-Programm zwischen der erfolgreichen Ausführung einer READ-Anweisung und der Ausführung der nächsten REWRITE-Anweisung für diese Datei geändert.

   2 = Doppelter Schlüssel.
   Es wurde versucht, einen Datensatz zu schreiben oder zurückzuschreiben, der einen doppelten Schlüssel innerhalb der indizierten Datei erzeugt hätte.

   3 = Datensatz nicht gefunden.
   Es wurde versucht, anhand eines Schlüssels auf einen Datensatz zuzugreifen, der in der Datei nicht vorhanden war.

   4 = Überschreitung der Bereichsgrenzen.
   Es wurde versucht, außerhalb der festgelegten Bereichsgrenzen einer indizierten Datei zu schreiben. Dies wird normalerweise vom Betriebssystem als schwerwiegender Fehler behandelt.

4. Enthält Zustandsanzeiger 1 den Wert 9, kann Zustandsananzeiger 2 folgende Werte haben:

Statusbyte 1	Statusbyte 2	Fehlermeldung	ohne FILE STATUS-Angabe Abbruch des LZS mit der Fehlernummer:
9	A	gesamte Datei gesperrt	065
9	C	Datei nicht eröffnet	067
9	D	Satz ist gesperrt	068
9	E	unzulässiges Argument für ISAM-Modul	069
9	F	zu viele Dateien offen	070
9	G	fehlerhaftes ISAM-Dateiformat	071
9	H	Dateiende	072
9	J	Kein aktueller Satzanzeiger	074
9	K	Dateiname zu lang	075
9	M	Interner ISAM-Fehler	077
9	N	unzulässiger Datensatzschlüssel	078
9	O	unzulässiger Primärer Datensatzschlüssel	079
9	P	Kein exklusiver Zugriff	080

Tabelle 6-1 Statusbyte 2 und zugehörige Fehlermeldungen

INVALID KEY-Bedingung:

Die INVALID KEY-Bedingung kann nach Ausführung einer START-, READ-, WRITE-, REWRITE- oder DELETE-Anweisung auftreten. Einzelheiten über das Auftreten dieser Bedingung sind bei den angeführten Anweisungen in diesem Kapitel beschrieben.

Wenn eine INVALID KEY-Bedingung erkannt wurde, führt das Betriebssystem nachfolgende Aktionen in der angegebenen Reihenfolge durch:

1. Ist für die Datei FILE STATUS angegeben, wird ein Wert in das FILE STATUS-Datenfeld übertragen, der die INVALID KEY-Bedingung anzeigt (vgl. Ein-Ausgabezustand in diesem Kapitel).
2. Enthält die Anweisung, die eine Schlüsselfehlerbedingung verursachte, die INVALID KEY-Angabe, wird die unter INVALID KEY stehende unbedingte Anweisung ausgeführt. Eine für diese Datei vorhandene USE-Prozedur wird nicht durchlaufen.
3. Ist INVALID KEY nicht angegeben, jedoch eine USE-Prozedur für diese Datei implizit oder explizit vorhanden, wird sie durchlaufen.

Tritt eine INVALID KEY-Bedingung auf, war die entsprechende Ein-Ausgabeanweisung erfolglos und die Datei befindet sich im gleichen Zustand wie vor Ausführung dieser Ein-Ausgabeanweisung.

AT END-Bedingung

Die AT END-Bedingung kann aufgrund der Ausführung einer READ Anweisung auftreten. Einzelheiten über die Ursachen dieser Bedingung werden bei der Beschreibung der READ-Anweisung in diesem Kapitel behandelt.

6.3 ENVIRONMENT DIVISION im Ein- Ausgabemodul

INPUT-OUTPUT SECTION

Der FILE-CONTROL-Paragraph

Funktion

Der FILE-CONTROL-Paragraph ordnet jeder Datei einen Namen zu und erlaubt die Beschreibung anderer dateibezogener Informationen (vgl. LEVEL II COBOL Bedienungsanleitung).

Allgemeines Format:

```
FILE-CONTROL. {dateisteuerungseintrag} ...
```

Dateisteuerungseintrag

Funktion

Der Dateisteuerungseintrag bezeichnet jede Datei mit Namen und definiert andere dateibezogene Informationen.

Allgemeines Format

```
SELECT dateiname

    ASSIGN TO    {externes dateiname-literal}
                 {dateibezeichner           }

                 [ , {externes dateiname-literal} ]
                     {dateibezeichner           }

[ ; RESERVE ganzzahl-1 [AREA ] ]
                       [AREAS]

 ; ORGANIZATION IS INDEXED

[ ; ACCESS MODE IS {SEQUENTIAL} ]
                   {DYNAMIC   }
                   {RANDOM    }

[ ; LOCK MODE IS   {EXCLUSIVE} ]
                   {AUTOMATIC }
                   {MANUAL    }

 ; RECORD KEY IS datenname-1

[;ALTERNATE RECORD KEY IS datenname-2 [WITH DUPLICATES]]...

[; FILE STATUS IS datenname-3].
```

Syntaxregeln

1. Die SELECT-Klausel muß der erste Eintrag im Dateisteuerungseintrag sein. Alle Klauseln, die der SELECT-Klausel folgen, können in beliebiger Reihenfolge geschrieben werden.
2. Jede in der DATA DIVISION beschriebene Datei darf nur einmal als Dateiname in der SELECT-Klausel angegeben werden. Jede im Dateisteuerungseintrag angegebene Datei muß einen Dateibeschreibungseintrag (z.B. FD) in der DATA DIVISION des Quellprogramms haben.
3. Ist die ACCESS MODE-Klausel nicht angegeben, wird ACCESS IS SEQUENTIAL angenommen.
4. datenname-3 muß in der DATA DIVISION als ein zwei Zeichen langes alphanumerisches Datenfeld definiert sein; er darf nicht in der FILE SECTION angegeben werden.
5. datenname-2 und datenname-3 können gekennzeichnet sein.
6. Die Datenfelder, auf die sich datenname-1 und datenname-2 beziehen, müssen in der Datensatzbeschreibung, die mit dateiname verknüpft ist, als alphanumerische Datenfelder definiert sein.
7. datenname-1 und datenname-2 dürfen keine Datenelemente beschreiben, deren Größe variabel ist (vgl. OCCURS-Klausel im Kapitel 3).
8. datenname-2 darf sich nicht auf ein Datenfeld beziehen, dessen linke Zeichenposition mit der linken Zeichenposition eines Datenfeldes übereinstimmt, auf das datenname-1 Bezug nimmt oder auf einen anderen datennamen-2, der mit dieser Datei verknüpft ist.

Allgemeine Regeln

1. Die ASSIGN-Klausel beschreibt die Verknüpfung der durch dateiname benannten Datei mit einem Speichermedium (vgl. LII COBOL Bedienungsanleitung). Die Erstzuweisung ist maßgebend. Nachfolgende Zuweisungen in einer ASSIGN-Klausel werden nur zur Dokumentation angegeben.

2. Mit RESERVE-Klausel kann der Benutzer die Anzahl der Ein-Ausgabebereiche bestimmen, die dem Zielprogramm vom Übersetzer zugeordnet werden sollen.
   Die RESERVE-Klausel wird nur zur Dokumentation benutzt.

3. Die ORGANIZATION-Klausel gibt den logischen Aufbau einer Datei an. Die Dateiorganisation wird bei der Erstellung der Datei festgelegt und kann danach nicht mehr geändert werden.

4. Ist ACCESS MODE IS SEQUENTIAL angegeben, wird auf die Datensätze in der Reihenfolge zugegriffen, die durch die Dateiorganisation vorgeschrieben ist. Bei indizierten Dateien entspricht diese Reihenfolge der Folge der aufsteigenden Datensatzschlüsselwerte innerhalb eines vorgegebenen Schlüssels.

5. Ist FILE STATUS angegeben, wird durch das Betriebssystem ein Wert in das durch datenname-2 angegebene Datenfeld übertragen und zwar nach Ausführung jeder Anweisung, die auf diese Datei entweder explizit oder implizit Bezug nimmt. Dieser Wert gibt den Zustand der Ausführung der Anweisung an.

6. Ist ACCESS MODE IS RANDOM angegeben, zeigt der Wert des RECORD-KEY-Datenfeldes den Datensatz an, auf den zugegriffen werden soll.

7. Ist ACCESS MODE IS DYNAMIC angegeben, kann auf die in der Datei vorhandenen Datensätze sequentiell oder wahlfrei zugegriffen werden.

8. Die RECORD KEY-Klausel gibt den primären Datensatzschlüssel für die Datei an. Die Werte des primären Datensatzschlüssels müssen unter den Datensätzen der Datei eindeutig sein. Dieser primäre Datensatzschlüssel bietet einen Zugriffsweg auf Datensätze einer indizierten Datei.

9. Die ALTERNATE RECORD KEY-Klausel gibt einen alternativen Datensatzschlüssel für diese Datei an. Dieser alternative Datensatzschlüssel bietet einen zweiten Zugriffsweg auf Datensätze einer indizierten Datei.

10. Die Datenbeschreibung von datenname-1 bzw. datenname-2 sowie die relativen Speicherplätze innerhalb eines Datensatzes müssen die gleichen sein, wie die, die bei Erstellung der Datei festgelegt wurden. Ebenso muß die Anzahl der alternativen Schlüssel für die Datei die gleiche sein wie bei Erstellung der Datei.

11. DUPLICATES gibt an, daß der Wert des zugehörigen alternativen Datensatzschlüssels innerhalb der Datensätze einer Datei verdoppelt werden kann. Ist DUPLICATES nicht angegeben, darf der Wert des zugehörigen alternativen Datensatzschlüssels zwischen den Datensätzen in der Datei nicht verdoppelt werden.

12. Ist dateibezeichner nicht explizit angegeben, wird er implizit definiert.

13. Die LOCK MODE-Klausel gibt an, ob und in welcher Form die Datei gesperrt werden soll.

    - EXCLUSIVE

      Die Datei wird vollständig für alle anderen Ausführungseinheiten gesperrt. Einzelne Sätze der Datei können nicht gesperrt werden.

    - AUTOMATIC

      Ein Satz wird automatisch immer dann gesperrt, wenn eine Ausführungseinheit auf ihn zugreift.

    - MANUAL

      Der Satz wird bei einem Zugriff nur dann gegen weitere Zugriffe geschützt, wenn dies in der Zugriffsanweisung ausdrücklich festgelegt ist (READ WITH LOCK).
      Siehe dazu auch LII COBOL Bedienungsanleitung Abschnitt 7.4.4.

I-O-CONTROL-Paragraph

Funktion

Der I-O CONTROL-Paragraph definiert die Ereignisse, bei deren Eintreten Wiederanlaufpunkte festzulegen sind und den Speicherbereich, der von verschiedenen Dateien gleichzeitig benutzt werden soll.

Allgemeines Format

```
I-O-CONTROL.

[;RERUN ON {dateiname-1   } EVERY {ganzzahl-1 RECORDS OF dateiname-2}]...
           {hersteller-   }       {ganzzahl-2 CLOCK-UNITS           }
           {name          }       {bedingungsname                   }

[;SAME [RECORD] AREA FOR dateiname-3 {dateiname-4} ... ] ... .
```

Syntaxregeln

1. Der I-O-CONTROL-Paragraph ist wahlweise.
2. dateiname-1 muß eine sequentiell organisierte Datei sein.
3. herstellername muß in der RERUN-Klausel angegeben sein, wenn ganzzahl-1 RECORDS oder ganzzahl-2 CLOCK UNITS definiert sind.
4. Sind mehrere ganzzahl-1 RECORDS angegeben, muß dateiname-2 eindeutige Namen haben.
5. Es darf nur eine RERUN-Klausel angegeben werden, die CLOCK-UNITS enthält.
6. Die beiden Formen der SAME-Klausel (SAME AREA, SAME RECORD AREA) werden nachfolgend getrennt behandelt.

   Es kann mehr als eine SAME-Klausel in einem Programm verwendet werden; dann gilt jedoch:

   a) Ein Dateiname darf nicht in mehreren SAME AREA-Klauseln vorkommen.

   b) Ein Dateiname darf nicht in mehreren SAME RECORD-Klauseln vorkommen.

c) Ein Dateiname darf gleichzeitig in einer SAME AREA- und in einer SAME RECORD AREA-Klausel vorkommen. In diesem Fall müssen alle Dateinamen, die in der SAME AREA-Klausel aufgeführt sind, auch in der SAME RECORD AREA-Klausel stehen. Die SAME RECORD AREA-Klausel darf darüber hinaus weitere Dateinamen enthalten, die nicht in der SAME-AREA-Klausel stehen.
Die Regel, daß nur eine der angegebenen Dateien in der SAME AREA-Klausel zu einem gegebenen Zeitpunkt eröffnet sein kann, hat Vorrang vor der Regel, daß alle der angegebenen Dateien in der SAME RECORD AREA-Klausel zu einem gegebenen Zeitpunkt eröffnet sein können.

7. Die in der SAME AREA- oder SAME RECORD AREA-Klausel angegebenen Dateien müssen nicht alle die gleiche Organisation bzw. den gleichen Zugriff haben.

Allgemeine Regeln

1. Die RERUN-Klausel wird nur zur Dokumentation benutzt.
2. Die SAME AREA-Klausel gibt an, daß mehrere Dateien während der Programmausführung einen Ein-Ausgabebereich benutzen sollen. Dieser Ein-Ausgabebereich umfaßt den gesamten Speicherbereich, der den angegebenen Dateien zugewiesen wird. Es ist daher nicht zulässig, daß mehr als eine Datei zum selben Zeitpunkt eröffnet ist (vgl. Syntaxregel 6 c).
3. Die SAME RECORD AREA-Klausel gibt an, daß mehrere Dateien denselben Speicherbereich für die Verarbeitung des aktuellen logischen Datensatzes verwenden müssen. Alle Dateien können zur selben Zeit eröffnet sein. Ein logischer Datensatz gilt als logischer Datensatz aller zur Ausgabe eröffneten Dateien, deren Dateiname in dieser SAME-RECORD AREA-Klausel aufgeführt sind. Er ist ebenfalls logischer Datensatz derjenigen Datei aus dieser Klausel, aus der die letzte Eingabe erfolgte. Dies entspricht einer impliziten Überlagerung aller Datensatzbereiche, wobei die Datensätze auf die am weitesten links stehende Zeichenposition ausgerichtet sind.

6.4 DATA DIVISION im Ein-Ausgabemodul

FILE SECTION

In einem COBOL-Programm stellt die Dateibeschreibung (FD) die höchste Organisationsstufe in der FILE SECTION dar. Der Kapitelüberschrift FILE SECTION folgt eine Dateibeschreibung, die aus einer Stufenbezeichnung (FD), einem Dateinamen und einer Anzahl unabhängiger Klauseln besteht.

Die FD-Klausel gibt an:

- wie groß die logischen und physischen Datensätze sind,
- ob Kennsätze vorhanden sind,
- welchen Wert die Kennsatzdatenfelder und welchen Namen die Datensätze haben, die die Datei betreffen.

Die Eintragung wird mit einem Punkt abgeschlosssen.

Aufbau der Datensatzbeschreibung

Eine Datensatzbeschreibung besteht aus einer Anzahl von Datenbeschreibungen, die die Merkmale eines bestimmten Datensatzes beschreiben. Jede Datenbeschreibung besteht aus einer Stufennummer, gegebenenfalls einem Datennamen und einer Folge von unabhängigen Klauseln. Eine Datensatzbeschreibung hat einen hierarchischen Aufbau. Daher können die bei einer Eintragung verwendeten Klauseln sehr unterschiedlich sein. Das hängt davon ab, ob untergeordnete Eintragungen nachfolgen oder nicht. Der Aufbau einer Datensatzbeschreibung wird im Abschnitt "Stufenkonzepte" im Kapitel 2 beschrieben.

Dateibeschreibung - vollständiges Eintragungsschema

Funktion

Die Dateibeschreibung bestimmt den physischen Aufbau, die Bezeichnung und die Datensatznamen einer Datei.

Allgemeines Format

```
FD dateiname

   [;BLOCK CONTAINS [ganzzahl-1] TO ganzzahl-2 {RECORDS    }]
                                               {CHARACTERS}

    [;RECORD CONTAINS [ganzzahl-3 TO] ganzzahl-4 CHARACTERS]

    ;LABEL {RECORD IS  } {STANDARD}
           {RECORDS ARE} {OMITTED }

   [; VALUE OF datenname-1 IS {datenname-2}
                              {literal-1  }

      [, datenname-3 IS {datenname-4}] ... ]
                        {literal-2  }

[; DATA {RECORD IS  }
        {RECORDS ARE} datenname-5 [, datenname-6] ...] .
```

Syntaxregeln

1. Stufenbezeichnung FD legt den Anfang einer Dateibeschreibung fest und muß vor dem Dateinamen stehen.
2. Die dem Dateinamen folgenden Klauseln sind in den meisten Fällen wahlweise. Die Reihenfolge ihres Auftretens ist beliebig. Alle Klauseln sind wahlweise, wenn der ANSI-Schalter gesetzt ist.
3. Eine Dateibeschreibung muß von einer Datensatzerklärung oder von mehreren Datensatzbeschreibungen gefolgt sein.

BLOCK CONTAINS-Klausel

Funktion

Die BLOCK-CONTAINS-Klausel gibt die Größe eines physischen Datensatzes an.

Allgemeines Format

```
BLOCK CONTAINS [ganzzahl-1 TO] ganzzahl-2 { RECORDS    }
                                          { CHARACTERS }
```

Allgemeine Regel

Diese Klausel ist nur zur Dokumentation erforderlich.

DATA RECORDS-Klausel

Funktion

Die DATA RECORDS-Klausel dient nur zur Dokumentation der Namen von Datensätzen und der zugehörigen Dateien.

Allgemeines Format

```
DATA { RECORD IS  } datenname-1 [, datenname-2] ...
     { RECORDS ARE}
```

Syntaxregel

datenname-1 und datenname-2, ... bezeichnen Datensätze. Jeder dieser Datensätze muß eine zugehörige Datensatzbeschreibung auf der Stufennummer 01 mit dem gleichen Namen haben.

Allgemeine Regeln

1. Die Angabe von mehr als einem Datennamen zeigt, daß die Datei mehr als eine Art von Datensätzen enthält. Diese Datensätze können von verschiedener Größe, verschiedenem Format usw. sein. Die Reihenfolge ihrer Auflistung ist nicht von Bedeutung.
2. Konzeptionell teilen sich alle Datensätze in einer Datei den gleichen Bereich. Dies wird auch dadurch nicht geändert, wenn mehr als eine Art von Datensätzen in der Datei vorhanden ist.

LABEL RECORDS-Klausel

Funktion

Die LABEL RECORDS-Klausel gibt an, ob Kennsätze vorhanden sind.

Format

```
LABEL { RECORD IS   } { STANDARD }
      { RECORDS ARE } { OMITTED  }
```

Allgemeine Regel

Diese Klausel wird nur zur Dokumentation verwendet.

RECORD CONTAINS-Klausel

Funktion

Die RECORD CONTAINS-Klausel gibt die Größe von Datensätzen an.

Format

```
RECORD CONTAINS [ganzzahl-1 TO] ganzzahl-2 CHARACTERS
```

Allgemeine Regel

Die Größe eines jeden Datensatzes wird vollständig in der Datensatzerklärung definiert, so daß diese Klausel niemals erforderlich wird. Die RECORD CONTAINS-Klausel wird nur zur Dokumentation angegeben.

VALUE OF-Klausel

Funktion

Die VALUE OF-Klausel dokumentiert den Inhalt eines Datenfeldes in den Kennsätzen einer Datei.

Allgemeines Format

```
VALUE OF datenname-1 IS { datenname-2 }
                        { literal-1   }

        [ , datenname-3 IS { datenname-4 } ] ...
                           { literal-2   }
```

Syntaxregeln

1. Datennamen können gekennzeichnet, dürfen jedoch nicht subskribiert oder indiziert sein. Sie dürfen keine Datenfelder sein, die mit USAGE IS INDEX beschrieben sind.
2. datenname-2, datenname-4 usw. müssen in der WORKING-STORAGE SECTION enthalten sein.

Allgemeine Regeln

1. Diese Klausel wird nur zur Dokumentation verwendet.
2. Bei einer Eingabedatei überprüft die entsprechende Kennsatzroutine, ob der Wert von datenname-1 gleich dem Wert von literal-1 bzw. datenname-2 ist, je nachdem, was angegeben wurde.

   Bei einer Ausgabedatei wird der Wert von datenname-1 dem Wert von literal-1 oder datenname-2 gleichgesetzt, je nachdem, was angegeben wurde.
3. Im obigen Format kann jedes genannte Literal durch eine figurative Konstante ersetzt werden.

6.5 PROCEDURE DIVISION im Ein-Ausgabemodul

CLOSE-Anweisung

Funktion

Durch die CLOSE-Anweisung wird die Verarbeitung von Dateien beendet.

Allgemeines Format

```
CLOSE dateiname-1 [WITH LOCK] [,dateiname-2 [WITH LOCK]] ...
```

Syntaxregel

Die in der CLOSE-Anweisung angegebenen Dateien brauchen nicht alle die gleiche Organisation bzw. die gleiche Zugriffsart zu haben.

Allgemeine Regeln

1. Eine CLOSE-Anweisung darf nur für eine Datei im OPEN-Modus ausgeführt werden.
2. Indizierte Dateien werden klassifiziert, als ob sie der Kategorie der nicht sequentiellen Eindatenträgerdatei bzw. Mehrdatenträgerdatei angehören würden. Die Ergebnisse der Ausführung der verschiedenen CLOSE-Anweisungen für die Dateitypen werden in nachfolgender Tabelle zusammengefaßt:

CLOSE-Anweisungsformat	Dateityp = nicht sequentielle Eindatenträgerdatei/ Mehrdatenträgerdatei
CLOSE	A
CLOSE WITH LOCK	A , B

Die in der Tabelle angegebenen Symbole sind nachfolgend definiert. Hängt die Definition davon ab, ob es sich um eine Eingabe-, Ausgabe- oder Ein-Ausgabedatei handelt, werden alternative Definitionen angegeben. In allen anderen Fällen gelten die Definitionen für Eingabe-, Ausgabe- und Ein-Ausgabedateien.

A = Dateiabschluß

Eingabe- und Ein-Ausgabedateien (sequentieller Zugriffsmodus):

Ist das Ende der Datei erreicht und sind Kennsätze für die Datei angegeben, werden die Kennsätze entsprechend den Standardkennsatzkonventionen des Betriebssystems verarbeitet. Das Verhalten der CLOSE-Anweisung ist undefiniert, wenn Kennsätze angegeben, aber nicht vorhanden sind, oder wenn Kennsätze nicht angegeben, aber vorhanden sind.

Ist das Ende der Datei erreicht und sind keine Kennsätze für die Datei angegeben, findet keine Kennsatzverarbeitung statt. Wird das Ende der Datei nicht erreicht, werden die vom Laufzeitsystem abhängigen Abschlußoperationen ausgeführt; es erfolgt jedoch keine Abschlußkennsatzverarbeitung.

Eingabe- und Ein-Ausgabedateien (wahlfreier oder dynamischer Zugriffsmodus); Ausgabedateien (wahlfreier, dynamischer oder sequentieller Zugriffsmodus):

Ist LABEL RECORDS für die Datei angegeben, werden die Kennsätze entsprechend den Kennsatzkonventionen des Betriebssystems verarbeitet. Das Verhalten der CLOSE-Anweisung ist undefiniert wenn LABEL RECORDS angegeben ist, aber keine Kennsätze vorhanden sind, oder wenn LABEL RECORDS nicht angegeben ist, aber Kennsätze vorhanden sind. Ist LABEL RECORDS für die Datei nicht angegeben, findet keine Kennsatzverarbeitung statt. Es werden Abschlußoperationen in Abhängigkeit vom Ausführungszeitsystem durchgeführt.

B = Dateisperrung:

Die Datei kann während des aktuellen Programmlaufs nicht wieder eröffnet werden.

3. Eine eröffnete Datei muß geschlossen werden, bevor eine STOP RUN-Anweisung ausgeführt wird. Eine eröffnete Datei in einem aufgerufenen Programm muß geschlossen werden, bevor eine CANCEL-Anweisung für dieses Programm ausgeführt wird.
4. Wurde eine CLOSE-Anweisung für eine Datei ausgeführt, kann keine andere Anweisung, die explizit oder implizit auf diese Datei Bezug nimmt, ausgeführt werden, bevor nicht eine erneute OPEN-Anweisung ausgeführt worden ist.
5. Nach erfolgreicher Ausführung einer CLOSE-Anweisung steht der der Datei zugeordnete Satzbereich nicht mehr zur Verfügung. Nach erfolgreicher Ausführung einer CLOSE-Anweisung ist die Verfügbarkeit des Satzbereiches undefiniert.
6. Ist WITH LOCK angegeben, kann die Datei während des aktuellen Ablaufs der Ausführungseinheit nicht wieder eröffnet werden.

Die COMMIT-Anweisung

Funktion

Die COMMIT-Anweisung wird bei Satzsperrung bei indizierten Dateien verwendet. Sie hebt Satzsperren auf und legt einen Ruhepunkt fest.

Allgemeines Format

```
COMMIT
```

Allgemeine Regel

Diese Anweisung gilt nur für Dateien mit Mehrsatzsperrung. Siehe dazu LII COBOL Bedienungsanleitung Abschnitt 7.4.4.

DELETE-Anweisung

Funktion

Durch die DELETE-Anweisung wird ein Datensatz einer Plattenspeicherdatei logisch gelöscht.

Allgemeines Format

```
DELETE dateiname RECORD [; INVALID KEY unbedingte-anweisung]
```

Syntaxregeln

1. Für eine DELETE-Anweisung, die sich auf eine Datei mit sequentiellen Zugriffsmodus bezieht, darf INVALID KEY nicht angegeben werden.
2. Für eine DELETE-Anweisung, die sich auf eine Datei mit nicht sequentiellem Zugriffsmodus bezieht, und für die keine anwendbare USE-Prozedur angegeben ist, muß INVALID KEY angegeben werden.

Allgemeine Regeln

1. Die in der DELETE-Anweisung angesprochene Datei muß sich während der Ausführung dieser Anweisung im Eröffnungsmodus I-O befinden (vgl. OPEN-Anweisung in diesem Kapitel).
2. Bei einer Datei, die sich im sequentiellen Zugriffsmodus befindet, muß die der DELETE-Anweisung vorausgegangene Ein-Ausgabe-Anweisung eine erfolgreich ausgeführte READ-Anweisung gewesen sein. Das Betriebssystem löscht logisch den durch die READ-Anweisung gelesenen Datensatz aus der Datei.
3. Bei einer Datei mit wahlfreiem oder dynamischen Zugriffsmodus löscht das Betriebssystem logisch den Datensatz, der durch den Inhalt des in der RELATIVE KEY-Klausel der Datei angegebenen Datenfeldes beschrieben wurde. Falls der durch den Schlüssel angegebene Datensatz nicht in der Datei ist, besteht eine INVALID KEY-Bedingung (vgl. INVALID KEY-Bedingung in diesem Kapitel).

4. Nach erfolgreicher Ausführung einer DELETE-Anweisung ist der in Frage kommende Datensatz logisch aus der Datei gelöscht, es kann nicht mehr auf ihn zugegriffen werden.
5. Die Ausführung einer DELETE-Anweisung beeinflußt nicht den Inhalt des zur Datei gehörenden Datensatzbereichs.
6. Der aktuelle Satzzeiger wird durch die Ausführung einer DELETE-Anweisung nicht betroffen.
7. Die Ausführung der DELETE-Anweisung wird der Inhalt des für die Datei in der FILE STATUS-Klausel angegebenen Datenfeldes fortgeschrieben (vgl. Ein-Ausgabezustand in diesem Kapitel).

OPEN-Anweisung

Funktion

Die OPEN-Anweisung eröffnet Dateien für die Verarbeitung. Sie führt die Prüfung, das Beschreiben von Kennsätzen und andere Ein-Ausgabe-Operationen durch.

Allgemeines Format

```
        ⎡INPUT  dateiname-1 [,dateiname-2] ...⎤
OPEN    ⎢OUTPUT dateiname-2 [,dateiname-4] ...⎥ ...
        ⎣I-O    dateiname-3 [,dateiname-6] ...⎦
```

Syntaxregeln

Die in der OPEN-Anweisung benannten Dateien müssen nicht alle die gleiche Organisation oder den gleichen Zugriff haben.

Allgemeine Regeln

1. Die erfolgreiche Ausführung einer OPEN-Anweisung bestimmt die Verfügbarkeit der Datei und führt dazu, daß sich die Datei im OPEN-Modus befindet.
2. Nach erfolgreicher Ausführung einer OPEN-Anweisung steht der zugehörige Datensatzbereich dem Programm zur Verfügung.
3. Vor der erfolgreichen Ausführung einer OPEN-Anweisung für eine Datei darf keine Anweisung ausgeführt werden, die explizit oder implizit auf die Datei Bezug nimmt.

4. Vor Ausführung einer der zulässigen Ein-Ausgabe-Anweisungen muß eine OPEN-Anweisung erfolgreich ausgeführt worden sein. In nachfolgender Tabelle sind die für einen OPEN-Modus zulässigen Anweisungen aufgeführt. Ein X im Schnittpunkt bedeutet erlaubt.

Dateizu-griffs-modus	Anweisung	OPEN-Modus		
		INPUT	OUTPUT	I-O
sequen-tiell	READ	X		X
	WRITE		X	
	REWRITE			X
	START	X		X
	DELETE			X
direkt	READ	X		X
	WRITE		X	X
	REWRITE			X
	START			
	DELETE			X
dynamisch	READ	X		X
	WRITE		X	X
	REWRITE			X
	START	X		X
	DELETE			X

Tabelle 6-2 Zulässige Kombinationen von Anweisungen und OPEN-Modus für indizierte I-O.

5. Alle Angaben INPUT, OUPUT, oder I-O sind bei verschiedenen OPEN-Anweisungen für eine Datei im selben Programm möglich. Bevor eine weitere OPEN-Anweisung nach der logisch ersten für dieselbe Datei in einem Programm durchlaufen werden kann, muß eine CLOSE-Anweisung durchlaufen worden sein.

6. Die Ausführung der OPEN-Anweisung bewirkt nicht, daß der erste Datensatz zur Verfügung steht oder daß er übergeben wird.

7. Der mit SELECT ASSIGN angegebene Dateiname wird wie folgt verarbeitet:

   a) INPUT bedeutet, daß die Datei als Eingabedatei verarbeitet werden soll. Ist INPUT angegeben, verursacht die Ausführung einer OPEN-Anweisung, daß der in der ASSIGN-Klausel angegebene Name entsprechend den Betriebssystemkonventionen für das Eröffnen von Eingabedateien überprüft wird.

   b) OUTPUT bedeutet, daß die Datei als Ausgabedatei verarbeitet werden soll. Ist OUTPUT angegeben, verursacht die OPEN-Anweisung, daß der in der ASSIGN-Klausel angegebene Name entsprechend den Betriebssystemkonventionen für das Eröffnen von Ausgabedateien geschrieben wird.

8. Die Dateibeschreibung für dateiname-1, dateiname-2, dateiname-5, dateiname-6 muß äquivalent zu der sein, die bei der Erstellung der Datei verwendet wurde.

9. Bei Dateien, die mit INPUT oder I-O eröffnet wurden, verursacht die OPEN-Anweisung, daß der aktuelle Satzzeiger auf den ersten Datensatz der Datei zeigt. Bei indizierten Dateien ist der primäre Datensatz der Bezugsschlüssel, der dazu verwendet wird, den ersten Datensatz zu bestimmen, auf den zugegriffen werden soll. Sind keine Datensätze in der Datei vorhanden, wird der aktuelle Satzzeiger so gesetzt, daß die nächste ausgeführte READ-Anweisung vom Format 1 für die Datei eine AT END-Bedingung zur Folge hat. Ist die Datei nicht vorhanden, verursacht OPEN INPUT eine Fehleranzeige.

10. I-O bedeutet, daß die Datei für Eingabe- und Ausgabeoperationen eröffnet werden soll. Ist die Datei nicht vorhanden, wird sie erstellt. Beim sequentiellen Zugriffsmodus wird sie dann für Eingabeoperationen verwendet. Jeder Versuch eines Schreibzugriffs führt zu einem Fehler.

11. Ist I-O angegeben und zeigt die LABEL RECORDS-Klausel an, daß Kennsätze vorhanden sind, werden bei der Ausführung der OPEN-Anweisung folgende Schritte durchlaufen:

    a) Die Kennsätze werden entsprechend den im Betriebssystemkonventionen für die Ein-Ausgabe von Kennsätzen überprüft.

    b) Die neuen Kennsätze werden entsprechend den Betriebssystemkonventionen für das Schreiben von Ein-Ausgabe-Kennsätzen geschrieben.

12. Bei erfolgreicher Ausführung einer OPEN-Anweisung mit OUTPUT-Angabe wird eine Datei erstellt. Zu diesem Zeitpunkt enthält die zugehörige Datei keine Datensätze. Existiert schon eine Datei mit dem gleichen Namen, wird sie gelöscht. Ist diese aber schreibgeschützt, tritt ein Fehler auf.

READ-Anweisung

Funktion

Bei sequentiellem Zugriff stellt die READ-Anweisung den nächsten logischen Datensatz aus einer Datei zur Verfügung. Bei wahlfreiem Zugriff stellt die READ-Anweisung einen angegebenen Datensatz aus der Plattenspeicherdatei zur Verfügung.

Allgemeine Formate

Format 1

```
READ dateiname [NEXT] RECORD [INTO bezeichner]
  [; WITH [KEPT] LOCK]
  [; AT END unbedingte-anweisung]
```

Format 2

```
READ dateiname RECORD [INTO bezeichner]
  [; KEY IS datenname]
  [; WITH [KEPT] LOCK]
  [; INVALID KEY unbedingte-anweisung]
```

Syntaxregeln

1. Die INTO-Angabe darf nicht verwendet werden, wenn die Datensatzbeschreibung für die Eingabedatei variable logische Datensätze enthält. Der durch bezeichner angegebene Speicherbereich darf nicht derselbe sein, wie der durch dateiname angegebene Datensatzbereich.
2. datenname muß der Name eines Datenfeldes sein, das als Datensatzschlüssel definiert ist, der wiederum mit dateiname verknüpft ist.

3. datenname kann gekennzeichnet sein.
4. Format 1 ohne NEXT-Angabe muß für alle Dateien im sequentiellen Zugriffsmodus verwendet werden.
5. Format 1 mit NEXT-Angabe muß für alle Dateien im dynamischen Zugriffsmodus verwendet werden, wenn Datensätze sequentiell wieder aufgefunden werden sollen.
6. Format 2 muß für alle Dateien im wahlfreien oder dynamischen Zugriffsmodus verwendet werden, wenn Datensätze wahlfrei wieder aufgefunden werden sollen.
7. Ist für dateiname keine USE-Prozedur spezifiziert, muß INVALID KEY oder AT END angegeben werden.

Allgemeine Regeln

1. Bevor eine READ-Anweisung für eine Datei durchgeführt werden kann, muß diese Datei mit INPUT oder I-O eröffnet worden sein (vgl. OPEN-Anweisung).
2. Der durch die READ-Anweisung im Format 1 zur Verfügung gestellte Datensatz wird wie folgt bestimmt:

   a) Der Datensatz, auf den der aktuelle Satzzeiger zeigt, wird zur Verfügung gestellt, wenn der aktuelle Satzzeiger durch eine START- oder OPEN-Anweisung positioniert worden ist und wenn auf den Datensatz nach wie vor auf dem Weg zugegriffen werden kann, den der aktuelle Satzzeiger anzeigt.

      Kann auf den Datensatz nicht mehr zugegriffen werden, weil er z.B. gelöscht ist oder der alternative Satzschlüssel geändert worden ist, wird der aktuelle Satzzeiger so fortgeschrieben, daß er auf den nächsten existierenden Datensatz in der Datei zeigt. Dieser Datensatz wird dann zur Verfügung gestellt.

   b) War der aktuelle Satzzeiger bei Ausführung einer vorhergehenden READ-Anweisung gesetzt, wird er so fortgeschrieben, daß er mit dem festgesetzten Bezugsschlüssel auf den nächsten vorhandenen Datensatz in dieser Datei zeigt. Dieser Datensatz wird dann zur Verfügung gestellt.

3. Die Ausführung einer READ-Anweisung bewirkt, daß der Inhalt des Datenfeldes fortgeschrieben wird, das in der FILE STATUS-Klausel der zugehörigen Dateibeschreibung angegeben wurde.

4. Ohne Rücksicht auf die verwendete Methode, Zugriffszeit mit Ausführungszeit zu überlagern, bleibt das Konzept der READ-Anweisung insofern unverändert, als ein Datensatz solange für das Zielprogramm verfügbar ist, bis eine andere der READ-Anweisung folgende Anweisung ausgeführt wird.

5. Wenn eine Datei mehr als einen logischen Datensatztyp enthält, teilen sich diese Datensätze automatisch den gleichen Speicherbereich. Dies entspricht einer impliziten Redefinition des Speicherbereiches. Der Inhalt von Datenfeldern, die außerhalb des Bereichs des aktuellen Datensatzes liegen, sind bei Abschluß der READ-Anweisung undefiniert.

6. Ist INTO angegeben, wird der gelesene Datensatz aus dem Datensatzbereich in den durch bezeichner angegebenen Bereich übertragen (impliziter MOVE). Die Übertragung findet entsprechend den Regeln für die MOVE-Anweisung ohne CORRESPONDING-Angabe statt. Der implizite MOVE wird nicht ausgeführt, wenn die Ausführung der READ-Anweisung erfolglos war.

   Jede mit bezeichner zusammenhängende Subskribierung oder Indizierung wird ausgewertet, nachdem der Datensatz gelesen wurde und unmittelbar bevor er in das Datenfeld übertragen wird.

7. Ist INTO angegeben, steht der gelesene Datensatz sowohl im Eingabebereich als auch in den durch bezeichner festgelegten Bereich zur Verfügung.

8. Ist zur Ausführungszeit einer READ-Anweisung vom Format 1 die Position des aktuellen Satzzeigers für diese Datei undefiniert, ist die Ausführung der READ-Anweisung erfolglos.

9. Ist bei Ausführung einer READ-Anweisung vom Format 1 kein nächster logischer Datensatz in der Datei vorhanden, tritt die AT END-Bedingung auf. Die Ausführung der READ-Anweisung wird als erfolglos betrachtet (vgl. Ein-Ausgabezustand).

10. Wird die AT END-Bedingung erkannt, laufen die folgenden Schritte in der angegebenen Reihenfolge ab:

    a) Der Inhalt des Datenfeldes aus der FILE STATUS-Klausel, falls für diese Datei angegeben, wird mit einem Wert versorgt, der die AT END-Bedingung anzeigt.

    b) Ist in der READ-Anweisung AT END angegeben, wird auf die unbedingte Anweisung nach der AT END-Angabe verzweigt. Eine für die Datei vorhandene USE-Prozedur wird nicht durchlaufen.

    c) Ist AT END in der READ-Anweisung nicht angegeben, muß entweder explizit oder implizit eine USE-Prozedur für diese Datei vorliegen. Diese Prozedur wird dann durchlaufen.

    Bei Auftreten der AT END-Bedingung ist die Ausführung der entsprechenden Ein-Ausgabe-Anweisung erfolglos.

11. Nach der erfolglosen Ausführung einer READ-Anweisung sind der Inhalt des zur Datei gehörigen Datensatzbereiches und die Position des aktuellen Satzzeigers undefiniert.
    Der Bezugsschlüssel ist ebenfalls undefiniert.

12. Nach Auftreten der AT END-Bedingung darf für diese Datei keine READ-Anweisung vom Format 1 gegeben werden, bevor nicht eine der folgenden Aktionen ausgeführt worden ist:

    a) eine erfolgreiche CLOSE-Anweisung, gefolgt von der Ausführung einer erfolgreichen OPEN-Anweisung für diese Datei,

    b) eine erfolgreiche START-Anweisung für diese Datei,

    c) eine erfolgreiche READ-Anweisung vom Format 2 für diese Datei.

13. Bei einer Datei im dynamischen Zugriffsmodus wird durch eine READ-Anweisung vom Format 1 mit NEXT-Angabe der nächste logische Datensatz in der Datei wieder aufgefunden (vgl. Allgemeine Regel 2).

14. Bei einer indizierten Datei, auf die sequentiell zugegriffen wird, stehen die Datensätze, die den gleichen Verdoppelungswert in einem alternativen Datensatzschlüssel (Bezugsschlüssel) haben, in der gleichen Reihenfolge, in der sie durch Ausführung von WRITE- oder REWRITE-Anweisungen, die solche Verdoppelungswerte erstellen, verfügbar werden.

15. Ist KEY in einer READ-Anweisung vom Format 2 nicht angegeben, wird der primäre Datensatzschlüssel als Bezugsschlüssel für das Wiederauffinden festgelegt. Ist der dynamische Zugriffsmodus angegeben, wird dieser Bezugsschlüssel auch zum Wiederauffinden durch nachfolgende READ-Anweisungen vom Format 1 für diese Datei verwendet.

16. Ist KEY in einer READ-Anweisung vom Format 2 einer indizierten Datei angegeben, wird datenname als Bezugsschlüssel für das Wiederauffinden festgelegt. Ist ACCESS IS DYNAMIC angegeben, wird dieser Bezugsschlüssel zum Wiederauffinden durch nachfolgende READ-Anweisungen vom Format 1 solange für diese Datei verwendet, bis ein anderer Bezugsschlüssel für diese Datei erstellt wird.

17. Die Ausführung einer READ-Anweisung vom Format 2 hat zur Folge, daß der Wert des Bezugsschlüssels so lange mit dem Wert im entsprechenden Datenfeld der gespeicherten Datensätze in der Datei verglichen wird, bis der erste Datensatz mit einem gleichen Wert gefunden wird. Der aktuelle Satzzeiger wird auf diesen Datensatz positioniert und dann zur Verfügung gestellt. Kann ein Datensatz auf diese Weise nicht identifiziert werden, tritt die INVALID KEY-Bedingung ein. Die Ausführung der READ-Anweisung ist erfolglos (vgl. die INVALID KEY-Bedingung in diesem Kapitel).

18. Die WITH KEPT LOCK-Angabe dient zum Sperren von Datensätzen. Siehe dazu LII COBOL Bedienungsanleitung Abschnitt 7.4.

REWRITE-Anweisung

Funktion

Die REWRITE-Anweisung ersetzt einen Datensatz logisch, der in einer Plattenspeicherdatei steht.

Allgemeines Format

```
REWRITE datensatzname [FROM bezeichner] [; INVALID KEY unbedingte-anweisung]
```

Syntaxregeln

1. datensatzname und bezeichner dürfen sich nicht auf den gleichen Speicherbereich beziehen.
2. datensatzname ist der Name eines logischen Datensatzes in der FILE SECTION der DATA DIVISION und kann gekennzeichnet sein.
3. Ist keine USE-Prozedur für Dateien im wahfreien oder dynamischen Zugriffsmodus vorhanden, muß INVALID KEY in der REWRITE-Anweisung angegeben sein.

Allgemeine Regeln

1. Die mit datensatzname verknüpfte Datei muß zur Ausführungszeit dieser Anweisung mit OPEN I-O eröffnet worden sein (vgl. OPEN-Anweisung in diesem Kapitel).

2. Für Dateien im sequentiellem Zugriffsmodus gilt:

   Eine REWRITE-Anweisung muß eine erfolgreich abgelaufene READ-Anweisung vorangegangen sein und zwar als letzte Ein-Ausgabeanweisung für die zugehörige Datei.
   Der Datensatz, auf den durch die READ-Anweisung zugegriffen worden ist, wird vom Betriebssystem logisch ersetzt.

3. Die Anzahl der Zeichenpositionen in dem durch datensatzname bezeichneten Datensatz muß mit der Anzahl der Zeichenpositionen in dem zu ersetzenden Datensatz übereinstimmen.

4. Der logische Datensatz, der durch eine erfolgreich abgelaufene REWRITE-Anweisung zurückgeschrieben wurde, steht im Satzbereich nicht mehr zur Verfügung; eine Ausnahme stellt die Verwendung der SAME RECORD AREA-Klausel dar. In diesem Fall steht der logische Datensatz dem Programm zur Verfügung, sowohl als Datensatz aller anderen Dateien, die in derselben SAME RECORD AREA-Klausel aufgeführt wurden, wie die zugehörigen Ein-Ausgabedateien, als auch der mit datensatzname verknüpften Datei.

5. Die Ausführung einer REWRITE-Anweisung mit der FROM-Angabe entspricht den folgenden Anweisungen:

   MOVE bezeichner TO datensatzname.
   REWRITE datensatzname.

   Der Inhalt des Speicherbereichs, beschrieben durch datensatzname, vor Ausführung der impliziten MOVE-Anweisung, hat keine Bedeutung für die Ausführung der REWRITE-Anweisung.

6. Durch die Ausführung einer REWRITE-Anweisung wird der aktuelle Satzzeiger nicht beeinflußt.

7. Die Ausführung einer REWRITE-Anweisung bewirkt, daß der Inhalt des in der FILE STATUS-Klausel angegebenen Datenfeldes fortgeschrieben wird (vgl. Ein-Ausgabezustand in diesem Kapitel).

8. Bei Dateien im sequentiellen Zugriffsmodus wird der zu ersetzende Datensatz durch den Wert des primären Datensatzschlüssels angegeben. Wird die REWRITE-Anweisung ausgeführt, muß der Wert im primären Datensatzschlüssel des zu ersetzenden Datensatzes der gleiche sein wie der Wert des primären Datensatzschlüssels des zuletzt aus dieser Datei gelesenen Datensatzes.

9. Bei Dateien im wahlfreien oder dynamischen Zugriffsmodus wird der zu ersetzende Datensatz durch das Datenfeld angegeben, das den primären Datensatzschlüssel enthält.

10. Der Inhalt des ALTERNATIVE RECORD KEY-Datenfeldes des ersetzten Datensatzes muß nicht mit dem zu ersetzenden Datensatz übereinstimmen. Das Betriebssystem benutzt den Inhalt des RECORD KEY-Datenfeldes während der Ausführung einer REWRITE-Anweisung so, daß nachfolgende Ausführungen des Datensatzes auf irgendeinem der angegebenen Satzschlüssel basieren können.

11. Die INVALID KEY-Bedingung tritt auf:

    a) Wenn der Zugriffsmodus sequentiell ist und der Wert, der im primären Datensatzschlüsselfeld des zu ersetzenden Datensatzes enthalten ist, nicht dem Wert des primären Datensatzschlüssels des letzten aus dieser Datei gelesenen Datensatzes entspricht.

    b) Wenn der Wert des primären Datensatzschlüsselfeldes nicht mit einem in dieser Datei gespeicherten Datensatz übereinstimmt.

    c) Wenn der Wert eines alternativen Datensatzschlüsselfeldes, für das DUPLICATES nicht angegeben wurde, mit einem in der Datei enthaltenen Datensatz übereinstimmt.

    Es wird nicht fortgeschrieben und die Daten im Datensatzbereich werden nicht beeinflußt (vgl. INVALID KEY-Bedingung in diesem Kapitel).

Die ROLLBACK-Anweisung

Funktion:

Die Rollback-Anweisung gilt für Mehrsatzsperrung bei indizierten Dateien. Sie bewirkt, daß nach der Änderung eines Satzes andere Benutzer auf diesen Satz nur zum Lesen zugreifen können.

Format:

```
ROLLBACK
```

Allgemeine Regeln

Siehe dazu die LII COBOL Bedienungsanleitung Abschnitt 7.4.4.

START-Anweisung

Funktion

Die START-Anweisung bietet eine Voraussetzung für logisches Positionieren in einer indizierten Datei und für das aufeinanderfolgende sequentielle Lesen von Datensätzen.

Allgemeines Format

```
START dateiname [ KEY { [ IS   EQUAL TO              ] } datenname ]
                      [ IS =                         ]
                      [ IS > GREATER THAN            ]
                      [                              ]
                      [ IS >                         ]
                      [ IS   NOT LESS THAN           ]
                      [ IS   NOT <                   ]

    [; INVALID KEY unbedingte-anweisung]
```

Anmerkung

Die erforderlichen Vergleichszeichen '>' und '<?' und '=' sind nicht unterstrichen, um Verwechslungen mit anderen Symbolen wie '≥' (größer gleich) zu vermeiden.

Syntaxregeln

1. dateiname muß der Name einer indizierten Datei sein.
2. dateiname muß eine Datei im dynamischen oder sequentiellen Zugriffsmodus bezeichnen.
3. datenname kann gekennzeichnet sein.
4. Die INVALID KEY-Bedingung muß vorhanden sein, falls keine entsprechende USE-Prozedur angegeben ist.
5. Ist KEY angegeben und dateiname der Name einer indizierten Datei, kann datenname sich beziehen auf:
   - ein Datenfeld, das als Datensatzschlüssel angegeben ist, der mit dateiname verknüpft ist;
   - ein alphanumerisches Datenfeld, das dem Datennamen eines Datensatzschlüssel untergeordnet ist, der mit dateiname verknüpft ist und dessen am weitesten links stehende Zeichenposition der am weitesten links stehenden Zeichenposition des Datenfeldes dieses Datensatzschlüssels entspricht.

Allgemeine Regeln

1. Die durch dateiname angegebene Datei muß zum Zeitpunkt der Ausführung der START-Anweisung durch INPUT oder I-O eröffnet sein (vgl. OPEN-Anweisung in diesem Kapitel).
2. Ist eine KEY-Angabe nicht vorhanden, wird 'IS EQUAL TO' angenommen.
3. Die Art des Vergleichs wird durch den Vergleichsoperator in der KEY-Angabe festgelegt. Der Schlüssel eines Satzes in der durch dateiname bezeichneten Datei wird mit dem Inhalt des durch datenname bezeichneten Datenfeldes verglichen
   (vgl. auch Allgemeine Regel 5).
   a) Der aktuelle Satzzeiger ist auf den ersten Datensatz positioniert, der die Vergleichsbedingung erfüllt.
   b) Falls für keinen Datensatz der Datei die Vergleichsbedingung erfüllt ist, tritt eine INVALID KEY-Bedingung auf. Die Ausführung der START-Anweisung ist erfolglos und die Position des aktuellen Satzzeigers undefiniert
   (vgl. INVALID KEY-Bedingung in diesem Kapitel).

4. Ist 'FILE STATUS' angegeben, wird nach Ausführung einer START-Anweisung der Inhalt des FILE STATUS-Datenfeldes fortgeschrieben (vgl. Ein-Ausgabezustand in diesem Kapitel).
   Bezeichnet dateiname eine indizierte Datei und haben die Operanden verschiedene Größen, wird so verglichen, als ob der längere der Operanden rechts abgeschnitten würde, so daß seine Größe dem des kürzeren entspricht.
   Es gelten alle anderen Regeln für nichtnumerischen Vergleich. Ausnahme: Ist eine PROGRAM COLLATING SEQUENCE-Klausel vorhanden, so hat das keine Auswirkung auf den Vergleich (vgl. Vergleich von nichtnumerischen Operanden).

5. Ist KEY angegeben, wird der Vergleich mit dem Datenfeld durchgeführt, auf das sich datenname bezieht (wie in Allgemeine Regel 3 beschrieben).

6. Ist KEY nicht angegeben, wird der Vergleich mit dem mit dateiname verknüpften RECORD KEY-Datenfeld durchgeführt (wie in Allgemeine Regel 3 beschrieben).

7. Nach Ausführung einer erfolgreichen START-Anweisung wird ein Bezugsschlüssel festgelegt und in nachfolgenden READ-Anweisungen vom Format 1 wie folgt verwendet (vgl. READ-Anweisung in diesem Kapitel).

   a) Ist KEY nicht angegeben, wird als Bezugsschlüssel der für dateiname angegebene primäre Datensatzschlüssel verwendet.

   b) Ist KEY angegeben und datenname als Datensatzschlüssel für dateiname spezifiziert, wird dieser Datensatzschlüssel als Bezugsschlüssel verwendet.

   c) Ist KEY angegeben und datenname nicht als Datensatzschlüssel für dateiname spezifiziert, wird der Datensatzschlüssel als Bezugsschlüssel verwendet, dessen am weitesten links stehende Zeichenposition der am weitesten links stehenden Zeichenposition des durch datenname angegebenen Datenfeldes entspricht.

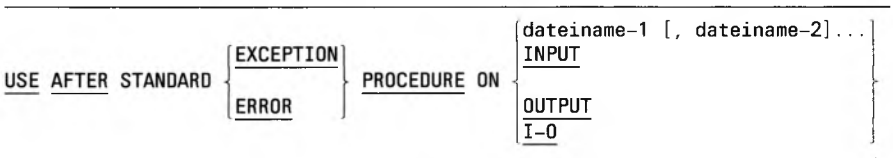
8. Ist die Ausführung einer START-Anweisung erfolglos, ist der Bezugsschlüssel undefiniert.

USE-Anweisung

Funktion

Die USE-Anweisung gibt zusätzlich zu den Standardprozeduren der Ein-Ausgabesteuerung Prozeduren für die Behandlung von Ein-Ausgabefehlern an.

Allgemeines Format



Syntaxregeln

1. Ist die USE-Anweisung angegeben, muß sie innerhalb der Prozedurvereinbarungen (DECLARATIVES) stehen. Der erste Eintrag in den Prozedurvereinbarungen ist die Kapitelüberschrift und zwar: Kapitelname, gefolgt von dem Wort SECTION und einem Punkt. Unmittelbar anschließend an die Kapitelüberschrift muß die USE-Anweisung geschrieben werden. Danach können Paragraphen folgen, die die Prozedur definieren.

2. Die USE-Anweisung selbst wird nicht ausgeführt. Sie vereinbart Prozeduren, die durchlaufen werden sollen, wenn ein Ein-Ausgabefehler für eine Datei auftritt.

3. Da das Format variiert werden kann, kann in verschiedenen USE-Anweisungen der gleiche Dateiname auftreten. Dies darf jedoch nicht den gleichzeitigen Aufruf mehrerer Prozedurvereinbarungen bewirken.

4. Die Wörter ERROR und EXCEPTION sind gleichwertig und können alternativ verwendet werden.

5. Die Dateien, die implizit angesprochen werden (INPUT, OUTPUT, I-O, EXTEND) oder explizit (dateiname-1, dateiname-2,...) brauchen nicht dieselbe Organisation und denselben Zugriff zu haben.

Allgemeine Regeln

1. Die USE-Prozeduren werden durchlaufen bei Auftreten einer AT END- oder INVALID KEY-Bedingung, wenn die Ein-Ausgabeanweisung, bei der die AT END- oder INVALID KEY-Bedingung auftrat, keine AT END- oder INVALID-Angabe enthält.

2. Nach Ausführung einer USE-Prozedur wird die Steuerung an die aufrufende Routine zurückgegeben.

3. Innerhalb einer USE-Prozedur darf nur auf Prozeduren außerhalb der Prozedurvereinbarungen (DECLARATIVES) mit der PERFORM-Anweisung zugegriffen werden.
   Auf Prozedurnamen, die einer USE-Anweisung untergeordnet sind, darf aus einer anderen Prozedur oder außerhalb der Prozedurvereinbarungen (DECLARATIVES) nur mit einer PERFORM-Anweisung Bezug genommen werden.

4. Innerhalb einer USE-Prozedur darf keine Anweisung ausgeführt werden, die eine Ausführung einer USE-Prozedur verursachen würde, die ihrerseits die Steuerung noch nicht an die aufrufende Routine zurückgegeben hatte.

WRITE-Anweisung

Funktion

Die WRITE-Anweisung stellt einen logischen Datensatz für eine Ausgabe- oder Ein-Ausgabedatei zur Verfügung.

Allgemeines Format

```
WRITE datensatzname [FROM bezeichner]
      [; INVALID KEY unbedingte-anweisung]
```

Syntaxregeln

1. datensatzname und bezeichner dürfen nicht den selben Speicherbereich belegen.
2. datensatzname ist der Name eines logischen Datensatzes in der FILE SECTION der DATA DIVISION und kann gekennzeichnet werden.
3. INVALID KEY muß angegeben werden, wenn keine entsprechende USE-Prozedur für die Datei vereinbart wurde.

Allgemeine Regeln

1. Bei Ausführung einer WRITE-Anweisung muß die Datei mit OUTPUT oder I-O eröffnet sein (vgl. OPEN-Anweisung in diesem Kapitel).

2. Der durch eine WRITE-Anweisung ausgegebene Datensatz steht im Datensatzbereich nur dann zur Verfügung, wenn:

   - die zum Datensatz gehörende Datei in einer SAME AREA-Klausel angegeben ist oder wenn
   - die Ausführung der WRITE-Anweisung aufgrund einer INVALID KEY-Bedingung erfolglos war.

   Der logische Datensatz ist für das Programm als Datensatz anderer Dateien ebenfalls verfügbar, die in der gleichen SAME RECORD AREA-Klausel wie die zugehörigen Ausgabedateien angegeben sind und auch für die mit datensatzname verknüpften Dateien.

3. Die Ausführung einer WRITE-Anweisung mit FROM-Angabe ist gleichbedeutend mit folgenden Anweisungen:

   MOVE bezeichner TO datensatzname.
   WRITE datensatzname.

   Die Übertragung findet entsprechend den Regeln für die MOVE-Anweisung ohne CORRESPONDING-Angabe statt.

   Der Inhalt des Datensatzbereichs vor Ausführung der impliziten MOVE-Anweisung hat keinen Einfluß auf die Ausführung der WRITE-Anweisung.

   Nach erfolgreicher Ausführung der WRITE-Anweisung steht die Information in dem durch bezeichner angesprochenen Bereich zur Verfügung, auch wenn die Information in dem durch datensatzname angesprochenen Bereich nicht zur Verfügung steht (vgl. Allgemeine Regel 2).

4. Der aktuelle Satzzeiger wird durch die Ausführung einer WRITE-Anweisung nicht beeinflußt.

5. Nach Ausführung einer WRITE-Anweisung wird der Wert des Datenfeldes einer für diese Datei vorhandenen FILE STATUS-Klausel fortgeschrieben (vgl. Ein-Ausgabezustand).

6. Die maximale Datensatzgröße wird beim Erstellen der Datei festgelegt. Sie darf danach nicht mehr verändert werden.

7. Die auf einer Platte erforderliche Anzahl von Zeichenpositionen zum Speichern eines logischen Datensatzes in einer Datei muß nicht mit der für diesen Datensatz im Programm definierten Anzahl übereinstimmen.

8. Durch die Ausführung der WRITE-Anweisung wird dem Betriebssystem ein logischer Datensatz zur Verfügung gestellt.

9. Die Ausführung einer WRITE-Anweisung bewirkt, daß der Inhalt des Datensatzbereichs verfügbar wird. Das Betriebssystem verwendet den Inhalt des Datensatzschlüssels so, daß nachfolgende Zugriffe auf den Datensatz mittels irgendeinem dieser angegebenen Datensatzschlüssel durchgeführt werden können.

10. Der Wert des primären Datensatzschlüssels muß innerhalb der Datensätze einer Datei eindeutig sein.

11. Das Datenfeld, das als primärer Datensatzschlüssel angegeben ist, muß durch das Programm vor Ausführung der WRITE-Anweisung auf den gewünschten Wert gesetzt werden.

12. Ist für eine Datei ACCESS MODE IS SEQUENTIAL angegeben, müssen die Datensätze dem Betriebssystem in aufsteigender Reihenfolge der primären Datensatzschlüsselwerte zur Verfügung gestellt werden.

13. Ist ACCESS IS RANDOM oder DYNAMIC angegeben, werden die Datensätze dem Betriebssystem in der Reihenfolge zur Verfügung gestellt, die im Programm angegeben ist.

14. Ist für eine indizierte Datei im Dateisteuerungseintrag ALTERNATE RECORD KEY angegeben, braucht der Wert des alternativen Datensatzschlüssels nur dann nicht eindeutig sein, wenn für das Datenfeld DUPLICATES angegeben ist. In diesem Fall ermöglicht das Betriebssystem Datensätze so zu speichern, daß beim Zugriff auf die Datensätze dieselbe Reihenfolge gilt wie sie dem Betriebssystem zur Verfügung gestellt wurden.

15. Eine INVALID KEY-Bedingung wird durch folgende Gründe verursacht:

    a) wenn ACCESS IS SEQUENTIAL für eine Datei angegeben ist, die mit OPEN OUTPUT eröffnet wurde, und der Wert des Datensatzschlüsselfeldes nicht größer ist als der Wert des primären Datensatzschlüssels des vorhergehenden Datensatzes.

    b) Wenn die Datei mit OUTPUT oder I-O eröffnet ist, und der Wert des primären Datensatzschlüssels dem Wert eines primären Datensatzschlüssels eines in der Datei bereits vorhandenen Datensatzes entspricht.

    c) Wenn die Datei mit OUTPUT oder I-O eröffnet ist, und der Wert eines alternativen Datensatzschlüssels, für den ein doppelter Schlüssel nicht zulässig ist, dem entsprechenden Datenfeld eines bereits in der Datei vorhandenen Datensatzes entspricht.

    d) Wenn der Versuch unternommen wurde, über die extern definierten Grenzen einer Datei hinaus zu schreiben.

16. Tritt eine INVALID KEY-Bedingung auf, ist die Ausführung der WRITE-Anweisung erfolglos. Der Inhalt des Datensatzbereiches steht weiterhin zur Verfügung.
    Ist FILE STATUS angegeben, wird das FILE STATUS-Datenfeld auf einen Wert gesetzt, der die Ursache der INVALID KEY-Bedingung anzeigt. Die Fortsetzung des Programms hängt von den Regeln der INVALID KEY-Bedingung ab (vgl. Ein-Ausgabezustand in diesem Kapitel).

7 Sortieren und Mischen

7.1 Einführung

Das SORT-MERGE-Modul bietet Möglichkeiten:

- eine oder mehrere Dateien zu sortieren
- eine oder mehrere Dateien zu mischen

Diese Dateien enthalten Anwenderschlüssel, diese Anwenderschlüssel dienen als Kriterium für das Mischen.
Optional kann ein Anwender jeden der einzelnen Datensätze durch Eingabe- oder Ausgabe-Prozeduren besonders bearbeiten. Diese besondere Bearbeitung kann erfolgen, bevor und/oder nachdem die Datensätze durch SORT sortiert oder nachdem die Datensätze durch MERGE gemischt worden sind.

Beziehung zur sequentiellen Ein- Ausgabe

Die in den USING- und GIVING-Angaben der SORT und MERGE-Anweisungen angegebenen Dateien müssen implizit oder explizit im FILE-CONTROL Paragraph mit einer sequentiellen Organisation beschrieben sein. Es darf keine Eingabe-/Ausgabe-Anweisung für die in der SORT-MERGE-Dateibeschreibung genannte Datei ausgeführt werden.

7.2 ENVIRONMENT DIVISION im SORT-MERGE Modul

INPUT-OUTPUT-SECTION

Der FILE-CONTROL Paragraph

Funktion

Der FILE-CONTROL Paragraph benennt jede Datei und ermöglicht die Angabe anderer dateibezogener Information.

Allgemeines Format

```
FILE CONTROL.      { dateieintragung}   ...
```

Dateibeschreibung

SELECT-Klausel

Funktion

Die Datei-Beschreibung benennt eine Sortier- oder Misch-Datei und führt die Verknüpfung der Datei mit einem Speichermedium ein.

Allgemeines Format

```
SELECT dateiname

ASSIGN TO {externes dateiname-literal} [ [,] {externes dateiname-literal} ]...
          {dateibezeichner           }       {dateibezeichner           }
```

Syntaxregeln

1. Jede in der DATA DIVISION beschriebene Sortier- oder Misch-Datei muß im FILE-CONTROL Paragraph genau einmal als Dateiname benannt sein. Jede in der FILE-CONTROL-Eintragung angegeben Sortier- oder Mischdatei muß eine SORT-MERGE-Dateibeschreibung in der DATA DIVISION besitzen.

2. Wenn dateiname für eine Sortier- oder Misch-Datei steht, darf nach dateiname im FILE-CONTROL Paragraph nur die ASSIGN-Klausel stehen.

   Die Erstzuweisung ist maßgebend. Nachfolgende Zuweisungen werden nur zur Dokumentation angegeben.

Allgemeine Regel

Die ASSIGN-Klausel spezifiziert die Verknüpfung der durch dateiname benannten Sortier- oder Misch-Datei mit einem Speichermedium.

Der I-O-CONTROL-Paragraph

Funktion

Der I-O-CONTROL Paragraph gibt den Speicherbereich an, der von den verschiedenen Dateien gemeinsam benutzt wird.

Allgemeines Format

```
I-O-CONTROL

     [ ; SAME { RECORD     }     AREA FOR dateiname-1
              { SORT       }
              { SORT-MERGE }

                { , dateiname-2 } ... ] ... .
```

Syntaxregeln

1. Der I-O-CONTROL-Paragraph ist optional.
2. In der SAME AREA-Klausel sind SORT und SORT-MERGE gleichbedeutend.
3. Wenn die SAME SORT AREA oder SAME SORT-MERGE AREA-Klausel verwendet wird, muß mindestens einer der Dateinamen für eine Sortier- oder Misch-Datei stehen. Dateien, die keine Sortier- oder Misch-Dateien darstellen, können in der Klausel ebenfalls benannt werden.
4. Die drei Formate der SAME-Klausel (SAME RECORD AREA, SAME SORT AREA, und SAME SORT-MERGE AREA) werden nachfolgend getrennt behandelt:

In einem Programm kann mehr als nur eine SAME-Klausel enthalten sein. Dann ist jedoch zu beachten:

a) ein Dateiname darf nicht in mehr als einer SAME RECORD AREA-Klausel erscheinen,

b) ein Dateiname, der für eine Sortier- oder Misch-Datei steht, darf nicht in mehr als einer SAME SORT AREA oder SAME SORT-MERGE AREA-Klausel erscheinen,

c) es müssen alle in der SAME AREA-Klausel genannten Dateien in der SAME SORT AREA oder SAME SORT-MERGE AREA-Klausel benannt werden, wenn ein Dateiname

- nicht für eine Sortier-Mischdatei steht und
- in einer SAME AREA-Klausel und in ein oder mehreren SAME SORT oder SAME SORT-MERGE AREA-Klauseln erscheint.

5. Die in der SAME SORT AREA, SAME SORT-MERGE AREA oder SAME RECORD AREA-Klausel benannten Dateien müssen nicht alle die gleiche Organisation oder die gleiche Zugriffart haben.

Allgemeine Regeln

1. Die SAME RECORD AREA-Klausel gibt an, daß zwei oder mehrere Dateien für die Verarbeitung des aktuellen logischen Datensatzes den gleichen Speicherbereich verwenden. Alle Dateien können gleichzeitig geöffnet werden. Ein logischer Datensatz in der SAME RECORD AREA wird als ein logischer Datensatz jeder geöffneten Ausgabedatei betrachtet, deren Dateiname in dieser SAME RECORD AREA-Klausel erscheint.

   Er wird ebenfalls als Datensatz der zuletzt gelesenen Eingabedatei betrachtet, deren Dateiname in dieser SAME RECORD AREA-Klausel erscheint. Dies entspricht einer impliziten Neudefinition des Bereiches, d.h., Datensätze werden nach der am weitesten links stehenden Zeichenposition ausgerichtet.

2. Wenn die SAME SORT AREA oder SAME SORT-MERGE AREA-Klausel verwendet wird, muß mindestens einer der Dateinamen für eine Sortier- oder Misch-Datei stehen. Dateien, die keine Sortier- oder Misch-Dateien sind, können in der Klausel ebenfalls benannt sein. Durch diese Klausel wird festgelegt, wie der Speicherbereich aufgeteilt wird:

   a) die SAME SORT AREA oder SAME SORT-MERGE AREA-Klausel benennt einen Speicherbereich, der für das Sortieren oder Mischen jeder benannten Sortier- oder Misch-Datei zur Verfügung gestellt wird. Daher steht der für das Sortieren oder Mischen zugewiesene Speicherbereich zur Wiederverwendung für Sortier- oder Mischvorgänge irgendwelcher anderen Sortier- oder Misch-Dateien zur Verfügung.

   b) Ferner können Speicherbereiche, die nicht Sortier-Mischdateien zugeordnet sind, nach Bedarf für Sortier- oder Mischvorgänge der in der SAME SORT AREA oder SAME SORT-MERGE AREA-Klausel benannten Sortier- oder Misch-Dateien zugeordnet werden.

   c) Andere Dateien als die Sortier- oder Misch-Dateien teilen sich nicht den gleichen Speicherbereich. Wenn der Anwender wünscht, daß diese Dateien den gleichen Speicherbereich untereinander teilen, muß er im Programm eine SAME AREA oder SAME RECORD AREA-Klausel für diese Dateien angeben.

   d) Während der Ausführung einer SORT oder MERGE-Anweisung, die auf eine in dieser Klausel benannte Sortier- oder Misch-Datei Bezug nimmt, darf keine in dieser Klausel benannte nicht-SORT-MERGE-Datei geöffnet werden.

7.3 DATA DIVISION im SORT-MERGE Modul

FILE SECTION

Eine SD-Dateibeschreibung bietet Information über die Größe und die Namen der Datensätze, die mit der zu sortierenden oder zu mischenden Datei verknüpft sind. Es gibt keine Kennsatzprozeduren, die der Anwender steuern kann. Für Sperren und interne Speicherung im Zusammenhang mit SORT und MERGE-Anweisungen gelten besondere Regeln.

Die SORT-MERGE-Dateibeschreibung

Funktion

Die Beschreibung der SORT-MERGE-Datei liefert Information über die physikalische Struktur, Identifizierung und Datensatznamen der zu sortierenden oder zu mischenden Datei.

Allgemeines Format

```
SD dateiname

   [;RECORD CONTAINS [ganzzahl-1 TO] ganzzahl-2 CHARACTERS]

     [;DATA { RECORD IS   } datenname-1 [,datenname-2] ...].
            { RECORDS ARE }
```

Syntaxregeln

1. Die Stufenbezeichnung SD identifiziert den Anfang der SORT-MERGE-Dateibeschreibung und muß vor dem Dateinamen stehen.
2. Die Klauseln, die nach der Datei stehen, sind optional, und die Reihenfolge ihres Auftretens ist unbedeutend. Sie dienen nur zur Dokumentation.
3. Nach der SORT-MERGE-Dateibeschreibung müssen eine oder mehrere Datensatzbeschreibungen stehen; es dürfen jedoch keine Eingabe- oder Ausgabe-Anweisungen für diese Datei ausgeführt werden.

Die DATA RECORDS-Klausel

Funktion

Die DATA RECORDS-Klausel dient nur zur Dokumentation von Datennamen und der zugehörigen Datei.

Allgemeines Format

```
DATA { RECORD IS  } datenname-1 [, datenname-2] ...
     { RECORDS ARE }
```

Syntaxregel

datenname-1 und datenname-2 sind Namen von Datensätzen und müssen auf der Stufennummer 01 mit den gleichen Namen beschrieben sein.

Allgemeine Regeln

1. Die DATA RECORDS-Klausel dient nur zur Dokumentation.
2. Wenn mehr als nur ein datenname vorhanden ist, bedeutet dies, daß die Datei mehr als nur einen Datensatztyp enthält. Diese Datensätze können von verschiedener Größe, verschiedenem Format usw. sein. Die Reihenfolge, in der sie aufgelistet sind, ist unwesentlich.
3. Konzeptionell teilen sich alle Datensätze in einer Datei den gleichen Bereich. Dies wird in keinem Fall geändert, wenn mehr als ein Datensatztyp in der Datei vorhanden ist.

Die RECORD CONTAINS-Klausel

Funktion

Die RECORD CONTAINS-Klausel gibt die Größe von Datensätzen an.

Allgemeines Format

```
RECORD CONTAINS [ganzzahl-1 TO] ganzzahl-2 CHARACTERS
```

Allgemeine Regeln

1. Die RECORD CONTAINS-Klausel dient nur zur Dokumentation.
2. Die Größe jedes Datensatzes wird vollständig in der Datensatzbeschreibung definiert; die Klausel ist daher nicht erforderlich. Bei ihrer Verwendung gelten die folgenden Regeln:

   a) ganzzahl-2 darf nur allein verwendet werden, wenn alle Datensätze in der Datei die gleiche Größe haben. In diesem Fall stellt ganzzahl-2 die genaue Anzahl der Zeichen im Datensatz dar. Wenn sowohl ganzzahl-1 als auch ganzzahl-2 angegeben sind, so beziehen sie sich auf die Anzahl von Zeichen im kleinsten Datensatz bzw. auf die maximale Anzahl von Zeichen im größten Datensatz.

   b) Die Größe wird in Anzahl von Zeichenpositionen angegeben, die für die Speicherung des logischen Datensatzes erforderlich ist, ungeachtet dessen, ob alle Arten von Zeichen für die Darstellung der Datenfelder im logischen Datensatz verwendet werden. Die Größe eines Datensatzes wird durch die Summe der Zeichen in allen Datenelementen mit fester Länge plus die Summe der maximalen Anzahl von Zeichen in den Datenelementen mit variabler Länge, die diesem Datensatz untergeordnet sind, bestimmt.
   Diese Summe kann von der tatsächlichen Größe des Datensatzes abweichen (vgl. Auswahl der Zeichendarstellung und des Zahlensystems in Kapitel 2).

7.4 PROCEDURE DIVISION im SORT-MERGE-Modul

Die MERGE-Anweisung

Funktion

Durch die MERGE-Anweisung werden zwei oder mehrere nach identischen Kriterien sortierte Dateien entsprechend einer Menge von angegebenen Schlüsseln gemischt. Außerdem werden Datensätze während der Verarbeitung in der durch Mischen entstandenen Ordnung einer Ausgabeprozedur oder Ausgabedatei zur Verfügung gestellt.

Allgemeines Format

```
MERGE dateiname-1 ON { ASCENDING  } KEY datenname-1 [,datenname-2]...
                     { DESCENDING }

[  ON { ASCENDING  } KEY datenname-3 [, datenname-4] ...  ] ...
      { DESCENDING }

[ COLLATING SEQUENCE IS alphabetname ]

USING dateiname-2, dateiname-3 [ , dateiname-4] ...

{ OUTPUT PROCEDURE IS sectionname-1 [ { THROUGH } sectionname-2 ] }
                                     { THRU    }                 }
{ GIVING dateiname-5                                             }
```

Syntaxregeln

1. dateiname-1 muß in einer SORT-MERGE-Dateibeschreibung in der DATA DIVISION beschrieben sein.
2. sectionname-1 steht für den Namen einer Ausgabeprozedur.

3. dateiname-2, dateiname-3, dateiname-4 und dateiname-5 müssen in einer Dateibeschreibung in der DATA DIVISION, jedoch nicht in einer SORT-MERGE-Dateibeschreibung beschrieben sein. Die tatsächliche Größe der logischen Datensätze, die für dateiname-2, dateiname-3, dateiname-4 bzw. dateiname-5 beschrieben ist, muß gleich der tatsächlichen Größe der für dateiname-1 beschriebenen logischen Datensätze sein.

   Wenn die Datenbeschreibungen der Datenelemente, aus denen diese Datensätze bestehen, nicht identisch sind, so liegt es in der Verantwortung des Programmierers, die entsprechenden Datensätze so zu beschreiben, daß die gleiche Anzahl Zeichenpositionen für die entsprechenden Datensätze zugewiesen werden.

4. Die Worte THRU und THROUGH sind gleichbedeutend.

5. datenname-1, datenname-2, datenname-3 und datenname-4 sind KEY-Datennamen und unterliegen den folgenden Regeln:

   a) Die durch KEY-Datennamen identifizierten Datenfelder müssen mit dateiname-1 verknüpften Datensätzen beschrieben werden.

   b) KEY-Datennamen können gekennzeichnet werden.

   c) Die durch KEY-Datennamen identifizierten Datenfelder dürfen keine Datenfelder mit variabler Länge sein.

   d) Wenn dateiname-1 mehr als eine Datensatzbeschreibung hat, dann dürfen die durch KEY-Datennamen identifizierten Datenfelder nur in einer der Datensatzbeschreibungen beschrieben sein.

   e) Keines der durch KEY-Datennamen identifizierten Datenfelder darf durch eine Eintragung oder einer dieser untergeordneten Eintragung beschrieben sein, die eine OCCURS-Klausel enthält.

6. In einer MERGE-Anweisung darf nicht mehr als ein Dateiname von einem Band mit mehreren Dateien erscheinen.

7. Dateinamen dürfen in der MERGE-Anweisung nicht wiederholt verwendet werden.

8. MERGE-Anweisungen können nicht im Vereinbarungsteil der PROCEDURE DIVISION oder in einer mit einer SORT oder MERGE-Anweisung verknüpften Eingabe- oder Ausgabeprozedur auftreten, sonst überall.

Allgemeine Regeln

1. Durch die MERGE-Anweisung werden alle in dateiname-2, dateiname-3 und dateiname-4 enthaltenen Datensätze gemischt. Die in der MERGE-Anweisung genannten Dateien dürfen bei Ausführung der Anweisung nicht gleichzeitig geöffnet sein. Diese Dateien werden automatisch durch die Mischoperation geöffnet und geschlossen, wobei alle impliziten Funktionen durchgeführt werden, wie z. B. die Ausführung irgendwelcher zugehöriger USE Prozeduren.
   Die Dateien werden geschlossen, als ob eine CLOSE-Anweisung ohne optionale Angaben für jede Datei ausgeführt worden wäre.

2. Die nach dem Wort KEY stehenden Datennamen müssen in der MERGE-Anweisung von links nach rechts in der Reihenfolge der absteigenden Bedeutung aufgelistet werden, ungeachtet dessen, wie sie in KEY Angaben unterteilt sind. Im Format ist datenname-1 der Hauptschlüssel, datenname-2 ist der zweitwichtigste Schlüssel usw.

   a) Wenn ASCENDING angegeben ist, erfolgt das Mischen vom niedrigsten Wert der Inhalte der durch KEY-datennamen identifizierten Datenfelder aufwärts bis zum höchsten Wert entsprechend den Regeln für den Vergleich von Operanden in einer Vergleichsbedingung.

   b) Wenn DESCENDING angegeben ist, erfolgt das Mischen vom höchsten Wert der Inhalte der durch KEY-Datennamen identifizierten Datenfelder abwärts bis zum niedrigsten Wert entsprechend der Regeln für den Vergleich von Operanden in einer Vergleichsbedingung.

3. Beim Vergleich nicht numerischer Operanden hat eine angegebene COLLATING SEQUENCE Vorrang vor der Programmsortierfolge.

4. Die Ausgabeprozedur muß aus einem oder mehreren Kapiteln bestehen, die im Programm aufeinanderfolgend erscheinen, und die nicht Teil irgendeiner anderen Prozedur sind. Um Datensätze für die Verarbeitung bereitzustellen, muß in der Ausgabeprozedur mindestens eine RETURN-Anweisung ausgeführt werden. Die Ausgabeprozedur darf nur über eine SORT- oder MERGE-Anweisung aufgerufen werden, nicht direkt. Die Ausgabeprozedur kann aus irgendwelchen Prozeduren bestehen, die zur Auswahl, Änderung oder Kopieren der Datensätze benötigt werden, wobei einer nach dem anderen in der nach Ausführung einer MERGE-Anweisung entstandenen Reihenfolge zurückgegeben wird, beginnend mit dateiname-1. Für Prozeduranweisungen in einer Ausgabeprozedur gelten folgende Einschränkungen:

   a) Die Ausgabeprozedur darf keine Sprünge an Punkte außerhalb der Ausgabeprozedur enthalten, d.h. ALTER, GO TO und PERFORM-Anweisungen in der Ausgabeprozedur dürfen sich nicht auf Prozedurnamen außerhalb der Ausgabeprozedur beziehen. COBOL-Anweisungen, die eine implizite Steuerung des Programmablaufes zu Vereinbarungen verursachen, sind jedoch zulässig.

   b) Die Ausgabeprozeduren dürfen keine SORT- oder MERGE-Anweisungen enthalten.

   c) Der Rest der PROCEDURE DIVISION darf keine Sprünge an Punkte innerhalb der Ausgabeprozeduren enthalten, ALTER, GO TO und PERFORM-Anweisungen im restlichen Teil der PROCEDURE DIVISION dürfen sich nicht auf Prozedurnamen innerhalb der Ausgabeprozedur beziehen.

5. Wenn eine Ausgabeprozedur angegeben ist, geht die Steuerung während der Ausführung der MERGE-Anweisung an diese Prozedur über. Durch den Compiler wird ein Rückgabemechanismus an das Ende des letzten Absatzes in der Ausgabeprozedur angefügt. Wenn die Steuerung an die letzte Anweisung in der Ausgabeprozedur übergeht, sorgt der Rückgabemechanismus für die Beendigung des Mischvorganges. Die Steuerung geht dann an die nächste ausführbare Prozedur über.

   Die MERGE Prozedur hat dann einen Punkt erreicht, an dem sie den nächsten Datensatz in der nach Ausführung einer MERGE-Anweisung entstandenen Reihenfolge anwählen kann, falls dies erforderlich ist. Mit Ausführung von RETURN-Anweisungen in der Ausgabeprozedur wird gleichzeitig der nächste Datensatz angefordert.

6. Die Segmentierung, wie in Kapitel 8 definiert, kann auf Programme angewandt werden, die eine MERGE-Anweisung enthalten. Es gelten jedoch die folgenden Einschränkungen:

   a) Wenn die MERGE-Anweisung in einem Abschnitt erscheint, der kein unabhängiges Segment darstellt, dann muß jede durch diese MERGE-Anweisung angesprochene Ausgabeprozedur entweder

      völlig innerhalb nicht-unabhängiger Segmente erscheinen

      oder

      völlig in einem einzelnen unabhängigen Segment enthalten sein.

   b) Wenn eine MERGE-Anweisung in einem unabhängigen Segment erscheint, dann muß eine durch diese MERGE-Anweisung angesprochene Ausgabeprozedur entweder

      völlig innerhalb nicht-unabhängiger Segmente

      oder

      völlig innerhalb des gleichen unabhängigen Segmentes wie die MERGE-Anweisung enthalten sein.

7. Wenn GIVING angegeben ist, werden alle gemischten Datensätze von dateiname-1 in dateiname-5 geschrieben.

8. Die Inhalte der Datenfelder werden verglichen, die durch alle zwischen den Datensätzen von mehreren Eingabedateien liegenden KEY datenname identifiziert werden.
   Sind die Inhalte entsprechend den Regeln der Vergleichbedingung gleich, so werden die Datensätze in Abhängigkeit von der spezifizierten Angabe entweder in dateiname-5 geschrieben oder an die Ausgabeprozedur zurückgegeben. Dies wird in der Reihenfolge durchgeführt, in der die zugehörigen Eingabedateien in der MERGE-Anweisung angegeben sind.

9. Die Ergebnisse der Mischoperation sind nur dann vorhersagbar, wenn die Datensätze in den durch dateiname-2, dateiname-3, ... benannten Dateien so geordnet sind, wie in der mit der MERGE-Anweisung verknüpften ASCENDING oder DESCENDING KEY-Klausel beschrieben.

Die RELEASE-Anweisung

Funktion

Durch die RELEASE-Anweisung werden Datensätze an den Anfang einer SORT-Operation gesetzt.

Allgemeines Format

```
RELEASE datensatzname    [ FROM bezeichner ]
```

Syntaxregeln

1. Eine RELEASE-Anweisung kann nur innerhalb einer mit einer SORT-Anweisung verknüpften Eingabeprozedur für eine Datei verwendet werden, deren SORT-MERGE-Dateibeschreibung datensatzname enthält (vgl. Die SORT-Anweisung).
2. datensatzname muß der Name eines logischen Datensatzes in der zuhörigen SORT-MERGE-Dateibeschreibung sein und kann gekennzeichnet sein.
3. datensatzname und bezeichner dürfen sich nicht auf den gleichen Speicherbereich beziehen.

Allgemeine Regeln

1. Durch die Ausführung einer RELEASE-Anweisung wird der durch Datensatzname benannte Datensatz an die Anfangsphase eines Sortiervorganges übergeben.

2. Wenn die FROM-Angabe verwendet wird, wird der Inhalt von bezeichner in datensatzname übertragen und anschließend der Inhalt von datensatzname in die Sortier-Datei übergeben. Die Übertragung von Dateien erfolgt entsprechend den Regeln für die MOVE-Anweisung ohne die CORRESPONDING Angabe. Die Information im Datensatzbereich steht danach nicht mehr zur Verfügung; die Information in dem mit bezeichner verknüpften Datensatzbereich ist jedoch verfügbar.

3. Nach Ausführung der RELEASE-Anweisung steht der logische Datensatz im Datensatzbereich nicht mehr zur Verfügung, sofern die SORT-MERGE-Datei nicht in einer SAME RECORD AREA-Klausel benannt ist. Der logische Datensatz steht dem Programm auch als Datensatz für andere Dateien in der gleichen SAME RECORD AREA-Klausel sowie für die mit Datensatzname verknüpfte Datei zur Verfügung. Wenn die Steuerung von der Eingabeprozedur übergeben wird, besteht die Datei aus all jenen Datensätzen, die durch die Ausführung von RELEASE-Anweisungen in diese übertragen worden sind.

Die RETURN-Anweisung

Funktion

Durch die RETURN-Anweisung werden sortierte Datensätze nach einer SORT-Operation oder gemischte Datensätze einer MERGE-Operation an das Programm übertragen.

Allgemeines Format

```
RETURN dateiname RECORD  [ INTO bezeichner ]

     ; AT END unbedingte-anweisung
```

Syntaxregeln

1. dateiname muß durch eine SORT-MERGE-Dateibeschreibung in der DATA DIVISION beschrieben sein.
2. Eine RETURN-Anweisung kann nur innerhalb einer Ausgabeprozedur für dateiname verwendet werden, wenn sie mit einer SORT/MERGE-Anweisung verknüpft ist.
3. Die INTO-Angabe darf nicht verwendet werden, wenn die Eingabedatei logische Datensätze verschiedener Größe, wie durch ihre Datensatzbeschreibungen festgelegt, enthält. Der mit bezeichner verknüpfte Speicherbereich und der mit dateiname verknüpfte Datensatzbereich dürfen nicht den gleichen Speicherbereich belegen.

Allgemeine Regeln

1. Wenn die logischen Datensätze einer Datei mit mehr als einer Datensatzbeschreibung beschrieben sind, teilen sich diese Datensätze automatisch den gleichen Speicherbereich; dies entspricht einer impliziten Neudefinition des Bereiches. Der Inhalt irgendwelcher Datenfelder, die außerhalb des Bereiches des aktuellen Datensatzes liegen, sind bei Abschluß der Ausführung der RETURN-Anweisung undefiniert.

2. Die Ausführung der RETURN-Anweisung hat zur Folge, daß der nächste Datensatz in der Reihenfolge für die Verarbeitung in den mit der Sortier- oder Misch-Datei verknüpften Datensatzbereichen zur Verfügung gestellt wird, wie durch die in der SORT- oder MERGE-Anweisung aufgelisteten Schlüssel angegeben.

3. Wenn INTO angegeben ist, wird der aktuelle Datensatz aus dem Eingabebereich in den durch bezeichner benannten Bereich übertragen, entsprechend den Regeln für die MOVE-Anweisung ohne die CORRESPONDING Angabe. Eine implizite Übertragung erfolgt nicht, wenn eine AT END-Bedingung besteht. Subskribierung oder Indizierung verknüpft mit bezeichner wird ausgewertet, nachdem der Datensatz zurückgegeben wurde, und unmittelbar bevor er in das Datenfeld übertragen wird.

4. Wenn die INTO-Angabe verwendet wird, stehen die Daten sowohl im Eingabe-Datensatzbereich als auch in dem mit bezeichner verknüpften Datenbereich zur Verfügung.

5. Wenn kein nächster logischer Datensatz für die Datei zum Zeitpunkt der Ausführung einer RETURN-Anweisung vorhanden ist, tritt eine AT END-Bedingung auf. Die Inhalte der mit der Datei verknüpften Datensatzbereiche sind beim Auftreten der AT END-Bedingung undefiniert. Nach Ausführung der mit der AT END-Angabe verknüpften unbedingten Anweisung darf keine RETURN-Anweisung als Teil der aktuellen Ausgabeprozedur ausgeführt werden.

Die SORT-Anweisung

Funktion

Durch die SORT-Anweisung wird eine Sortier-Datei durch Ausführung von Eingabeprozeduren oder durch Übertragung von Datensätzen aus einer anderen Datei erstellt. Die Datensätze werden in der Sortier-Datei, entsprechend einer Menge von angegebenen Schlüsseln, sortiert und nach einem Sortiervorgang jedem Datensatz aus der Sortier-Datei in der nach Ausführung einer SORT-Anweisung entstandenen Reihenfolge irgendwelchen Ausgabeprozeduren oder einer Ausgabedatei zur Verfügung gestellt.

Allgemeines Format

```
SORT dateiname-1 ON { ASCENDING  } KEY datenname-1 [,datenname-2]...
                    { DESCENDING }

    [ ON { ASCENDING  } KEY datenname-3 [, datenname-4] ... ] ...
         { DESCENDING }

    [ COLLATING SEQUENCE IS alphabetname ]

  { INPUT PROCEDURE IS sectionname-1 [ { THROUGH } sectionname-2 ] }
                                     { THRU    }
  { USING dateiname-2 [, dateiname-3] ...                         }

  { OUTPUT PROCEDURE IS sectionname-3 [ { THROUGH } sectionname-4 ] }
                                      { THRU    }
  { GIVING dateiname-4                                            }
```

Syntaxregeln

1. dateiname-1 muß in einer SORT-MERGE-Dateibeschreibung in der DATA DIVISION beschrieben sein.
2. sectionname-1 steht für den Namen einer Eingabeprozedur und sectionname-3 steht für den Namen einer Ausgabeprozedur.

3. dateiname-2, dateiname-3 und dateiname-4 müssen in einer Dateibeschreibung in der DATA DIVISION, jedoch nicht in einer SORT-MERGE-Dateibeschreibung beschrieben sein. Die tatsächliche Größe der logischen Datensätze, die für dateiname-2, dateiname-3 und dateiname-4 beschrieben ist, muß der tatsächlichen Größe der logischen Datensätze entsprechen, die für dateiname-1 beschrieben ist. Wenn die Datenbeschreibungen der Datenelemente, aus denen diese Datensätze bestehen, nicht identisch sind, so liegt es in der Verantwortung des Programmierers, die entsprechenden Datensätze so zu beschreiben, daß die gleiche Anzahl Zeichenpositionen für die entsprechenden Datensätze zugewiesen werden.

4. datenname-1, datenname-2, datenname-3 und datenname-4 sind KEY-Datennamen und unterliegen den folgenden Regeln:

   a) Die durch KEY-dDtennamen identifizierten Datenfelder müssen in den mit dateiname-1 verknüpften Datensätzen beschrieben werden.

   b) KEY-Datennamen können gekennzeichnet werden.

   c) Die durch KEY-Datennamen identifizierten Datenfelder dürfen keine Datenfelder mit variabler Länge sein.

   d) Wenn dateiname-1 mehr als eine Datensatzbeschreibung hat, dann dürfen die durch KEY-Datennamen identifizierten Datenfelder nur in einer der Datensatzbeschreibungen beschrieben sein.

   e) Keines der durch KEY-Datennamen identifizierten Datenfelder darf durch eine Eintragung oder einer dieser untergeordneten Eintragung beschrieben sein, die eine OCCURS-Klausel enthält.

5. Die Worte THRU und THROUGH sind gleichbedeutend.
6. SORT-Anweisungen können nur nicht im Vereinbarungsteil der PROCEDURE DIVISION oder in einer mit einer SORT oder MERGE-Anweisung verknüpften Eingabe- oder Ausgabeprozedur erscheinen, sonst überall.
7. In der SORT-Anweisung kann nicht mehr als ein Dateiname von einem Band mit mehreren Dateien erscheinen.

Allgemeine Regeln

1. Die PROCEDURE DIVISION darf mehr als eine SORT-Anweisung enthalten, die überall erscheinen können, ausgenommen

   a) im Vereinbarungsteil

      oder

   b) in den mit einer SORT- oder MERGE-Anweisung verknüpften Eingabe- oder Ausgabe-Prozeduren.

2. Die nach dem Wort KEY stehenden Datennamen müssen von links nach rechts in der SORT-Anweisung in der Reihenfolge absteigender Bedeutung aufgelistet sein, ungeachtet dessen, wie sie in KEY-Angaben unterteilt sind. Im Format ist datenname-1 der Hauptschlüssel, datenname-2 ist der zweitwichtigste Schlüssel usw.

   a) Wenn ASCENDING angegeben ist, erfolgt die Sortierung vom niedrigsten Wert der Inhalte der durch KEY-Datennamen identifizierten Datenfelder aufwärts bis zum höchsten Wert entsprechend den Regeln für den Vergleich von Operanden in einer Vergleichsbedingung.

   b) Wenn DESCENDING angegeben ist, erfolgt die Sortierung vom höchsten Wert der Inhalte der durch KEY-Datennamen identifizierten Datenfelder bis zum niedrigsten Wert entsprechend den Regeln für den Vergleich von Operanden in einer Vergleichsbedingung.

3. Beim Vergleich nichtnumerischer Operanden hat eine angegebene COLLATING SEQUENCE Vorrang vor der Programmsortierfolge.

4. Die Eingabeprozedur muß aus einer oder mehreren Kapiteln bestehen, die im Quellprogramm aufeinanderfolgend erscheinen, und die nicht Teil irgendeiner anderen Ausgabeprozedur sind. Um Datensätze an die durch dateiname-1 benannte Datei zu übertragen, muß in der Eingabeprozedur mindestens eine RELEASE-Anweisung ausgeführt werden. Die Steuerung darf nicht an die Eingabeprozedur weitergegeben werden, wenn gerade eine zugehörige SORT-Anweisung ausgeführt wird.

   Die Eingabeprozedur kann aus irgendwelchen Prozeduren bestehen, die zur Auswahl, Änderung oder zum Kopieren der Datensätze benötigt werden. Für Prozeduranweisungen in einer Eingabeprozedur gelten folgende Einschränkungen:

   a) Die Eingabeprozedur darf keine SORT- oder MERGE-Anweisungen enthalten.

   b) Die Eingabeprozedur darf keine Sprünge an Punkte außerhalb der Eingabeprozedur enthalten; ALTER, GO TO und PERFORM-Anweisungen in der Eingabeprozedur dürfen sich nicht auf Prozedurnamen außerhalb der Eingabeprozedur beziehen. COBOL-Anweisungen, die eine implizite Steuerung des Programmablaufes zu Vereinbarungen verursachen, sind jedoch zulässig.

   c) Der Rest der PROCEDURE DIVISION darf keine Sprünge an Punkte innerhalb der Eingabeprozeduren enthalten; ALTER, GO TO und PERFORM-Anweisungen im restlichen Teil der PROCEDURE DIVISION dürfen sich nicht auf Prozedurnamen innerhalb der Eingabeprozedur beziehen.

5. Wenn eine Eingabeprozedur angegeben ist, geht die Steuerung an die Eingabeprozedur über, bevor dateiname-1 durch die SORT-Anweisung sortiert wird. Durch den Compiler wird ein Rückgabemechanismus an das Ende des letzten Absatzes in der Eingabeprozedur angefügt. Wenn die Steuerung an die letzte Anweisung in der Eingabeprozedur übergeht, sind die Datensätze, die an dateiname-1 übergeben worden sind, sortiert.

6. Die Ausgabeprozedur muß aus einem oder mehreren Kapiteln bestehen, die in einem Quellprogramm aufeinanderfolgend erscheinen, und die nicht Teil irgendeiner anderen Prozedur sind. Um sortierte Datensätze für die Verarbeitung bereitzustellen, muß in der Ausgabeprozedur mindestens eine RETURN-Anweisung ausgeführt werden. Die Ausgabeprozedur darf nur über eine SORT-Anweisung aufgerufen werden, nicht direkt. Die Ausgabeprozedur kann aus irgendwelchen Prozeduren bestehen, die zur Auswahl, Änderung oder zum Kopierung der Datensätze benötigt werden, wobei einer nach dem anderen in sortierter Ordnung von der Sortier-Datei zurückgegeben werden. Für Prozeduranweisungen in einer Ausgabeprozedur gelten folgende Einschränkungen:

   a) Die Ausgabeprozedur darf keine SORT oder MERGE-Anweisungen enthalten.

   b) Die Ausgabeprozedur darf keine Sprünge an Punkte außerhalb der Ausgabeprozedur enthalten; ALTER, GO TO und PERFORM-Anweisungen in der Ausgabeprozedur dürfen sich nicht auf Prozedurnamen außerhalb der Ausgabeprozedur beziehen. COBOL-Anweisungen, die eine implizite Steuerung des Programmablaufes zu Vereinbarungen verursachen, sind jedoch zulässig.

   c) Der Rest der PROCEDURE DIVISION darf keine Steuerungsübertragungen an Punkte innerhalb der Ausgabeprozeduren enthalten; ALTER, GO TO und PERFORM-Anweisungen im restlichen Teil der PROCEDURE DIVISION dürfen sich nicht auf Prozedurnamen innerhalb der Ausgabeprozeduren beziehen.

7. Wenn eine Ausgabeprozedur angegeben ist, geht die Steuerung an diese über, nachdem dateiname-1 durch die SORT-Anweisung sortiert wurde. Durch den Compiler wird ein Rückgabemechanismus an das Ende des letzten Absatzes in der Ausgabeprozedur angefügt. Wenn die Steuerung an die letzte Anweisung in der Ausgabeprozedur übergeht, sorgt der Rückgabemechanismus für die Beendigung des Sortiervorganges. Die Steuerung geht dann an die nächste ausführbare Anweisung nach der SORT-Anweisung über. Vor dem Übergang zur Ausgabeprozedur erreicht die SORT-Prozedur dann einen Punkt, an dem sie den nächsten Datensatz in sortierter Reihenfolge, wenn erforderlich, anwählen kann. Mit Ausführung von RETURN-Anweisungen in der Ausgabeprozedur wird gleichzeitig der nächste Datensatz angefordert.

8. Die Segmentierung, wie in Kapitel 8 definiert, kann auf Programme angewendet werden, die eine SORT-Anweisung enthalten. Es gelten jedoch die folgenden Einschränkungen:

   a) Wenn die SORT-Anweisung in einem Abschnitt erscheint, der kein unabhängiges Segment darstellt, dann muß jede durch diese SORT-Anweisung angesprochene Eingabe- oder Ausgabeprozedur entweder völlig innerhalb nicht-unabhängiger Segmente erscheinen oder völlig in einem einzelnen unabhängigen Segment enthalten sein.

   b) Wenn eine SORT-Anweisung in einem unabhängigen Segment erscheint, dann muß eine durch diese SORT-Anweisung angesprochene Eingabe- oder Ausgabeprozedur völlig innerhalb nichtunabhängiger Segmente oder völlig innerhalb des gleichen unabhängigen Segmentes wie die SORT-Anweisung enthalten sein.

9. Wenn USING angegeben ist, werden alle Datensätze in dateiname-2 und dateiname-3 automatisch in dateiname-1 übertragen. Zum Zeitpunkt der Ausführung der SORT-Anweisung dürfen dateiname-2 und dateiname-3 nicht geöffnet sein. Durch die SORT-Anweisung wird die Bearbeitung von dateiname-2 und dateiname-3 automatisch eingeleitet, die logischen Datensätze zur Verfügung gestellt und die Bearbeitung schließlich wieder beendet. Diese impliziten Funktionen werden dadurch erfüllt, daß irgendwelche verknüpften USE-Prozeduren ausgeführt werden.

   Die Dateien werden so geschlossen, als ob eine CLOSE-Anweisung ohne optionale Angaben für jede Datei ausgeführt worden wäre. Die SORT-Anweisung erfüllt auch automatisch die impliziten Funktionen, wodurch die Datensätze aus dem Dateibereich von dateiname-2 und dateiname-3 in den Dateibereich von dateiname-1 übertragen werden und die Übergabe von Datensätzen am Anfang der Sortierung erfolgt.

10. Wenn GIVING angegeben ist, werden alle sortierten Datensätze von dateiname-1 als implizite Ausgabeprozedur für diese SORT-Anweisung automatisch in dateiname-4 geschrieben. Zum Zeitpunkt der Ausführung der SORT-Anweisung darf die Datei mit dateiname-4 nicht geöffnet sein. Durch die SORT-Anweisung wird die Bearbeitung von dateiname-4 automatisch eingeleitet, die logischen Datensätze werden zur Verfügung gestellt und die Bearbeitung schließlich wieder beendet. Diese impliziten Funktionen werden dadurch erfüllt, daß irgendwelche verknüpften USE Prozeduren ausgeführt werden.
    Die Dateien werden so geschlossen, als ob eine CLOSE-Anweisung ohne optionale Angaben für die Datei ausgeführt worden wäre. Die SORT-Anweisung erfüllt auch automatisch die impliziten Funktionen, wodurch die sortierten Datensätze nach dem Sortiervorgang zurückgegeben werden und die Datensätze aus dem Datenbereich für dateiname-1 in den Datenbereich von dateiname-4 übertragen werden.

8 Segmentierung

8.1 Allgemeine Beschreibung

Die Segmentierung bei LII COBOL ermöglicht es, zum Zeitpunkt der Übersetzung Angaben für die Überlagerung des Programms zu machen. Segmentierung ist nur in der PROCEDURE DIVISION möglich, die Angaben zur Segmentierung werden in der PROCEDURE DIVISION und der ENVIRONMENT DIVISION gemacht.

Bei der Segmentierung können permanente unabhängige Segmente angegeben werden.
Alle permanenten Segmente müssen im Quellprogramm hintereinander stehen.
Man kann überlagerbare und nicht überlagerbare Segmente auch mischen, d.h. der nicht überlagerbare Teil des Quellprogramms kann Segmente enthalten, die überlagert werden können.

8.2 Organisation

Die PROCEDURE DIVISION eines Quellprogramms wird üblicherweise als eine Folge von mehreren Kapiteln geschrieben, von denen jedes aus einer Anzahl von Anweisungen besteht, die gemeinsam eine bestimmte Funktion ausführen. Wird die Segmentierung benutzt, muß die ganze PROCEDURE DIVISION in Kapitel eingeteilt werden. Zusätzlich muß bei jedem Kapitel angegeben werden, ob es zum festen Teil oder zu einem der unabhängigen Segmente des Zielprogramms gehören soll. Segmentierung ändert jedoch nichts an der Notwendigkeit, Prozedurnamen durch Kennzeichnung eindeutig zu machen.

8.2.1 Fester (nicht überlagerbarer) Teil

Der feste Teil wird definiert als der Teil des Zielprogramms, der logisch so behandelt wird, als ob er stets im Speicher vorhanden wäre. Dieser Teil des Programms besteht aus festen permanenten Segmenten und festen überlagerbaren Segmenten.

Ein festes permanentes Segment ist ein Segment im festen Teil, das nicht durch irgendeinen anderen Teil des Programms überlagert werden kann.

Ein festes überlagerbares Segment ist ein Segment im festen Teil, das, obwohl es logisch so behandelt wird, als ob es immer im Speicher vorhanden wäre, durch ein anderes Segment überlagert werden kann, um die Speicherausnutzung zu optimieren. Die Anzahl der festen permanenten Segmente im festen Teil kann durch Verwendung einer speziellen Einrichtung, die sogenannte SEGMENT-LIMIT-Klausel (vgl. SEGMENT-LIMIT in diesem Kapitel), verändert werden. Ein solches Segment, das vom Programm aufgerufen wird, wird stets im zuletzt verwendeten Zustand ('Last used') zur Verfügung gestellt.

8.2.2 Unabhängige Segmente

Ein unabhängiges Segment wird als Teil des Zielprogramms definiert, das entweder durch ein festes überlagerbares Segment oder ein anderes unabhängiges Segment überlagert werden kann bzw. dieses überlagern kann. Ein unabhängiges Segment befindet sich im Ursprungszustand:

- wenn es zum ersten Mal angesprungen wird
- wenn das Segment im Rahmen der Ablauffolge implizit angesprungen wurde,
- wenn die Steuerung an dieses Segment übertragen wird aufgrund einer impliziten Steuerungsübertragung zwischen einer SORT oder MERGE-Anweisung, und zwar in einem Segment mit einer anderen Segmentnummer und einer verknüpften Eingabe- oder Ausgabeprozedur in diesem unabhängigen Segment,
- wenn die Steuerung explizit an dieses Segment von einem Segment mit einer anderen Segmentnummer (mit der in 2. Punkt unten angemerkten Ausnahme) übertragen wird.

Bei aufeinanderfolgenden Steuerungsübertragungen an das Segment befindet sich ein unabhängiges Segment im zuletzt verwendeten Zustand, wenn

- die Steuerung implizit an das Segment von einem Segment mit einer anderen Segmentnummer (ausgenommen wie im 1. Punkt oben angemerkt) übertragen wurde,
- die Steuerung explizit an dieses Segment aufgrund der Ausführung einer EXIT PROGRAM-Anweisung für dieses Segment übertragen wurde.

8.3 Allgemeine Regeln

SECTIONs, die segmentiert werden sollen, werden unter Verwendung eines Segmentnummernsystems und entsprechend der nachfolgenden Kriterien klassifiziert:

Logische Anforderungen:

SECTIONs, die zu jeder Zeit verfügbar sein müssen, oder sehr häufig aufgerufen werden, werden normalerweise eingeteilt, als würden sie zu einem der permanenten Segmente gehören. SECTIONs, die weniger häufig verwendet werden, werden normalerweise eingeteilt, als würden sie entweder zu einem der überlagerbaren festen Segmente oder zu einem der unabhängigen Segmente gehören. Dies hängt jeweils von den logischen Erfordernissen des Programmablaufs ab.

Häufigkeit der Verwendung:
Allgemein kann gesagt werden: Je häufiger eine SECTION aufgerufen wird, desto niedriger sollte die Segmentnummer sein; je weniger eine SECTION aufgerufen wird, desto höher sollte die Segmentnummer sein.

Beziehungen zu anderen SECTIONs:
SECTIONs, die häufig miteinander in Verbindung stehen, sollten dieselbe Segmentnummer erhalten. Alle SECTIONs mit derselben Segmentnummer bilden ein Programmsegment für sich.

8.4 Segmentierungssteuerung

Die logische Folge des Programms ist die gleiche wie die physikalische, ausgenommen bei ganz bestimmten Programmverzweigungen. Die Steuerung kann innerhalb eines Quellprogramms an irgendeinen Paragraphen in einer SECTION übertragen werden, d. h., es ist nicht zwingend erforderlich, die Steuerung an den Anfang einer SECTION zu übergeben.

8.4.1 Struktur der Programmsegmente

Segmentnummer

Die Klassifizierung von SECTIONs erfolgt mit Segmentnummern. Die Segmentnummer ist in der SECTION-Überschrift enthalten.

Allgemeines Format

```
sectionname  SECTION [segmentnummer].
```

Syntaxregeln

1. segmentnummer muß eine Ganzzahl im Wertebereich zwischen 0 und 99 sein.
2. Wenn in der SECTION-Überschrift keine Segmentnummer angegeben ist, so wird als Segmentnummer 0 angenommen.
3. SECTIONs im Vereinbarungsteil müssen Segmentnummern unter 50 enthalten.

Allgemeine Regeln

1. Alle SECTIONs, die die gleiche Segmentnummer besitzen, bilden ein Programmsegment, müssen jedoch im Quellprogramm nicht physikalisch aufeinanderfolgen.
2. Segmente mit den Segmentnummern 0 bis 49 gehören zum festen Teil des Zielprogramms.
3. Segmente mit den Segmentnummern 50 bis 99 sind unabhängige Segmente.

SEGMENT-LIMIT

Allgemeines Format

Die SEGMENT-LIMIT-Klausel erscheint im OBJECT-COMPUTER Paragraph und hat das folgende Format:

```
[,SEGMENT-LIMIT IS segmentnummer]
```

Syntaxregel

Segmentnummer muß eine Ganzzahl im Wertebereich zwischen 1 und 49 sein.

Allgemeine Regeln

1. Die SEGMENT-LIMIT-Klausel wird nur zur Dokumentation angegeben.
2. Wenn die SEGMENT-LIMIT-Klausel angegeben ist, werden nur solche Segmente als permanente Segmente des Zielprogramms betrachtet, die Segmentnummern von 0 bis zu der als SEGMENT-LIMIT angegebenen Segmentnummer ausschließlich besitzen.
3. Die Segmente, die Segmentnummern vom SEGMENT-LIMIT bis 49 besitzen, sind überlagerbare feste Segmente.
4. Wenn keine SEGMENT-LIMIT-Klausel angegeben ist, sind alle Segmente mit den Segmentnummern 0 bis 49 permanente Segmente des Zielprogramms.

8.5 Einschränkungen des Programmablaufs

Wenn die Segmentierung verwendet wird, unterliegen die ALTER, PERFORM, MERGE und SORT-Anweisungen den nachfolgend genannten Einschränkungen.

Die ALTER-Anweisung

Eine GO TO-Anweisung in einer SECTION, deren Segmentnummer größer oder gleich 50 ist, darf nicht durch eine ALTER-Anweisung in einer SECTION mit einer anderen Segmentnummer angesprochen werden.

Alle anderen Verwendungsmöglichkeiten der ALTER-Anweisung sind zulässig und werden durchgeführt, selbst wenn sich die GO TO Anweisung, auf die sich die ALTER Anweisung bezieht, in einem festen überlagerbaren Segment befindet.

Die PERFORM-Anweisung

Die PERFORM-Anweisung, die in einer SECTION erscheint, die nicht in einem unabhängigen Segment liegt, kann (neben irgendwelchen Vereinbarungs-SECTIONS, deren Ausführung innerhalb dieses Bereiches erfolgt), nur einen der folgenden Punkte innerhalb ihres Bereiches enthalten:

- SECTIONs und/oder Paragraphen, die vollständig in einem oder mehreren nicht-unabhängigen Segmenten enthalten sind,
- SECTIONs und/oder Paragraphen, die vollständig in einem einzelnen unabhängigen Segment enthalten sind.

Die MERGE-Anweisung

Wenn in einer SECTION, die nicht in einem unabhängigen Segment liegt, die MERGE-Anweisung erscheint, dann muß jede durch diese MERGE-Anweisung angesprochene Ausgabeprozedur

- völlig innerhalb nicht-unabhängiger Segmente erscheinen,

  oder

- völlig in einem einzelnen unabhängigen Segment enthalten sein.

Wenn eine MERGE-Anweisung in einem unabhängigen Segment erscheint, dann muß jede durch diese MERGE-Anweisung angesprochene Ausgabeprozedur

- völlig innerhalb nicht-unabhängiger Segmente

  oder

- völlig innerhalb des gleichen unabhängigen Segments wie diese MERGE-Anweisung enthalten sein.

Die SORT-Anweisung

Wenn eine SORT-Anweisung in einer SECTION erscheint, die nicht in einem unabhängigen Segment liegt, so muß jede durch diese SORT-Anweisung angesprochene Eingabe- oder Ausgabeprozedur

- völlig innerhalb nicht-unabhängiger Segmente liegen

  oder

- völlig in einem einzelnen unabhängigen Segment enthalten sein.

Wenn eine SORT-Anweisung in einem unabhängigen Segment erscheint, dann muß jede durch diese SORT-Anweisung angesprochene Eingabe- oder Ausgabeprozedur

- völlig innerhalb nicht-unabhängiger Segmente

  oder

- völlig innerhalb des gleichen unabhängigen Segments wie diese SORT-Anweisung enthalten sein.

8.6 Zusätzliche Zwischencode-Dateien

Wenn die Segmentierung verwendet wird, werden vom Übersetzer zusätzliche Dateien wie folgt erzeugt:

dateiname.Inn	-	zusätzliche Zwischencode-Dateien; eine für jedes unabhängige Segment
dateiname.ISR	-	Inter-Segment-Referenztabelle; eine pro segmentiertem Programm.
dateiname.Dnn	-	Inhaltsverzeichnis-Datei; eines für jedes unabhängige Segment, ausgenommen dem letzten.

wobei:

"dateiname" der Name der Datei ist (ohne die Ergänzung), in der der Hauptteil des Programms gespeichert wird;

"nn" eine Segmentnummer ist, die das entsprechende Segment identifiziert.

Hinweis

Die Dateien dateiname.Dnn werden einzig und allein durch den Übersetzer geschrieben und verwendet und müssen nach der Kompilierung nicht erhalten werden. Die Dateien dateiname.Inn und dateiname.ISR müssen als Teil des Zielprogrammes gespeichert und, wenn das Programm kopiert wird, ebenfalls kopiert werden.

9 Programmkommunikation

9.1 Einführung in das Programmkommunikationsmodul

Mit den Anweisungen der Programmkommunikation kann ein Programm mit einem oder mehreren Programmen kommunizieren. Der Programmierer hat die Möglichkeit, modular zu programmieren. Jedes Modul wird dann beim Aufruf dynamisch durch das Ausführungszeitsystem geladen. Die Kommunikation erfolgt

- indem die Steuerung von einem Programm an ein anderes in der Ausführungseinheit übertragen wird;
- über gleiche Datenbereiche verschiedener Programme.

Es gibt 2 Arten von Programmen:

Aufrufendes Programm:

Das **aufrufende Programm** übergibt die Steuerung über die CALL-Anweisung an das aufgerufene Programm. Außerdem kann es eine Liste von Datennamen übergeben, die im aufrufenden Programm definiert sind und auf die sich das aufgerufene Programm beziehen kann.

Aufgerufenes Programm:

Das **aufgerufene Programm** wird am Anfang der PROCEDURE DIVISION angesprungen. Um die Steuerung an das aufrufende Programm zu der Anweisung, die unmittelbar der CALL-Anweisung folgt, zurückzugeben, wird die EXIT PROGRAM-Anweisung verwendet.

Das aufgerufene Programm kann Datenfelder des aufrufenden Programms verändern. Diese Datenfelder werden im aufrufenden sowie im aufgerufenen Programm angegeben. Für die Datenfelder im aufgerufenen Programm wird kein Speicherplatz reserviert; statt dessen übergibt das aufrufende Programm die Adressen seiner Datenfelder dem aufgerufenen Programm. Ändert das aufgerufene Programm diese Datenfelder, so ändert es sie im aufrufenden Programm.

Ein aufgerufenes Programm kann zugleich ein aufrufendes sein; d.h. ein aufgerufenes Programm kann wiederum ein anderes Programm aufrufen. Ein aufgerufenes Programm kann jedoch nicht ein Programm aufrufen, von dem es selbst aufgerufen worden ist.

Die Reihenfolge der Rücksprünge von mehreren aufgerufenen Programmen erfolgt umgekehrt zur Reihenfolge der Ansprünge; d.h. der Rücksprung aus einem aufgerufenem Programm führt immer erst zum aufrufenden Programm zurück, von dem der direkte Ansprung erfolgte und nicht zu einem früher aufrufenden Programm.

9.2 DATA DIVISION

LINKAGE SECTION

In der LINKAGE SECTION werden die Daten beschrieben, die sowohl vom aufrufenden, als auch vom aufgerufenen Programm gelesen und geändert werden können.

In der LINKAGE SECTION des aufgerufenen Programms definierte Datenfelder können innerhalb der PROCEDURE DIVISION des aufgerufenen Programms nur dann angesprochen werden, wenn sie in der CALL-Anweisung der USING-Angabe angegeben sind.

Die LINKAGE SECTION ist genauso wie die WORKING-STORAGE SECTION aufgebaut, d. h., sie wird mit einer SECTION-Überschrift eingeleitet, gefolgt von Datenbeschreibungen für nicht benachbarte Datenfelder und/oder Datensatzbeschreibungen.

Jeder in der LINKAGE SECTION definierte Datensatzname und nicht benachbarte Datenfeld-Name muß im aufgerufenen Programm eindeutig sein, da er nicht gekennzeichnet werden kann.

Von den in der LINKAGE SECTION definierten Datenfeldern können nur datenname-1, datenname-2, ... in der USING-Angabe der PROCEDURE DIVISION Überschrift in der PROCEDURE DIVISION angesprochen werden. Das können auch Datenfelder sein, die diesem Datennamen untergeordnet sind, und Bedingungsnamen und/oder Indexnamen, die mit diesen Datennamen verknüpft und/oder diesen Datenfeldern untergeordnet sind.

Nicht benachbarte LINKAGE STORAGE

Datenfelder in der LINKAGE SECTION, die keine hierarchische Beziehung zueinander haben, müssen nicht in Datensätze zusammengefaßt werden und werden als nicht benachbarte Datenelemente klassifiziert und definiert. Jedes dieser Datenfelder wird in einer getrennten Datenbeschreibung definiert, die mit der Sonderstufennummer 77 beginnt.

Die folgenden Daten-Klauseln sind für jede Datenbeschreibung erforderlich:

* Stufennummer 77
* Datenname
* Die PICTURE-Klausel oder die USAGE IS INDEX-Klausel

Andere Datenbeschreibungsklauseln sind wahlweise und können verwendet werden, um die Beschreibung des Datenelementes, sofern notwendig, zu vervollständigen.

LINKAGE Datensätze

Datenelemente in der LINKAGE SECTION, die eine ganz bestimmte hierarchische Beziehung zueinander haben, müssen in Datensätze entsprechend den Regeln für die Bildung von Datensatzbeschreibungen zusammengefaßt werden. Jede Klausel, die in einer Eingabe- oder Ausgabe-Datensatzbeschreibung verwendet wird, kann in einer LINKAGE SECTION verwendet werden.

Anfangswerte:

Die VALUE-Klausel darf in der LINKAGE SECTION nicht angegeben sein, außer bei Bedingungsnamen-Eintragungen (Stufe 88).

9.3 PROCEDURE DIVISION

Die PROCEDURE DIVISION-Überschrift

Die PROCEDURE DIVISION wird durch die folgende Überschrift identifiziert und muß damit beginnen:

```
PROCEDURE DIVISION [USING datenname-1 [,datenname-2]...].
```

Die USING-Angabe darf nur geschrieben werden, wenn das Zielprogramm unter der Steuerung einer CALL-Anweisung ablaufen soll und die CALL-Anweisung im aufrufenden Programm eine USING-Angabe enthält.

Jeder der Operanden, der in der USING-Angabe in der PROCEDURE DIVISION-Überschrift enthalten ist, muß als ein Datenfeld in der LINKAGE SECTION des Programms definiert sein, in dem diese Überschrift auftritt, und dieses Datenfeld muß die Stufennummer 01 oder 77 besitzen.

In einem aufgerufenen Programm werden LINKAGE SECTION-Datenfelder entsprechend ihrer im aufgerufenen Programm vorgegebenen Datenbeschreibungen verarbeitet.

Wenn die USING-Angabe vorhanden ist, bezieht sich das Zielprogramm bei datenname-1 der PROCEDURE DIVISION Überschrift im aufgerufenen Programm und datenname-1 in der USING-Angabe der CALL Anweisung im aufrufenden Programm auf dieselben Daten. In den Beschreibungen muß eine gleiche Anzahl von Zeichenpositionen definiert sein; sie müssen jedoch nicht den gleichen Namen haben. In ähnlicher Weise gibt es eine gleichartige Beziehung zwischen datenname-2 in der USING-Angabe des aufgerufenen Programms und datenname-2, ... in der USING-Angabe der CALL-Anweisung im aufrufenden Programm. Ein Datenname darf nicht mehr als einmal in der USING-Angabe in der PROCEDURE DIVISION Überschrift des aufgerufenen Programms erscheinen; ein vorgegebener Datenname kann jedoch häufiger als nur einmal in der gleichen USING-Angabe einer CALL Anweisung erscheinen.

Unter LEVEL II COBOL kann die Anzahl der Operanden in der PROCEDURE DIVISION USING-Anweisung und der CALL USING-Anweisung verschieden sein. Wenn sie verschieden sind, dann werden die Operanden in den beiden Anweisungen von links nach rechts so lange miteinander verglichen, bis in einer der beiden Anweisungen keine Operanden mehr vorhanden sind.

Wenn die verbleibenden noch nicht verglichenden Operanden in der CALL-Anweisung stehen, so werden sie ignoriert; wenn die verbleibenden noch nicht verglichenen Operanden in der PROCEDURE DIVISION-Anweisung stehen, so stehen sie dem Programm nicht zur Verfügung, und es tritt, wenn sie zur Ausführungszeit angesprochen werden, ein Ausführungszeitfehler auf.

Da nur zur Ausführungszeit festgestellt werden kann, daß Operanden nicht übereinstimmen, wird diese Erweiterung nicht durch die FLAG-Anweisung des Übersetzers gekennzeichnet.

Die CALL-Anweisung

Funktion

Durch die CALL-Anweisung wird die Steuerung von einem Zielprogramm an ein anderes in der Ausführungseinheit übertragen.

Allgemeine Formate

Format 1

```
CALL {bezeichner-1} [USING datenname-1[,datenname-2]...]
     {literal-1   }

     [ON OVERFLOW unbedingte-anweisung ]
```

Format 2

```
CALL {literal-2   } [USING datenname-3[,datenname-4]...]
     {bezeichner-2}
```

Syntaxregeln

1. literal-1 muß ein nichtnumerisches Literal sein, das einen Programmnamen bezeichnet, der ein durch eine CALL-Anweisung aufgerufenes Unterprogramm enthält.
2. bezeichner-1 muß als ein alphanumerisches Datenfeld USAGE DISPLAY definiert sein.
3. Die USING-Angabe darf in der CALL-Anweisung nur dann enthalten sein, wenn in der PROCEDURE DIVISION-Überschrift des aufgerufenen Programms eine USING-Angabe steht und die Anzahl der Operanden in jeder USING-Angabe identisch ist.
4. Jeder der in der USING-Angabe stehenden Operanden muß als ein Datenfeld in der FILE SECTION, WORKING-STORAGE SECTION oder LINKAGE SECTION definiert sein, aber nicht mit der Stufen-

nummer 88 (aber: muß die Stufennummer 01 oder 77 haben, wenn der ANSI Schalter gesetzt ist). datenname-1, datenname-2, ... können gekennzeichnet werden, wenn sie Datenfelder ansprechen, die in der FILE SECTION definiert sind.

5. literal-2 muß als ein alphanumerisches Literal definiert sein, das einen numerischen Wert enthält.

6. bezeichner-2 muß als ein alphanumerisches Datenfeld mit einem numerischen Wert definiert sein, z. B. CALL "3" oder CALL D-NAM, wobei D-NAM als alphanumerisch und USAGE DISPLAY definiert ist und einen numerischen Wert enthält.

Allgemeine Regeln

1. Das durch den Wert von literal-1 oder bezeichner-1 identifizierte Programm ist ein aufgerufenes Unterprogramm; das durch literal-2 oder bezeichner-2 identifizierte Programm ist eine aufgerufene Laufzeit-Unterroutine; das Programm, das die CALL-Anweisung enthält ist das aufrufende Programm.

2. Durch die Ausführung einer CALL-Anweisung wird die Steuerung an das aufgerufene Programm übergeben.

3. Im Format 1 ruft man ein Programm auf. Das Programmodul wird dann von der Platte in den Speicher geladen

   — wenn es das erste Mal in einer Ausführungseinheit aufgerufen wird,

   — wenn es das erste Mal nach einer CANCEL-Anweisung wieder aufgerufen wird.

   Bei allen anderen Aufrufen des aufgerufenen Programms bleibt der Status des Programms der gleiche, wie er nach der letzten Programmbeendigung war. Dies gilt für alle Datenfelder, den Status und das Positionieren aller Dateien und für alle änderbaren Schalterstellungen.

4. In Format 2 befindet sich ein Programm immer in dem Status, den es das letzte Mal hatte, als es vom Ausführungszeitsystem aufgerufen wurde.

5. Wenn während der Ausführung einer CALL-Anweisung festgestellt wird, daß der noch zur Verfügung stehende Teil des Ausführungszeit-Speichers für das aufgerufene Programm nicht ausreicht, wird der nächste sequentielle Befehl ausgeführt. Wenn ON OVERFLOW angegeben worden ist, wird die zugehörige unbedingte Anweisung ausgeführt, bevor der nächste Befehl ausgeführt wird.

6. Aufgerufene Programme können CALL-Anweisungen enthalten. Ein aufgerufenes Programm darf jedoch keine CALL-Anweisung enthalten, durch die das aufrufende Programm direkt oder indirekt aufgerufen wird.

7. Die durch die USING-Angabe der CALL-Anweisung angegebenen Datennamen benennen die Datenfelder, die einem aufrufenden Programm zur Verfügung stehen, die auch das aufgerufene Programm eventuell benutzt. Die Reihenfolge des Auftretens von Datennamen in der USING-Angabe der CALL Anweisung und der USING-Angabe der PROCEDURE DIVISION Überschrift ist kritisch. Übereinstimmende Datennamen beziehen sich auf eine bestimmte Datenmenge, die dem aufgerufenen und dem aufrufenden Programm zur Verfügung steht.

   Die Übereinstimmung bezieht sich auf die Zeichenpositionen und nicht auf den Namen.

   Bei Indexnamen wird keine derartige Übereinstimmung festgestellt. Indexnamen im aufgerufenen und im aufrufenden Programm beziehen sich immer auf getrennte Indizes.

8. Die CALL-Anweisung kann überall in einem segmentierten Programm erscheinen. Deshalb befindet sich, wenn eine CALL Anweisung in einer SECTION mit einer Segmentnummer größer oder gleich 50 erscheint, dieses Segment im zuletzt verwendeten Status, wenn die EXIT PROGRAM-Anweisung die Steuerung an das aufrufende Programm zurückgibt.

Die CANCEL-Anweisung

Funktion

Durch die CANCEL-Anweisung werden Speicherbereiche freigegeben, die durch das angesprochene Programm belegt sind.

Allgemeines Format

```
CANCEL { bezeichner-1 } [ bezeichner-2 ] ...
       { literal-1    } [ literal-2    ]
```

Syntaxregeln

1. literal-1, literal-2, ... muß ein nichtnumerisches Literal sein, das denselben Programmnamen enthält, der in der entsprechenden CALL-Anweisung angegeben ist.
2. bezeichner-1, bezeichner-2, ... muß jeweils als ein alphanumerisches Datenfeld so beschrieben werden, daß dessen Wert ein Programmname sein kann.

Allgemeine Regeln

1. Nach Ausführung einer CANCEL-Anweisung besitzt das angesprochene Programm keine logische Beziehung mehr zur Ausführungseinheit. Eine nachfolgend ausgeführte CALL-Anweisung, die das gleiche Programm aufruft, hat zur Folge, daß das Programm in seinem ursprünglichen Zustand ausgeführt wird. Die mit den benannten Programmen verknüpften Speicherbereiche werden freigegeben, so daß sie durch das Betriebssystem wieder zur Verfügung gestellt werden.

2. Zuerst muß ein Programm mit einer EXIT PROGRAM-Anweisung beendet worden sein, bevor sich eine CANCEL-Anweisung auf dieses Programm bezieht.

3. Eine logische Beziehung zum gelöschten Unterprogramm wird nur durch die Ausführung einer nachfolgenden CALL-Anweisung wieder hergestellt.

4. Ein aufgerufenes Programm wird entweder dadurch gelöscht, daß es als Operand einer CANCEL-Anweisung angesprochen wird, oder durch Beendigung der gesamten Ausführungseinheit, an der das Programm teilnimmt.

5. Es passiert nichts, wenn eine CANCEL-Anweisung ein Programm benennt, das in dieser Ausführungseinheit nicht aufgerufen worden ist oder das zwar aufgerufen, jedoch bereits gelöscht ist. Die Steuerung geht dann an die nächste Anweisung über.

Die EXIT PROGRAM-Anweisung

Funktion

Die EXIT PROGRAM-Anweisung markiert das logische Ende eines aufgerufenen Programms.

Allgemeines Format

```
EXIT PROGRAM .
```

Syntaxregeln

gelten in diesem Fall nur, wenn der ANSI Schalter gesetzt ist.

1. Die EXIT PROGRAM-Anweisung muß in einem eigenen Satz erscheinen.
2. Der EXIT PROGRAM Satz muß der einzige Satz im Paragraph sein.

Allgemeine Regel

Eine Ausführung einer EXIT PROGRAM-Anweisung in einem aufgerufenen Programm hat zur Folge, daß die Steuerung an das aufrufende Programm übergeben wird. Die Ausführung einer EXIT PROGRAM-Anweisung in einem Programm, das nicht aufgerufen wurde, verhält sich so, als ob die Anweisung eine EXIT-Anweisung wäre.

10 Bibliothek

10.1 Einführung

Das Bibliotheksmodul bietet eine Möglichkeit, Text von einer Quell-Anwenderbibliotheksdatei zu kopieren. Diese wird unter Verwendung eines Quelltext-Editors erstellt.

LII COBOL Bibliotheken bestehen aus Plattendateien. Sie enthalten Quelltext, der dem Übersetzer zur Verfügung gestellt werden soll. Die Interpretation der COPY-Anweisung hat zur Folge, daß Text in das Quellprogramm eingefügt wird, wo es durch den Compiler als Teil des Quellprogramms behandelt wird. Jedes Auftreten eines vorgegebenen Literals, Bezeichners, Wortes oder einer Wortgruppe im Bibliothekstext kann durch solchen Text während des Kopiervorganges ersetzt werden. Das Bibliotheksmodul kann auch mehr als eine COBOL Bibliothek während der Übersetzungszeit verarbeiten.

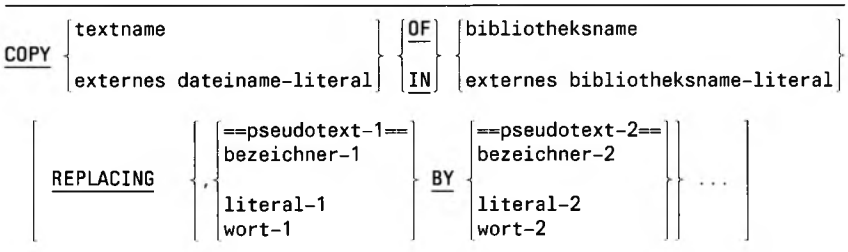
10.2 DATA DIVISION

Die COPY-Anweisung

Funktion

Die COPY-Anweisung fügt Text in ein LII COBOL Quellprogramm ein.

Allgemeines Format



Syntaxregeln

1. Wenn mehr als eine COBOL Bibliothek während der Übersetzung zur Verfügung steht, muß textname durch den Bibliotheksnamen gekennzeichnet werden, in der sich der Text befindet. Einzelheiten über Bibliotheksdateien werden in der LII COBOL Bedienungsanleitung behandelt.
2. Vor der COPY-Anweisung muß ein Leerzeichen und danach das Trennzeichen Punkt stehen.
3. pseudotext-1 darf weder Null sein, noch darf er nur aus Leerzeichen oder nur aus Kommentarzeilen bestehen.
4. pseudotext-2 kann Null sein.

5. Zeichenfolgen in pseudotext-1 und pseudotext-2 können fortgesetzt werden. Beide Zeichen eines Pseudotext-Begrenzers müssen sich jedoch auf der gleichen Zeile befinden.

6. wort-1 oder wort-2 kann ein einzelnes COBOL-Wort sein.

7. Eine COPY-Anweisung kann in einem Quellprogramm überall dort auftreten, wo eine Zeichenfolge oder ein Trennzeichen auftreten kann, mit der Ausnahme, daß eine COPY-Anweisung nicht in einer COPY-Anweisung auftreten darf.

8. textname definiert einen eindeutigen externen Dateinamen, der den Regeln für Programmiererwörter entspricht (es ist zu beachten, daß Kleinschreibung in Großschreibung umgesetzt wird). externes dateiname-literal und externes bibliotheksname-literal steht für ein alphanumerisches Literal **in Anführungszeichen**, das dem Betriebssystemregeln für die Bildung von Dateinamen bzw. Bibliotheksnamen entspricht.

Allgemeine Regeln

1. Die Übersetzung eines Quellprogramms, das eine COPY-Anweisung enthält, entspricht logisch der Verarbeitung aller COPY-Anweisungen, bevor das Quellprogramm übersetzt wird.

2. Die Verarbeitung einer COPY-Anweisung hat zur Folge, daß der mit textname verknüpfte Bibliothekstext in das Quellprogramm einkopiert und die gesamte COPY-Anweisung logisch ersetzt wird, wobei mit dem reservierten Wort COPY begonnen und mit dem Interpunktionszeichen Punkt einschließlich geendet wird.

3. Wenn REPLACING nicht angegeben ist, so wird der Bibliothekstext ungeändert kopiert.

   Wenn REPLACING angegeben ist, wird der Bibliothekstext kopiert und wenn pseudotext-1, bezeichner-1, wort-1 bzw. literal-1 im Bibliothekstext auftritt, wird es durch den entsprechenden pseudotext-2, bezeichner-2, wort-2 bzw. literal-2 ersetzt.

4. Für die Prüfung der Übereinstimmung werden bezeichner-1, wort-1 bzw.literal-1 wie Pseudo-Text behandelt, der nur bezeichner-1, wort-1 bzw. literal-1 enthält.

5. Die Vergleichsoperation bei der Festlegung von Textersetzung erfolgt in folgender Art und Weise:

   Jedes Trennzeichen, Komma, Strichpunkt und/oder Leerzeichen, vor dem am weitesten links stehenden Bibliothekstextwort wird in das Quellprogramm einkopiert. Beginnend mit dem am weitesten links stehenden Bibliothekstextwort und dem ersten pseudotext-1, wird der gesamte Operand der REPLACING-Angabe Angabe, der vor dem reservierten Wort BY steht, mit einer entsprechenden Anzahl von benachbarten Bibliothekstextworten verglichen.

   pseudotext-1, bezeichner-1, wort-1 oder literal-1 entsprechen dem Bibliothekstext nur dann, wenn Sie der geordneten Folge der Bibliothekstextwörter Zeichen für Zeichen entsprechen. Zur Prüfung der Übereinstimmung, wird jedes Auftreten eines Trennzeichens, Kommas oder Strichpunkts in pseudotext-1 oder im Bibliothekstext so betrachtet, als ob es sich um ein einzelnes Leerzeichen handeln würde. Eine Ausnahme dazu stellt pseudotext-1 dar, wenn dieser nur aus dem Trennzeichen Komma oder Strichpunkt besteht, und am Vergleich als Textwort teilnimmt. Jede Folge von einem oder mehreren Leerzeichen als Trennzeichen wird wie ein einzelnes Leerzeichen betrachtet.

   Wenn keine Gleichheit auftritt, wird der Vergleich mit jedem nachfolgenden pseudotext-1, bezeichner-1, wort-1 bzw. literal-1, sofern vorhanden, in der REPLACING-Angabe wiederholt, bis entweder eine Gleichheit gefunden wird oder es keinen nächsten nachfolgenden REPLACING-Operanden gibt.

   Wenn alle Operanden der REPLACING-Angabe verglichen worden sind und keine Übereinstimmung erzielt wurde, so wird das am weitesten links stehende Bibliothekstextwort in das Quellprogramm einkopiert. Das nächste nachfolgende Bibliothekstextwort wird dann als das am weitesten links stehende Bibliothekstextwort betrachtet und der Vergleichszyklus beginnt erneut mit dem ersten pseudotext-1, bezeichner-1, wort-1 bzw. literal-1, das in der REPLACING-Angabe steht.ben

   Wird zwischen pseudotext-1, bezeichner-1, wort-1 oder literal-1 und dem Bibliothekstext eine Übereinstimmung erzielt, wird der entsprechende pseudotext-2, bezeichner-2, wort-2 bzw. literal-2 in das Quellprogramm übertragen. Das Bibliothekswort, das unmittelbar nach dem am weitesten rechts stehenden Textwort steht,

das am Vergleich teilgenommen hat, wird dann als das am weitesten links stehende Bibliothekstextwort betrachtet. Der Vergleichszyklus beginnt erneut mit dem ersten pseudotext-1, bezeichner-1, wort-1 bzw. literal-1, das in der REPLACING-Angabe steht.

Die Vergleichsoperation wird fortgeführt, bis das am weitesten rechts stehende Textwort im Bibliothekstext entweder am Vergleich teilgenommen hat oder als das am weitesten links stehende Bibliothekstextwort, das an einem vollständigen Vergleichszyklus teilgenommen hat, betrachtet wird.

6. Eine im Bibliothekstext und pseudotext-1 auftretende Kommentarzeile, wird beim Vergleich als ein einzelnes Leerzeichen betrachtet. In pseudotext-2 und im Bibliothekstext erscheinende Kommentarzeilen werden unverändert in das Quellprogramm einkopiert.

7. Testhilfezeilen sind innerhalb des Bibliothekstextes und in pseudotext-2 erlaubt. Testhilfezeilen in pseudotext-1 sind nicht zulässig; Textworte in einer Testhilfezeile nehmen am Vergleich so teil, als ob das "D" nicht im Indikatorbereich geschrieben wäre. Wenn in einer Testhilfezeile eine COPY-Anweisung angegeben ist, dann erscheint der Text, der aus der Verarbeitung der COPY-Anweisung resultiert, so, als ob er in Testhilfezeilen angegeben worden wäre.

   Ausnahme: Kommentarzeilen im Bibliothekstext erscheinen als Kommentarzeilen im resultierenden Quellprogramm.

8. Der aus einer vollständigen Verarbeitung einer COPY-Anweisung resultierende Text darf keine COPY-Anweisung enthalten.

9. Die syntaktische Richtigkeit des Bibliothekstextes kann nicht unabhängig bestimmt werden.

   Die syntaktische Richtigkeit des gesamten COBOL Quellprogramms kann nicht festgelegt werden, bevor nicht alle COPY-Anweisungen vollständig verarbeitet worden sind.

10. Ein Bibliothekstext muß den Regeln für das COBOL Referenzformat entsprechen.

11. Für die Übersetzung werden Textworte nach der Ersetzung in das Quellprogramm entsprechend den Regeln für das Referenzformat, wie in Kapitel 1 beschrieben, eingesetzt.

11 Testhilfe und Fehlersuche (Debug-Mode)

11.1 Einführung

Standard ANSI COBOL Testhilfe bietet ein Mittel, durch das der Anwender die Bedingungen beschreiben kann, unter denen Prozeduren während der Ausführung des Zielprogramms überwacht werden sollen.

Ein anderes Testhilfeprodukt steht ebenfalls zur Verfügung und ruft ein Programm auf den Bildschirm auf durch Anzeige des Quellcodes während der Ausführungszeit, wobei der Cursor von COBOL Quellanweisung zu Quellanweisung bewegt wird. ANIMATIOR ist ein voll interaktives symbolisches Testhilfewerkzeug, das ermöglicht:

- Setzen von Haltepunkten
- Überprüfen und Ändern von Daten
- Ändern des Programmablaufs

Hier in diesem Kapitel wird das Standard ANSI 74 COBOL DEBUG Modul beschrieben.

11.2 Standard ANSI COBOL DEBUG

Die Entscheidung darüber, was überwacht und welche Information angezeigt werden soll, bestimmt der Programmierer.

Die Sprachenmerkmale, die das COBOL Testhilfemodul unterstützen, sind die folgenden:

- der Kompilierzeit-Schalter WITH DEBUGGING MODE
- der Ausführungszeit-Schalter
- die USE FOR DEBUGGING-Anweisung
- das Sonderregister DEBUG-ITEM
- Testhilfezeilen

Das reservierte Wort DEBUG-ITEM ist der Name für ein Sonderregister, das vom Übersetzer generiert wird, der die Testhilfeeinrichtung unterstützt. Jedem Programm wird jeweils nur ein DEBUG-ITEM zugewiesen. Die Namen der untergeordneten Datenfelder im DEBUG-ITEM sind ebenfalls reservierte Worte.

Kompilierzeit-Schalter

Die DEBUGGING MODE-Klausel wird als Teil des SOURCE-COMPUTER Paragraphen in der ENVIRONMENT DIVISION geschrieben. Sie dient als Kompilierzeit-Schalter für im Programm geschriebene Testhilfe-Anweisungen.

Wenn die WITH DEBUGGING MODE-Klausel im Programm angegeben ist, werden alle Testhilfe-SECTIONs und alle Testhilfezeilen, wie in diesem Kapitel des Handbuches beschrieben, übersetzt.

Wenn DEBUGGING MODE in einem Programm nicht angegeben ist, werden alle Testhilfe-SECTIONs und Testhilfezeilen so übersetzt, als ob diese Kommentarzeilen wären. Die Syntax wird nicht überprüft.

COBOL Testhilfe - Ausführungszeit-Schalter

Ein Ausführungszeit-Schalter aktiviert den durch den Übersetzer eingefügten Testhilfecode dynamisch. Dieser Schalter kann im Programm nicht direkt angesprochen werden; er wird außerhalb der COBOL Umgebung gesteuert. Wenn der Schalter auf "EIN" steht, sind die Auswirkungen jeglicher im Quellprogramm geschriebene USE FOR DEBUGGING-Anweisungen zulässig. Wenn der Schalter auf "AUS" steht, werden alle in der USE FOR DEBUGGING-Anweisung beschriebenen Auswirkungen unterdrückt. Eine Neukompilierung des Quellprogramms ist für den Einbau oder Ausbau dieser Einrichtung nicht erforderlich.

Der Ausführungszeitschalter hat keine Auswirkung auf die Ausführung des Zielprogramms, wenn die WITH DEBUGGING MODE-Klausel im Quellprogramm zur Übersetzungszeit nicht angegeben war.

Der Ausführungszeit-Schalter wird in der LII COBOL Bedienungsanleitung beschrieben.

11.3 ENVIRONMENT DIVISION in der COBOL Testhilfe

Die WITH DEBUGGING MODE-Klausel

Funktion

Die WITH DEBUGGING MODE-Klausel gibt an, daß alle Testhilfe-SECTIONs und alle Testhilfezeilen übersetzt werden sollen. Wenn diese Klausel nicht angegeben ist, werden alle Testhilfezeilen und SECTIONs so übersetzt, als ob sie Kommentarzeilen wären.

Allgemeines Format

```
SOURCE-COMPUTER. übersetzungsanlage [WITH DEBUGGING MODE].
```

Allgemeine Regeln

1. Wenn die WITH DEBUGGING MODE-Klausel im SOURCE-COMPUTER-Paragraph der CONFIGURATION SECTION eines Programms angegeben ist, werden alle USE FOR DEBUGGING-Anweisungen und alle Testhilfezeilen übersetzt.
2. Wenn die WITH DEBUGGING MODE-Klausel im SOURCE-COMPUTER-Paragraph der CONFIGURATION SECTION eines Programms nicht angegeben ist, werden jegliche USE FOR DEBUGGING-Anweisungen und alle damit verknüpften Testhilfe-SECTIONs sowie alle Testhilfezeilen so behandelt, als ob sie Kommentarzeilen wären.

11.4 PROCEDURE DIVISION in der COBOL Testhilfe

Die USE FOR DEBUGGING-Anweisung

Funktion

Die USE FOR DEBUGGING-Anweisung identifiziert die Datenfelder des Anwenders, die durch die zugehörige Testhilfe-SECTION überwacht werden sollen.

Allgemeines Format

```
sectionname SECTION [segmentnummer] .

                          ┌[ALL REFERENCES OF] bezeichner-1┐
                          │dateiname-1                     │
USE FOR DEBUGGING ON      {                                }
                          │prozedurname-1                  │
                          │ALL PROCEDURES                  │
                          └                                ┘

  ┌  [ALL REFERENCES OF] bezeichner-2 ┐
  │   dateiname-2                     │
, │                                   │ ...  .
  │   prozedurname-2                  │
  └   ALL PROCEDURES                  ┘
```

Syntaxregeln

1. Testhilfe-SECTIONs, sofern angegeben, müssen unmittelbar im Anschluß an die DECLARATIVES-Überschrift erscheinen.
2. Mit Ausnahme der USE FOR DEBUGGING-Anweisung selbst darf sich nichts auf irgendeine nichtvereinbarende Prozedur innerhalb der Testhilfe-SECTION beziehen.
3. Außerhalb des Bereiches der Testhilfe-SECTIONs erscheinende Anweisungen dürfen keine in Testhilfe-SECTIONs definierten Prozedurnamen ansprechen.
4. Mit Ausnahme der USE FOR DEBUGGING-Anweisung selbst können in einer vorgegebenen Testhilfe-SECTION erscheinende Anweisungen Prozedurnamen, die in einer anderen USE-Prozedur definiert sind, nur mit einer PERFORM-Anweisung ansprechen.
5. In Testhilfe-SECTIONs definierte Prozedurnamen dürfen nicht in USE FOR DEBUGGING-Anweisungen erscheinen.
6. Jeder vorgegebene Bezeichner, Dateiname oder Prozedurname kann nur in einer USE FOR DEBUGGING-Anweisung erscheinen und darf auch nur einmal in dieser Anweisung auftreten.
7. Die ALL PROCEDURES-Angabe kann in einem Programm nur einmal erscheinen.
8. Wenn ALL PROCEDURES angegeben ist, dürfen prozedurname-1, prozedurname-2, ... nicht in irgendeiner USE FOR DEBUGGING-Anweisung angegeben werden.
9. Wenn die Datenbeschreibung des durch bezeichner-1, bezeichner-2, ... benannten Datenfeldes eine OCCURS-Klausel enthält oder einer Datenbeschreibung untergeordnet ist, die eine OCCURS-Klausel enthält, müssen bezeichner-1, bezeichner-2, ... ohne die normalerweise erforderliche Subskribierung oder Indizierung angegeben werden.
10. Bezugnahmen auf das Sonderregister DEBUG-ITEM dürfen nur aus einer Testhilfe-SECTION erfolgen.

Allgemeine Regeln

1. In den folgenden Allgemeinen Regeln gelten alle Bezugnahmen auf bezeichner-1, prozedurname-1 bzw. dateiname-1 ebenso für bezeichner-2, prozedurname-2 bzw. dateiname-2.

2. Dadurch, daß eine Anweisung in einer Testhilfe-SECTION erscheint, erfolgt nicht automatisch auch die Ausführung einer Testhilfe-SECTION.

3. Wenn dateiname-1 in einer USE FOR DEBUGGING-Anweisung angegeben ist, wird diese Testhilfe-SECTION begonnen:

   - nach der Ausführung irgendwelcher OPEN oder CLOSE-Anweisungen, die sich auf dateiname-1 beziehen, und
   - nach der Ausführung einer READ-Anweisung, die nicht aus der Ausführung einer verknüpften AT END oder INVALID KEY-Anweisung resultiert, und
   - nach der Ausführung einer DELETE oder START-Anweisung, die sich auf dateiname-1 bezieht.

4. Wenn prozedurname-1 in einer USE FOR DEBUGGING-Anweisung angegeben ist, wird diese Testhilfe-SECTION ausgeführt:

   - unmittelbar vor jeder Ausführung der genannten Prozedur

     und

   - unmittelbar nach der Ausführung einer ALTER-Anweisung, die sich auf prozedurname-1 bezieht.

5. Durch die ALL PROCEDURES-Angabe treten die in der Allgemeinen Regel 4 beschriebenen Auswirkungen für jeden Prozedurnamen im Programm auf, mit Ausnahme der Prozedurnamen, die in einer Testhilfe-SECTION erscheinen.

6. Wenn für Bezeichner die ALL REFERENCES OF-Angabe gemacht wurde, wird diese Testhilfe-SECTION für jede Anweisung, die sich explizit auf bezeichner-1 bezieht, jedesmal ausgeführt, wenn folgendes auftritt:

   - unmittelbar vor einer WRITE oder REWRITE-Anweisung und bei der FROM-Angabe nach einer impliziten Übertragung, oder
   - bevor eine GO TO-Anweisung mit einer DEPENDING ON-Angabe ausgeführt wird und die Testhilfe-SECTION gestartet wird, die den Prozedurnamen des Ansprungs enthält, oder
   - eine PERFORM-Anweisung, in der bezeichner-1 durch eine VARYING, AFTER oder UNTIL-Angabe angesprochen wird, unmittelbar nach jeder Initialisierung, Änderung oder Auswertung des Inhaltes dieses durch bezeichner-1 benannten Datenfeldes, oder
   - eine andere COBOL-Anweisung, die unmittelbar nach Ausführung dieser Anweisung steht.

   Wenn bezeichner-1 in einer Angabe steht, die nicht ausgeführt oder ausgewertet wird, wird auch die zugehörige Testhilfe-SECTION nicht ausgeführt.

7. Wenn bezeichner-1 ohne ALL REFERENCES OF angegeben ist, wird diese Testhilfe-SECTION jedesmal ausgeführt, wenn folgendes auftritt:

   - unmittelbar vor der Ausführung einer WRITE oder REWRITE-Anweisung und nach der Ausführung einer impliziten Übertragung bei einer FROM-Angabe, oder
   - einer PERFORM-Anweisung, in der bezeichner-1 durch eine VARYING, AFTER, oder UNTIL-Angabe angesprochen wird, unmittelbar nach jeder Initialisierung, Änderung oder Auswertung dieses Inhaltes dieses durch bezeichner-1 benannten Datenfeldes, oder
   - unmittelbar nach der Ausführung einer anderen COBOL-Anweisung, die den Inhalt des durch bezeichner-1 benannten Datenfeldes explizit anspricht und dies Datenfeld verändert.

   Wenn bezeichner-1 in einer Angabe steht, die nicht ausgeführt oder ausgewertet wird, wird auch die zugehörige Testhilfe-SECTION nicht ausgeführt.

8. Die zugehörige Testhilfe-SECTION wird für einen bestimmten Operanden nur einmal ausgeführt, ungeachet dessen, wie oft der Operand angegeben ist. Bei einer PERFORM-Anweisung, die eine schrittweise Ausführung einer angesprochenen Prozedur zur Folge hat, wird die zugehörige Testhilfe-SECTION für jeden Schritt genau einmal ausgeführt.

   Jedes unbedingte Verb in einer unbedingten Anweisung benötigt eine gesonderte Anweisung als Testhilfe.

9. Ein Bezugnahme auf dateiname-1, bezeichner-1, prozedurname-1 oder als Kennzeichner stellt keinen Bezug zu diesem Datenfeld für die Testhilfe dar.

10. Zu jeder Ausführung einer Testhilfe-SECTION gehört das Sonderregister DEBUG-ITEM, das Information über die Bedingungen liefert, unter denen die Testhilfe-SECTION abläuft. DEBUG-ITEM wird wie folgt implizit beschrieben:

```
01  DEBUG-ITEM.
    02  DEBUG-LINE          PICTURE IS X (6).
```

```
02  FILLER              PICTURE IS X VALUE SPACE.
02  DEBUG-NAME          PICTURE IS X (30).
02  FILLER              PICTURE IS X VALUE SPACE.
02  DEBUG-SUB-1         PICTURE IS S9999 SIGN IS
                    LEADING SEPARATE CHARACTER.
02  FILLER              PICTURE IS X VALUE SPACE
02  DEBUG-SUB-2         PICTURE IS S9999 SIGN IS
                    LEADING SEPARATE CHARACTER.
02  FILLER              PICTURE IS X VALUE SPACE.
02  DEBUG-SUB-3         PICTURE IS S9999 SIGN IS
                    LEADING SEPARATE CHARACTER.
02  FILLER              PICTURE IS X VALUE SPACE.
02  DEBUG-CONTENTS      PICTURE IS X (n).
```

11. Vor jeder Ausführung einer Testhilfe-SECTION wird das durch DEBUG-ITEM angesprochene Datenfeld mit Leerzeichen aufgefüllt. Die Inhalte der dem DEBUG-ITEM untergeordneten Datenfelder werden unmittelbar bevor die Steuerung an diese Testhilfe-SECTION übergeben wird, entsprechend den folgenden Allgemeinen Regeln fortgeschrieben. Ein Datenfeld, das nicht in den folgenden Allgemeinen Regeln angegeben ist, enthält nur Leerzeichen.

    Die Fortschreibung wird in Übereinstimmung mit den Regeln für die MOVE-Anweisung durchgeführt; die einzige Ausnahme ist die Übertragung an DEBUGGING-CONTENTS, wobei die Übertragung genauso behandelt wird, als ob es sich um eine alphanumerische zu alphanumerische elementare Übertragung handeln würde, ohne Konvertierung der Daten von einer Form der internen Darstellung in eine andere.

12. In DEBUG-LINE steht die entsprechende Zeilenzahl des COBOL Quellprogramms. Dadurch besteht die Möglichkeit, eine ganz bestimmte Quellanweisung zu identifizieren.

13. DEBUG-NAME enthält die ersten 30 Zeichen des Namens, der die Testhilfe-SECTION anstößt.

    Alle Kennzeichner des Namens werden in DEBUG-NAME durch das Wort IN oder OF getrennt.

    Subskripte oder Indizes, sofern vorhanden, werden nicht in DEBUG-NAME angegeben.

14. Wenn ein Datenfeld beim Ablauf der Testhilfe-SECTION subskribiert oder indiziert ist, wird die Nummer jeder Stufe in DEBUG-SUB-1, DEBUG-SUB-2 bzw. DEBUG-SUB-3 angegeben.

15. DEBUG-CONTENTS ist ausreichend groß beschrieben, die in den folgenden Allgemeinen Regeln geforderten Daten aufnehmen zu können.

16. Wenn die erste Ausführung der ersten nichtvereinbarenden Prozedur im Programm die Ausführung der Testhilfe-SECTION zur Folge hat, bestehen die nachfolgend genannten Bedingungen:

    - DEBUG-LINE identifiziert die erste Anweisung dieser Prozedur.
    - DEBUG-NAME enthält den Namen dieser Prozedur.

- DEBUG-CONTENTS enthält "START PROGRAM".

17. Wenn prozedurname-1 in einer ALTER-Anweisung eine Testhilfe-SECTION einleitet, bestehen die nachfolgend genannten Bedingungen:

    - DEBUG-LINE identifiziert die ALTER-Anweisung, die auf prozedurname-1 Bezug nimmt.
    - DEBUG-NAME enthält prozedurname-1.
    - DEBUG-CONTENTS enthält den anwendbaren Prozedur-Namen, verknüpft mit der TO-Angabe der ALTER-Anweisung.

18. Wenn die GO TO-Anweisung der Testhilfe-SECTION die Steuerung übergibt, bestehen die nachfolgend genannten Bedingungen:

    - DEBUG-LINE identifiziert die GO TO-Anweisung, deren Ausführung die Steuerung an prozedurname-1 überträgt.
    - DEBUG-NAME enthält prozedurname-1.

19. Wenn prozedurname-1 in der INPUT- oder OUTPUT-Angabe einer SORT- oder MERGE-Anweisung die Testhilfe-SECTION auslöst, bestehen die nachfolgend genannten Bedingungen:

    a) DEBUG-LINE identifiziert die SORT- oder MERGE-Anweisung, die sich auf prozedurname-1 bezieht.

    b) DEBUG-NAME enthält prozedurname-1.

    c) DEBUG-CONTENTS enthält:

        - "SORT INPUT", wenn sich prozedurname-1 auf die INPUT-Angabe einer SORT-Anweisung bezieht.
        - "SORT OUTPUT", wenn sich prozedurname-1 auf die OUTPUT-Angabe einer SORT-Anweisung bezieht.
        - "MERGE OUTPUT", wenn sich prozedurname-1 auf die OUTPUT-Angabe einer MERGE-Anweisung bezieht.

20. Wenn die Übertragung der Steuerung zusammen mit einer PERFORM-Anweisung die Ausführung der Testhilfe-SECTION mit prozedurname-1 zur Folge hat, bestehen die nachfolgend genannten Bedingungen:

    - DEBUG-LINE identifiziert die PERFORM-Anweisung, die sich auf prozedurname-1 bezieht.
    - DEBUG-NAME enthält prozedurname-1.
    - DEBUG-CONTENTS enthält "PERFORM LOOP".

21. Wenn prozedurname-1 eine USE Prozedur ist, die nicht ausgeführt wird, bestehen die nachfolgend genannten Bedingungen:

    - DEBUG-LINE identifiziert die Anweisung, die die Ausführung der-USE Prozedur zur Folge hat.
    - DEBUG-NAME enthält prozedurname-1.
    - DEBUG-CONTENTS enthält "USE PROCEDURE".

22. Wenn der vorhergehende Paragraph die Steuerung an prozedurname-1 übergibt und dies die Testhilfe-SECTION anstößt, bestehen die nachfolgend genannten Bedingungen:

    - DEBUG-LINE bezeichnet die vorhergehende Anweisung,
    - DEBUG-NAME enthält prozedurname-1,
    - DEBUG-CONTENTS enthält "FALL THROUGH".

23. Wenn die Bezugnahme auf dateiname-1 die Ausführung der Testhilfe-SECTION zur Folge hat, dann

    - identifiziert DEBUG-LINE die Quellanweisung, die sich auf dateiname-1 bezieht,
    - enthält DEBUG-NAME den Namen von dateiname-1,
    - enthält DEBUG-CONTENTS für READ den ganzen Speicherbereich des Datensatzes,
    - enthält DEBUG-CONTENTS bei allen anderen Bezugnahmen auf dateiname-1 Leerzeichen.

24. Wenn eine Bezugnahme auf bezeichner-1 die Ausführung der Testhilfe-SECTION zur Folge hat, dann

    - identifiziert DEBUG-LINE die Quellanweisung, die sich auf bezeichner-1 bezieht,
    - enthält DEBUG-NAME den Namen von bezeichner-1,
    - enthält DEBUG-CONTENTS den Inhalt des durch bezeichner-2 benannten Datenfeldes zu dem Zeitpunkt, an dem die Steuerung an die Testhilfe-SECTION übergeht (vgl. Allgemeine Regeln 6 und 7).

Testhilfezeilen

Eine Testhilfezeile ist jede Zeile mit einem "D" im Indikatorbereich der Zeile. Jede Testhilfezeile, die nur aus Leerzeichen von Rand A bis Rand R besteht, wird als eine leere Zeile betrachtet.

Der Inhalt einer Testhilfezeile muß gewährleisten, daß ein syntaktisch korrektes Programm erstellt wird, wobei die eventuell vorhandenen Testhilfezeilen als Kommentarzeilen betrachtet werden.

Es wird angenommen, daß eine Testhilfezeile alle Merkmale einer Kommentarzeile hat, wenn die WITH DEBUGGING MODE-Klausel im SOURCE-COMPUTER Paragraph nicht angegeben ist.

Aufeinanderfolgende Testhilfezeilen sind zulässig. Die Fortsetzung von Testhilfezeilen ist zulässig mit der Ausnahme, daß jede Fortsetzungszeile ein "D" im Indikatorbereich haben muß, und daß innerhalb von Zeichenfolgen kein Zeilenumbruch erfolgt.

Im Programm ist eine Testhilfezeile nur nach dem OBJECT-COMPUTER-Paragraph zulässig.

12 Programmiertechnik und Programm dimensionierung

12.1 Programmiertechniken

Obwohl COBOL im wesentlichen frei geschrieben ist, hat der Anwender viele Vorteile, wenn er sich selbst Regeln auferlegt. Diese Regeln sind:

1. Sie erhalten einen kompakteren und effektiveren Code, wenn Sie häufig angesprochene Variablen in die ersten 256 Byte der WORKING-STORAGE SECTION legen.
2. Verwenden Sie Subskripte so sparsam wie möglich! Jedes Subskript benötigt so viel Speicherplatz wie ein normaler Befehl.
3. Für ACCEPT und DISPLAY generiert der Übersetzer einen Befehl für jeden elementaren Teil des Datennamens, der angegeben oder gelesen werden soll. Deshalb sollten Feldergruppen als einzelnes Feld neu definiert werden, um sie mit DISPLAY anzugeben. Vermeiden Sie beim ACCEPT viele kleine Felder!
4. Für ein Datenelement, das nicht explizit angesprochen wird, sollte FILLER anstelle eines Datennamens verwendet werden, da das Wort FILLER im Datenverzeichnis zu einem Zeichen komprimiert wird.
5. Halten Sie die Anzahl der Ziffern in einem numerischen Feld möglichst klein!
6. Übertragen Sie mit einer MOVE-Anweisung Datengruppen und nicht viele einzelne Datenelemente.

Nützliche Hinweise

Wenn Sie interaktive Programme schreiben, sollten Sie die folgenden Möglichkeiten von LII COBOL beachten:

1. Mit der CURSOR IS Einrichtung und der ACCEPT-Anweisung ist es einfach, nach einem Datenanforderungsmenü in Abhängigkeit von der Cursor-Position bedingt zu programmieren. Sie müssen dabei lediglich den Cursor an die benötigte Position bewegen, um die Anforderung zu beantworten oder RETURN-Taste drücken.

2. Mit der Angabe ACCEPT FROM CONSOLE können dem Programm durch die Aufrufzeile Parameter übergeben werden (vgl. Die ACCEPT-Anweisung in Kapitel 2).

3. Wenn die STOP Literal Anweisung in einem Programm verwendet wird, wird die Ausführung des Programms bei dieser Anweisung angehalten das Literal wird auf dem Bildschirm ausgegeben. Wenn Sie dann die RETURN-Taste drücken, läuft die Ausführung weiter.

Bitte beachten Sie: Wenn eine Zeichenfolge, mit der RETURN-Taste abgeschlossen, gelesen werden soll, hält möglicherweise das Programm nicht an. Das kann daran liegen, daß bei der Eingabe des letzten Kommandos die RETURN-Taste versehentlich zweimal gedrückt wurde.

12.2 Dimensionierung

Allgemeines zur Programmdimensionierung

Drei Aspekte bestimmen die Größe eines Programms:

— der Quellcode

— das Datenverzeichnis

— der übersetzte Code

Die Höchstzahl an Quellprogrammanweisungen pro Programm ist beschränkt durch den verfügbaren Speicherplatz

- für das Datenverzeichnis
- das erzeugte Programm zu laden.

Das Datenverzeichnis enthält für jeden durch den Programmierer im Programm definierten Namen eine Eintragung. Einzelheiten sind im nächsten Abschnitt erläutert.

Die Größe eines ausführbaren Programms, das vom Übersetzer generiert wurde, berechnet sich so:

Die Summe der Datensatzgröße aller Dateien in Bytes

- die Datensatzgröße für jeden Datensatz der WORKING-STORAGE SECTION in Bytes
- die Anzahl der Zeichen aller Literale in der PROCEDURE DIVISION
- 60 Bytes pro Datei
- 300 Bytes Kontrollbereich
- 6 Bytes pro COBOL Befehl mit den folgenden Kennzeichnern:

  auf eine ACCEPT/DISPLAY-Anweisung werden drei Bytes pro Datenelement im ACCEPT/DISPLAY-Datennamen addiert,

  für jedes Subskript, das in einer Anweisung verwendet wird, werden 7 Bytes addiert,

  für einen implizit erzeugten Vergleich, wie z. B. durch PERFORM UNTIL oder READ AT END, werden 6 Bytes addiert.

Datenverzeichnis

Das Datenverzeichnis wird während der Übersetzung des Programms erzeugt. Jeder vom Programmierer festgelegte Name hat eine Eintragung in diesem Verzeichnis. Tabelle 12-1 zeigt die Byte-Anzahl, die für jede Eintragung benötigt wird.

vom Programmierer festgelegter Name	Anzahl Bytes *
Dateiname	18 + n
Datensatzname	8 + n
Schlüsselname	8 + n
Statusname	8 + n
Paragraphname	6 + n
Gruppenname	8 + n **
Alphanumerisch < 32 Zeichen	7 + n **
Alphanumerisch > 32 Zeichen	8 + n **
Numerische Ganzzahl	7 + n **
Numerisch, aber keine Ganzzahl	8 + n **
Numerisch druckaufbereitet	7 + n + x

* n = Anzahl der Zeichen im vom Programmierer festgelegten Namen

Bei FILLER ist n = 1

x = Anzahl der Zeichen in einem PICTURE, nach zusammenhängenden Wiederholungen

z.B.:

9999.9	= 3 Bytes
9(4).9	= 3 Bytes
Z(2)9(4).9(3)	= 4 Bytes

** Wenn das Datenfeld innerhalb der ersten 256 Bytes der WORKING-STORAGE SECTION liegt, ist 1 Byte abzuziehen.

Wenn das Datenfeld mit einer OCCURS-Klausel verknüpft ist, sind 4 Bytes dazu zu addieren.

Wenn das Datenfeld einem mit der OCCURS-Klausel beschriebenen Datenfeld untergeordnet ist, sind 2 Bytes zu addieren.

Tabelle 12-1 Datenverzeichnis - Eintragungsgröße

Stichwörter