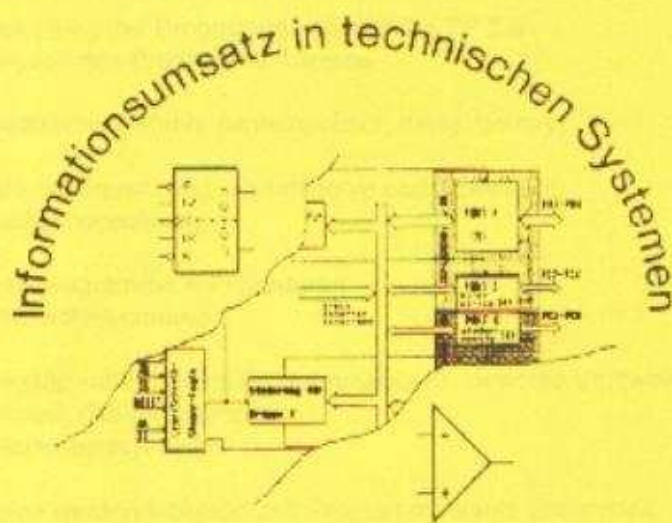


TECHNIK



Einführung in Turbo-Pascal 3

Arbeitsblätter zum Selbststudium

Peter Grimm

Gesamtschule Castrop-Rauxel

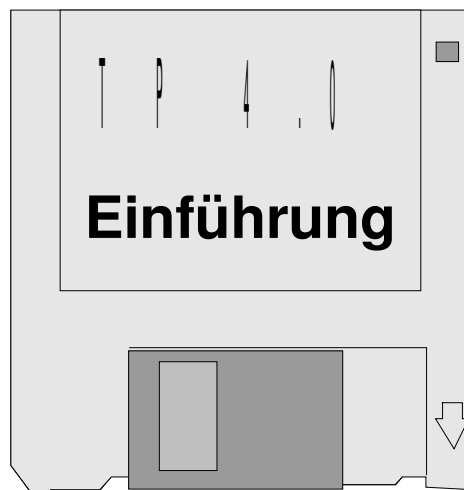
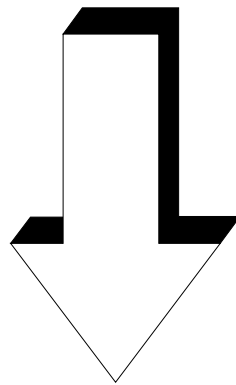
3/94

Herausgeber:

Technik-Unterricht: Forum e.V. (TUF)
Memelstraße 75 47057 Duisburg

TECHNIK

Turbo-Pascal 4.0



Auf den nachfolgenden Blättern wird der Versuch unternommen, die Programmieroberfläche von Turbo-Pascal 4.0 so vorzustellen, daß der Schüler, quasi im Selbststudium, die wesentlichen Schritte erarbeitet, um funktionsfähige Technikprogramme zu schreiben. Die vorliegenden Blätter können und sollen kein Buch ersetzen, bieten aber auf 26 kopierbaren Seiten eine m.E. brauchbare Zusammenfassung an. Wer sich bereits für Turbo 5.0 oder höher entschieden hat, kann sicherlich den größten Teil durch leichtes Modifizieren übernehmen.

Themenübersicht:

Blatt 1	Umgang mit dem Computer auf DOS-Ebene	
Blatt 2-5	Benutzung der Programmieroberfläche TP 4.0	<i>Eingabe des Programms Adresse</i>
Blatt 6-7	Zusätzliche Befehle (writeln, clrscr, delay, gotoxy)	
Blatt 7	Programmoptik	
Blatt 8	Variablenbenutzung, Variablentyp und Zuweisung	<i>Flächenberechnung 1</i>
Blatt 9	Unterprogramme = Prozeduren	<i>Flächenberechnung 2</i>
Blatt 10	Eingabe während des Programmablaufs, gewollte Verzweigung bzw. gewollter Abbruch des Programms	<i>Flächenberechnung 3</i>
Blatt 11	Daten werden während des Programmablaufs übermittelt	<i>Flächenberechnung 4</i>

Beginn der Programmierung technischer Objekte

Blatt 12	Zahlensysteme	<i>Dezimale Eingabe, duale Ausgabe</i>
Blatt 13	Menütechnik	<i>Beispiel eines Lauflichtprogramms</i>
Blatt 14	Include-Dateien	<i>Verbesserung Lauflichtprogramm</i>
Blatt 15	Erstellen einer Unit	<i>Verbesserung Lauflichtprogramm</i>
Blatt 16-17	Zusätzliche, nützliche Befehle	
Blatt 18	Projekte	

Anhang

Blatt 19-20	Turtle-Grafik (<i>alternativer Einstieg, spez. beim Prozedurkonzept</i>)
Blatt 21-22	Erweiterung der Technik-Unit
Blatt 23	Befehle der Technik-Unit
Blatt 24	Lesbarkeit von Programmen

DOS-Befehle

Benutzung des Computers, Handling

- | | | |
|-----|----------------------------------|---|
| 1.1 | Computer ohne Festplatte | Computer einschalten, Systemmeldung abwarten
Systemdiskette einlegen, der Computer meldet sich nach einiger Zeit mit A> bzw. A:\> |
| 1.2 | Computer mit Festplatte | Der Computer meldet sich nach dem Bootvorgang im allgemeinen mit C> bzw. C:\> |
| 2.1 | dir + <RET> eingeben | der Rechner gibt das Inhaltsverzeichnis des Laufwerks aus |
| 2.2 | dir /p + <RET> | die Bildschirmausgabe hält nach einer Seite autom. an |
| 2.3 | dir /w + <RET> | alle Dateien werden platzsparend nebeneinander ausgegeben |
| 3.1 | b: + <RET> | umschalten nach Laufwerk B> |
| 3.2 | c: + <RET> | umschalten nach Laufwerk C> usw. |
| 4.1 | cd Name | Wechselt in das Unterverzeichnis Name |
| 4.2 | cd .. | Wechselt in das nächst höhere Verzeichnis |
| 4.3 | cd \ | Wechselt in das Hauptverzeichnis |
| 5.1 | type Name.* | gibt Textzeichen der Datei auf dem Bildschirm aus
z.B. type Test.pas |
| 5.2 | type Name.* > pt1: | gibt Textzeichen der Datei auf dem Drucker aus |
| 6.1 | del Name.Endung | löscht Datei |
| 6.2 | del *.Endung | löscht alle Dateien mit gleicher Endung |
| 6.3 | del *.* | löscht alle Dateien im Verzeichnis |
| 7.1 | copy a:\name.* b:\ | kopiert Datei von Laufwerk a nach b |
| 7.2 | copy a:*.* b:\ | kopiert alle Dateien eines Verzeichnisses von Laufw. a nach b |
| 8. | name.com
name.exe
name.bat | sind ausführbare Dateien, der Name kann ohne Endung mit anschließendem <RET> eingegeben werden
Stapeldatei, die mehrere ausführbare Dateien enthalten kann |
| 9. | format b: | formatiert eine neue Diskette in Laufwerk b,
das Formatprogramm muß sich im Verzeichnis befinden |

Wichtig: Im allgemeinen können Befehlswörter in Groß- oder Kleinschrift eingegeben werden.

Der Computer ist ein Arbeitsgerät mit vielfältigen Fähigkeiten. Im Technikunterricht sollen mit seiner Hilfe Lampen, Motoren, Schrittmotoren, Relais und Roboter gesteuert, Ströme, Spannungen, Temperaturen gemessen werden.

Wie wird das gemacht? Im Prinzip ganz einfach, die einzelnen Arbeitsanweisungen, Arbeitsschritte müssen dem Computer nur verständlich (Programmiersprache) mitgeteilt werden. Der Computer (das Programm) arbeitet dann alle Anweisungen hintereinander ab.

Wir wollen jetzt nacheinander die Oberfläche und einige Befehlsörter der Programmiersprache Pascal kennenlernen.

Start und Handhabung der Programmieroberfläche von TP 4.0

- a) Systemdiskette mit Turbo 4.0 einlegen bzw. ins Turbo-Verzeichnis der Festplatte wechseln, siehe Befehle unter DOS
- b) turbo eingeben und <RET> Turbo-Pascal meldet sich mit einer Menue-, Informations- und Hotkey-Zeile

```

TP 4.0
File Edit Run Compile Options
Line 10 Col 1 Insert Indent C:ADRESSE.PAS

Program Adresse;
begin
  write ("Schulname");
  write ("PLZ Ort");
  write ("Straße Nr.");
end.

F1-Help F2-Save F3-Load F5-Zoom F6-Output F9-Make F10-Main me NUM

```

- c) Menü-Zeile (1. Zeile) Kann durch die Alt- bzw. F10-Taste angewählt, anschließend können die verschiedenen Optionen mit den Cursor- und der Ret-Taste aktiviert werden
- d) Informationszeile (2. Zeile) z.B. Spalten-, Zeilenzahl, Name der aktuellen Datei
- e) Hotkey-Zeile (letzte Zeile) direkt anwendbare Befehle z.B. F2 = Datei speichern, F1 bzw. ctrl + F1 = Hilfesystem
- f) In Laufwerk b (bzw. a) wird eine Datendiskette eingelegt Auf dieser Diskette werden alle Programme abgespeichert und von dieser wieder geladen.
- g) Umschalten auf die Datendiskette In die Menü-Zeile wechseln (siehe Pkt. c), anschließend unter File die Option change dir anwählen und die Zeichen b: bzw. a: eingeben

Programmeingabe

Unter einem Programm versteht man die Summe aller sinnvollen Anweisungen, die hintereinander abgearbeitet zu dem gewünschten Ziel (Erfolg) führen.

Ein Pascal Programm steht immer zwischen der Klammer **begin end**.

Wir wollen an dem folgenden Beispiel lernen, wie man ein Programm mit der Programmiersprache Turbo-Pascal erstellt und ausführt.

Program Adresse;

```
begin
write ('Schulname');
write ('Plz Ort');
write ('Straße Nr. ');
end.
```

Die Summe aller Anweisungen (Programm) steht immer innerhalb der Klammer begin - end.
write ('...'); schreibt den Inhalt der runden Klammern auf den Bildschirm, beliebiger Text muß in 'Hochkommastriche' eingebunden werden.

1. **Neues Programm**

Beim ersten Start von TP ist der Arbeitsschirm leer und in der Informationszeile steht noname.pas. Bei späteren Programmstarts befindet sich die zuletzt bearbeitete Datei im Editor. Soll eine Neue Datei angelegt werden, dann muß der Menüpunkt **File** und anschließend **New** angewählt werden.
2. **Editor**

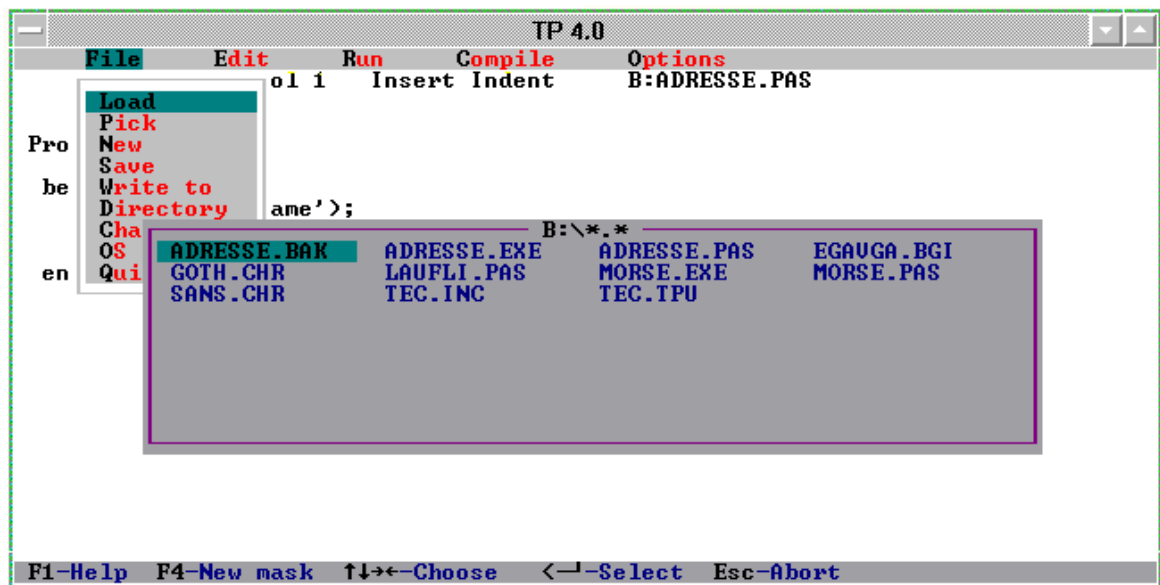
Der EDITOR ist das eigentliche Schreibprogramm, es kann über die Menüzeile angewählt werden. Mit Hilfe des Editors kann das Programm eingegeben werden. Tippfehler lassen sich leicht mit den Korrekturtasten (Pfeil, DEL) beheben. Mit Hilfe der Cursortasten kann jede Stelle des Textes erreicht werden.
3. **Alt + C** für compilieren

Der Computer kann bis jetzt noch nicht unterscheiden, ob es sich um einen beliebigen Text oder um ein Programm handelt. Der Compiler übersetzt den Text in einen Code, den der Rechner versteht. Ist die Übersetzung ohne Fehler, dann erscheint eine Compiliermitteilung und die Aufforderung, eine Taste zu drücken. Wir können weiterarbeiten. Sollte sich ein Fehler eingeschlichen haben, dann wird der Compilierungsvorgang automatisch unterbrochen, wir befinden uns wieder im EDITOR und der Cursor befindet sich im allgemeinen in der Nähe des Fehlers. Wenn der Fehler behoben ist, versuchen wir erneut zu compilieren. Dieses Vorgehen muß so lange wiederholt werden, bis keine Fehlermeldung mehr beim Übersetzen auftritt.
4. **Alt + R** für run

Endlich, das Programm kann ausgeführt werden. Der Rechner arbeitet die Befehle ab und führt sie nacheinander aus. Das Ergebnis ist meist (wie in unserem Beispiel) auf dem Bildschirm sichtbar. Nach der Programmausführung erscheint wieder eine Systemmeldung bzw. nach Tastendruck das turbo-Menue.

Programme sichern und laden

5. **sichern**
F2 bzw. write
unter dem Menüpunkt File
Durch Betätigung wird automatisch das Programm bzw. der Text unter dem Namen abgespeichert, der in der Informationszeile angezeigt wird. Dieser Befehl sollte in Abständen von 10 bis 15min benutzt werden, damit nicht durch unvorhersehbare Ereignisse die Tipparbeit von zwei Stunden verloren geht.
6. **sichern mit neuem Namen**
write to
unter dem Menüpunkt File
Hier muß ein Name für das Programm eingegeben werden. Unter diesem Namen wird das Programm später gesichert und wieder geladen. Der Programmname sollte einen Hinweis auf das Programm geben. Otto, Paul oder einfach nur a sind keine hilfreichen Namen. Leider dürfen die Namen nur 8 Buchstaben lang sein, also gut überlegen.
7. **laden**
F3 bzw. load
unter dem Menüpunkt File
Mit Hilfe der Cursortasten kann die zu wählende Datei ausgewählt und mit Return geladen werden.



In dem gezeigten Beispiel wurde vorab unter dem Menü File die Option Change dir angewählt und das neue Verzeichnis b: eingegeben. Da "normalerweise" nur *.pas Dateien angezeigt werden, mußte die Ladeoption *.pas in *.* umgeändert werden. Die Bedeutung der unterschiedlichen Endungen kann dir sicherlich dein Fachlehrer erläutern.

exe - Datei erstellen

Mit Hilfe des Turbo-Compilers lassen sich selbststartende Programme erstellen. Die Datei wird nicht mehr im RAM des Computers, sondern direkt auf der Diskette (Festplatte) als sogenannte exe-Datei (ProgrammEndung) abgelegt. Das so compilierte Programm läßt sich nun auch ohne die Programmierumgebung von Turbo-Pascal starten (siehe Pkt.8 DOS-Befehle Blatt 01).

- | | | |
|----|---------------------|---|
| 8. | Compilereinstellung | Unter dem Menüpunkt Compiler die Option Destination Memory anwählen und in Destination Disk ändern. Nach Anlegen der Exedatei sollte die Option wieder umgekehrt werden. |
| 9. | exe-Datei | Dies entspricht im Prinzip der Vorgehensweise wie unter Pkt. 3 beschrieben, allerdings mit dem Unterschied, daß die Datei nun mit der Endung .exe auf dem gewählten Laufwerk abgelegt wird. Um das Programm zu starten, muß die Oberfläche von Turbo verlassen werden. Soll mit Turbo-Pascal weitergearbeitet werden, dann muß Pkt. 8 erneut durchgeführt werden. Diesmal wird aber die Option Memory angewählt, anschließend kann wie gewohnt gearbeitet werden. |

Besondere Tastenkombinationen

- | | | |
|-----|---|--|
| 10. | ctrl + <y> | löscht eine ganze Zeile |
| 11. | Block markieren, kopieren, verschieben | |
| | markieren | |
| | ctrl + <K> + | an den Textanfang des zu kopierenden Blocks gehen und die Tastenkombination eingeben, dann mit der Cursor-Taste das Ende des Blocks markieren und dann |
| | ctrl + <K> + <K> | eingeben, der Block ist jetzt markiert und befindet sich im Zwischenspeicher |
| | kopieren | |
| | ctrl + <K> + <C> und
ctrl + <K> + <K> | mit dem Cursor an die Stelle gehen, bei der der zu kopierende Text eingesetzt werden soll, anschließend nacheinander die beiden Tastenkombinationen eingeben |
| | verschieben | |
| | ctrl + <K> + <V> und
ctrl + <K> + <K> | mit dem Cursor an die Stelle gehen, an die der Text verschoben werden soll, anschließend nacheinander die beiden Tastenkombinationen eingeben |

12. ctrl + <K> + <R>

read block from file:

Name einer Datei + <RET> eingeben.

Komfortable Methode, um alte Programme, Programmteile in ein bestehendes Programm zu kopieren

13. ctrl + <K> + <W>

write block to file:

Name einer Datei + <RET> eingeben.

Komfortable Methode, um Teile eines Programms, die vorher nach Pkt.11 markiert wurden, auf der Diskette abzuspeichern.

Aufgaben - Befehle

Wir wollen jetzt neue Befehle kennenlernen. Benutze hierzu das 1. Programm (Adresse) in der Form von Blatt 02 (Adresse der Schule). Das Programm kann wie unter Pkt.7 Blatt 03 geladen werden.

0. (* *)
{ }

Kommentartext, kein Semikolon, kann an jeder beliebigen Stelle ins Programm eingefügt werden, hat keinen Einfluß auf das Programm

Schreibe ab jetzt jeweils in die erste Zeile Deines Programms eine Kommentartextzeile, in der Dein Name und das Datum der Programmerstellung steht.

Achtung: Denke daran, daß jeder Befehl mit einem Semikolon abgeschlossen wird!

1. writeln ('....');

Ändere in dem Programm die Anweisung write in writeln.
Halte Deine Beobachtungen schriftlich fest!

Achtung: Aus programmtechnischen Gründen ist es sinnvoll, Befehle in Units (Sammlung von Befehlen) auszulagern. Will man diese Befehle benutzen, dann muß die entsprechende Unit (Befehlssammlung) angegeben werden. Eine der wichtigsten Units ist die Unit CRT. Hinter jedem neuen Befehl wird nun angegeben, ob hierfür eine Unit in das Programm eingebunden werden muß.

2. clrscr; {uses crt}

Wir wollen den Befehl clrscr kennenlernen.

a) füge clrscr hinter begin ein

b) füge clrscr vor end ein

Halte Deine Beobachtung schriftlich fest.

Das Programm nimmt nun folgende Gestalt an:

Program Adresse;

uses crt;

{beachte die Anordnung des Befehls für}
{zukünftige Programme}

begin

clrscr;
write ('Schulname');
write ('Plz Ort');
write ('Straße Nr.');

{erfordert die Unit crt}

end.

- 3. **delay (500);** {uses crt} Füge vor jedem writeln den Befehl delay ein, benutze dabei auch verschiedene Zahlenwerte. Was bewirkt der Befehl?
- 4. **gotoxy (20,11);** {uses crt} Mit diesem Befehl kann vorab bestimmt werden, an welcher Stelle des Bildschirms die Ausgabe mit write erfolgen soll. Waagrecht (x-Richtung) können in einer Zeile 80 Zeichen und senkrecht 25 Zeilen ausgegeben werden. *Prinzipiell könnte eine bestimmte Position des Bildschirms auch durch mehrere writeln; Ausgaben und durch Leerzeichen innerhalb der writeln-Anweisung z.B. writeln (' Schulname'); erfolgen.* Das 1. Programm (Adresse) soll nun so umgeschrieben werden, daß die Anschrift in der Mitte des Bildschirms erscheint. Wir benutzen hierzu den komfortablen Befehl gotoxy (x-Wert, y-Wert);

5. Programmoptik

Produkte (hierzu gehören auch Programme) können umso besser verkauft werden, wenn das äußere Erscheinungsbild dem Kunden (Käufer) gefällt. Programme sollten nicht nur funktionieren, sondern dem Benutzer möglichst viel Bedienungskomfort und ein ansprechendes Erscheinungsbild bieten. Wir wollen ab jetzt Programme dadurch aufwerten, daß die Bildschirmausgabe mit Hilfe von Rahmen optisch ansprechend gestaltet wird. Leider sind die unten angeführten nicht direkt durch die Tastatur erreichbar (wie z.B. 176). Da die Zeichen aber versteckt in der ASCII-Tabelle sind, können wir sie durch einen kleinen Trick 177 auf dem Bildschirm zaubern. Hierzu ist es erforderlich, die Funktion der Nummernblöcke zu verwenden. Bei 178 im Numerik-Block (z.B. bei Laptops) muß mit Hilfe der Funktionstaste auf den Numerik-Block (zusätzliche Zeichen) umgeschaltet werden.

```

201 ┌───┐ ┌───┐ ┌───┐ 187
    │   │ │   │ │   │
186 │   │ │   │ │   │
    │   │ │   │ │   │
204 └───┘ └───┘ └───┘ 185
    │   │ │   │ │   │
200 └───┘ └───┘ └───┘ 188
    │   │ │   │ │   │
    └───┘ └───┘ └───┘
    205
    203
    206
    202
    
```

```

218 ┌───┐ ┌───┐ ┌───┐ 196
    │   │ │   │ │   │ 194
    │   │ │   │ │   │ 197
195 └───┘ └───┘ └───┘ 180
    │   │ │   │ │   │
192 └───┘ └───┘ └───┘ 217
    │   │ │   │ │   │
    └───┘ └───┘ └───┘
    193
    
```

Vorgehensweise: ALT-Taste festhalten und nacheinander die Zahlen 6 und 5 (manchmal auch 065, ASCII-Code 65) eingeben, danach die ALT-Taste loslassen. Auf dem Bildschirm müßte das Zeichen A erscheinen. A ist das 65. Zeichen in der ASCII-Tabelle. B hat den Wert 66, C = 67 usw.. Wenn Ihr die in den Skizzen angeführten dreistelligen Zahlen eingibt, erhaltet ihr die abgebildeten Zeichen.

```

213 ┌───┐ ┌───┐ ┌───┐ 184 214 ┌───┐ ┌───┐ ┌───┐ 183 220
    │   │ │   │ │   │ 210 219 ┌───┐ ┌───┐ 219
    │   │ │   │ │   │ 215 221 ┌───┐ ┌───┐ 222
198 └───┘ └───┘ └───┘ 181 199 └───┘ └───┘ └───┘ 182
    │   │ │   │ │   │ 211 └───┘ └───┘ └───┘ 189 223
212 └───┘ └───┘ └───┘ 190 210
    │   │ │   │ │   │ 208
    └───┘ └───┘ └───┘
    205
    209
    216
    207
    
```

Aufgabe: Das unter 4. (Schuladressenausgabe mit gotoxy) beschriebene Programm soll mit einem Rahmen (freie Auswahl) versehen und bildschirmzentriert ausgegeben werden.

Wir wollen nun ein kleines Rechenprogramm schreiben

Das Programm soll die Fläche eines Rechtecks berechnen, hierzu ist es erforderlich, Länge und Breite festzulegen. Denkt dabei an die Formel zur Berechnung der Fläche, sie lautet:

$$\text{Fläche} = \text{Länge} * \text{Breite} \quad \text{oder kürzer}$$

$$A = l * b$$

l und b sind Variablen, sie können jeden beliebigen Wert annehmen, A ist ebenfalls eine Variable, sie speichert das Ergebnis der Berechnung von l * b. Im Fach Mathematik verwendet man diese Begriffe automatisch. Wollen wir diese Berechnung mit einem Pascal-Programm durchführen lassen, dann müssen wir einige Vorkehrungen treffen, damit dies im Programm reibungslos abläuft.

1. Schritt: Wir müssen dem Programm mitteilen, welche Variablen wir benutzen wollen
2. Schritt: Wir müssen mitteilen, welche Eigenschaften die Variablen besitzen, d.h. dürfen sie ganze Zahlen, Dezimalzahlen, Text, einzelne Buchstaben oder logische Aussagen enthalten

Beispiel:

```

Program Flaechenberechnung;      (* workfile: Rechteck.pas *)
uses crt;                        (* wegen clrscr erforderlich *)
var Flaechen, Laenge, Breite : integer; (* 1. und 2. Schritt beachtet, die Variablen dürfen *)
                                     (* nur ganze Zahlen 'integer' enthalten *)

begin
  clrscr;
  Laenge := 5;                    (* Laenge wird der Wert 5 zugewiesen *)
  Breite := 6;                    (* Breite wird der Wert 6 zugewiesen *)
  Flaechen := Laenge * Breite;    (* das Ergebnis aus Laenge*Breite wird der *)
                                     (* Variablen Flaechen zugewiesen *)

  write ('Fläche = ',Flaechen, ' qcm');
end.

```

- Aufgaben:
- a) Gib das Programm ohne Kommentartext ein und überprüfe das Ergebnis. Verändere die Werte von Laenge und Breite.
 - b) Lösche alle überflüssigen Leerzeichen im Programm und führe es aus. Das Programm müßte in etwa so aussehen:
Program Flaechenberechnung; uses crt; var Flaechen, Laenge, Breite : integer; begin clrscr; Laenge := 5; Breite := 6; Flaechen := Laenge * Breite; write ('Fläche = ',Flaechen, ' qcm'); **end.**
 Was hat sich an der Programmausführung geändert? Wie sollte man Deiner Meinung nach Programme schreiben?
 - c) Gib für Laenge den Wert 5.5 ein und überprüfe das Ergebnis.
 - d) Ändere innerhalb des Programms den Begriff integer in real und führe das Programm aus. Ändere anschließend die Werte von Laenge und Breite in 2.2 und 3.3 und überprüfe das Ergebnis.

- e) Ergänze das Programm durch die geforderte Kommentartextzeile mit Namen. Gib auf dem Bildschirm erläuternden Text aus, der die Funktion des Programms erläutert.
- f) real- und integer Größen können auch formatiert ausgegeben werden z.B. **write (Flaeche:6:2)**. Für real-Größen bedeutet dies rechtsbündige Ausgabe in ein 6 Stellen großes Feld mit 2 Nachkommastellen. Ergänze die Ausgabe in Deinem Programm durch eine entsprechende Formatausgabe. Gib den Text und die Ergebnisse in der Bildschirmmitte aus.
- g) Ergänze das Programm so, daß neben der Flächenausgabe auch der Umfang des Rechtecks ausgegeben wird.
- h) Schreibe ein neues Programm, das Flächeninhalt und Umfang eines rechtwinkligen Dreiecks ausgibt. Benutze die Variablen Grundlinie und Hoehe. Für die Umfangsberechnung wird ein Befehl zur Wurzelberechnung benötigt, siehe Pkt.5.
6. **sqrt (Zahl);** berechnet die Wurzel aus Zahl
- i) Schreibe ein Programm, daß Flächeninhalt und Umfang eines Kreises ausgibt. Für die Berechnung wird der Quadrat-Befehl benötigt, siehe Pkt. 6.
7. **sqr (Zahl);** berechnet die Quadratzahl von Zahl

Wir wollen jetzt die Programme für die Rechteck- und die Dreiecksberechnung in einem Programmablauf durchführen. Dazu bedienen wir uns der Unterprogrammtechnik.

8. **procedure Rechteck;** Unterprogramm mit dem Namen Rechteck. Im Prinzip wie ein Hauptprogramm, mit dem Unterschied, daß end ein Semikolon bekommt. Zusätzlich bekommt das Unterprogramm einen Namen (hier Rechteck). Eine Procedure muß immer vor einem Aufruf definiert werden.
begin
 (* Folge von Anweisungen *)
end;

Das Neue Programm sieht dann wie folgt aus:

```

Program Flaechenberechnung; (* Name des Programms *)
uses crt;
var Flaechen, Laenge, Breite : real; (* Definition der Variablen *)

Procedure Rechteck; (* Name des Unterprogramms *)
begin
  (* Folge von Anweisungen *)
end; (* wichtig, end; *)

begin (* Start des Hauptprogramms *)
  clrscr;
  Rechteck; (* Aufruf des Unterprogramms *)
end.

```

- j) Ergänze das Programm durch die Procedure Dreieck und rufe die Prozedur im Hauptprogramm auf. Versuche den Funktionsablauf des Programms Schritt für Schritt zu verstehen.

Die bisher geschriebenen Programme sind nach einem Durchlauf beendet. Das folgende Programm soll nun so ergänzt werden, daß es mehrere Berechnungen ermöglicht und definiert abgebrochen werden kann. Dazu sind weitere Befehle erforderlich:

- | | | |
|-----|--|--|
| 9. | var Taste : Char; | definiert eine Variable, die nur ein Zeichen des ASCII-Codes zulassen. |
| 10. | Taste := Readkey;
{uses crt} | Sobald der Befehl readkey im Programm auftaucht, stoppt die Programmausführung. Das Echo auf dem Bildschirm wird unterdrückt, Return ist zur Bestätigung nicht erforderlich, das Programm reagiert auf Tastendruck |
| 11. | if taste = '1' then
begin
(* Folge von Anweisungen *)
end; | Vergleichsbefehl, Durchführung wenn Vergleich stimmt, sonst nächster Befehl, bei einer Anweisung kann begin end fortfallen |
| 12. | repeat
(*Folge von Anweisungen *)
until Taste = 'q'; | Wiederhole Befehlsfolgen bis Taste q für quit gedrückt |

Das Programm zur Berechnung von Fläche und Umfang) könnte grob folgende Form annehmen:

Program Flaechenberechnung;

uses crt;

var Flaechen, Laenge, Breite : real;

Taste : char;

(* weitere Variablen falls erforderlich *)

Procedure Rechteck;

Procedure Dreieeck;

(* Auflistung der vollständigen Unter- *)

(* programme, die zur Berechnung *)

(* erforderlich sind *)

(* Hauptprogramm *)

begin

repeat

clrscr;

writeln ('Wollen Sie eine <R>echteck oder <D>reiecksberechnung durchführen?');

Taste := Readkey;

if taste = 'r' then Rechteck;

if taste = 'd' then Dreieeck;

until taste = 'q';

(* q für Quit *)

end.

- k) Übertrage das Programm, optimiere die Bildschirmausgabe, so daß beide Unterprogramme unabhängig auf dem Bildschirm angezeigt werden.
- l) Ergänze das Programm durch die Berechnung von Kreisfläche und Umfang

Noch komfortabler wäre das Programm, wenn die Werte für Länge, Breite usw. während des Programmablaufs eingegeben werden könnten. Dies gestattet der read-Befehl.

13. **read (Laenge);** Wird dieser Befehl in das Programm eingefügt, dann stoppt der Programmablauf, eine Eingabe über Tastatur wird erwartet, die mit Return abgeschlossen wird. Füge den Befehl in das Programm ein und überprüfe.(benutze auch **readln**)
- m) Füge den Befehl so oft wie erforderlich in die Prozeduren ein und überprüfe die Funktionsfähigkeit des Programms.

Zusatzaufgaben

- n) Ergänze Dein Berechnungsmenü durch die Volumenberechnung von Quader, Kegel und Kugel (siehe in der Formelsammlung nach).
 - o) Schreibe ein Programm, das die Zinsen von einem festangelegten Kapital berechnet (nach 1, 2, 3, n-Jahren).
 - p) Eine Celsius-Temperatur soll in Fahrenheit umgewandelt werden. Zum Vergleich der beiden Skalen sind Gefrier- und Siedepunkt des Wassers angegeben:
- | | Celsius | Fahrenheit |
|---------------|---------|------------|
| Gefrierpunkt: | 0 | 32 |
| Siedepunkt: | 100 | 212 |

Programmierung technischer Objekte

(z.B. Lauflicht, Ampel, Motor)

Zur Steuerung von Verbrauchern können mit Hilfe der Druckerschnittstelle 8-Leitungen aktiviert werden. Der wichtigste Befehl ist:

14. **Port [888] := 0..255;** Aktiviert eine Leitung der Druckerschnittstelle, die Leitungen sind im Dualzahlensystem codiert (siehe Blatt11). Eine 1 aktiviert die erste Leitung, eine 2 die 2. und eine 4 die 3. Leitung usw..
- q) Schreibe ein Programm, daß nacheinander eine Leuchtdiode am Ausgangsport aktiviert (Lauflicht).
 - r) Schreibe mind. 4 verschiedene Lauflichtprozeduren und bediene sie per Menü. (siehe Blatt 9)
 - s) Schreibe ein Programm für eine Ampelsteuerung.
 - t) Schreibe ein Programm für 2 Ampeln (Kreuzungsbetrieb)
 - u) Schreibe ein Programm für eine Anforderungsampel (Fußgänger)
 - v) Erstelle ein Programm zur Ansteuerung einer 7-Segmentanzeige

Die Programme können zur Not mit Ctrl + C bzw. Ctrl + Untbr abgebrochen werden. Die Ctrl-Taste hat auch oft die Bezeichnung Strg.

Zahlensysteme - Gegenüberstellung

Dezimal	Hexadezimal	Dual/Binär
0	0	0000 0000
1	1	0000 0001
2	2	0000 0010
3	3	0000 0011
4	4	0000 0100
5	5	0000 0101
6	6	0000 0110
7	7	0000 0111
8	8	0000 1000
9	9	0000 1001
10	A	0000 1010
11	B	0000 1011
12	C	0000 1100
13	D	0000 1101
14	E	0000 1110
15	F	0000 1111
16	10	0001 0000

Aufgabenbeispiele:

Ermittle die unbekanntten Zahlen.

dez	hex	bin
24		
42		
	24	
	42	
		0101 1100
		1010 1001

Bei den **Dezimalzahlen** hat jede Stelle einen um den Faktor 10 größeren Wert.

Beispiel: 37256 Die Zahl setzt sich aus folgenden Summanden zusammen

$$\begin{array}{r}
 3 \cdot 10000 + 7 \cdot 1000 + 2 \cdot 100 + 5 \cdot 10 + 6 \cdot 1 \\
 3 \cdot 10^4 + 7 \cdot 10^3 + 2 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0
 \end{array}$$

Die **Dualzahlen** haben die Wertigkeit zur 2er Potenz

$$\begin{array}{cccccccc}
 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1
 \end{array}
 \quad (8\text{-bit Zahl}),$$

die nächst höhere Stelle hätte die Wertigkeit 256 usw.

Beispiel: Welchen Dezimalwert hat die 8-bit Dualzahl 0010 0100

$$0 \cdot 128 + 0 \cdot 64 + 1 \cdot 32 + 0 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 = 36$$

Hexadezimalzahlen werden immer aus 4-bit Dualzahlen gebildet (siehe obige Übersicht).

Ganze Zahlen können in Turbopascal in verschiedenen Formen angemeldet werden, dies hängt mit der Platzreservierung zusammen. Es sollte immer der geringste Platzbedarf angestrebt werden. Wir unterscheiden die Typen

byte	:	0 .. 255
integer	:	-32768 bis + 32768
word	:	0 .. + 65535
longint	:	-2147483648 bis +2147483648

Menütechnik (am Beispiel von Aufgabe 14r Blatt 10, mögliche Lösung)**Program Laufflicht;**

```

uses crt;                { Einbinden zusätzlicher Befehle, z.B. clrscr }

const   io = 888;        { Konstantendeklaration, wegen Platzersparnis }
var     Taste : char;
        z : word;

procedure Lauf1;
begin
  clrscr;
  writeln('----- Laufflicht 1 ----- Stop durch Tastendruck -----');
  repeat
    port [io] := 1;      writeln ('Lampe-1');      delay(z);
    port [io] := 2;      writeln ('Lampe-2');      delay(z);
    port [io] := 4;      writeln ('Lampe-3');      delay(z);
    port [io] := 8;      writeln ('Lampe-4');      delay(z);
    {weitere Ausgaben sind 16, 32, 64 und 128}
  until keypressed;
end;

procedure Lauf2;
begin
  clrscr;
  writeln('----- Laufflicht 2 ----- Stop durch Tastendruck -----');
  repeat
    port [io] := 129;    writeln ('Lampe 1+8');    delay(z);
    port [io] := 66;    writeln ('Lampe 2+7');    delay(z);
    port [io] := 36;    writeln ('Lampe 3+6');    delay(z);
    {weitere Ausgaben sind 24, 36 und 66}
  until keypressed;
end;

{ ---- Hauptprogramm ---- }

begin
  z := 5000;              {Festlegen der Verzögerungszeit}
  Port[io] := 0;         {definiertes Ausschalten der LED-Anzeige}
  repeat
    clrscr;
    writeln ('Taste <1> oder <2> oder <ESC> = Ende eingeben');
    Taste := Readkey;
    case taste of
      '1' : Lauf1;
      '2' : Lauf2;
    end;
  until taste= #27;
end.

```

Das Programm hat den Nachteil, daß die Repeatschleife vollständig abgearbeitet werden muß, bevor keypressed wirksam wird. Abbruch evtl. möglich durch <ctrl> + <c> bzw. <ctrl> + <Untbr>.

Eine Verbesserung des vorab vorgestellten Programms ist nur möglich, wenn eine andere Verzögerungsprozedur zur Verfügung steht, die ein jederzeitiges Abbrechen gestattet. Im Folgenden werden zwei Methoden vorgestellt, die das Einbinden von immer wiederkehrenden, nützlichen Prozeduren und Funktionen beinhalten.

Methode 1 Include-Dateien

Der nachfolgende Programmteil wird im Editor geschrieben und unter write to mit dem Namen **tec.inc** (.inc diesmal mitangeben) abgespeichert. Die Endung .inc wurde frei gewählt, damit sie nicht mit den "normalen" Pascal Dateien verwechselt wird.

```
function zeit (wert : word) : boolean;                                {Delayersatz, mit dem }
  var    n : word;                                                  {Vorteil eines jederzeiti-}
  begin                                                                 {gen Abbruchs.        }
    zeit:=false;                                                    {Die Funktion hat den }
    for n:= 1 to wert do if keypressed then begin                    {Wert true (wahr), wenn}
                                                                    {eine Taste betätigt wird.}
                                                                    zeit:=true;
                                                                    exit;
                                                                    end;
  end;

procedure warte (k:word);
  begin
    if zeit(k) then exit;
  end;
```

Nachdem die Datei abgespeichert wurde, kann das Lauflichtprogramm von Blatt 12 wieder geladen und entsprechend geändert werden. Damit die einzelnen Programme nicht verwechselt werden, könnte man das neue Programm z.B. unter write to mit dem Namen Laufli_1 abspeichern.

Vorzunehmende Programmänderungen:

1. Im Programmkopf wird nach uses crt; die Zeile {\$I tec.inc} eingefügt,

Program Lauflicht_1;

```
  uses crt;                { Einbinden zusätzlicher Befehle, z.B. clrscr }
  {$I tec.inc}             { Einbinden der obigen Datei, Befehl warte}
  const    io = 888;       { Konstantendeklaration, wegen Platzersparnis }
  {sonst weiter wie gehabt}
```

2. delay(z); wird im gesamten Programm durch den neuen Befehl warte(z); ersetzt.

Nachteil dieser Methode: Es kann vorkommen, daß bei einem Syntax-Fehler im Hauptprogramm die Datei geschlossen und automatisch die Includedatei geöffnet wird, der Fehler aber nicht in der Includedatei liegt, sondern im Hauptprogramm. D.h. unter dem Menüpunkt File muß die alte Datei (Hauptprogramm) wieder geladen werden.

Vorsicht: Keine unbeabsichtigten Änderungen in der Includedatei vornehmen.

Methode 2 Erstellen einer Unit

Der nachfolgende Programmteil wird im Editor geschrieben und unter write to mit dem Namen **tec.pas** abgespeichert.

```

unit tec;                                     { Name = Dateiname }
interface                                     { Beginn Deklarationsblock}

uses crt;                                       {für keypressed erforderlich}
function zeit (wert : word) : boolean;
procedure warte (k:word);

implementation                               { Start Methodenblock }

function zeit (wert : word) : boolean;        { Delayersatz, mit dem }
var n : word;                                  { Vorteil eines jederzeiti-}
begin                                          { gen Abbruchs. }
    zeit:=false;                               { Die Funktion hat den }
    for n:= 1 to wert do if keypressed then begin { Wert true (wahr), wenn}
                                                zeit:=true;      { eine Taste betätigt wird.}
                                                exit;
    end;

procedure warte (k:word);
begin
    if zeit(k) then exit;
end;

begin
end.

```

Nach dem Abspeichern wird unter dem Menüpunkt Compiler die Option Destination Memory in Destination Disk geändert. Anschließend wird der Compilierungsvorgang mit Compile eingeleitet. Bei fehlerfreier Durchführung wird auf dem Datenträger eine Datei mit dem Namen **tec.tpu** angelegt. Diese Datei enthält die Funktion warte und wird genauso benutzt wie die Unit CRT. Für das weitere Arbeiten ist es sinnvoll, die Compileroption (Disk) wieder rückgängig (Memory) zu machen.

Für das Hauptprogramm wird nun eine neue Datei angemeldet und später unter dem Namen Laufli_2 abgespeichert.

Vorzunehmende Programmänderungen:

1. Im Programmkopf wird uses crt; durch uses crt, tec; ersetzt,

Program Lauflicht_2;

```

uses crt, tec;                                { Einbinden zusätzlicher Befehle, z.B. clrscr, warte }
{sonst weiter wie gehabt, $I tec.inc entfällt selbstverständlich}

```

2. delay(z); wird im gesamten Programm durch den neuen Befehl warte(z); ersetzt.

Vorteil dieser Methode: Es gibt kein unbeabsichtigtes Verändern der tpu-Datei.

TP 4.0 Ausgewählte Befehle**array**

Mit array werden Datenfelder vereinbart.
 Beispiel:

```
var Werte : array[1..10] of integer;
begin
  Wert[1] := 27;
  Wert[4] := 302;
  write( Wert[4] );
end.
```

Dieser Befehl bietet sich z.B. an, wenn Meßwerte in zeitlichen Abständen aufgenommen und abgespeichert werden sollen.

case..of

Komfortabler Auswahlselektor ersetzt if
 Beispiel:

```
var Taste : char;
begin
  read(kbd,taste);
  case taste of
    '?' : writeln(' Hilfe ');
    'x' : writeln (' x wurde gedrückt ');
  end;
end.
```

Wie bei *if* ist auch die Alternative *else* möglich.

cos, sin

Berechnung der Funktionswerte, Winkel im Bogenmaß angeben
 Beispiel:

```
writeln (sin(pi));
```


 pi ist eine vordefinierte REAL-Konstante

div, mod

Division ermittelt Vorkommawert bzw. Rest
 Beispiel:

```
writeln (10 div 3);      {Ergebnis = 3}
writeln (10 mod 3);     {Ergebnis = 1}
```

exit, halt

Beenden eines Unterprogramms z.B. Prozedur; Programmende

exp

Exponentialfunktion
 Beispiel:

```
Write ( exp(1) );      { Ergebnis = Eulerzahl e }
```

for..to..do

Zählschleife
 Beispiel:

```
var n : integer;
begin
  for n:=1 to 255 do begin
    port[888] := n;
  end;
end.
```


 Alternativ kann *for..downto..do* gewählt werden.

int

Bestimmung des Ganzzahlwertes einer Variablen
 Beispiel:

```
write ( int(pi) );      { Ergebnis = 3.000   REAL-Zahl }
```

sound	{uses crt}	Einschalten der Tonausgabe über den Systemlautsprecher
nosound	{uses crt}	Ausschalten der Tonausgabe (wichtig)
		Bsp.: begin sound (500); {Tonhöhe} delay (200); {Tondauer} nosound; end.
round		Wert runden Bsp.: write (round(2.53)); { Ergebnis = 3 }
str	Bsp.:	Konvertierung eines Zahlenwertes in einen String var wort : string[10]; lzahl : integer; Rzahl : real; begin lzahl := 576; Rzahl := 5.76; str (lzahl:3, wort); { 3 = Stellenzahl } writeln (wort); str (Rzahl:1:2, wort); { 1 = Vorkommazahl } writeln (wort); { 2 = Nachkommazahl } end.
textbackground	{uses crt}	Wahl der Hintergrundfarbe Textbackground (0..7);
textcolor	{uses crt}	Wahl der Zeichenfarbe Textcolor (0..15);
		Durch Kombination verschiedener Werte lassen sich schöne Bildschirmeffekte realisieren, z.B. blinkende oder reverse Darstellung der Zeichenausgabe.
trunc		Berechnung der Vorkommazahl vom Typ integer Bsp.: write (trunc (pi));
val	Bsp.:	Konvertierung eines Strings in einen Zahlenwert var Zahl : real oder integer; Fehler : integer; Wort : string[6]; begin wort := ' 12.3' ; val (wort,zahl,fehler); {Im Fehlerfall hat Fehler} writeln (zahl,' ',fehler); {den Wert 3 sonst 0 } end.
wherex	{uses crt}	Ermittelt die aktuelle x-Cursorposition
wherey	{uses crt}	Ermittelt die aktuelle y-Cursorposition
	Bsp.:	write(wherex, wherey);
window	{uses crt}	Textausgabefenster
	Bsp.:	window (1,1,80,25); = max. Größe

Projektthemen:

- 01. Zitterallye**
Modell und Softwareerstellung, evtl. Gruppenwettbewerb
- 02. Kreuzungs-, Fußgängerampel**
Modellbau und Softwareerstellung, Modellerweiterung durch das Erfassen mehrerer Kreuzungen (grüne Welle)
- 03. Morseprogramm**
Version-1: Bau eines Lichtmorsegerätes plus Ansteuersoftware
Version-2: Bau eines Lichtmorseempfängers plus Ansteuersoftware
- 04. Parkplatzerfassungssystem**
Modell und Softwareerstellung
Version-1: Nur zahlenmäßige Erfassung
Version-2: Bildschirmanzeige der belegten und freien Parkplätze
Version-3: Bildschirmleitsystem zu den freien Parkplätzen
- 05. Wettererfassungssystem**
Aufbereitung der physikalischen Größen (Temperatur, Windstärke, Feuchtigkeit, Niederschlag, Luftdruck), Abspeichern der erfaßten Daten in gleichen zeitlichen Abständen. Achtung: AD-Wandler erforderlich
- 06. Temperaturregelung eines Modellhauses**
Modell und Softwareerstellung, Achtung: AD-Wandler erforderlich
- 07. Aufzugssteuerung**
Modell und Softwareerstellung
- 08. Serielle Nachrichtenübertragung**
Verbindung der Rechner über den Centronics-Port (z.B. Data1 von Rechner A auf den Dateneingang DE1 von Rechner B und umgekehrt)

Mit dem Projekt verbunden ist die Aufgabe, eine saubere (Schreibmaschine, Computer) Projektbeschreibung zu erstellen. Die Beschreibung sollte prinzipiell die folgenden Kapitel enthalten:

- I Titel, Bezeichnung**
- II Autoren, Start und Ende des Projekts**
- III Zielaufgabe des Gerätes**
- IV Funktionsbeschreibung**
- V Skizze, Schaltpläne**
- VI Bauteilliste, Kosten**
- VII Baubeschreibung, Zeitumfang, Schwierigkeiten**
- VIII Kritik, Verbesserungsvorschläge, wie gehts weiter**

Turtle-Grafik

Mit Hilfe der Turtle-Grafik kann auf anschaulich einfache Weise der Umgang mit Prozeduren, Funktionen, Wertübergabe eingeübt werden. Die Turtle-Befehle können benutzt werden, wenn die Unit graph3 eingebunden wird.

An den Beispielprogrammen soll einmal der prinzipielle Programmaufbau gezeigt werden:

program Grafik1;

```
uses graph3;                                {Einbinden der benötigten unit}

{---Hauptprogramm---}

begin
  graphmode;                                {Grafik einschalten}
  showturtle;                               {Turtle zeigen, nicht unbedingt erforderlich}
  forwd (50);
  turnleft (90); forwd (50);                {hoffentl. selbsterklärend, sonst <Strg> + <F1>}
end.
```

program Grafik2;

```
uses crt, graph3;                           {Einbinden der benötigten units}

var Taste : char;                            {Variablendeklaration}

procedure grafikbildschirm;
begin
  graphmode;                                 {Grafik einschalten}
  showturtle;                               {Turtle zeigen, nicht unbedingt erforderlich}
  forwd(50);
  turnright(90); forwd(50);                 {hoffentl. selbsterklärend, sonst <Strg> + <F1>}
end;

procedure textbildschirm;
begin
  textmode(BW80);                           {Textmodus einschalten}
  gotoxy (20,12); write ('weiterhin g und t drücken, <ESC> = Ende');
end;

{---Hauptprogramm---}

begin
  clrscr;
  gotoxy(20,5); write ('Mit Hilfe der Tasten <g> und <t> kann zwischen');
  gotoxy(20,7); write ('Grafik- und Text-Bildschirm umgeschaltet werden. ');
  repeat
    Taste := readkey;                        {Lesebefehl ohne Bildschirmecho und Return}
    if Taste = 'g' then grafikbildschirm;
    if Taste = 't' then textbildschirm;
  until taste = #27;                          {Code der Taste ESC}
end.
```

Turtle-Grafikbefehle

```
const      North = 0;      East = 90;
           South = 180;     West = 270;
```

```
procedure Graphics;
```

```
procedure GraphMode;
```

```
procedure GraphColorMode;
```

```
procedure HiRes;
```

```
procedure HiResColor(Color: Integer);
```

```
procedure Palette(N: Integer);
```

```
procedure GraphBackground(Color: Integer);
```

```
procedure GraphWindow(X1,Y1,X2,Y2: Integer);
```

```
procedure Plot(X,Y,Color: Integer);
```

```
procedure Draw(X1,Y1,X2,Y2,Color: Integer);
```

```
procedure ColorTable(C1,C2,C3,C4: Integer);
```

```
procedure Arc(X,Y,Angle,Radius,Color: Integer);
```

```
procedure Circle(X,Y,Radius,Color: Integer);
```

```
procedure GetPic(var Buffer; X1,Y1,X2,Y2: Integer);
```

```
procedure PutPic(var Buffer; X,Y: Integer);
```

```
function GetDotColor(X,Y: Integer): Integer;
```

```
procedure FillScreen(Color: Integer);
```

```
procedure FillShape(X,Y,FillCol,BorderCol: Integer);
```

```
procedure FillPattern(X1,Y1,X2,Y2,Color: Integer);
```

```
procedure Pattern(var P);
```

```
procedure Back(Dist: Integer);
```

```
procedure ClearScreen;
```

```
procedure Forwd(Dist: Integer);
```

```
function Heading: Integer;
```

```
procedure HideTurtle;
```

```
procedure Home;
```

```
procedure NoWrap;
```

```
procedure PenDown;
```

```
procedure PenUp;
```

```
procedure SetHeading(Angle: Integer);
```

```
procedure SetPenColor(Color: Integer);
```

```
procedure SetPosition(X,Y: Integer);
```

```
procedure ShowTurtle;
```

```
procedure TurnLeft(Angle: Integer);
```

```
procedure TurnRight(Angle: Integer);
```

```
procedure TurtleDelay(Delay: integer);
```

```
procedure TurtleWindow(X,Y,W,H: Integer);
```

```
function TurtleThere: Boolean;
```

```
procedure Wrap;
```

```
function Xcor: Integer;
```

```
function Ycor: Integer;
```

Erweiterung der begonnenen **Technik-Unit**

unit tec;

interface

uses crt,dos;

```
{ *****
Allgemeine Prozeduren und Funktionen
***** }
```

TYPE

```
Switch=(on,off); {Software-Switches}
Regs=Registers;
t_speicher = array [1..8] of byte;
c_titel = char;
t_titel = string[8];
```

var

```
zeichensatz : array [0..255] of t_speicher
              absolute $f000:$fa6e;
```

```
procedure farbe (a,b:byte);
procedure rvs;
procedure rvsb;
procedure ton(frequenz, dauer:integer);
```

```
function zeit (wert:word):boolean;
procedure warte (k:word);
procedure cursor(CuSw:Switch);
```

```
procedure leseeintrag (eintrag: t_speicher);
procedure bigchar (titel:c_titel; posx, posy :byte);
procedure bigtitel (titel : t_titel; posy :byte);
```

Implementation

```
procedure farbe (a,b:byte);
begin
  textcolor(a);
  textbackground(b);
end;
```

```
procedure rvs;
begin
  textbackground(7);
  textcolor(8);
end;
```

```
procedure rvsb;
begin
  textbackground(7);
  textcolor(24);
end;
```

```
procedure ton(frequenz, dauer:integer);
begin
  sound(frequenz);
  delay(dauer);
  nosound;
end;
```

```
function zeit (wert:word):boolean;
var n: word;
begin
  zeit:=false;
  for n:= 1 to wert do if keypressed then
    begin
      zeit:=true;
      exit;
    end;
end;
```

```
procedure warte(k:word);
begin
  if zeit(k) then exit;
end;
```

```
procedure cursor(CuSw:Switch);
{ *****
Schaltet den Cursor im Textmodus ein
(CuSw=on) oder aus (CuSw=off)
***** }
```

```
var Reg:Regs;
begin
  Reg.AX:=$0100;
  IF CuSw=on THEN Reg.CX:=$0607 ELSE
    Reg.CX:=$2607;
  Intr($10,Reg);
  { 0c0d bei hgc }
  { 0607 bei cga }
end;
```

```

procedure leseeintrag (eintrag: t_speicher);
var
  pixel_posy, pixel_posx : 1..8;
  pixel                   : byte;

begin
  lowvideo;
  for pixel_posy := 1 to 7 do
    begin
      pixel := eintrag [pixel_posy];
      for pixel_posx := 8 downto 1 do
        begin
          gotoxy (pixel_posx,pixel_posy);
          if odd (pixel) then write ('Ü')
            else write (' ');
          pixel := pixel div 2
        end
      end;
    highvideo
  end;
end;

```

```

{*****
  Gibt Großbuchstaben an Position x,y aus
  *****}

```

```

procedure bigchar (titel:c_titel; posx,posy:byte);

```

```

var
  n, x : byte;

begin
  x:= posx+2 ;
  window (x-2,posy,x + 7,posy +8);
  leseeintrag (zeichensatz [ord (titel)]);
  window (1,1,80,25)
end;

```

```

{*****
  Gibt Text zentriert in y-Position aus
  *****}

```

```

procedure bigtitel (titel : t_titel; posy :byte);

```

```

var
  n, laenge,posx,x : byte;

begin
  laenge := length (titel);
  posx := ((80-8*laenge) div 2 ) and $00ff;
  for n := 1 to laenge do
    begin
      x:= posx + 8 * (n-1);
      window (x-2,posy,x + 7,posy +8);
      leseeintrag (zeichensatz [ord (titel [n])])
    end;
  window (1,1,80,25)
end;

```

```

begin
end.

```

Befehle der Unit Tec (uses tec;)

Warte (Wert);	alternativer delay-Befehl
Farbe (Stiftfarbe,Hintergrundfarbe);	
rvs;	Revers, entspricht Farbe mit den Werten (8,7)
rvsb;	Revers blink, entspricht Farbe mit den Werten (24,7)
normvideo;	Grundeinstellung
Ton(frequenz,dauer);	Tonausgabe über Systemlautsprecher
Cursor(on bzw. off);	Schaltet den Cursor ein bzw. aus
Bigchar ('x',x,y);	Gibt Großbuchstaben an Position x,y aus
Bigtitel ('Wort',y);	Gibt Wort zentriert in Zeile y aus
Drahmen (x,y,b,h);	Zeichnet Rahmen mit doppeltem Strich Start linke obere Ecke, Position x,y b= Breite und h = Höhe
Erahmen (x,y,b,h);	Zeichnet Rahmen mit einfachem Strich Start linke obere Ecke, Position x,y b = Breite und h = Höhe
Tc_Rahmen1('Titel');	Zeichnet Rahmen mit Programm-Titel
Hotkeyzeile ('Text');	Fügt in Tc_Rahmen untere Zeile ein

Lesbarkeit von Programmen

Aufgabe 1

Übertrage die Programmbeispiele 1..3 und führe sie aus. Wodurch unterscheiden sich die Programme? Halte die Ergebnisse schriftlich fest.

Beispiel-1

```
uses crt, graph3; var i : integer; procedure q;
begin i:=1; repeat i:=i+1; forwd (40);
turnleft (90); until i=5; end; begin graphmode;
showturtle; q; end.
```

Beispiel-2

```
uses crt, graph3;
var k : integer;
procedure a;
begin
k:=2;
repeat
k:=k+2;
forwd (60);
turnright (90);
until k=10;
end;
begin
graphmode;
showturtle;
a;
end.
```

Beispiel-3

```
uses crt, graph3;
var zaehler : integer;
procedure quadrat;
begin
zaehler:=0;
repeat
zaehler:=zaehler+1;
forwd (50);
turnright (90);
until zaehler=4;
end;
begin
graphmode;
showturtle;
quadrat;
end.
```

Aufgabe 2

Zkizziere das Ergebnis des nachfolgenden Programmablaufs. Welche sinnvolleren Namen fallen Dir für kreis, k und r ein.

```
uses graph3;
var k : integer;
procedure kreis (r : integer);
begin
k:=0;
repeat
k:=k+1;
forwd (40);
turnright (90);
until k=4;
if r > 4 then halt;
turnright(20);
kreis(r+1);
end;
begin
graphmode;
showturtle;
turtledelay(200);
kreis (1);
end.
```