Fortran-Kurs technisch orientiert

Einführung in die Programmierung
mit Fortran 77

von
Prof. Dipl.-Ing. Günter Schmitt

8. Auflage

R. Oldenbourg Verlag München Wien 1992

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Schmitt, Günter:
Fortran-Kurs technisch orientiert : Einführung in die Programmierung mit Fortran 77 / von Günter Schmitt. – 8. Aufl. – München ; Wien : Oldenbourg, 1992
ISBN 3-486-22210-4

Unveränderter Nachdruck der 7. Auflage

Gesamtherstellung: Hofmann Druck, Augsburg

ISBN 3-486-22210-4

Inhaltsverzeichnis

1 Einführung 9
1.1 Was ist elektronische Datenverarbeitung? 9
1.2 Wie entsteht ein FORTRAN-Programm? 13
1.3 Wie arbeitet das Betriebssystem? 16
1.4 Zur Geschichte des FORTRAN 18

2 Grundlagen des FORTRAN 21
2.1 Allgemeine Eingaberegeln 21
2.2 Die Darstellung ganzer Zahlen durch INTEGER-Größen 23
2.3 Die Darstellung reeller Zahlen durch REAL-Größen 24
2.4 Einfache Dateneingabe und Datenausgabe im Dialog 27
2.5 Die Programmierung von Formeln 29
2.6 Die wichtigsten Steueranweisungen 34
2.7 Die Struktur der einfachen Verarbeitungsschleife 35
2.8 Übungen zum Abschnitt Grundlagen 39
2.9 Die häufigsten Fehler 40

3 Programmverzweigungen 42
3.1 Die Vergleichsoperationen 42
3.2 Die einseitig bedingte Anweisung, das logische IF 44
3.3 Der Aufbau von Blockstrukturen durch das Block-IF 45
3.4 Programmverzweigungen mit dem arithmetischen IF 52
3.5 Programmverzweigungen mit dem GOTO-Befehl 54
3.6 Übungen zum Abschnitt Programmverzweigungen 58

4 Programmschleifen 60
4.1 Die Eingabeschleife 60
4.2 Der Aufbau von Zählschleifen mit der DO-Anweisung 61
4.3 Die Programmierung von Schleifen mit bedingten Sprungbefehlen . . . 65
4.4 Die Programmierung von Näherungsschleifen (Iteration) 68
4.5 Übungen zum Abschnitt Programmschleifen 71

5 Indizierte Variablen (Felder) 73
5.1 Die Arbeit mit eindimensionalen Feldern 75
5.2 Übungen zum Abschnitt eindimensionale Felder 85
5.3 Die Arbeit mit mehrdimensionalen Feldern 86
5.4 Übungen zum Abschnitt mehrdimensionale Felder 90
5.5 Die häufigsten Fehler bei der Verwendung von Feldern 91

6 Unterprogrammtechnik 93
6.1 Standard-Unterprogramme 93
6.2 Die Funktionsanweisung 95
6.3 FUNCTION-Unterprogramme 98
6.4 SUBROUTINE-Unterprogramme 101
6.5 Die Verwendung von Feldern in Unterprogrammen 103
6.6 Übungen zum Abschnitt Unterprogrammtechnik 108
6.7 Die Arbeit mit externen Unterprogrammen 109

6.8 Weitere Möglichkeiten der Unterprogrammtechnik 113
6.9 Die grafische Darstellung von Unterprogrammen 120
6.10 Fehler bei der Verwendung von Unterprogrammen 122

7 Die Eingabe und Ausgabe von Daten 124
7.1 Die listengesteuerte Ein/Ausgabe über die Konsole 124
7.2 Die formatgesteuerte (formatierte) Ein/Ausgabe über die Konsole . . 124
7.3 Die Arbeit mit Datendateien 132
7.4 Die Arbeit mit zeilenorientierten Dateien 136
7.5 Die Arbeit mit sequentiellen Datendateien (Banddateien) 138
7.6 Die Arbeit mit Datendateien im direkten Zugriff 145
7.7 Die Arbeit mit internen Dateien im Arbeitsspeicher 151
7.8 Übungen zum Abschnitt Ein/Ausgabe von Daten 153

8 Datentypen 154
8.1 Allgemeines 154
8.2 Größen vom Typ DOUBLE PRECISION 156
8.3 Größen vom Typ COMPLEX 159
8.4 Größen vom Typ LOGICAL 165
8.5 Textverarbeitung mit Größen vom Typ CHARACTER 171
8.6 Übungen zum Abschnitt Datentypen 178

9 Aufgabensammlung 179
9.1 Numerische Integration 179
9.2 Fourieranalyse 182
9.3 Interpolation und Darstellung von Funktionen 183
9.4 Elektrotechnische Aufgaben 188

10 Ergänzende und weiterführende Literatur 190

11 Lösungen der Übungsaufgaben 191

12 Anhang 216
Mathematische Bibliotheksfunktionen 216
Mathematische Einbaufunktionen 217
Sinnbilder für Programmablaufpläne und Datenflußpläne 218
Sinnbilder für Struktogramme 220
Verzeichnis der Beispiele und Aufgaben auf der Programmdiskette . . 221

13 Stichwortverzeichnis 230

Vorwort zur 8. Auflage

Die vorliegende Auflage orientiert sich weiterhin an dem Standard Fortran 77, dessen Sprachumfang in einführenden Lehrveranstaltungen nicht immer voll ausgeschöpft werden kann. Die neueren Versionen der Fortran Compiler haben inzwischen fast alle den Datentyp "COMPLEX" implementiert, so daß die Beispiele und Übungen des Abschnitts 8.3 direkt getestet werden können. Mit Hilfe der dort abgebildeten Unterprogramme lassen sich der Zahlenbereich und die Genauigkeit der komplexen Rechnung erweitern. Ich hoffe, daß damit die Diskussion um die komplexe Rechnung, die man in anderen Programmiersprachen schmerzlich vermißt, abgeschlossen ist.

Groß-Umstadt, im März 1992 Günter Schmitt

Vorwort zur 6. Auflage

Aufgrund von Leserzuschriften möchte ich besonders darauf hinweisen, daß in der von mir verwendeten Version 2.0 des FORTRAN-Compilers die Variablen vom Typ COMPLEX (Abschnitt 8.3) nicht verfügbar sind. Es wird jedoch gezeigt, wie die komplexen Funktionen mit Hilfe von Unterprogrammen realisiert werden können. Allen Lesern, die mich darauf aufmerksam gemacht haben, sei an dieser Stelle herzlich gedankt.

Groß-Umstadt, im Dezember 1988 Günter Schmitt

Vorwort zur 4. Auflage

Die vorliegende 4. Auflage wurde der Entwicklung der Programmiersprache FORTRAN und der Softwaretechnik angepaßt, ohne auf die bewährte Struktur und Darstellungsweise des Werkes zu verzichten. Die Lochkarte wurde dem Museum für Datenverarbeitung übergeben und durch das Bildschirmgerät ersetzt, mit dem heute sowohl der Ingenieur am Arbeitsplatzrechner (Personal Computer) als auch der Programmierer von Großrechnern seine Programme eingibt und testet.

Der Abschnitt "Einführung" vergleicht die datenverarbeitenden Geräte Mensch, Taschenrechner, Mikrocomputer und Großrechner und erklärt die Bedeutung des Betriebssystems.

Das Ziel des Abschnitts "Grundlagen" ist es, den Leser möglichst schnell in die Lage zu versetzen, einfache FORTRAN-Programme erfolgreich entwerfen und testen zu können. Nichts ist beim Erlernen einer Programmiersprache wichtiger als die ersten Erfolgserlebnisse, selbst auf die Gefahr hin, daß die ersten Lösungen noch nicht ausgereift sind.

Die folgenden Abschnitte bringen eine systematische Einführung in die verschiedenen Programmiertechniken von Programmverzweigungen, Schleifen, Feldern und Unterprogrammen. Die Beispiele und Übungsaufgaben sind möglichst einfach und leicht durchschaubar gehalten und können alle im Dialog getestet werden.

Während die ersten Abschnitte nur mit ganzen und reellen Zahlen arbeiten, folgen nun Abschnitte über weitere Möglichkeiten der Ein/Ausgabe, über die Arbeit mit externen Speichern und über weitere Datentypen.

Die Aufgabensammlung zeigt Beispiele und Aufgaben aus verschiedenen Gebieten wie z.B. Mathematik, Elektrotechnik und Datenverarbeitung; für alle Aufgaben enthält der Abschnitt 11 Lösungsvorschläge.

Die vorliegende Neubearbeitung orientiert sich am Standard "FORTRAN 77", ohne jedoch auf alle Sprachelemente dieses Standards oder herstellerspezifische Spracherweiterungen einzugehen. Alle Beispiele und Lösungen wurden an einem Arbeitsplatzrechner (IBM AT) getestet. Frau Deinaß vom Verlag R. Oldenbourg danke ich für die seit langem bewährte gute Zusammenarbeit.

Groß-Umstadt, im Juli 1985 Günter Schmitt

1 Einführung

Aller Anfang ist schwer - auch der Einstieg in die Datenverarbeitung. Diese Einführung erklärt Ihnen den Aufbau eines Rechners und einige Grundbegriffe der Datenverarbeitung. Dies geschieht mit Hilfe eines einführenden Beispiels, das wir mit verschiedenen Geräten der Datenverarbeitung, dem rechnerlosen Menschen, dem Taschenrechner, dem programmierbaren Taschenrechner, dem Großrechner und dem neusten Kind der Datenverarbeitung, dem Arbeitsplatzrechner oder Personal Computer, lösen werden.

1.1 Was ist elektronische Datenverarbeitung?

Der folgende Abschnitt beantwortet diese Frage anhand eines einfachen Beispiels, das sicher allen Lesern vertraut ist. Bei der Auswertung von Meßergebnissen haben Sie die Aufgabe, aus fünf Meßwerten den Mittelwert zu bilden. **Bild 1-1** zeigt die Liste mit den fünf Zahlen für jede Meßreihe. In die letzte Spalte tragen Sie als Ergebnis das arithmetische Mittel ein.

Messung	Nr.1	Nr.2	Nr.3	Nr.4	Nr.5	Mittel
10.00	14,3	14,4	12,3	13,0	14,4	
10.30	17,4	15,5	13,9	14,7	16,1	
11.00	19,5	14,9	16,0	17,5	18,3	
11.30						

Bild 1-1: Die Daten des einführenden Beispiels

Sehen wir Ihnen bei der Arbeit zu:

Als **rechnerloser Mensch** nehmen Sie Papier und Bleistift zur Hand, addieren schriftlich die fünf Meßwerte einer Zeile, dividieren die Summe schriftlich durch die Konstante 5 und schreiben das Ergebnis in die letzte Spalte der Tabelle. Dann fahren Sie mit der nächsten Zeile fort, bis keine Zahlen mehr da sind.

In der Fachsprache der Datenverarbeitung ausgedrückt haben Sie die veränderlichen (variablen) Zahlen oder **Eingabedaten** mit dem Eingabegerät Auge vom Datenträger Liste gelesen und auf den Arbeitsspeicher Schmierpapier übertragen. Dort haben Sie gerechnet. Dabei wurde die Zwischensumme durch die konstante Zahl 5 dividiert. Dann haben Sie das Ergebnis, die **Ausgabedaten** , mit dem Ausgabegerät Kugelschreiber auf dem Datenträger Liste ausgegeben.

Daten sind Mitteilungen über bestimmte Dinge; sie bestehen aus Zahlen oder aus Zeichen wie z.B. Texten.

Variablen sind veränderliche Daten, deren Zahlenwert erst während der Datenverarbeitung bekannt wird.

Konstanten sind Daten, deren Zahlenwert bereits vor Beginn der Arbeit feststeht und die sich im Laufe der Verarbeitung auch nicht verändern.

Daten verarbeiten heißt Daten übertragen, speichern, berechnen und auswerten.

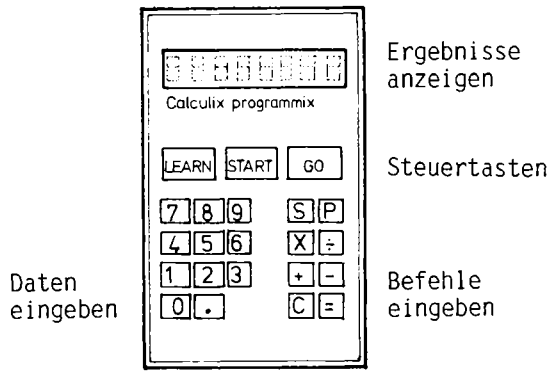

Bild 1-2: Taschenrechner (programmierbar)

Als moderner Mensch rechnen Sie nicht mehr mit Papier und Bleistift, sondern benutzen einen **Taschenrechner (Bild 1-2)** . Für jede Meßreihe müssen Sie nun die Daten (Zahlenwerte) und Funktionstasten für die Rechenoperationen eingeben. Die Rechentasten (+ , - , x und /) sind Anweisungen an den Taschenrechner, Zahlen (Daten) miteinander zu verknüpfen. Bei umfangreichen Formeln wie z.B. bei der Berechnung von Filterkurven wird es nun sehr lästig, bei jeder Rechnung neben den Zahlen auch noch die Rechenoperationen eingeben zu müssen. Sie werden daher zu einem **programmierbaren Taschenrechner** greifen. Dieser kennt z.B. die beiden Betriebsarten "LEARN" zur Eingabe der Rechenschritte und "START" zur Eingabe der Daten, die mit den gespeicherten Rechenschritten berechnet werden. In unserem einfachen Beispiel bestehen die Rechenschritte (Befehle) aus Additionsbefehlen, der Konstanten 5 und dem Divisionsbefehl. Sie bilden eine Arbeitsanweisung (Programm), das im Taschenrechner gespeichert wird und das sich beliebig oft auf die eingegebenen Daten anwenden läßt.

Ein **Befehl** ist eine Anweisung an den Rechner, etwas Bestimmtes zu tun z.B. zwei Zahlen zu addieren.

Mehrere Befehle bilden zusammen mit konstanten Daten ein **Programm** z.B. zur Berechnung des Mittelwertes aus fünf Zahlen.

Liegt das Programm im **Programmspeicher** des Rechners, so kann es beliebig oft auf verschiedene Daten angewendet werden.

Sicherlich werden Sie im Laufe der Zeit mehrere Programme für Ihren programmierbaren Taschenrechner schreiben und sich eine Programmbibliothek in einem Aktenordner anlegen. Vor jedem Programmlauf müssen Sie das Programm der Bibliothek entnehmen, laden und starten. Meßreihen lassen sich nach verschiedenen Gesichtspunkten auswerten wie z.B. Mittelwert bilden, Standardabweichung berechnen oder die Funktion grafisch darstellen. Sie werden also auch Ihre Meßreihen in einem Aktenordner aufbewahren und, natürlich mit Hilfe von Programmen, nach verschiedenen Gesichtspunkten auswerten. In der Datenverarbeitung legt man Programme und Daten auf magnetischen Speichern ab. Dazu gehören Magnetkarten (bei Taschenrechnern), Floppy Disks oder Disketten bei Personal Computern und Plattenlaufwerke und Magnetbandstationen bei Großrechnern. Ergebnisse können ebenfalls auf einem Magnetspeicher ausgegeben werden.

Eine **Datei** ist eine Sammlung von Programmen oder Daten auf einem - meist magnetischen - Speicher.

Wenn Sie als Physiker in einem Kernforschungsinstitut Meßreihen auswerten müssen, so werden Sie es vermutlich mit einem Taschenrechner nicht weit bringen, wenn bei einem Experiment die Daten schneller anfallen als Sie sie verarbeiten können. Zunächst werden Sie versuchen, die Meßwerte nicht mehr einzeln von einem Meßgerät abzulesen und aufzuschreiben, sondern automatisch zu erfassen und auf einem Magnetband zu speichern. Für die Auswertung steht Ihnen ein Großrechner mit mehr als 1 Million Rechenoperationen pro Sekunde zur Verfügung, den Sie natürlich auch verwenden, wenn Sie für eine theoretische Überprüfung Ihrer Meßergebnisse z.B. Formeln auswerten. Die Programme werden Sie sowohl bei der Auswertung von Meßreihen als auch bei der Berechnung von Formeln selbst schreiben müssen, weil nur Sie das Rechenverfahren auswählen und die Ergebnisse kritisch beurteilen können. Mit den technischen Einzelheiten des Großrechners können und wollen Sie sich nicht beschäftigen, Sie interessieren sich nur für Ihre Forschungen. Sie wollen also nicht mehr wie bei der Programmierung eines Taschenrechners die einzelnen Rechenschritte, sondern nur die auszuwertende Formel eingeben. Dazu benötigen Sie ein Programm, das die Formel aus der wissenschaftlichen Schreibweise in die Befehle der Rechenanlage übersetzt. Ein derartiges Übersetzungsprogramm nennt man einen Compiler.

Die Programmierung von technisch-wissenschaftlichen Anwendungen geschieht in der Regel in einer aufgabenorientierten Programmiersprache, deren Befehle durch einen **Compiler** in die Befehle der Rechenanlage übersetzt werden.

Wie ist nun ein **Großrechner** aufgebaut? **Bild 1-3** zeigt die Funktionseinheiten Eingabe, Zentraleinheit und Ausgabe.

Die wichtigsten **Eingabegeräte** sind eine Tastatur für die Bedienung und die Eingabe von Programmen bei der Programmentwicklung sowie Magnetband- und Plattengeräte für die Eingabe von Daten und Programmen aus Dateien. Dazu gehört auch der Compiler für die Übersetzung von Programmen aus einer höheren Programmiersprache in die Maschinensprache. In der Ecke steht vielleicht

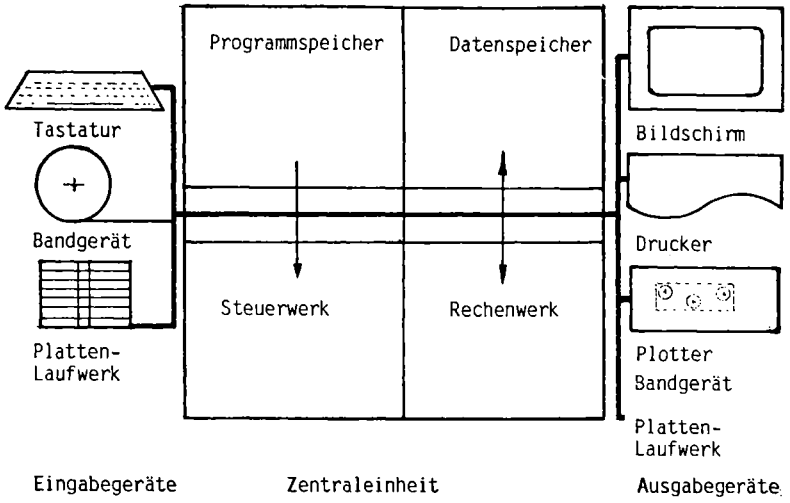

Bild 1-3: Aufbau eines Großrechners

noch ein Kartenleser, um auch noch ältere Programme aus der Lochkartenzeit laden zu können.

Die **Zentraleinheit** besteht aus einem Programmspeicher für das auszuführende Programm und einem Steuerwerk, das sich die Maschinenbefehle einzeln holt, decodiert und ausführt. Die Rechenwerk verarbeitet die Daten. Zwischenergebnisse werden kurzzeitig im Datenspeicher aufbewahrt.

Die wichtigsten **Ausgabegeräte** sind ein Bildschirmgerät für die Bedienung und die Programmentwicklung, Drucker und Plotter für Listen und Zeichnungen sowie magnetische Speicher für die Aufbewahrung von Ergebnissen, die weiterverarbeitet werden sollen.

Vielleicht steht an Ihrem Arbeitsplatz aber auch ein Personal Computer, Ihr persönlicher Rechner, der nur etwa 10 000 bis 100 000 Rechenoperationen in der Sekunde ausführen kann. **Bild 1-4** zeigt den Aufbau eines **Arbeitsplatzrechners** . Der Mikroprozessor enthält das Steuerwerk und das Rechenwerk. Bei den Speicherbausteinen unterscheidet man Festwertspeicher (EPROM) für Betriebsprogramme und Schreib/Lesespeicher (RAM) für Anwenderprogramme und Daten sowie für das Betriebssystem. Über Peripheriebausteine sind ein Bildschirmgerät für die Bedienung, der Drucker für die Datenausgabe und magnetische Speicher (Diskettenlaufwerke) angeschlossen. Auch mit diesem Rechner können Sie in aufgabenorientierten Sprachen wie z.B. FORTRAN programmieren. Nur arbeitet der Arbeitsplatzrechner bei der Übersetzung und Ausführung des Programms langsamer als der Großrechner.

Das folgende einführende Beispiel sowie alle Programme dieses Buches können sowohl auf einem Großrechner (z.B. UNIVAC 1100) als auch auf einem Arbeitsplatzrechner (z.B. IBM PC) übersetzt und ausgeführt werden.

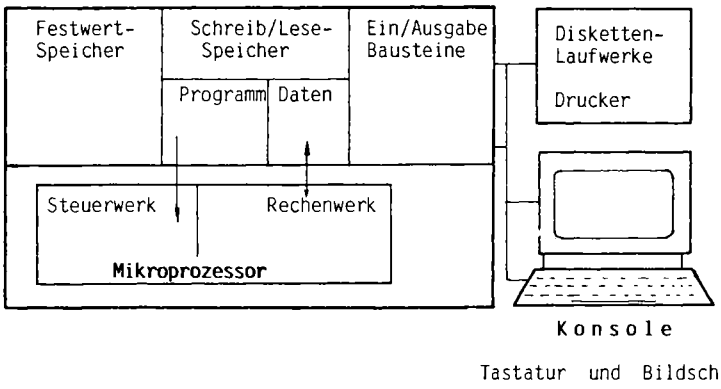

Bild 1-4: Aufbau eines Arbeitsplatzrechners (Personal Computer)

1.2 Wie entsteht ein FORTRAN-Programm?

Aufgabe:
Man entwickle ein FORTRAN-Programm, das aus fünf einzulesenden Zahlen den Mittelwert berechnet und ausgibt. Es sollen beliebig viele Meßreihen zu je fünf Meßwerten verarbeitet werden. Für den Mittelwert Null werde das Programm beendet.

Bild 1-5: Aufgabenstellung des einführenden Beispiels

Die Aufgabenstellung **Bild 1-5** verlangt die Berechnung und Ausgabe des Mittelwertes aus fünf einzulesenden Meßwerten einer Meßreihe. Damit das Programm nicht für jede Meßreihe neu geladen und gestartet werden muß, soll es solange Werte lesen und verarbeiten, bis es durch eine Endebedingung abgebrochen wird. Sind keine Eingabedaten mehr vorhanden, so werden fünf Nullen eingegeben, die den Mittelwert Null ergeben und damit das Programm abbrechen. Dabei muß sichergestellt sein, daß der Mittelwert Null nicht als Ergebnis einer zulässigen Dateneingabe erscheint. Wir werden später bessere Möglichkeiten kennenlernen, ein Programm anzuhalten. Die Lösung der Aufgabe wird nun in einem Struktogramm entsprechend **Bild 1-6** grafisch dargestellt.

Das Struktogramm zeigt eine Lösung als Dialogprogramm. Auf dem Bildschirm erscheint zunächst eine Meldung für den Benutzer, daß das Programm auf die Eingabe von Daten über die Tastatur wartet. Die Ergebnisse erscheinen auf dem Bildschirm und können gleichzeitig auch auf dem Drucker ausgegeben werden. Der Anhang enthält eine Zusammenstellung der Sinnbilder für Struktogramme.

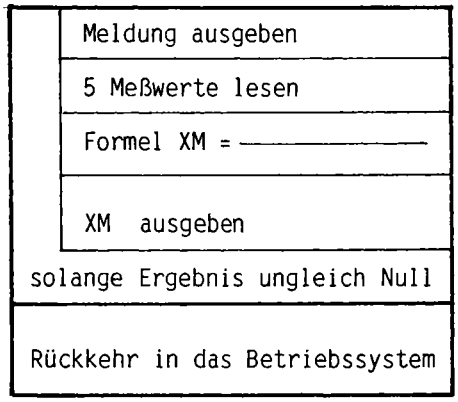

Bild 1-6: Struktogramm des einführenden Beispiels

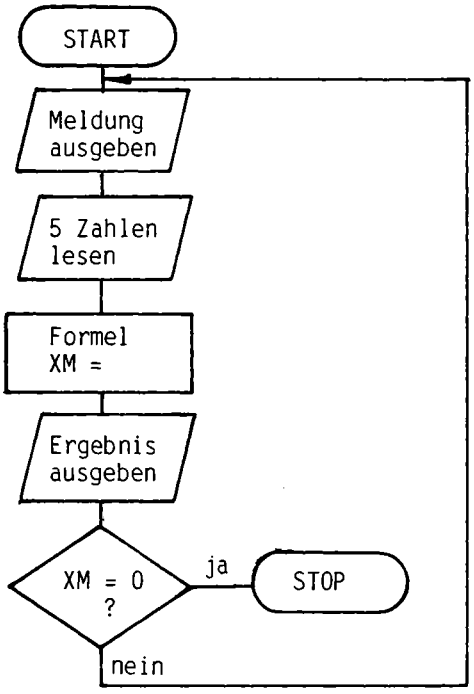

Bild 1-7: Programmablaufplan des einführenden Beispiels

Der Programmablaufplan **Bild 1-7** zeigt im Gegensatz zum Struktogramm bereits die einzelnen Befehle. Der Anhang enthält eine Zusammenstellung der Sinnbilder für Programmablaufpläne. Nun folgt die Programmierung auf einem Entwurfsformular, das bereits die Eingabevorschriften der Programmiersprache FORTRAN berücksichtigt. Wir wollen das Programm **Bild 1-8** kurz erklären.

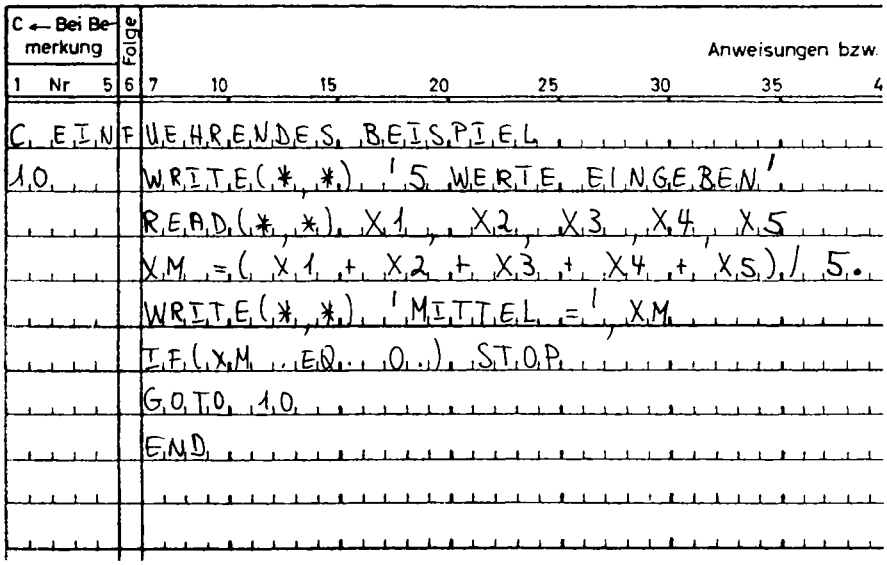

Bild 1-8: FORTRAN-Programm des einführenden Beispiels

Die 1. Zeile beginnt mit einem C in Spalte 1 und ist eine Kommentarzeile, die das Programm beschreibt und keine Anweisung an den Rechner darstellt.

Die 2. Zeile enthält eine Marke 10 und eine WRITE-Anweisung, die die Meldung "5 WERTE EINGEBEN" auf dem Bildschirm erscheinen läßt.

Die 3. Zeile besteht aus einer Leseanweisung READ, mit der 5 Dezimalzahlen gelesen, in Dualzahlen umgewandelt und abgespeichert werden.

Die 4. Zeile besteht aus der Formel zur Berechnung des arithmetischen Mittelwertes aus 5 Meßwerten. Das Ergebnis wird in der Speicherstelle XM abgelegt.

Die 5. Zeile enthält eine Ausgabeanweisung, mit der das Ergebnis zusammen mit dem Text "MITTEL =" auf dem Bildschirm erscheint.

Die 6. Zeile enthält die Abbruchbedingung. Wenn sich ein Mittelwert Null ergibt, so stoppt das Programm.

Ist der Mittelwert ungleich Null und damit die Abbruchbedingung nicht erfüllt, so setzt der Sprungbefehl GOTO der 7. Zeile das Programm an der Sprungmarke 10 mit der Ausgabe einer neuen Meldung fort.

Die 8. Zeile mit der END-Anweisung zeigt dem Übersetzer, dem FORTRAN-Compiler, das Ende der Programmzeilen an.

Das Programm muß nun in den Rechner eingegeben, übersetzt, geladen und gestartet werden. Dies geschieht mit Hilfe des Betriebssystems.

1.3 Wie arbeitet das Betriebssystem?

Bei einem (programmierbaren) Taschenrechner muß der Benutzer alle Arbeiten, die für die Bedienung des Gerätes erforderlich sind, selbst ausführen. Dazu muß er das Programm in der Programmbibliothek suchen, in den Rechner eingeben und dann starten. Die Ergebnisse sind von der Anzeige abzulesen und auf die Liste zu übertragen. Bei einem Arbeitsplatzrechner oder gar Großrechner würden diese einzelnen Handgriffe zu unerträglich langen Verarbeitungszeiten führen. Daher läßt man in den Rechnern Programme ablaufen, die die Arbeit der **Anlage** steuern und überwachen. Sie bilden das Betriebssystem. Die Programme des Betriebssystems liegen entweder fest im Arbeitsspeicher oder werden bei Bedarf von den magnetischen Speichern geladen. Der Benutzer ruft sie mit Kommandos auf. Bei einem Arbeitsplatzrechner könnten diese Befehle wie folgt aussehen:

```
EDLIN  TEST.FOR
```

Der Editor EDLIN ist ein Systemprogramm, mit dessen Hilfe das FORTRAN-Programm in den Rechner eingegeben und in einer Datei gespeichert wird. Mit seiner Hilfe lassen sich Programmzeilen eingeben, ändern, löschen und einfügen. Das FORTRAN-Programm erhält in dem vorliegenden Beispiel den frei gewählten Namen TEST mit dem Zusatz FOR als Abkürzung für FORTRAN.

```
FOR1  TEST,,,;
```

Der 1. Durchlauf des Compilers FOR1 prüft die Programmzeilen und gibt gegebenenfalls Fehlermeldungen aus. Der erzeugte Maschinencode wird auf der Diskette abgelegt.

```
FOR2
```

Der 2. Durchlauf des Compilers FOR2 optimiert den Maschinencode, um eine möglicht geringe Laufzeit des Programms zu erzielen.

```
LINK  TEST,,;
```

Der Binder LINK verbindet das übersetzte Maschinenprogramm mit Hilfsprogrammen wie z.B. Unterprogrammen des Betriebssystems für die Ein/Ausgabe.

```
TEST
```

Das fertige Programm wird durch Eingabe des Namens von der Diskette in den Arbeitsspeicher geladen und gestartet. Es meldet sich auf dem Bildschirm mit der Nachricht "5 WERTE EINGEBEN".

```
B>EDLIN TEST.FOR
New file
*I
     1:*C Einführendes Beispiel Mittelwert
     2:*10        WRITE(*,*) '5 WERTE EINGEBEN'
     3:*          READ(*,*) X1,X2,X3,X4,X5
     4:*          XM = (X1 + X2 + X3 + X4 + X5) / 5
     5:*          WRITE(*,*) 'MITTEL =', XM
     6:*          IF(XM.EQ.0) STOP
     7:*          GOTO 10
     8:*          END
     9:*^Z
*E

B>FOR1 TEST,,,;

IBM Personal Computer FORTRAN Compiler
Version 2.00
(C)Copyright IBM Corp 1982, 1984
(C)Copyright Microsoft Corp 1982, 1984

 Pass One    No Errors Detected
              8 Source Lines

B>FOR2

 Code Area Size = #012E   (  302)
 Cons Area Size = #0024   (   36)
 Data Area Size = #001A   (   26)

 Pass Two    No Errors Detected.

B>LINK TEST,,;

IBM Personal Computer Linker
Version 2.20 (C)Copyright IBM Corp 1981, 1982, 1983, 1984


B>TEST
5 WERTE EINGEBEN
1 2 3 4 5
MITTEL =          3.0000000
5 WERTE EINGEBEN
17.3 22.5 4.9 11.7 30
MITTEL =         17.2800000
5 WERTE EINGEBEN
1.2E3 3.6E5 33E4 12E-4 2.24E5
MITTEL =     183040.0000000
5 WERTE EINGEBEN
0  0  0  0  0
MITTEL =  0.000000E+000
Stop - Program terminated.

B>
```

Bild 1-9: Testlauf des einführenden Beispiels (IBM PC)

Der in **Bild 1-9** dargestellte Testlauf entstand an einem Arbeitsplatzrechner und zeigt die Eingabe, die Übersetzung, die Verbindung mit Systemprogrammen und den Start des einführenden Beispiels. Alle Eingaben des Benutzers wurden nachträglich unterstrichen. Es wurden drei Meßreihen mit je fünf Meßwerten eingegeben und zwar in den Zahlendarstellungen ganzzahlig (z.B. 1), reell (z.B. 17.3) und in der Exponentenschreibweise (z.B. 1.2E3). Die Meßreihe 0 0 0 0 0 ergab den Mittelwert 0 und beendete das Programm.

Bei der Arbeit mit dem Betriebssystem eines Großrechners gibt es andere Betriebssystemkommandos, aber die Arbeit im Prinzip die gleiche:

FORTRAN-Programm mit dem Editor eingeben,
FORTRAN-Programm mit dem Compiler prüfen und übersetzen,
übersetztes Programm mit Systemprogrammen verbinden,
fertiges Maschinenprogramm laden und starten sowie
falls im Programm vorgesehen Daten eingeben.

An einem Arbeitsplatzrechner arbeitet normalerweise nur ein Benutzer gleichzeitig, und der Rechner führt auch nur ein Programm gleichzeitig aus. Bei Großrechnern unterscheidet man mehrere Betriebsarten:

Großrechner arbeiten so schnell, daß besonders bei der Eingabe und Ausgabe von Daten erhebliche Wartezeiten entstehen können. Während dieser Pausen läßt man andere Programme arbeiten. Im **Timesharing-Betrieb** laufen mehrere Programme zeitlich verschachtelt in einer einzigen Zentraleinheit ab. Das Betriebssystem teilt jedem Programm je nach Wichtigkeit (Priorität) Rechenzeit zu. Damit können z.B. 50 Benutzer an Bildschirmgeräten an einem Großrechner arbeiten. Durch die zeitliche Verschachtelung hat jeder den Eindruck, der Rechner arbeite nur für ihn. Laufen jedoch gleichzeitig rechenintensive Programme mit höherer Priorität, so kann es zu Wartezeiten kommen.

Die höchste Priorität haben **Echtzeitprogramme,** die z.B. für die Steuerung einer Raffinerie die ankommenden Daten sofort verarbeiten müssen. Programme werden heute fast ausschließlich im **Dialogbetrieb** mit einem Bildschirmgerät eingegeben und getestet. Bildschirmgeräte erhalten daher die nächsthöhere Priorität, denn Programmierer sind hochbezahlte Arbeitskräfte, die immer an ganz dringenden Projekten arbeiten. Bei der Wirkverarbeitung laufen die Anwendungsprogramme oft im **Stapelbetrieb,** d.h. sie werden ohne Priorität in der Reihenfolge ausgeführt, in der die Start-Kommandos eingegeben wurden. Die Programme des Betriebssystems haben die allerhöchste Priorität, denn sie müssen ja den Ablauf und den Betrieb der gesamten Rechenanlage überwachen.

1.4 Zur Geschichte des FORTRAN

Nichts geschieht von allein, auch nicht in der Datenverarbeitung. Daher benötigt auch ein Computer eine Arbeitsanweisung, die Programm genannt wird und die in einer dem Rechner verständlichen Darstellung abgefaßt sein muß. In der Steinzeit der Datenverarbeitung wurde binär programmiert. Das bedeutet, daß alle Daten und Befehle durch zwei Zustände verschlüsselt (codiert) werden müssen, also durch "Loch" bzw. "Nicht Loch" eines Papierstreifens oder durch "Schalter oben" bzw. "Schalter unten" eines Schalterfeldes. Dann wurde der Assembler erfunden. Der Programmierer arbeitet mit symbolischen Befehlen wie z.B. LDA für "Lade Akkumulator" oder JMP für "Springe zu einem bestimmten Befehl". Die Eingabe erfolgt über eine Schreibmaschinentastatur. Ein Übersetzungsprogramm, der Assembler, wandelt die symbolischen Befehle um in den binären Code der Maschine. Da jeder Rechnertyp eine eigene Maschinensprache hat, lassen sich Assemblerprogramme nicht zwischen verschiedenen Rechnertypen austauschen. Beispiel:

Assemblerbefehl des Rechners XYZ 85:

```
LDA  ADAT  ; Lade den Akkumulator mit dem Inhalt von ADAT
```

Übersetzter Maschinenbefehl des Rechners XYZ 85:

```
00111010010011111000010001
```

Dann wurde etwa um 1954 ein Formelübersetzer (FORmula TRANslator oder FORTRAN) geschaffen, der Formeln aus der mathematisch-technischen Schreibweise in den Maschinencode übersetzen konnte. Beispiel:

Formel:

$$x = \frac{a + b}{c - d}$$

FORTRAN-Schreibweise:

```
X = (A + B) / (C - D)
```

Übersetzte Befehle in der Assemblerschreibweise:

```
LDA  CDAT  ; Lade den Akkumulator mit dem Inhalt von CDAT
SUB  DDAT  ; Subtrahiere vom Akkumulator den Inhalt von DDAT
STA  RETT  ; Speichere die Differenz in eine Hilfsspeicherstelle
LDA  ADAT  ; Lade den Akkumulator mit dem Inhalt von ADAT
ADD  BDAT  ; Addiere zum Akkumulator den Inhalt von BDAT
DIV  RETT  ; Dividiere den Akkumulator durch den Inhalt von RETT
STA  XDAT  ; Speichere das Ergebnis in die Speicherstelle XDAT
```

Zu dem eigentlichen Formelübersetzer treten Eingabebefehle (READ) zur Umwandlung der eingegebenen Dezimalzahlen in die interne binäre Darstellung und Ausgabebefehle (WRITE), mit denen die Ergebnisse aus der internen binären Darstellung wieder in Dezimalzahlen umgewandelt werden. Ein Rechner kann mehr als nur rechnen, er kann auch Zahlen miteinander vergleichen und ist auch imstande, Entscheidungen über den weiteren Ablauf des Programms zu treffen. Dies geschieht durch einfache Kontrollbefehle wie z.B. "Setze das Programm bei einem bestimmten Befehl fort" (GOTO) oder "Verzweige nur dann, wenn ...." (IF). Das Übersetzungsprogramm, das Formeln, Ein/Ausgabeanweisungen und Kontrollbefehle in den binären Maschinencode übersetzt, wird als FORTRAN-Compiler bezeichnet.

Im Laufe der Jahre wurde FORTRAN weiterentwickelt und um weitere Datentypen (z.B. CHARACTER), Datenstrukturen (z.B. Felder), Programmstrukturen (z.B. DO-Schleife) und Ein/Ausgabebefehle (z.B. Plattendateien) erweitert. Der Umfang der Programmiersprache FORTRAN ist heute in den amerikanischen Normen (ANSI X3.9 - 1978) und deutschen Normen (DIN 66027, 1980) festgelegt. Dieser Standard wird als FORTRAN 77 bezeichnet. Damit ist es möglich, ein FORTRAN-Programm unabhängig von einem Maschinentyp zu erstellen und FORTRAN-Programme zwischen verschiedenen Rechnertypen auszutauschen.

In der Praxis hat es sich jedoch gezeigt, daß trotz dieser Normung die FORTRAN-Compiler der verschiedenen Hersteller Unterschiede haben; meist handelt es sich um Spracherweiterungen gegenüber dem Standard-FORTRAN 77.

Neben FORTRAN entstanden im Laufe der Jahre weitere Programmiersprachen. Dazu gehören COBOL für kaufmännische Anwendungen, BASIC als einfache Einsteigersprache für Hobbyanwendungen und PASCAL als Programmiersprache der Informatik. Und dann gibt es noch 100 weitere Programmiersprachen, von denen jeder Anwender behauptet, sie sei die beste. Und welche ist nun wirklich die beste? Lassen Sie mich dazu die Computersprachen mit den Menschensprachen vergleichen. Italienisch für Opernarien, Französisch für Liebesschwüre, Deutsch für philosophischen Tiefgang und Englisch für Big Business. Und wofür nun FORTRAN?

FORTRAN als dienstälteste anwendungsorientierte Programmiersprache wird heute vorwiegend für die Programmierung von Problemen der Wissenschaft und Technik verwendet. Sie ist leicht erlernbar und entspricht damit der Arbeitsweise des Ingenieurs, für den eine Programmiersprache kein Selbstzweck, sondern ein Werkzeug ist, mit dem er seine Probleme lösen will. Gerade im Bereich der Technik existieren umfangreiche Programmbibliotheken und Programmsammlungen, in denen der FORTRAN-Programmierer fertige Lösungen oder Anregungen für eigene Programme findet. Die neuen Kontrollstrukturen (IF ... ELSE) des FORTRAN 77 ermöglichen es, die modernen Methoden der "Strukturierten Programmierung" auch in FORTRAN anzuwenden.

2 Grundlagen des FORTRAN

Dieser Abschnitt führt Sie in die Grundlagen der Programmierung und der FORTRAN-Sprache ein. Am Ende können Sie Ihre ersten einfachen Programme schreiben und im Dialog am Bildschirm testen.

2.1 Allgemeine Eingaberegeln

In FORTRAN können Sie folgende **Zeichen** verwenden:

Ziffern	0 bis 9 für Zahlen
Große Buchstaben	A bis Z für Namen und Texte
Kleine Buchstaben	a bis z für Namen und Texte
Sonderzeichen	* Stern als Multiplikationszeichen
	. Punkt als Dezimalpunkt
	, Komma als Trennzeichen
	/ Schrägstrich als Divisionszeichen
	+ Plus als Vorzeichen und Rechenzeichen
	- Minus als Vorzeichen und Rechenzeichen
	' Apostroph als Begrenzung von Texten
	$ Dollarzeichen
	() runde Klammern
	Leerzeichen (blank)

In Texten sind alle Sonderzeichen der verwendeten Rechenanlage außer dem Apostroph, dem Begrenzungszeichen, erlaubt. Kleine Buchstaben werden, außer bei Texten, in große Buchstaben umgewandelt.

FORTRAN kennt fest vereinbarte **Namen** oder Kennwörter wie z.B. READ für das Lesen von Daten oder WRITE für die Datenausgabe. Andere Namen wie z.B. die Namen von Speicherstellen vergeben Sie selbst. Für diese frei wählbaren Namen gelten allgemein folgende Regeln:

Sie dürfen aus höchstens sechs Buchstaben oder Ziffern bestehen,
das erste Zeichen muß ein Buchstabe sein,
Sonderzeichen als Bestandteil des Namens sind verboten und
Kennwörter sollten nicht als frei wählbare Namen verwendet werden.

Alle **Daten** werden im Rechner binär gespeichert. Man unterscheidet ganze Zahlen (INTEGER), reelle Zahlen (REAL), doppelt genaue Zahlen (DOUBLE PRECISION), komplexe Zahlen (COMPLEX), logische Größen (LOGICAL) und Textzeichen (CHARACTER). Für veränderliche Daten (Variablen) vergeben Sie einen symbolischen Namen, unter dem diese Größe im Programm angesprochen werden kann. Beispiel:

```
READ(*,*) X1, X2, X3, X4, X5
```

Die fünf einzulesenden Zahlen werden unter den frei gewählten symbolischen Namen X1, X2, X3, X4 und X5 im Arbeitsspeicher des Rechners abgelegt. Für diese Variablen geben Sie nach dem Start des Programms dezimale Zahlenwerte ein; Hilfsprogramme des Betriebssystems rechnen sie in das duale Zahlensystem um und legen sie im Speicher ab.

Daten, die bereits bei der Programmierung bekannt sind, heißen Konstanten. Sie erscheinen z.B. als Dezimalzahlen in Formeln. Beispiel:

```
XM = (X1 + X2 + X3 + X4 + X5) / 5.
```

Die Zahl 5 ist eine Konstante. Sie erscheint als Dezimalzahl in einer Formel.

Die **Eingaberegeln** des FORTRAN stammen noch aus der Lochkartenzeit. An die Stelle der 80spaltigen Lochkarte tritt heute die Bildschirmzeile. Die in **Bild 2-1** an einem Beispiel dargestellte Spalteneinteilung wird bei der Eingabe durch den Editor des Betriebssystems unterstützt. Mit Hilfe der Tabulatortaste kann der Cursor (Schreibmarke) des Bildschirmgerätes auf bestimmte Spalten gesetzt werden.

Fachhochschule der DBP Dieburg — FORTRAN – Programmschema

Programm: Bild 2-1 FORTRAN-Eingabeformular — Ablochungshinweise: Zeichen / Lochung — Blatt 1 von 1

Programmierer: Schmitt — Datum: 11.6.85

C ← Bemerkung	Nr.	folgt	Anweisungen bzw. Vereinbarungen	Kennung
C			BEISPIEL FÜR EINGABEFORMULAR	
	10		WRITE(*,*) 'FÜNF WERTE EINGEBEN'	
			READ(*,*) X1,X2,X3,X4,X5	
			XM = (X1+X2+X3+X4+X5)/5.	
			WRITE(*,*) 'MITTEL=',XM	
			IF(XM.EQ.0) STOP	
			GOTO 10	
			END	

Bild 2-1: FORTRAN-Eingabeformular

Kommentare werden durch den Buchstaben "C" oder das Zeichen "*" in Spalte 1 gekennzeichnet. Der nachfolgende Text wird vom Compiler als Anmerkung aufgefaßt und nicht in Maschinenbefehle übersetzt.

Marken sind ganze Zahlen von 1 bis 99999 und stehen in den Spalten 1 bis 5. Sie bezeichnen Sprungziele oder FORMAT-Beschreibungen. Sie werden vom Programmierer vergeben und sind **nicht** wie in BASIC identisch mit den Zeilennummern des Editors! Eine Marke darf nur einmal im Programm erscheinen.

FORTRAN-Anweisungen stehen in den Spalten 7 bis 72. Durch das Einfügen von Leerzeichen, außer in Texten, lassen sich Programmzeilen besser lesen. Die Anweisung muß nicht in Spalte 7 beginnen; Strukturblöcke lassen sich durch Einrücken besser überblicken. Jede FORTRAN-Anweisung steht auf einer neuen Zeile. Reicht eine Zeile nicht aus, so kann die Anweisung auf der nächsten Zeile fortgesetzt werden.

Fortsetzungszeilen erhalten ein beliebiges Zeichen außer "0" oder Leerzeichen in Spalte 6 der Eingabezeile. Es sind bis zu 19 Fortsetzungszeilen möglich. Kommentarzeilen lassen sich nicht fortsetzen; jede Kommentarzeile muß mit einem "C" oder "*" in Spalte 1 beginnen.

In den Spalten 73 bis 80 der Eingabezeile können beliebige Zeichen oder Zahlen stehen, die vom Compiler nicht beachtet werden. Bei langen FORTRAN-Anweisungen besteht die Gefahr, daß man in die Spalten 73 bis 80 gerät und daß der Compiler diesen Teil der Eingabe nicht beachtet.

2.2 Die Darstellung ganzer Zahlen durch INTEGER-Größen

Das Kind beginnt sein mathematisches Leben mit ganzen Zahlen: 1 Daumen, 2 Bonbons, 3 Finger usw.; erst später lernt es das Nichts, die Null, und die Schulden, die negativen Zahlen, kennen. Ganze positive und negative Zahlen einschließlich der Null werden in FORTRAN als INTEGER-Größen bezeichnet. An vielen Stellen müssen Sie INTEGER-Größen verwenden:

Einheit der Ein/Ausgabeanweisungen READ und WRITE,
Marken für Sprungziele und FORMAT-Beschreibungen,
Exponenten von reellen Zahlen und
bei älteren Compilern Laufgrößen von DO-Schleifen und Indizes von Feldern.

INTEGER-Zahlen werden im Rechner als Dualzahlen mit Vorzeichen gespeichert; sie werden jedoch dezimal ein- und ausgegeben. Der zulässige Zahlenbereich hängt von der verwendeten Maschine ab und kann dem Handbuch des Herstellers entnommen werden. Reelle Zahlen (REAL-Größen) können Stellen hinter dem Dezimalkomma enthalten und werden im Rechner daher dual in der Mantisse-Exponent-Darstellung abgespeichert. Da der Compiler für INTEGER- und für REAL-Zahlen unterschiedliche Rechenbefehle erzeugen muß, ist es nötig, diese beiden Zahlentypen im Programm voneinander zu unterscheiden.

INTEGER-Konstanten sind ganze Zahlen ohne Dezimalpunkt, dabei kann ein positives Vorzeichen entfallen. Beispiele für INTEGER-Konstanten:

```
-100
4711
0
-2147483647   (kleinste Zahl bei 4 Bytes Speichergröße)
+2147483647   (größte Zahl bei 4 Bytes Speichergröße)
```

Einige Rechner speichern eine INTEGER-Zahl in 36 Bits ab und haben einen größeren INTEGER-Zahlenbereich! Für die Dateneingabe von INTEGER-Zahlen während des Programmlaufes gelten die gleichen Regeln.

Wenn eine **Variable** als ganze Zahl in der INTEGER-Zahlendarstellung verarbeitet werden soll, so müssen Sie entweder den Typ besonders vereinbaren oder Sie müssen den Anfangsbuchstaben des Namens entsprechend der Namensregel wählen.

Bei der expliziten (erklärten) Typvereinbarung folgt auf das Kennwort INTEGER

eine Liste von Variablennamen, die als INTEGER-Größen behandelt werden sollen.

```
INTEGER    Liste von Variablennamen
```

Beispiele:

```
INTEGER ZAHL,WERTE,NR
```

In der Indexschreibweise und bei einfachen Zählern verwendet man in der Mathematik sehr oft die Buchstaben i, j, k , l, m oder n. Daher gibt es in FORTRAN die **Namensregel** , die allen Variablen, die mit den Buchstaben

```
I , J , K , L , M  oder  N
```

beginnen, den Typ INTEGER zuweist; alle anderen Anfangsbuchstaben kennzeichnen REAL-Größen. Beispiele für die Namensregel:

INTEGER-Größen sind die Variablen IN , J , KO , LAENG , MONAT und N

REAL-Größen sind die Variablen X , Y , Z , ANZ , RX , X3 und OTTO

Vergibt man die Namen von Variablen entsprechend der Namensregel (implizite oder vorausgesetzte Typvereinbarung), so können besondere explizite Typvereinbarungen entfallen. Durch eine IMPLICIT-Vereinbarung läßt sich die Namensregel auf weitere Anfangsbuchstaben ausdehnen.

2.3 Die Darstellung reeller Zahlen durch REAL-Größen

Reelle Zahlen können Stellen hinter dem Dezimalkomma enthalten. Wie bei Taschenrechnern arbeitet man in FORTRAN mit einem Dezimalpunkt anstelle des Kommas. Ergeben sich bei wissenschaftlichen Rechnungen sehr große oder sehr kleine Zahlen, so schreibt man sie bequemer als Zehnerpotenz. Anstelle des Basis 10 steht in FORTRAN der Kennbuchstabe E für Exponent.

REAL-Konstanten schreibt man in FORTRAN immer als Dezimalzahl mit einem Dezimalpunkt. In der Exponentenschreibweise steht der Kennbuchstabe E vor dem ganzzahligen Exponenten zur Basis 10. Beispiele für REAL-Konstanten:

```
+13.5        für 13,5
13.5         für 13,5
-3.1415927   für - 3,1415927
1.3E+4       für 1,3·10⁴
```