4 Aufbau eines Programms

Ein Programm in Turbo Pascal ist nach genau festgelegten syntaktischen Regeln zu schreiben, es ist z. B. in bestimmter Weise in einzelne Teile zu untergliedern. Was bei der Gliederung des Programms zu beachten ist und was die einzelnen Teile enthalten, wird in den Abschnitten 4.1 bis 4.3 beschrieben. Dann wird der Anweisungsteil des Programms genauer betrachtet: In Abschnitt 4.4 werden einfache Anweisungen und ihre Wirkungen beschrieben. Daraus läßt sich dann schon ein erstes Programm erstellen.

Am Ende dieses Kapitels sind alle Voraussetzungen bereitgestellt, mit denen Sie eigene Programme schreiben und ablaufen lassen können.

4.1 Programmkopf

Ein Programm wird in Turbo Pascal – wie in anderen Programmiersprachen – zeilenweise geschrieben, die Zeile ist das Grundelement des geschriebenen Programmtextes. Die erste Zeile eines Programms enthält meist einen Namen für das Programm. Die Zeile mit dem Programmnamen wird Programmkopf genannt. Dem Programmkopf folgt der eigentliche Programmblock, der sich in einen Deklarationsteil und einen Anweisungsteil gliedert (Bild 4.1).

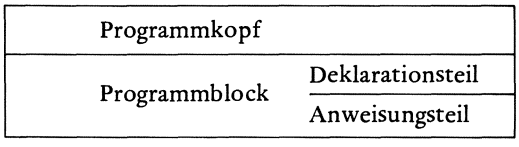

Bild 4.1 Ein Programm besteht aus einem Kopf und einem Block, der einen Deklarationsteil und einen Anweisungsteil enthält.

Die Kopfzeile eines Programms beginnt mit dem reservierten Wort Program, dem ein Name für das Programm folgt. Diesen Programmnamen kann der Benutzer selbst festlegen, er muß wie andere Bezeichner

(s. Abschnitt 3.5) mit einem Buchstaben beginnen und darf nur alphanumerische Zeichen enthalten (s. Abschnitt 3.1). Ein reserviertes Wort (s. Abschnitt 3.3) darf nicht als Name eingesetzt werden.

Man wird den Programmnamen so wählen, daß er einen Hinweis darauf liefert, wie das Programm einzusetzen ist und was es bewirkt. Die folgenden Beispiele zeigen, wie der Programmkopf aussehen kann:

```
Program Anfang;
Program Text_Verarbeitung;
Program Datei_Verwaltung;
```

Der Programmkopf wird mit einem Semikolon ; abgeschlossen.

Anmerkung: Turbo Pascal weicht von anderen Pascal-Versionen ab, indem es nicht zwingend einen Programmkopf fordert. Der Benutzer tut aber gut daran, eine Kopfzeile voranzusetzen, schon um ein Programm von Programmteilen wie Funktionen oder Prozeduren zu unterscheiden. Der im Kopf enthaltene Programmname gibt ihm auch Hinweise auf die Verwendungsmöglichkeiten und auf den Namen des Files, in dem das Programm auf der Diskette abgespeichert ist. Schließlich lassen sich dem Programmnamen in Klammern Parameter anfügen, die z. B. für den Datenaustausch mit Peripheriegeräten von Bedeutung sind. So wird im Beispiel

```
Program Druck (Input, Printer);
```

ein Ausgabefile festgelegt.

4.2 Deklarationsteil

Die Anweisungen eines Programms sollen die Verarbeitung der eingegebenen Daten steuern, so daß man die geforderten Ausgabedaten erhält. Damit der Rechner die Anweisungen interpretieren kann, muß ihm der Programmersteller sagen, welche Objekte er verarbeiten soll und von welchem Typ die Objekte sind. Das ist z. B. für die Bereitstellung von Speicherplatz und für eine richtige Interpretation der Operationszeichen von Bedeutung. Das letztere soll am Beispiel des Operators + verdeutlicht werden. Sind die Operanden Zahlen, dann müssen sie addiert werden. Handelt es sich um Zeichenketten, dann sind sie miteinander zu verketten.

Bei Zeichenketten muß man außer der Angabe des Typs auch die maximale Anzahl der darin enthaltenen Zeichen angeben, um den

Speicherplatz möglichst gut dem Bedarf anzupassen. Für jedes Zeichen ist genau 1 Byte = 8 Bit im Speicher erforderlich (s. Abschnitt 3.2), so daß die Länge der Zeichenkette den Speicherbedarf in Byte angibt.

Im Deklarationsteil sind alle Konstanten und Variablen, die im Programm vorkommen, mit Namen und Typ anzugeben, damit man im Anweisungsteil auf sie zugreifen kann. Will man für eine Variable einen anderen Typ verwenden, als standardmäßig verfügbar ist (s. Kapitel 5), dann kann man vorher eigene Typen einführen.

Werden in einem Programm Marken (Labels) gesetzt, mit denen man Sprünge bei der Bearbeitung steuern kann, dann sind diese ebenfalls zu deklarieren.

Schließlich werden im Deklarationsteil eines Programms die Bausteine eingeführt, die man als Teilprogramme im Anwendungsteil einsetzen will, das sind Funktionen und Prozeduren. Über ihren Einsatz und über ihre Deklaration wird in Abschnitt 4.4 etwas gesagt und dann in den Abschnitten 6.4 und 6.5 genauer informiert.

Damit ergibt sich die folgende Unterteilung für den Deklarationsteil eines Programms:

1. Labeldeklaration
2. Konstantendefinition
3. Typendefinition
4. Variablendeklaration
5. Deklaration von Funktionen und Prozeduren

Die genannten Definitionen und Deklarationen können in Turbo Pascal (abweichend vom Standard Pascal) in beliebiger Reihenfolge und auch mehrfach im Deklarationsteil vorkommen. Sie können auch fehlen, und im Sonderfall kann der Deklarationsteil leer sein.

Welche Regeln beim Definieren und beim Deklarieren zu beachten sind, soll am Beispiel der Konstanten, Variablen und Labels gezeigt werden.

4.2.1 Konstantendefinition

Die Definition von Konstanten wird mit dem dafür reservierten Wort Const eingeleitet. Jeder Konstanten, die Sie im Programm verwenden wollen, müssen Sie einen Namen geben (s. Abschnitt 3.5) und (mit dem Gleichheitszeichen =) einen Wert zuweisen. Der Wert kann eine Zahl, eine Zeichenkette oder ein Wahrheitswert sein.

Die folgenden Beispiele zeigen, wie Konstanten definiert werden:

```
Const  MaxByte=255;
       MWS=0.14;
       Sprache='Turbo Pascal';
       LetzterBuchstabe='Z';
       OK=True;
```

Anmerkungen zur Schreibweise der Werte:

1. Zahlenkonstanten werden wie in der Mathematik geschrieben, statt des Kommas steht der Dezimalpunkt.
2. Textkonstanten werden in Hochkommata ' eingeschlossen. So kann man die Zeichenkette '123' von der Zahl 123 oder die Konstante 'Wert' vom Bezeichner Wert unterscheiden.
3. Als Wahrheitswerte kann man die Konstanten True und False verwenden. Man braucht sie nicht in Hochkommata zu setzen, da sie als Standardkonstanten verfügbar sind.
4. Die Zuweisung des Wertes zum Bezeichner erfolgt mit dem Gleichheitszeichen =, die einzelnen Definitonen werden mit einem Semikolon ; abgeschlossen.

In Turbo Pascal sind die folgenden Standardkonstanten verfügbar, auf sie kann man ohne vorherige Definition zurückgreifen:

```
PI=3.1415926536;
MaxInt=32767;
False= (Wahrheitswert falsch)
True= (Wahrheitswert wahr)
```

Das Beispiel PI gibt eine Begründung für die Definition von Konstanten: Man kann beim Schreiben von Programmen Platz und Zeit sparen, wenn man einen kurzen Konstantennamen statt eines längeren einsetzen kann. Daher wird man auf eine Definition LetzterBuchstabe='Z' verzichten, wenn nicht andere Gründe vorliegen.

4.2.2 Variablendeklaration

Der Wert einer Konstanten bleibt im gesamten Programm unverändert, dagegen kann man einer Variablen nacheinander verschiedene Werte zuweisen. Eine Variable wird deklariert, indem man einen Namen festlegt und (nach einem Doppelpunkt :) angibt, von welchem Typ die Werte sein müssen, die man der Variablen zuweisen kann. Die Variablen-

deklarationen leitet man mit dem dafür reservierten Wort Var ein und schließt jeweils mit einem Semikolon ; ab.

Beispiele für Variablendeklarationen:

```
Var  Zahl:Real;
     I,J,K: Integer;
     Zeile: String(.80.);
```

Anmerkungen zur Schreibweise:

1. Die Zuordnung des Typs geschieht mit einem Doppelpunkt : zwischen Variablenname und Typname.
2. Man kann wie im zweiten Beispiele mehrere Variablen gleichen Typs zusammengefaßt deklarieren: Man führt die Variablennamen durch Kommata getrennt auf und gibt dann den gemeinsamen Typ an.

In den Beispielen kommen als Zahlentypen Real (Kommazahl) und Integer (Ganzzahl) vor. Sie sind wie String (Zeichenkette) Standardtypen in Turbo Pascal, brauchen also nicht vorher definiert zu werden. Die Standardtypen werden im Kapitel 5 genauer untersucht. Beim Typ String ist die maximale Anzahl der Zeichen anzugeben, sie wird in den Doppelzeichen (. und .) oder in eckigen Klammern angefügt.

4.2.3 Labeldeklaration

Vor jede Anweisung des Programms kann ein Label gesetzt werden, das als Adresse für einen Sprung (mit Goto) dient. Ein Label besteht aus einem Labelnamen, der in Turbo Pascal ein Bezeichner (s. Abschnitt 3.5) oder eine Zahl sein darf, und wird mit einem Doppelpunkt abgeschlossen. Die Deklaration von Labels wird durch das reservierte Wort Label eingeleitet.

Beispiel für die Deklaration mehrerer Labels:

```
Label NeuAnfang, 18, Exit;
```

Wie man Labels einsetzt, wird in Abschnitt 4.4.3 beschrieben.

4.3 Anweisungsteil

Im Anweisungsteil eines Programms dürfen nur solche Bezeichner vorkommen, die entweder standardmäßig verfügbar sind oder die im Deklarationsteil eingeführt wurden. Der Compiler weist bei der Überprüfung

des Programms auf syntaktische Fehler alle ihm unbekannten Bezeichner zurück.

Im Anweisungsteil des Programms sind die Anweisungen zusammengefaßt, mit denen alle Vorgänge wie die Eingabe und die Ausgabe von Daten oder die Verarbeitungsschritte gesteuert werden. Welche Regeln beim Schreiben der Anweisungen zu beachten sind, wird im Abschnitt 4.4 an einfachen Anweisungen eingeführt. Hier sei nur angemerkt, daß der Anweisungsteil die Form einer Verbundanweisung (s. Abschnitt 6.1) hat, er wird mit dem reservierten Wort Begin eingeleitet und mit dem Wort End abgeschlossen. Der dem Wort End folgende Punkt schließt den Programmblock ab:

```
Begin
    Anweisungen
End.
```

Der Anweisungsteil wird in Abschnitt 4.4 mit einfachen Anweisungen ausgefüllt, so daß ein lauffähiges Programm entsteht.

4.4 Einfache Anweisungen

Dieser Abschnitt soll zeigen, wie man einfache Anweisungen wie eine Wertzuweisung, einen Prozeduraufruf (z.B. zur Eingabe oder zur Ausgabe von Daten) oder einen Sprung in Turbo Pascal formuliert. Damit können Sie dann schon kleine Programme schreiben.

Zur Definition einer Konstanten führt man einen Namen ein und weist einen Wert zu:

```
Const  MWS = 0.14;
```

Mit dem Wert ist der Konstanten zugleich auch ein Typ zugeordnet worden, er ergibt sich aus dem Wert. So erhält die Konstante MWS mit dem Wert 0.14 den Typ Real (Kommazahl).

Anmerkung: Über typisierte Konstanten informiert der Abschnitt 9.2.

Bei der Deklaration einer Variablen werden nur ein Name und ein Typ festgelegt, aber noch kein Wert zugewiesen. Der Typ gibt lediglich an, aus welcher Menge die Werte zu nehmen sind, die der Variablen zugewiesen werden können. Der Programmierer muß dafür sorgen, daß einer Variablen vor ihrer ersten Verwendung ein Anfangswert zugewiesen wird, er muß sie initialisieren. Der Wert der Variablen kann dann im Laufe der Programmbearbeitung durch andere ersetzt werden. Greift

die Bearbeitung auf die Variable zu, d. h. kommt in einer Anweisung der Bezeichner der Variablen vor, dann wird jeweils der aktuelle Wert verarbeitet.

In Turbo Pascal gibt es unterschiedliche Möglichkeiten, einer Variablen einen Wert zuzuweisen. Man kann den Wert von außen zuweisen, das geschieht mit einer Eingabeanweisung. Der Wert läßt sich aber auch intern zuweisen, das leistet eine Anweisung, die man Wertzuweisung nennt. Diese soll zunächst untersucht werden.

4.4.1 Wertzuweisung

Die interne Wertzuweisung wird in Turbo Pascal mit dem Doppelzeichen := geschrieben, man könnte es als nach links gerichteten Pfeil interpretieren und aussprechen als „erhält zugewiesen". Links vom Zuweisungszeichen := schreibt man den Namen der Variablen, die den Wert erhalten soll.

Die Anweisung

```
Preis := 19.90;
```

weist der Variablen Preis, für die der Typ Real vereinbart sein muß, den Wert 19.90 zu.

Die Zuweisung eines konstanten Wertes ist der einfachste Fall. Meist wird auf der rechten Seite des Zuweisungszeichens ein Term (s. Kapitel 5) stehen.

Beispiele für die Zuweisung von Termen:

```
Zahl  := Zahl + 1;
```
Der bisherige Wert von Zahl wird um 1 erhöht.

```
Preis := Netto + Netto * MWS;
```
Zum Nettopreis wird der Mehrwertsteuerbetrag addiert, damit ergibt sich der Wert von Preis.

```
Zeile := Sprache + ' ist leicht zu erlernen';
```
Wenn Sprache den Wert 'Turbo Pascal' hat, dann erhält Zeile den Wert 'Turbo Pascal ist leicht zu erlernen'. Zeile und Sprache sind vom Typ String deklariert.

Der rechts stehende Term wird zunächst ausgewertet, sein Wert dann der Variablen zugewiesen, deren Name links steht.

Bei der Formulierung einer Wertzuweisung ist darauf zu achten, daß die Variable den Typ hat, der sich bei der Termauswertung ergibt. Bei der Überprüfung eines Programms auf syntaktische Korrektheit meldet der Compiler dem Benutzer einen Fehler, wenn der Typ nicht übereinstimmt.

4.4.2 Prozeduraufruf

Eine Prozedur ist ein Unter- oder Teilprogramm in einem anderen Programm. Eine Prozedur kann von verschiedenen Stellen her mit ihrem Namen aufgerufen werden und erledigt dann eine Folge von Bearbeitungsschritten. Turbo Pascal stellt eine Reihe von Standardprozeduren zur Verfügung. Wie der Benutzer weitere Prozeduren definieren kann und was beim Übergeben von Parametern zu beachten ist, wird in Abschnitt 6.4 genauer untersucht.

Hier sollen die Prozeduren zur Eingabe und zur Ausgabe von Daten sowie einige nützliche Standardprozeduren angegeben werden.

4.4.2.1 Eingabeanweisung

Einer Variablen kann ein Wert auch von außen, d. h. von einem Eingabegerät wie Tastatur, Diskettenlaufwerk oder Markierungskartenleser, zugewiesen werden. Das geschieht in Turbo Pascal durch den Aufruf der Read-Prozedur. Die Eingabe eines Wertes für die Variable Name vom Typ String wird mit der Anweisung

```
Read (Name);
```

veranlaßt.

Mit dem Aufruf der Eingabeprozedur Read wird zunächst die Eingabe einer Zeichenkette über die Tastatur in einen Zwischenspeicher, den Eingabepuffer, gesteuert. Dieser kann bis zu 127 Zeichen aufnehmen, das entspricht einer Zeile. Der Pufferinhalt erscheint auf dem Bildschirm und kann dort korrigiert werden. Mit dem Drücken der Returntaste wird der Inhalt des Eingabepuffers in den Arbeitsspeicher aufgenommen und der Variablen Name zugewiesen.

Bei der Deklaration einer Stringvariablen (s. Abschnitt 4.2.2) wird eine maximale Anzahl von Zeichen festgelegt. Hat der eingegebene Text weniger Zeichen, dann nimmt die Variable alle auf. Die aktuelle Anzahl der Zeichen wird mit abgespeichert (auf Platz 0), sie steht also für wei-

tere Verarbeitungsvorgänge wie die Ausgabe zur Verfügung. Hat der eingegebene Text mehr Zeichen als die Maximalzahl angibt, dann werden die überzähligen Zeichen nicht mit aufgenommen, sondern bleiben im Puffer.

Das ist anders bei einer Readln-Anweisung, Readln ist eine Verkürzung von Read Line. Mit dem Aufruf

```
Readln(Name)
```

wird das gleiche bewirkt wie mit dem Aufruf der Read-Prozedur, nämlich die Aufnahme eines Textes und die Wertzuweisung an die Variable Name, doch wird der Teil des Pufferinhalts, der über die deklarierte Maximalzahl von Zeichen im String Name hinausgeht, abgeschnitten und geht verloren.

Eine genauere Untersuchung der Prozeduren Read und Readln erfolgt in Abschnitt 8.4, dort wird der hier dargestellte Sonderfall der Tastatureingabe verallgemeinert auf die Eingabe von anderen Eingabegeräten aus.

Es bleibt anzumerken, daß mit der Read- oder der Readln-Prozedur auch Zahlenwerte oder Wahrheitswerte aufgenommen werden können. Mit einem einzigen Aufruf kann man auch die Werte für mehrere Variable eingeben. Das Beispiel

```
Readln(Zahl,X,Name);
```

zeigt, daß man die Variablennamen mit Kommata getrennt schreibt. Die im Aufruf genannten Variablen können auch von unterschiedlichem Typ sein. Ihre Werte werden über Tastatur nacheinander in der Reihenfolge der Nennung geschrieben und durch Leerzeichen voneinander getrennt.

Man kann die Prozeduren Read und Readln auch ohne die Angabe von Variablen aufrufen. Steht an einer Stelle des Programms der Aufruf

```
Readln;
```

dann wird an dieser Stelle der Bearbeitungsablauf unterbrochen, bis die Returntaste gedrückt wird.

4.4.2.2 Ausgabeanweisung

Die Werte von Variablen oder von Termen lassen sich mit der Write-Prozedur an ein Ausgabegerät wie Bildschirm, Drucker oder Diskettenlaufwerk geben.

Mit dem Aufruf

```
Write(Name);
```

wird der aktuelle Wert der Variablen Name auf dem Bildschirm ausgegeben. Wünscht man die Ausgabe über den Drucker, dann muß man seine Adresse Lst zusätzlich im Aufruf nennen:

```
Write(Lst,Name);
```

Die Ausgabe auf dem Bildschirm beginnt an der Stelle, an der der Cursor steht. Hat Name den Wert 'Turbo', dann werden die fünf Buchstaben ohne die Hochkommata geschrieben, der Cursor steht an der Stelle rechts vom o:

```
Turbo_
```

Man kann auch konstante Zahlen oder Zeichenketten an die Write-Prozedur übergeben, auch Terme aus Konstanten, Variablen und Funktionsaufrufen sind zugelassen. Vor der Ausgabe wird dann der Wert des im Aufruf genannten Terms ausgerechnet. Sollen mehrere Werte ausgegeben werden, kann man sie in einen Aufruf durch Kommata getrennt schreiben.
Beispiel:

```
Write(Name,'␣Pascal␣',Text);
```

Hat Text den Wert 'macht Spaß!', dann erscheint auf dem Bildschirm

```
Turbo Pascal macht Spaß!_
```

Das gleiche leistet die Ausgabeanweisung Writeln (von Write Line), doch steht der Cursor anschließend nicht hinter dem letzten Zeichen, sondern am Anfang der nächsten Zeile. Mit dem Aufruf

```
Writeln;
```

ohne Übergabe eines Wertes kann man eine Leerzeile schreiben.

Beim Schreiben der Ausgabeanweisung muß man beachten, daß auch Leerzeichen des Textes (in Hochkommata eingeschlossen) explizit als Textkonstanten geschrieben werden müssen. Hat T den Wert 'Turbo' und P den Wert 'Pascal', dann liefert die Anweisung

```
Writeln(T,' ',P);
```

die Ausgabe

```
Turbo Pascal
_
```

Die Ausgabe auf dem Bildschirm und mit dem Drucker läßt sich formatieren, d. h. auf einen gewünschten Platz in der Zeile bringen. Man gibt ihn in der Ausgabeanweisung nach dem Wert mit einem Doppelpunkt : an, wieviel Stellen die Ausgabe einnehmen soll. Mit

```
Write(T:15,P:7);
```

ergibt sich das Schirmbild

```
          Turbo Pascal_
```

Der Wert wird rechtsbündig in den verfügbaren Platz geschrieben.
Die Formatangabe ist besonders wichtig, wenn man Zahlen untereinander anordnen will. Bei Ganzzahlen wird wie bei Texten eine Stellenanzahl angegeben, bei Kommazahlen braucht man zwei Formatangaben: Die erste Zahl gibt an, wieviel Stellen insgesamt zum Schreiben frei sind, die zweite gibt die Anzahl der Nachkommastellen an. Hat Preis den Wert 19.90, dann liefert der Aufruf

```
Write ('Preis:',Preis:10:2,' DM');
```

das Schirmbild

```
Preis:      19.90 DM_
```

Die Formatierungsmöglichkeiten sollte der Benutzer ausschöpfen, um die Ausgabe auf dem Bildschirm ansprechend und übersichtlich zu gestalten.

4.4.2.3 Standardprozeduren

Die folgenden Standardprozeduren von Turbo Pascal lassen sich für die Bildschirmgestaltung und für den Programmablauf einsetzen:

4.4.2.3.1 ClrScr (Clear Screen)
Der Aufruf ClrScr löscht den Bildschirm und setzt den Cursor an den Anfangspunkt oben links.

4.4.2.3.2 ClrEoL (Clear End of Line)
Der Aufruf ClrEoL löscht alle Zeichen der Zeile rechts vom Cursor.

4.4.2.3.3 GotoXY
Der Aufruf GotoXY (XWert, YWert) erfordert zwei Parameter vom Typ Integer oder Byte und setzt den Cursor auf die Position, die mit XWert (Spaltennummer 1..80) und YWert (Zeilennummer 1..24) angegeben ist.

4.4.2.3.4 Delay
Der Aufruf Delay (Pause) erfordert einen Parameter vom Typ Integer oder Byte und bewirkt eine Pause im Programmablauf, deren Dauer (in ms) von Pause festgelegt ist.

4.4.3 Sprunganweisung Goto

Beim Ablauf eines Programms werden die Anweisungen sequentiell bearbeitet, wenn nicht eine Abweichung von der Reihenfolge einprogrammiert wird (s. Kapitel 6). Die Anweisung

```
Goto Labelwert;
```

bewirkt beim Programmablauf einen Sprung zum angegebenen Label, der vorher deklariert sein muß (s. Abschnitt 4.2.3).
Ein Sprung mit Goto zu einem Label ist nur innerhalb eines Programmblocks möglich. Mit dieser Anweisung kann man also nicht aus einer Prozedur heraus zu einem Label des aufrufenden Programms springen.

4.5 Kommentare

Mit den bisher eingeführten Anweisungen lassen sich schon einfache Programme schreiben. Auch wenn sie noch kurz und übersichtlich sind, sollte man die Programme schon durch Kommentare ergänzen, damit sie auch von anderen gelesen und verstanden werden können. Und der Verfasser eines Programms ist nach längerer Zeit ganz froh, wenn Kommentare ihn darauf hinweisen, was das Programm bewirkt und wie es aufgebaut ist.
Ein Kommentar wird durch Einschließen in geschweifte Klammern oder (wie durchgängig in diesem Buch) in Doppelzeichen aus runden Klammern und Stern kenntlich gemacht.

Beispiele für Kommentare:

```
Name:String(.24.); (* nimmt den Namen auf *)
ClrScr; (* löscht den Bildschirm *)
```

Hinweis: Man sollte nach (* und vor *) ein Leerzeichen setzen, damit der Kommentar nicht als Anweisung (Direktive) für Compiler oder Hilfsprogramme interpretiert wird. Solche Direktiven werden nämlich wie Kommentare in (* und *) eingeschlossen (s. Abschnitt 9.4).
Kommentare werden bei der Bearbeitung eines Programms einfach übersprungen, sie haben keine Wirkung auf die Abfolge beim Programmablauf. Man kann sie an beliebiger Stelle einsetzen; sie trennen auch Bezeichner und andere Sprachelemente. Wenn man hinreichend Kommentare einfügt, dann braucht man keine weiteren Erläuterungen zum Programm. Das zeigen die beiden Beispielprogramme in Abschnitt 4.6.

4.6 Starten des Programmablaufs

Mit den beschriebenen einfachen Anweisungen können Sie schon kleine eigene Programme schreiben. Wie Sie dabei vorgehen, ist schon in den Einzelheiten beschrieben worden und braucht nur noch einmal stichwortartig zusammengefaßt zu werden:

1. Betriebssystem CP/M starten (s. Abschnitt 1.1),
2. Sprachsystem mit TURBO aufrufen (s. Abschnitt 2.1),
3. Editor mit E aufrufen (s. Abschnitt 2.2.4).

Beispiel für ein Schirmbild nach Eingabe des Kommandos:

```
>E
 Work file name:_
```

Sie geben nun einen Filenamen für Ihr Programm ein und können dann im Editor zu schreiben beginnen (s. Abschnitt 2.3). Ist das Programm fertig geschrieben, verlassen Sie den Editor mit Ctrl-K Ctrl-D und gehen zurück in die Kommandoebene von Turbo Pascal. Nun können Sie mit dem Kommando C den Compiler aufrufen, der das Programm auf syntaktische Fehler hin untersucht und in Maschinencode übersetzt. Findet der Compiler einen Fehler, dann gibt er auf dem Bildschirm eine Fehlermeldung aus. Mit der Taste ESC gelangen Sie automatisch in den Editor, der Cursor steht an der fehlerhaften Stelle.

Ist das Programm schließlich frei von syntaktischen Fehlern, dann können Sie es mit dem Kommando R starten. Vorher sollten Sie es mit dem Kommando S auf der Diskette abspeichern.

Um Ihnen Anregungen für eigene Programme zu geben und um zu zeigen, wie man Programme durch eingefügte Kommentare besser lesbar machen kann, werden zwei Beispielprogramme abgedruckt. Sie wurden mit dem Hilfsprogramm TLIST gedruckt, die reservierten Wörter sind dabei unterstrichen worden.

Das folgende Programm Anfang soll zeigen, wie ein Programm aus Programmkopf, Deklarationsteil und Anweisungsteil aufgebaut ist. Der Anweisungsteil enthält nur Eingabe- und Ausgabeanweisungen.

```
Program Anfang;
(* Begrüßung des Benutzers im Sprachsystem *)
Var Name:String(.20.);
 (* Variable zur Aufnahme des Benutzernamens *)
Begin   (* Beginn des Anweisungsteils *)
ClrScr; (* Löscht den Bildschirm *)
Writeln('Willkommen in Turbo-Pascal !':40);
Writeln('--------------------------':40);
Writeln;  (* Liefert Leerzeile *)
Write('Bitte geben Sie Ihren Namen ein: ');
Readln(Name);
    (* PC nimmt den Namen auf *)
Writeln;  (* Leerzeile *)
Writeln('Sie werden sehen, ',Name,',');
Writeln('daß Turbo-Pascal leicht zu erlernen');
Writeln('und vielseitig anzuwenden ist.');
Writeln;   (* Leerzeile *)
Writeln('Und nun, ',Name,' , guten Start !!');
Read; (* Mit dem Drücken einer Taste endet die
         Bearbeitung des Programm, es meldet
         sich wieder das Sprachsystem mit  > *)
End. (* Abschluß des Programms *)
```

An diesem einfachen Beispiel läßt sich schon erkennen, daß eingefügte Kommentare die Lesbarkeit verbessern.

Das folgende Beispielprogramm läßt sich zur Codierung einsetzen. Es liefert für ein eingegebenes Zeichen die Ordnungszahl und umgekehrt. Damit können Sie z. B. die Codes der Steuerzeichen ermitteln, die Sie von der Tastatur aus eingeben. Diese Codes werden z. B. dann benötigt, wenn Sie den Cursor auf dem Bildschirm vom Programm aus steuern wollen. Die Programmbeispiele der folgenden Abschnitte enthalten die Steuercodes des Alphatronic PC. Durch welche Codes sie bei Ihrem PC zu ersetzen sind, können Sie mit dem folgenden Programm ermitteln.

```
Program Codierung_Decodierung;
(* im American Standard Code for Information Interchange *)
Var Ordnungszahl:Byte;
    Zeichen:Char;
Begin ClrScr; (* löscht den Bildschirm *)
  Writeln('Codierung von Zeichen');
  Writeln('----------------------');
  Writeln('A',Ord('A'):7);
  Writeln('Z',Ord('Z'):7);
  Writeln('a',Ord('a'):7);
  Write('Geben Sie selbst ein Zeichen ein: ');
  Readln(Zeichen);  (* Eingabe eines Zeichens *)
  Writeln(Zeichen,Ord(Zeichen):7);
  Write('Noch ein Zeichen: ');Readln(Zeichen);
  Writeln(Zeichen,Ord(Zeichen):7);
  Writeln;  (* Leerzeile *)
  Writeln('Decodierung der Ordnungszahl');
  Writeln('----------------------------');
  Writeln(' 67',Chr(67):5);
  Writeln('$4F',Chr($4F):5);
  Writeln(' 68',#68:5);
  Writeln('$45',#$45:5);
  Write('Geben Sie eine Ordnungszahl ein: ');
  Readln(Ordnungszahl);
  Writeln(Ordnungszahl:3,Chr(Ordnungszahl):5);
  Write('Noch eine Ordnungszahl: ');Readln(Ordnungszahl);
  Writeln(Ordnungszahl:3,Chr(Ordnungszahl):5);Writeln;
  Writeln('Drücken Sie eine beliebige Taste,');
  Writeln('dann können Sie mit R das Programm neu starten.');
  Read;  (* Druck einer Taste beendet und
            bewirkt Rückkehr ins Sprachsystem *)
End.  (* Programmende *)
```

Das Programm enthält die unterschiedlichen Schreibweisen, die in Kapitel 3 für Zahlen und für Zeichen eingeführt wurden. Die Wiederholung ist im Programm fest vorgegeben, sie kann nicht vom Benutzer gesteuert werden. Der Wunsch nach benutzergesteuerter Wiederholung legt die Einführung geeigneter Steuerstrukturen nahe. Mit der Wiederholungssteuerung, die in Abschnitt 6.3 eingeführt wird, können Sie das Programm ergänzen.

In den folgenden Abschnitten werden die Datentypen und die Steuerstrukturen von Turbo Pascal entwickelt, ohne daß jeweils vollständige Programme angegeben sind. Sie sollten die entwickelten Programmteile jeweils in ein lauffähiges Programm einbinden, um mit den Sprachelementen vertraut zu werden und um Sicherheit in der Syntax zu gewinnen.

Wie Ihre Programme aussehen können, sollen die beiden kleinen Beispielprogramme zeigen. Sie können sich auch an zwei umfangreicheren Programmen orientieren, die vollständig abgedruckt werden. Der Abschnitt 6.6 enthält ein Programm, mit dem sich einfache Texte bearbeiten lassen, und der Abschnitt 7.4.4 ein Programm zur Verwaltung einer Datenbank.