Programmieren in Turbo-Pascal 7.0[1]

von Simon Blank

1. Der Syntax von PASCAL

Die einfachsten Grundelemente einer Programmiersprache sind die erlaubten Zeichen (*Alphabet*), die vordefinierten Worte, sowie die damit zusammenhängenden Namensregelungen.

1.1 Im Programm erlaubte Zeichen

Das Grundalphabet der Programmiersprache PASCAL sind 7 Bit ASCII-Zeichen. Das bedeutet, daß Sonderzeichen, wie Umlaute oder grafische Zeichen beispielsweise, innerhalb von Funktionsnamen nicht zulässig sind:
0 1 2 3 4 5 6 7 8 9
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
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
< > = . ^ @ * + - / , : ; { } ( ) [ ] # $ & _ ’

Zu obiger Liste kommen noch die sogenannten *Whitespaces* hinzu. Das sind Leerzeichen, der Tabulatorschritt und der Zeilenumbruch. Die Einschränkung, daß nur die oben angegebenen Zeichen benutzt werden dürfen, gilt natürlich nur für Konstruktionen, die innerhalb der Programmiersprache eingesetzt werden.

In Zeichenketten können also alle beliebigen, darstellbaren Zeichen verwendet werden, also auch Sonderzeichen, Umlaute und grafische Zeichen! Bsp.: „write(’ äöü sind erlaubt’);“

1.2 Namensregeln

Um Variablen, Funktionen und Prozeduren zu beschreiben, vergibt man Namen. Die Gestaltung dieser Bezeichner unterliegen festen Regeln.

1.3 Regeln für Bezeichner

Natürlich beziehen sich diese Regeln nur auf Namen, die im Programm genutzt werden, jedoch nicht auf das, was Sie innerhalb von Zeichenketten eintragen.

- Ein Name beginnt mit einem Buchstaben oder einem Unterstrich.
- Im Namen sind Ziffern, Buchstaben und der Unterstrich erlaubt, nicht aber Leerzeichen.
- Namen können beliebig lang sein, es werden aber nur bis zu 62 Zeichen unterschieden.

1.4 Namenskonventionen

Die kürzesten Bezeichner bestehen somit aus einem einzigen Buchstaben, sind aber auch nicht übermäßig aussagekräftig. Das andere Extrem wären sehr lange Namen. Solche Programme würden ein Programm auch nicht lesbarer machen, zumal die wirkliche Unterscheidungsgrenze ja bei 62 Zeichen liegt.
Sie müssen also im Rahmen der gegebenen Möglichkeiten einen Kompromiß schließen zwischen guter Tippbarkeit und guter Lesbarkeit. So hat es sich eingebürgert, daß man kurze, aus Verben und Hauptwörter bestehende Namen benutzt, z.B.:

---

[1] Quelle: „Turbo-Pascal I und II“ von Dr. Lasch, Dr. Mißler-Behr und Dr. Paul für Siemens AG und „Das Einmaleins der Pascalprogrammierung“ von Ulrich Cuber und Heino Wenzel, ECON Verlag.

- diese_funktion_macht_was()
- DieseFunktion MachtWas()

2. Deklaration von Konstanten und Variablen

Eine Größe, der man während des Programmablaufs unterschiedliche Werte zuweisen kann, nennt man eine Variable. Für Variable muß der Programmierer

- ihren Namen, im PASCAL auch **Bezeichner** genannt, festlegen und
- ihren **Typ**, der angibt, wie etwas gespeichert bzw. beim Lesen interpretiert werden soll

angeben.

Werte, die während eines Programmablaufs stehts gleich bleiben, d.h. nur den vom Programmierer im Programm angegebenen Wert behalten, nennt man Konstante, bzw. **benannte Konstante**. Auch sie müssen durch Angabe von Namen und Wert im Deklarationsteil aufgeführt werden. Eine Tpyenangabe wird für Konstanten nicht verlangt, da der jeweilige Typ aus dem angegebenen Wert abgelesen wird. Konstanten können während des Programmablaufs nicht geändert werden, d.h., die Konstante a=40 kann nicht durch a:=100 einen neuen Wert bekommen.

Konstanten-Deklaration:

const *Konstantenname = Wert*;

z.B.: const e = 2.781828; drei = 3; Wort = 'Pascal';

Einige Konstanten sind schon vordefiniert, also dem Compiler bereits bekannt, zum Beispiel:

Pi = 3.14....	**Maxint** = 32767
true = Wahrheitswert: wahr	**false** = Wahrheitswert: falsch

Variablen-Deklaration:

var *Variablenname : Datentype*;

Standardisierte Datentypen:

- Ganze Zahlen (Festkommazahlen):

shortint	Länge: 01 Byte	Min: -128	Max: 127
byte	Länge: 01 Byte	Min: 0	Max: 256
integer	Länge: 02 Bytes	Min: -32768	Max: 32767
word	Länge: 02 Bytes	Min: 0	Max: 65536
longint	Länge: 04 Bytes	Min: -2147483648	Max: 2147483647

Vorsicht Grenzprobleme: Immer dann, wenn Sie Werte benutzten, die größer oder kleiner als die angegebenen Grenzwerte sind, kommt es mit **integer** zu Problemen. Handelt es sich um eine direkte Wertezuweisung, spielt der Compiler schon nicht mit, das heißt, Sie können einer Variable vom **integer**-Typ keinen Wert 50.000 zuweisen!
Problematischer wird es, wenn Sie im Rahmen von Berechnungen die Grenzen des **integer**-Typs verlassen. Der Compiler gibt hierbei keine Fehlermeldung aus. Wurde *vor* dem Compilieren unter „Option-Compiler" in der Rubrik „Laufzeitfehler" die Option „Überlauffehler" aktiviert, so bricht das Programm während der

Ausführung mit der Fehlermeldung „Fehler 215: Arithmetik-Überlauf“ ab[2]. Ist dies nicht der Fall, so wird das Programm zu einer Endlosschleife.
Beispiel: *„while i < 50000 do begin...end;“ mit „var i : integer;“*: wird i=32767 erreicht, „springt“ der Wert auf -32768. Die Programmschleife wird nun wieder bis 32767 durchlaufen usw.

- Gleitkommazahlen:

real	Länge:	06 Bytes	Min:	2,9E-39	Max:	1,7E38	Genauigkeit:	12 Stellen
single	Länge:	04 Bytes	Min:	1,5E-45	Max:	3,4E38	Genauigkeit:	08 Stellen
double	Länge:	08 Bytes	Min:	5,0E-324	Max:	1,7E308	Genauigkeit:	16 Stellen
extended	Länge:	10 Bytes	Min:	1,9E-4951	Max:	1,1E4932	Genauigkeit:	20 Stellen
comp	Länge:	08 Bytes	Min:	(-2E63)+1	Max:	(2E63)-1	Genauigkeit:	20 Stellen

Alle Gleitkommatypen außer dem **real**-Typ benötigen einen numerischen Coprozessor. Fehlt dieser, so muß er emuliert, d.h. imitiert werden. Um den Coprozessor bzw. dessen Emulation zu aktivieren, muß man im Benutztermenü „Option-Compiler“ aufrufen und dort in der Rubrik Gleitkommaberechnungen die Optionen 80x87-Code und Emulation durch anklicken anwählen. Mindestens eine der beiden Optionen ist Turbo Pascal 7.0 normalerweise aktiviert.

2.1 Wertezuweisungen

Die Zuweisungen von Werten auf Variable kann durch Einlesen von Werten (z.B.: readln(A)) oder durch den Zuweisungsoperator (z.B.: A:= 4.0; B:= B + 5.0; B=: A + B) erfolgen.
Bei Zuweisungen muß im Prinzip darauf geachtet werden, nur genau gleiche Datentypen einander zuzuweisen oder sonst mit einander zu verknüpfen. Es gibt aber eine Reihe von Abmilderungen dieser strengen Regel. So darf eine ganzzahlige Größe einer Gleitkommagröße zugewiesen werden, aber nicht umgekehrt. Ganzzahlige Größen dürfen anderen ganzzahligen Größen zugewiesen werden, Gleitkommazahlen anderen Typen von Gleitkommazahlen.
Doch Vorsicht: Es kommt zu **fehlerhaften Zuweisungen**, wenn Sie einer Variable einen Wert zuweisen, der **außerhalb** dessen **Definitionsbereich** liegt. Ordnet man beispielsweise einer **byte**-Variable A eine **word**-Variable mit B:= 50000 zu, so hat A den Wert 80 (statt 50000)! Der Compiler bemerkt den Fehler nicht. Das Programm wird jedoch bei einer sog. Bereichsüberschreitung mit der Fehlermeldung „Fehler 201: Bereichsüberschreitung“ abgebrochen, wenn vor dem Compilieren unter „Option-Compiler“ in der Rubrik „Laufzeitfehler“ die Option „Bereichsüberprüfung“ aktiviert ist.

2.2 Stringfunktionen

- Zeichen/Zeichenketten:

char:	einzelne Zeichen	Länge: 1 Byte
string:	Zeichenketten mit einer Länge zwischen 1 und 255	Länge: 1 Byte

Eine Erweiterung des **string**-Typs ermöglicht schon zu Beginn eine Angabe über die Größe des zur Verfügung stehenden Speicherplatzes. Allerdings gilt auch hier, daß die maximale Stringlänge für die PASCAL-Strings nur zwischen 1 und 255 liegen kann:

```
var     Name: string[30];   {maximale Länge der Eingabe: 30 Zeichen}
```

[2] Damit eventuell änderte Optionen über die aktuelle Sitzung hinaus Gültigkeit bewahren, müssen die Einstellungen unter „Option-Speichern TURBO.TP“ gespeichert werden. Optionen, die erst nach dem Compilieren des Programms verändert werden, werden bei der Programmausführung nicht berücksichtigt. Damit die Änderungen für das bereits compilierte Programm gültig werden, muß dieses mit „ALT-F9“ erneut compiliert werden.

Abgesehen von den Einschränkungen mit den 255 Zeichen und der Tatsache, daß string[3] und string[4] unterschiedliche Datentypen sind, können Sie mit Zeichenketten in PASCAL recht bequem arbeiten.
Letzteres gilt vor allem auch wegen der Vielzahl an vordefinierten Funktionen, deren Beschreibung Sie mit Hilfe finden können *(Routinen zur Verwendung von Strings (PASCAL-Stil)).* Einige wichtige Funktionen zeigt Ihnen die folgende Tabelle:

Funktion	**Einsatz**
Lenght(s)	Liefert die Länge eines Strings
Concat(s1, s2)	Liefert den verketteten String zurück
Copy(s,. p, l)	Liefert als Ergebnis l Zeichen aus s ab der Position p
Pos(s1, s2)	Liefert die Position, an der s1 in s2 zum ersten Mal vorkommt
Insert(s1, s2, p)	Fügt in s2 den String s1 ab der Position p ein
Delete(s, p, l)	Entfernt in s ab der Stelle p die Anzahl von l Zeichen
Str(v, s);	Verwandelt den numerischen Wert v in eine Zeichenkette
Val(s, v. p)	Verwandelt eine Zeichenkette s in einen numerischen Wert in v. In p steht, ob die Umwandlung erfolgreich war. Wenn das so ist, dann steht hier der Wert 0.

3. Sprünge

Mit Sprüngen ist es so eine Sache. Es gibt Fälle, da kann man nicht anders, um ein gutes und effektvolles Programm zu schreiben. Aber wenn man sie zu großzügig einsetzt, dokumentiert man lediglich, daß wenig Mühe auf den Programmentwurf verwandt wurde. Für Programme mit zu vielen Sprüngen hat sich der Name „Spaghetticode“ eingebürgert (vor allem der Gebrauch von „goto“ sollte vermieden werden!).
Seien Sie mit Sprüngen sparsam, dann werden Ihre Programme mit Sicherheit gut strukturiert sein. Und wenn Sie einen Sprung einführen, überlegen Sie lieber noch einmal, ob er nicht vermeidbar ist. Und wenn nicht, dann dokumentieren Sie ihn zumindest zu gut es nur geht.

Hier ist eine Liste von Sprungbefehlen:

Befehle	**Einsatz**
Break	Beendet eine for-, while- oder repeat-Anweisung
Continue	Setzt eine for-, while- oder repeat-Anweisung fort
Exit	Verläßt den momentanen Block (begin...end;) mit sofortiger Wirkung
Halt	Bricht die Ausführung des Programms ab und führt einen Rücksprung zur Betriebssystemebene durch (z.B. zu MS-DOS, WINDOWS...)
RunError	Beendet die Programmausführung

Syntax: if ... then break;...

4. Ausgabe von Daten auf dem Drucker

Um Daten auf dem Drucker auszugeben, wird die Unit „PRINTER“ benötigt. Diese definiert eine Textdatei namens lst und bindet sie an den Druckerport LPT1. Der Ausdruck erfolgt dann mit folgenden Befehlen:

- write(lst, Liste);
- writeln(lst, Liste);

Bis auf die Tatsache, daß nur druckbare Zeichen auf dem Drucker ausgegeben werden können, gelten für die Druckerausgabe die gleichen Regeln wie für die Bildschirmausgabe. Diese unterscheiden sich also nur durch „lst,“ von einander - läßt man „lst,“ weg, so erfolgt die Ausgabe also auf dem Bildschirm.