QBasic-Materialsammlung für den Informatikunterricht

von Wolfgang Gutenbrunner

E-Mail: w.gutenbrunner@eduhi.at
Webseite: http://home.eduhi.at/teaching/Gutenbrunner/Material.htm

PDF-Konvertierung von Thomas Antoni – www.qbasic.de

Inhalt

Kurze Einführung in das Programmieren in QBasic 2
1. Befehls-Zeilen; Kommentare 2
2. Variablen-Namen – Wert-Zuweisung 2
3. Bildschirmausgabe – Eingabe 3
4. Verzweigungen und Programmschleifen 4
5. Zufallszahlen 6
6. „Indizierte“ Variable, Felder 7
7. Grafik 7
8. Unterprogramme 8
9. Geräusche und Töne 9

Einführungsbeispiele in das Programmieren mit QBasic 10
10. PROGRAMM 1: „Endlosschleife“ 10
11. PROGRAMM 2: „Schleife mit Abbruchbedingung“ 10
12. PROGRAMM 3: „Zählschleife“; „Zufallszahlen“ 11
13. PROGRAMM 4: „Grafik aus Kreisen“ 12
14. PROGRAMM 5: „Zufallsrechtecke in Zufallsfarben“ 13

KNOBELEI 1 aus der ORF-Computerbox 14
KNOBELEI 2 aus der ORF-Computerbox 16
KNOBELEI 3 aus der ORF-Computerbox 19
Programmiersprachen 25

Kurze Einführung in das Programmieren in QBasic

1. Befehls-Zeilen; Kommentare

Im Allgemeinen steht jeder Basic-Befehl in einer eigenen Zeile. Werden mehrere Basic-Befehle in eine Zeile geschrieben, müssen sie jeweils durch einen „Doppelpunkt“ getrennt werden. So beginnt man ein Programm oft mit folgenden zwei Befehlen:

```
CLS : CLEAR
```

CLS („clear screen“) löscht den Bildschirm; CLEAR löscht alle Variablen.

Zum besseren Verständnis eines Programms können Kommentare eingefügt werden; damit diese beim Programmablauf selbst nicht stören, muss ihnen der Befehl **REM** („remark“) oder ein „Hochkomma“ (= ' = <Shift> + <#>) vorangestellt werden. Z. B.:

```
REM Programmtitel =  Lottozahlen
CLS : CLEAR          'Löschen von Bildschirm und Variablen
```

2. Variablen-Namen – Wert-Zuweisung

Die ***Variablen*** stehen für Zahlen oder Texte, die sich der Computer unter diesen Bezeichnungen „merkt“. Variablen-Namen können fast beliebig gewählt werden. Sie können bis zu 40 Zeichen lang sein, dürfen nicht mit einer Ziffer beginnen und keine Sonderzeichen (auch keine Umlaute, Bindestriche, Abstände) enthalten. Variable für Text enden mit dem Dollar-Zeichen (= $ = <Shift> + <4>).

Die **Wert-Zuweisung** erfolgt über das „Ist gleich“-Zeichen.
Z. B.:

```
Vorname$ = "Wolfgang"
Zeichen$ = "f"
Laenge = 12
Breite = Laenge / 2
Flaeche = Laenge * Breite
Umfang = 2*(Laenge + Breite)
```

3. Bildschirmausgabe – Eingabe

Mit dem Befehl **LOCATE** *Zeile*, *Spalte* kann vor einem Ausgabe-Befehl (PRINT) oder einem Eingabe-Befehl (INPUT) der Cursor an eine bestimmte Stelle des Bildschirms gesetzt werden. Das ist wichtig zum Aufbau geeigneter „Bildschirm-Masken“. Der Text-Bildschirm hat 25 Zeilen (1 bis 25 von oben nach unten) und 80 Spalten (1 bis 80 von links nach rechts). Z. B.:

LOCATE 1, 1 : PRINT "A"	das A steht ganz links oben;
LOCATE 25, 40 : PRINT "Wolfi"	der Name beginnt unten in der Mitte.

COLOR *Vordergrund*, *Hintergrund* setzt die Schriftfarbe und die Hintergrundfarbe, wobei für die beiden Variablen Werte von 0 bis 15 zu setzen sind.
(0 = schwarz, 1 = blau, 2 = grün, 3 = türkis, 4 = rot, 5 = violett, 6 = braun, 7 = weiß, 8 = grau, 9 = hellblau, 10 = hellgrün, 11 = helltürkis, 12 = hellrot, 13 = hellviolett, 14 = gelb, 15 = helles Weiß).

BEEP gibt einen kurzen Piepston am eingebauten Lautsprecher aus. Längerer Ton:
Z. B.: FOR i = 1 TO 50 : BEEP : NEXT i

Der **PRINT**-Befehl gibt Texte (in Anführungszeichen) oder Variableninhalte (Variable ohne Anführungszeichen!) am Bildschirm aus. Endet der Befehl mit „Strichpunkt“ (= ; = <Shift> + <.>), wird die nächste Bildschirm-Ausgabe gleich daneben gesetzt; nach einem „Beistrich“ wird ein bestimmter größerer Abstand gehalten; endet der PRINT-Befehl ohne ein solches Zeichen (oder mit „Doppelpunkt“), springt der Cursor an den Anfang der nächsten Zeile.
Z. B.: *Laenge* sei die Variable für die Länge eines Rechteks und habe den Wert 12:

PRINT "Länge"	Länge
PRINT *Laenge*	12
PRINT "Die Länge ist "; *Laenge***; "cm!"**	Die Länge ist 12 cm!

Der **INPUT**-Befehl ermöglicht Eingaben während des Programmablaufs. Das Programm hält dazu an, bis die Eingabetaste gedrückt wurde. Ein eventueller Kommentar (unter Anführungszeichen!) wird vorher am Bildschirm ausgegeben. Die Eingabe wird unter der angegebenen Variablen gespeichert. Z. B.:

INPUT "Wie lang soll das Rechteck sein"; Laenge
INPUT "Gib bitte die gewünschte Anzahl ein:", Anzahl

Endet der Kommentar mit Strichpunkt, wird ein Fragezeichen am Bildschirm gezeigt, wo die Eingabe erfolgen soll; bei einem Beistrich wird das Fragezeichen unterdrückt.

Mit dem Befehl **INKEY$** kann ein einzelner Tastendruck eingelesen werden (auf den das Programm dann sofort reagiert – ohne die Eingabe-Taste). Allerdings wartet das Programm nicht auf diesen Tastendruck. Daher muss der INKEY$-Befehl in eine „Warteschleife“ verpackt werden.

... **Befehl7** **DO WHILE** *Taste$* **= ""** *Taste$* **= INKEY$** **LOOP** **IF** *Taste$* **= "e" THEN END** **Befehl8** **...**	Beispiel: Das Programm „wartet“ (DO WHILE ... LOOP) nach dem *Befehl7* so lange, bis irgendeine Taste gedrückt wurde (bis die Variable *Taste$* irgendetwas enthält, nicht mehr gleich „leer“ ist) Wenn der Tastendruck ein „kleines E“ war, endet das Programm; ansonsten setzt es mit dem *Befehl8* fort.

4. Verzweigungen und Programmschleifen

Die „bedingte Verzweigung“ **IF ... THEN(... ELSE)**: **Wenn – dann – sonst**:
Wenn die Bedingung erfüllt ist, wird der „Befehl1“ ausgeführt, ansonsten der „Befehl2“; anschließend setzt das Programm bei „Befehl3“ fort: **...** **IF** *Bedingung* **THEN** *Befehl1* **ELSE** *Befehl2* *Befehl3*

Oft genügt auch ein „**wenn – dann**“: Ist die Bedingung erfüllt, wird der „Befehl1“ ausgeführt, ansonsten eben nicht; das Programm setzt dann in jedem Fall mit dem „Befehl2“ fort:

```
...
IF Bedingung THEN Befehl1
Befehl2
...
```

Z. B.: Ist die Bedingung erfüllt, endet das Programm; ansonsten geht es mit dem nächsten Befehl weiter:

```
IF Bedingung THEN END
Befehl ...
```

Es können sowohl nach dem THEN wie auch nach dem ELSE mehrere Befehle kommen; der ganze Block muss dann mit END IF abgeschlossen werden:

```
...
IF Bedingung THEN
   Befehl
   Befehl
   ...
ELSE
   Befehl
   Befehl
   ...
END IF
```

Die **DO ... LOOP** – Schleife:

„Endlosscheife“:

```
DO
   Befehl
   Befehl
   . . .
LOOP
```

Programmabruch über die Tastenkombination <Strg> + <Pause>!

Ausstieg mit EXIT DO:

```
DO
   Befehl
...
   IF Bedingung THEN EXIT DO
   Befehl
   ...
LOOP
```

Die **DO ... LOOP** – Schleife mit Abbruchbedingung:

Tu **solange** die Bedingung erfüllt ist

```
DO WHILE Bedingung
   Befehl
   Befehl
   ...
LOOP
```

Tu **bis** die Bedingung erfüllt ist

```
DU UNTIL Bedingung
   Befehl
   Befehl
   ...
LOOP
```

Die **FOR … NEXT** – Schleife = „Zählschleife“:

Die folgende Schleife wird z. B. 100 mal durchlaufen, (falls nicht vorher schon die eingebaute Abbruchbedingung erfüllt ist):

```
FOR Variable = 0 TO 100
   Befehl
   Befehl
   ...
   IF Beding. THEN EXIT FOR
   Befehl
   ...
NEXT Variable
```

Reine Zählschleifen dienen zur Verlangsamung des Programm-Ablaufs:

```
FOR Zaehler = 1 TO 5000
NEXT Zaehler
```

Man kann auch z. B. in „Dreierschritten“ zählen: 0, 3, 6, 9, 12, ...:

```
FOR x = 0 TO 30 STEP 3
   Befehl(e)
NEXT x
```

Oder „herunterzählen“:

```
FOR z = 20 TO 0 STEP  1
   Befehl(e)
NEXT z
```

5. Zufallszahlen

Neustart des „Zufallsgenerators“ mit dem Befehl **RANDOMIZE TIMER**.

Die Funktion **RND** („random“) liefert eine Zufallszahl (mit 7 Nachkommastellen), die größer 0 und kleiner 1 ist. So könnte **x = RND** für die Variable **x** etwa die Zufallszahl 0,0147284 oder auch 0,804582 liefern.

Die Funktion **INT** („integer“) verkleinert eine Komma-Zahl zur nächstkleineren „Ganzen Zahl“ (schneidet die Nachkommastellen einer positiven Zahl ab); z. B. ergibt der Ausdruck INT (3,4780329) den Wert 3 oder INT (0,9659372) den Wert 0; dementsprechend liefert die Befehlszeile *Variable* **= INT (RND)** auf jeden Fall für die Variable den Wert 0.

Wenn also **RND** eine Zahl zwischen 0,0000001 und 0,9999999 ergibt, dann liefert **RND * 45** eine Zahl zwischen 0,00000045 und 44,9999955. Demnach erhält die Variable in der Befehlszeile *Variable* **= INT (RND * 45)** einen zufälligen Wert zwischen 0 und 44. Will man eine Zahl von 1 bis 45 haben (z. B.: „Lotto 6 aus 45“), benötigt man daher den Befehl *Variable* **= INT (RND * 49) + 1**.

6. „Indizierte“ Variable, Felder

Variable können auch von der Art x_0, x_1, x_2, x_3, ... sein: das ist immer dieselbe Variable „x“ mit jeweils einem anderen „Index“. In der Basic-Schreibweise wäre das x(1), x(2), x(3), ... Nur wenn mehr als 11 „Indizes“ benötigt werden, muss man die Variable zunächst „dimensionieren“.

Z. B.: **DIM a(25)** erlaubt die Bildung der Variablen a(0) bis a(24).

RANDOMIZE TIMER : DIM a(25) **FOR i = 0 TO 24** ** a(i) = INT (RND * 6) + 1** **NEXT i** **FOR r = 1 TO 24** ** PRINT a(j);** **NEXT r**	Das Programm würfelt 25 mal und merkt sich alle Ergebnisse: Das Programm weist den Variablen a(0) bis a(24) zufällige Werte zwischen 1 und 6 („Würfelzahlen“) zu. Dann druckt es alle diese Zahlen nebeneinander auf den Bildschirm

7. Grafik

SCREEN 0 schaltet den Text-Schirm ein und damit die Grafik aus.

SCREEN 8 schaltet einen Grafikschirm mit 640 Punkten (waagrecht) mal 200 Punkten (senkrecht; y) und 16 möglichen Farben ein. Werte: x von 0 bis 639; y von 0 bis 199; Farbe: 0 bis 15.

PSET (x, y), F Punkt setzen auf Koordinate (x, y); F = Farbe (0 bis 15).

LINE (x_1, y_1) (x_2, y_2), F Linie ziehen von $P_1 = (x_1, y_1)$ nach $P_2 = (x_2, y_2)$.

LINE (x_1, y_1) (x_2, y_2), F, B Rechteck (b für „box“) zeichnen, in dem die angegebene Linie eine Diagonale wäre.

LINE (x_1, y_1) (x_2, y_2), F, BF Gefülltes Rechteck (BF für „box“ und „fill“).

CIRCLE (x, y), r, F Kreis mit Mittelpunkt in (x, y) und Radius = r.

PAINT (x, y), F Eine geschlossene (!) Fläche in der Farbe F ausfüllen („paint“ = malen), wobei der Punkt (x, y) innerhalb (!) dieser Fläche liegen muss.

8. Unterprogramme

Durch die Verwendung von Unterprogrammen wird ein Programm übersichtlicher und verständlicher. Besonders dann, wenn z. B. bestimmte Berechnungen innerhalb eines Programms mehrmals in gleicher Weise auftreten, wird man sich ein Unterprogramm dafür schreiben und dies an den entsprechenden Stellen des Hauptprogramms immer wieder aufrufen. Die Unterprogramme werden einfach unterhalb des Hauptprogramms in beliebiger Reihenfolge an den Programmtext angehängt.

Unterprogramme sind also Programmteile, die vom Hauptprogramm aus beliebig aufgerufen werden können. Ein Unterprogramm beginnt mit einem Variablennamen, der mit einem Doppelpunkt endet. Der Aufruf vom Hauptprogramm aus erfolgt mit dem Befehl **GOSUB** *Unterprogrammname*.

Der letzte Befehl des Unterprogramms ist der Befehl **RETURN**. Damit geht der Programmablauf wieder in das Hauptprogramm zurück; dieses wird an der Stelle fortgesetzt, die auf den Unterprogramm-Aufruf folgt. Die vom Unterprogramm berechneten oder abgeänderten Variableninhalte stehen dann im Hauptprogramm zur Verfügung.
Beispiel: Mehrmals in einem Programm muss der „Pythagoras“ angewendet werden:

Hauptprogramm:	**Unterprogramm:**
a = 24.8 : b = 17.4 **GOSUB Pythagoras** **PRINT c** **...**	**Pythagoras:** ** c = SQR (a * a + b * b)** **RETURN**

9. Geräusche und Töne

Mit dem Befehl **SOUND** *Frequenz*, *Dauer* können beliebige Geräusche, Töne und Tonfolgen einprogrammiert werden. Der Wert für Frequenz kann von 37 bis 32767 (Hz) gehen.

```
Sirene:
  i=500
  DO WHILE INKEY$ = ""
    SOUND i, 0.1
      IF i = 800 THEN si = 1
      IF i = 500 THEN si = 0
      IF si = 0 THEN i = i + 1
      IF si = 1 THEN i = i   1
  LOOP
RETURN
```

Das Unter-Programm **Sirene:** läuft, bis irgendeine Taste gedrückt wurde:
Auf- und abschwellender Heulton:
Wenn 800 Hz erreicht sind (si = 1), gehts abwärts (i = i – 1);
wenn 500 Hz erreicht sind (si = 0), gehts aufwärts (i = i + 1).

Diese Unterprogramm wird vom Hauptprogramm aus mit
GOSUB *Sirene*
aufgerufen.

Der Kammerton a_1 etwa schwingt mit 440 Hz. Eine Oktave höher verdoppelt sich der Wert: Wenn d_1 295 Hz hat, dann hat d_2 590 Hz. Ungefähre Ton-Frequenzen:

c_1		d		e	f		g		a		h	c_2
263	cis des	295	dis es	330	350	fis ges	394	gis as	440	ais b	494	526
	279		312			371		416		466		

Tonlänge (Dauer): Wenn man einen Ganzton mit dem Wert 16 annimmt, so wird dieser nicht ganz eine Sekunde dauern (1 sek entspricht ca. 18.2). Halb-, Viertel-, Achtel-Noten sind dann mit dementsprechend kleineren Werten anzusetzen. Für Klangeffekte können auch sehr kleine Werte eingesetzt werden (ab etwa 0.03).

Pause: Für Frequenz den Wert 0 setzen. Z. B.: Achtelpause: SOUND 0, 2.

Einführungsbeispiele in das Programmieren mit QBasic

10. PROGRAMM 1: „Endlosschleife“

Jeder Befehl steht in einer eigenen Zeile. QBASIC-Befehle werden vom Programm „QBASIC“ selbst in Großbuchstaben umgewandelt (Kontrolle der Schreibweise!).

Texte nach einem ' (= Hochkomma = <Shift>+<#> [gleich links von <Return>]) werden vom Programm ignoriert (dienen zur Programm-Erläuterung).

Das Programm wird Zeile für Zeile (also Befehl für Befehl) abgearbeitet. Mit dem Befehl „DO - - - LOOP“ wird eine endlose Wiederholung der eingeschlossenen Befehlsfolge erzwungen. (Abbruch nur mit <Strg>+<Pause> [oder <Strg>+<C>]!)

Das folgende Programm zählt von 0 aufwärts (0, 1, 2, 3, ...)!

```
CLS                         'clear screen (Bildschirm löschen).
x = 0                       'Startwert für die Variable [x].
DO                          'Beginn der „Endlosschleife“.
        PRINT x;                    'Wert von [x] auf Bildschirm drucken.
        x = x + 1                   'Wert von [x] um eins erhöhen.
LOOP                        'Rücksprung zum Beginn der „Endlosschleife“.
```

11. PROGRAMM 2: „Schleife mit Abbruchbedingung“

Mit den Befehlen „DO UNTIL Bedingung - - - LOOP“ („tu bis [Bedingung] erfüllt“) oder „DO WHILE Bedingung - - - LOOP“ („tu solange [Bedingung] erfüllt“) programmiert man „Schleifen“, die ihren Ablauf unter ganz bestimmten Bedingungen von selbst „abbrechen“.

Das folgende Programm zählt von 1 weg in Zweierschritten (also 1, 3, 5, ...) und bricht ab, wenn 10 000 überschritten ist: die letzte ausgedruckte Zahl ist demnach die Zahl 9 999.

```
CLS                         'clear screen (Bildschirm löschen).
x = 1                       'Startwert für die Variable [x].
DO UNTIL x > 10000          'Beginn der „bedingten Schleife“; Überprüfung,
                            'ob die Abbruchbedingung (x > 10 000) schon
                            'erfüllt ist (wenn ja: Ende des Programms!).
        PRINT x;                    'Wert von [x] auf Bildschirm drucken.
        x = x + 2                   'Wert von [x] um zwei erhöhen.
LOOP                        'Rücksprung zum Beginn der „Schleife“.
```

12. PROGRAMM 3: „Zählschleife“; „Zufallszahlen“

Schreibt man zwei (oder mehr) Befehle in eine Zeile, muss man sie durch **:** (Doppelpunkt) voneinander trennen! Texte nach dem Befehl „REM“ (= remark = Bemerkung) werden vom Programm ignoriert und dienen zur Erläuterung des Programms.
Ein Programm sollt möglichst gut gegliedert und ausführlich erläutert sein. Man sollte sich auch nach längerer Zeit noch im Programmtext auskennen und eventuell Verbesserungen durchführen und Erweiterungen einfügen können!

Das folgende Programm (eine Anwendung des „Zufallsgenerators“) „würfelt“ beliebig oft (man gibt die Anzahl der „Würfe“ am Anfang ein) und stellt fest, wie oft der „Einser“, „Zweier“, ..., „Sechser“ gewürfelt wurde:

```
CLS                                          'Bildschirm löschen.
CLEAR                                        'Alle Variablen auf 0 setzen.

REM Eingabe (Anzahl der Würfe):              'Bemerkung.
PRINT                                        'Leerzeile am Bildschirm.
PRINT                                        'Leerzeile am Bildschirm.
INPUT „Wie oft soll ich würfeln“; Anzahl     'Warten auf eine Eingabe (mit gleichzeitiger
                                             'Textausgabe) am Bildschirm; Übernahme der
                                             'eingegebenen Zahl in die Variable [Anzahl].
PRINT                                        'Leerzeile am Bildschirm.
PRINT                                        'Leerzeile am Bildschirm.
PRINT „Nur Geduld! Ich würfle ja schon!!“    'Textausgabe am Bildschirm.
RANDOMIZE TIMER                              'Zufallsgenerator einschalten.

REM Nun wird gewürfelt:                      'Bemerkung.
FOR i = 1 TO Anzahl                          'Beginn der „Zählschleife“: diese wird sooft
                                             'durchlaufen, bis [i] den Wert [Anzahl] erreicht.
    x = INT(RND * 6) + 1                         '[x] = Zufallszahl von 1 bis 6.
    IF x = 1 THEN Einser = Einser + 1            '
    IF x = 2 THEN Zweier = Zweier + 1            '
    IF x = 3 THEN Dreier = Dreier + 1        'Es wird „mitgezählt“, wie oft die einzel-
    IF x = 4 THEN Vierer = Vierer + 1        'nen Zahlen „kommen“.
    IF x = 5 THEN Fuenfer = Fuenfer + 1          '
    IF x = 6 THEN Sechser = Sechser + 1          '

NEXT i                                       'Rücksprung zum Beginn der „Zählschleife“.

REM Ausgabe der Ergebnisse:                  'Bemerkung.
PRINT                                        'Leerzeile am Bildschirm.
PRINT                                        'Leerzeile am Bildschirm.
PRINT „Der Einser wurde“; Einser; „mal gewürfelt“    '
PRINT „Der Zweier wurde“; Zweier; „mal gewürfelt“    '
PRINT „Der Dreier wurde“; Dreier; „mal gewürfelt“    'Die Ergebnisse des Würfelns (wie oft
PRINT „Der Vierer wurde“; Vierer; „mal gewürfelt“    'ist jede Zahl gekommen?) werden am
PRINT „Der Fünfer wurde“; Fuenfer; „mal gewürfelt“   'Bildschirm ausgegeben.
PRINT „Der Sechser wurde“; Sechser; „mal gewürfelt“  '
```

13. PROGRAMM 4: „Grafik aus Kreisen“

Grafikbefehle in einer „bedingten Schleife“. Experimentiere mit Abänderungen der Zuwächse oder Verminderungen von x, y und r im Hauptteil des Programms!

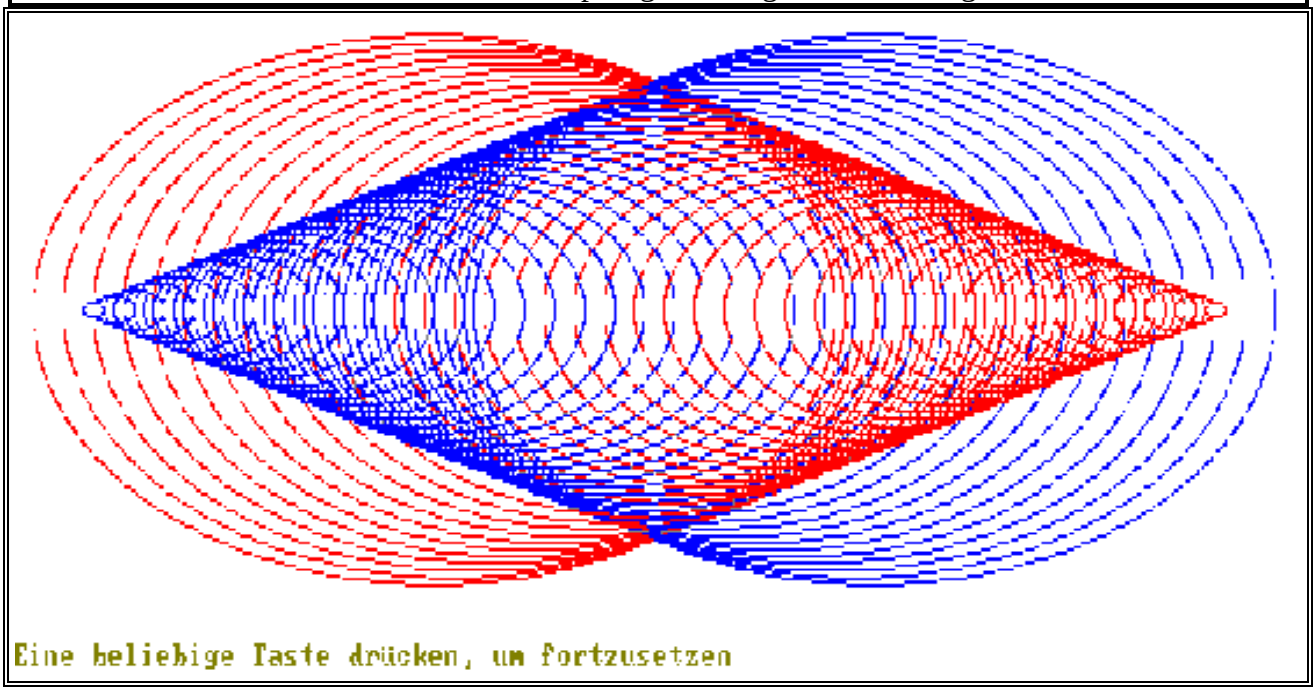
```
REM Einstellungen:                  'Bemerkung.
CLS                                 'Bildschirm löschen.
SCREEN 8                            'Grafikschirm 640 x 200; 16 Farben.
COLOR 6, 15                         'Vordergund (Text) braun (6), Hintergrund weiß (15).

REM Startwerte:                     'Bemerkung.
x1 = 210                            'x-Startwert für Mittelpunkt1.
y1 = 90                             'y-Startwert für Mittelpunkt1.
r1 = 200                            'Startwert für Radius1.
x2 = 430                            'x-Startwert für Mittelpunkt2.
y2 = 90                             'y-Startwert für Mittelpunkt2.
r2 = 200                            'Startwert für Radius2.

REM Hauptteil:                      'Bemerkung.
DO WHILE r1 > 0                     'Beginn der „bedingten Schleife“: Überprüfung, ob die
                                    'Bedingung [r1 > 0] (noch) erfüllt ist; wenn nicht: Ende
                                    'des Programms!

    CIRCLE (x1, y1), r1, 12         'Zeichne Kreis1 in rot (12).
    CIRCLE (x2, y2), r2, 9          'Zeichne Kreis2 in blau (9).
    x1 = x1 + 10                    'Mittelpunkt1 verschiebt sich nach rechts.
    x2 = x2 - 10                    'Mittelpunkt2 verschiebt sich nach links.
    y1 = y1                         'Mittelpunkt1 und Mittelpunkt2 bleiben auf.
    y2 = y2                         'gleicher Höhe (verschieben sich waagrecht).
    r1 = r1 - 5                     'Radius1 wird kleiner.
    r2 = r2 - 5                     'Radius2 wird kleiner.

LOOP                                'Rücksprung zum Beginn der „bedingten Schleife“.
```



14. PROGRAMM 5: „Zufallsrechtecke in Zufallsfarben“

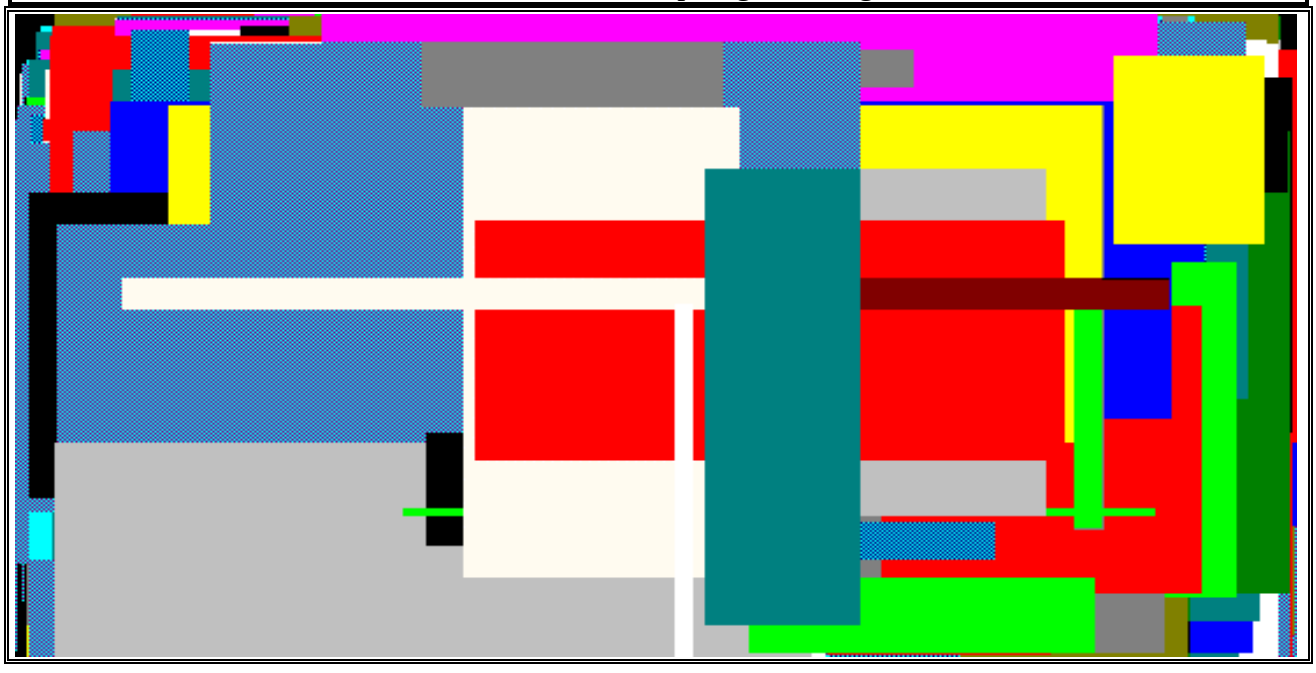
Es werden („gefüllte“) Rechtecke verschiedenster Größe, Form und Lage in den verschiedensten Farben zufällig erzeugt. Das sieht am Bildschirm recht hübsch aus!

```
REM Einstellungen:                      'Bemerkung.
CLS                                     'Bildschirm löschen.
SCREEN 8                                'Grafikschirm 640 x 200, 16 Farben.
COLOR 0, 15                             'Textfarbe schwarz (0), Hintergrundfarbe weiß (15).

REM Hauptteil:                          'Bemerkung.
RANDOMIZE TIMER                         'Zufallsgenerator starten.
DO                                      'Beginn der „Endlosschleife“.
        x1 = INT(RND * 640)             'Zufallswert für [x1] von 0 bis 639.
        y1 = INT(RND * 200)             'Zufallswert für [y1] von 0 bis 199.
        x2 = INT(RND * 640)             'Zufallswert für [x2] von 0 bis 639.
        y2 = INT(RND * 200)             'Zufallswert für [y2] von 0 bis 199.
        Farbe = INT(RND * 16)               'Zufallswert für [Farbe] von 0 bis 15.
        LINE (x1, y1) - (x2, y2), Farbe, BF 'Rechteck (gefüllt) in [Farbe] zeichnen.

                FOR i = 1 TO 200                '„Warteschleife“ zur Verlangsamung
                NEXT i                          'des Programm-Ablaufs.

LOOP UNTIL INKEY$ <> ""                 'Rücksprung zum Beginn der „Endlosschleife“.
```



Vieles kann man in QBASIC durch die Verwendung des Menü-Punktes „Hilfe“ lernen! Versuche, auftretende Probleme mit „Hilfe“ zu lösen! Der „Index“ gibt dir darüber hinaus eine alphabetische Übesicht über alle QBASIC-Befehle - jeden einzelnen kannst du dir ausführlich erklären lassen!
Falls sich jemand für das Programmieren in QBasic (oder Quick-Basic) näher interessiert, dem sei folgendes Buch empfohlen: Anatol Gardner: „Qbasic und

Quick-Basic - Der schnelle Weg zum erfolgreichen Programmieren“, dtv-Taschenbuch, 194,- öS.

KNOBELEI 1 aus der ORF-Computerbox

Aufgabe aus der ORF-COMPUTERBOX (Teletext 464) vom 16. 07. 1998.

Aufgabenstellung:

Herr Ernst Kleiner aus Riedlingen sandte uns folgende Aufgabe:
Eine Transmission (Kraftübertragung durch zwei über einen Riemen verbundene Räder) hat 500 mm Achsenabstand und 2 Räder von je 100 mm Radius (Übersetzung 1:1).
Schreiben Sie ein Programm, das den Durchmesser von 2 anderen auf den gleichen Achsen montierten Rädern für eine Übersetzung von 2:1 (bei gleicher Riemenlänge) berechnet.
(Einsendeschluss: 04. 08. 1998).

Anmerkung: Die Aufgabenstellung ist ein sehr gutes Beispiel für die Nützlichkeit eines Computer-Programms: Es wäre sehr schwierig, das Problem rein mathematisch exakt lösen zu wollen, während die (fast beliebig genaue) Annäherung an den gesuchten Wert durch das Programm leicht verständlich und relativ einfach erscheint.

Überlegung:

Mein Programm ist in QBasic geschrieben und erklärt sich wohl selbst:

```
DECLARE FUNCTION ArcusCosinus! (x!)
DECLARE SUB LaengenBerechnung (Laenge1!, Radius!, AchsenAbstand!)

CLS : CLEAR
CONST PI = 3.141592654#

AchsenAbstand = 500
REM Berechnung der AusgangsLänge:
Radius0 = 100
Laenge0 = 2 * Radius0 * PI + 2 * AchsenAbstand

REM Der gesuchte Radius wird nun durch
REM "Intervall-Halbierung" angenähert:
RadiusAlt = 100
RadiusNeu = 50
Radius = RadiusNeu
CALL LaengenBerechnung(Laenge1, (Radius), (AchsenAbstand))

DO UNTIL ABS(RadiusAlt - RadiusNeu) < .0001
   Radius = RadiusNeu
   CALL LaengenBerechnung(Laenge1, (Radius), (AchsenAbstand))
   IF Laenge1 < Laenge0 THEN
      PRINT "Der Radius"; Radius; "ist zu klein!"
      RadiusNeu = Radius + ABS(Radius - RadiusAlt) / 2
      RadiusAlt = Radius
   ELSEIF Laenge1 > Laenge0 THEN
      PRINT "Der Radius"; Radius; "ist zu groß!"
      RadiusNeu = Radius - ABS(Radius - RadiusAlt) / 2
      RadiusAlt = Radius
   END IF
LOOP

REM Ausgabe (auf drei Dezimalen gerundet):
PRINT : PRINT
Radius = INT(Radius * 1000 + .5) / 1000
PRINT "Gesuchter Radius ="; Radius; "mm beim kleinen"
PRINT "             und ="; Radius * 2; "mm beim großen Rad."

END


FUNCTION ArcusCosinus (x)
ArcusCosinus = ATN(SQR(1 - x * x) / x)
END FUNCTION


SUB LaengenBerechnung (Laenge1, Radius, AchsenAbstand)
        REM Berechnung der Länge des Riemens in Ab-
        REM hängigkeit von AchsenAbstand und Radius
        REM unter der Annahme "Übersetzung" = 2 : 1
Winkel = ArcusCosinus(Radius / AchsenAbstand)
GrosserTeilKreis = (4 * Radius * PI / 360) * (360 - 2 * Winkel * 180 / PI)
KleinerTeilKreis = (2 * Radius * PI / 360) * (2 * Winkel * 180 / PI)
ZweiGeradenStrecken = 2 * Radius * TAN(Winkel)
Laenge1 = GrosserTeilKreis + ZweiGeradenStrecken + KleinerTeilKreis
END SUB
```

KNOBELEI 2 aus der ORF-Computerbox

Aus der ORF-COMPUTERBOX (Teletext 464) vom 13. 08. 1998.

Aufgabenstellung:

Herr Martin Hofmaier aus Lunz am See stellt folgende Aufgabe:
Ein Blumenbeet soll folgende Form erhalten: an zwei gegenüberliegenden Seiten eines Rechteckes werden Halbkreise, an den beiden anderen Seiten Quadrate angefügt. Die äußere Begrenzung soll 10 * Pi m lang sein. Welche Abmessungen muss das Beet bekommen, damit möglichst WENIG Gartenfläche verbraucht wird.
(Ihr Programm zur Berechnung der Abmessungen und der Fläche senden Sie bitte an den ORF Text, Computerbox, 1136 Wien. Einsendeschluß ist der 1. 9. 1998.)

Anmerkung: Die Aufgabenstellung ist kein sehr gutes Beispiel für die Nützlichkeit eines Computer-Programms: Es ist recht leicht, durch Ableitung der Funktion F(a, b) – indem man die erste Ableitung gleich 0 setzt – eine exakte mathematische Lösung zu finden. Der Computer ist in diesem Fall völlig unnötig!

Überlegung:

Bei b = 10 ist a = 0. Bei fallendem b steigt das a, bis es bei b = 0 ca. den Wert a ≈ 5,2359878 erreicht. Gleichzeitig wird bei fallendem b die Fläche zunächst kleiner und dann wieder größer:

Grenzwerte; Minimum:
Bei b = 0 m ist a = $\frac{10p}{6}$ ≈ 5,2359878 m und es ergibt sich für F = 2a² ≈ 54,831136 m².

Bei b = 10 m ist a = 0 m und es ergibt sich für F = $\frac{b^2}{4} * p$ ≈ 78,539816 m².
Dazwischen gibt es ein Minimum der Fläche – für dieses ist der Wert von b (und damit auch von a) zu ermitteln.

Mein Programm dafür ist in QBasic geschrieben und leicht zu verstehen:

Der Wert von b wird – von 10 abwärts – jeweils um 1 verkleinert, bis die Fläche nicht mehr kleiner sondern wieder größer wird. Vom vorletzten Wert ausgehend wird b nun um 0,1 verkleinert, bis die Fläche nicht mehr kleiner sondern wieder größer wird. Vom vorletzten Wert ausgehend wird b nun um 0,01 verkleinert, bis .....

```
CLS : CLEAR
CONST pi# = 3.14159265#
b1# = 0: b2# = 0: d# = 10
DO UNTIL d# < .000000001#
   b2# = b1# + d#
   d# = d# / 10
   DO
      b1# = b2#
      a1# = (10 - b1#) * pi# / 6
      b2# = b1# - d#
      a2# = (10 - b2#) * pi# / 6
      F1# = a1# * b1# + 2 * a1# * a1# + b1# * b1# * pi# / 4
      F2# = a2# * b2# + 2 * a2# * a2# + b2# * b2# * pi# / 4
      u# = 6 * a1# + b1# * pi#
      LOCATE 10, 10: PRINT "b ="; INT(b1# * 100000000) / 100000000;
      LOCATE 11, 10: PRINT "d = "; d#;
      LOCATE 12, 10: PRINT "a ="; a1#;
      LOCATE 14, 10: PRINT "F ="; F1#
      LOCATE 15, 10: PRINT "u ="; INT(u# * 100000000) / 100000000
      FOR i = 1 TO 10000: NEXT i
      IF F2# >= F1# THEN EXIT DO
   LOOP
LOOP
```

Anmerkung: Bei einer ersten Analyse der Funktion F(a, b) – etwa mit Hilfe des untenstehenden Prográmmchens – kommt man leicht auf die Vermutung, das Minimum der Fläche müsse dann erreicht sein, wenn das Rechteck ein Quadrat, wenn also a = b ist. Diese Vermutung ist zunächst naheliegend aber falsch. Das Minimum liegt – laut den beiden Ableitungen – bei a = $\frac{10p}{3+2p}$ und b = $\frac{20p-30}{3+2p}$;

wäre a = b, dann müsste (wegen 6a + bπ = 10π) gelten a = b = $\frac{10p}{6+p} \approx 3{,}4365923$ m

und es wäre dann die Fläche mit F = $a^2 * (3+\frac{p}{4}) \approx 44{,}706182$ m² etwas größer als das Minimum von F ≈ 44,698063 m².

```
REM Hilfsprogramm zur ersten Untersuchung
REM des Verhaltens der Funktion F(a, b):
CLS : CLEAR
CONST pi = 3.14159
b = 10
FOR b = 0 TO 10 STEP .1
   a = (10 - b) * pi / 6
   F = a * b + 2 * a * a + b * b * pi / 4
   LPRINT TAB(10); "b ="; b;
   LPRINT TAB(20); "   a ="; a;
   LPRINT TAB(40); "   F ="; F
NEXT b
```

KNOBELEI 3 aus der ORF-Computerbox

Ferien-Programm: ORF Teletext, Computerbox (S. 465), 16. 07. 1998:

Das folgende BASIC-Programm **Dat WTag.bas** von Christian Krupica aus 3143 Pyhra berechnet nach Eingabe eines Datums den zugehörigen Wochentag.Verbesserungen (Strukturierung) von W. Gutenbrunner, Juli 1998.

```
DECLARE SUB Intro ()
DECLARE SUB TestSchaltJahr (Jahr&, SchaltJahr%)
DECLARE SUB Eingabe (Tag%, Monat%, Jahr&)
DECLARE SUB UeberPruefen (Tag%, Monat%, Jahr&, Test%)
DECLARE SUB Berechnung (Tag%, Monat%, Jahr&, WochenTag%)
DECLARE SUB Ausgabe (WochenTag%)
CLS: SCREEN 8
CALL Intro
Antwort$ = ""
DO UNTIL Antwort$ = "n" OR Antwort$ = "N"
   Test% = 0
   DO WHILE Test% = 0
      CALL Eingabe(Tag%, Monat%, Jahr&)
      CALL UeberPruefen((Tag%), (Monat%), (Jahr&), Test%)
   LOOP
   CALL Berechnung((Tag%), (Monat%), (Jahr&), WochenTag%)
   CALL Ausgabe((WochenTag%))
   LOCATE 23, 40: COLOR 7
   PRINT "Noch einmal (j/n)? ": Antwort$ = ""
   DO UNTIL Antwort$ = "j" OR Antwort$ = "J" OR Antwort$ = "n" OR Antwort$ = "N"
      Antwort$ = INKEY$
   LOOP
LOOP
CLS: END
```

```
SUB Intro
   RANDOMIZE TIMER
   FOR Zaehler& = 1 TO 1000
      COLOR 15
      LOCATE 10, 22: PRINT "  Dieses Programm errechnet Ihnen"
      LOCATE 12, 22: PRINT "bei beliebiger Eingabe eines Datums"
      LOCATE 14, 22: PRINT "    den dazugehörigen Wochentag!"
      COLOR 13
      LOCATE 17, 29: PRINT "Christian Krupica 1996"
      LOCATE 18, 29: PRINT "W. Gutenbrunner, 07/98"
      COLOR 2
      x% = INT(RND * 620)
      y% = INT(RND * 200)
      PSET (x%, y%)
   NEXT Zaehler&
END SUB


SUB TestSchaltJahr (Jahr&, SchaltJahr%)
   SchaltJahr% = 0
   IF Jahr& / 4 = INT(Jahr& / 4) THEN SchaltJahr% = 1
   IF Jahr& / 100 = INT(Jahr& / 100) THEN SchaltJahr% = 0
   IF Jahr& / 400 = INT(Jahr& / 400) THEN SchaltJahr% = 1
   IF Jahr& = 0 THEN SchaltJahr% = 0
END SUB
```

```
SUB Eingabe (Tag%, Monat%, Jahr&)
   CLS : COLOR 14
   Tag% = 0
   DO WHILE (Tag% < 1) OR (Tag% > 31)
      LOCATE 8, 30: PRINT "                "
      LOCATE 8, 30: INPUT "Tag: ", Tag%
      IF Tag% > 31 THEN
         LOCATE 18, 20: PRINT "Mehr als 31 Tage pro Monat gibt es nicht!"
         FOR Zaehler& = 1 TO 1000000: NEXT Zaehler&
         LOCATE 18, 20: PRINT "                                          "
      END IF
   LOOP
   Monat% = 0
   DO WHILE (Monat% < 1) OR (Monat% > 12)
      LOCATE 10, 28: PRINT "                "
      LOCATE 10, 28: INPUT "Monat: ", Monat%
      IF Monat% > 12 THEN
         LOCATE 18, 20: PRINT "Es gibt nur 12 Monate!"
         FOR Zaehler& = 1 TO 1000000: NEXT Zaehler&
         LOCATE 18, 20: PRINT "                          "
      END IF
   LOOP
   Jahr& = 0
   DO WHILE (Jahr& < 1)
      LOCATE 12, 29: PRINT "                "
      LOCATE 12, 29: INPUT "Jahr: ", Jahr&
      IF Jahr& < 1582 THEN
         LOCATE 18, 20: PRINT "Der gregorianische Kalender wurde"
         LOCATE 19, 20: PRINT "erst im J. d. H. 1582 eingeführt!"
         FOR Zaehler& = 1 TO 1000000: NEXT Zaehler&
         LOCATE 18, 20: PRINT "                                          "
         LOCATE 19, 20: PRINT "                                          "
```

```
         Jahr& = 0
      ELSEIF Jahr& > 4000 THEN
         LOCATE 18, 20: PRINT "So weit in die Zukunft?"
         LOCATE 19, 20: PRINT "Ist das noch sinnvoll???"
         FOR Zaehler& = 1 TO 1000000: NEXT Zaehler&
         LOCATE 18, 20: PRINT "                                        "
         LOCATE 19, 20: PRINT "                                        "
         Jahr& = 0
      END IF
   LOOP
END SUB


SUB Berechnung (Tag%, Monat%, Jahr&, WochenTag%)
   CALL TestSchaltJahr((Jahr&), SchaltJahr%)
   IF Monat% = 1 THEN LET TageFuerVormonate% = 0
   IF Monat% = 2 THEN LET TageFuerVormonate% = 31
   IF Monat% = 3 THEN LET TageFuerVormonate% = 59
   IF Monat% = 4 THEN LET TageFuerVormonate% = 90
   IF Monat% = 5 THEN LET TageFuerVormonate% = 120
   IF Monat% = 6 THEN LET TageFuerVormonate% = 151
   IF Monat% = 7 THEN LET TageFuerVormonate% = 181
   IF Monat% = 8 THEN LET TageFuerVormonate% = 212
   IF Monat% = 9 THEN LET TageFuerVormonate% = 243
   IF Monat% = 10 THEN LET TageFuerVormonate% = 273
   IF Monat% = 11 THEN LET TageFuerVormonate% = 304
   IF Monat% = 12 THEN LET TageFuerVormonate% = 334
   VorTageImJahr% = Tag% - 1 + TageFuerVormonate%
   TageSeit0& = (365 * Jahr&) + VorTageImJahr% + INT(Jahr& / 4) - INT(Jahr& / 100) +
                                                                  INT(Jahr& / 400)
   IF VorTageImJahr% < 60 THEN
      IF SchaltJahr% = 1 THEN TageSeit0& = TageSeit0& - 1
```

```
   END IF
   WochenSeit0! = TageSeit0& / 7
   GanzeWochenSeit0& = INT(WochenSeit0!)
   Differenz! = WochenSeit0! - GanzeWochenSeit0&
   ZusaetzlicheTage% = INT((Differenz! * 7) - .1) + 1
   COLOR 10, 0
   LOCATE 18, 20: PRINT "Seit Beginn unserer Zeitrechnung vergingen"
   LOCATE 19, 20: PRINT "bis vor das angegebene Datum:"; TageSeit0&; "Tage,"
   LOCATE 20, 20: PRINT "das sind:"; GanzeWochenSeit0&; "Wochen und";
                                                    ZusaetzlicheTage%; "Tag(e)."
   WochenTag% = ZusaetzlicheTage%
END SUB


SUB UeberPruefen (Tag%, Monat%, Jahr&, Test%)
   CALL TestSchaltJahr((Jahr&), SchaltJahr%)
   IF SchaltJahr% = 1 THEN LOCATE 12, 45: PRINT "(Schaltjahr)"
   IF SchaltJahr% = 0 THEN LOCATE 12, 45: PRINT "(Kein Schaltjahr)"
   IF Tag% > 30 AND (Monat% = 4 OR Monat% = 6 OR Monat% = 9 OR Monat% = 10) THEN
      LOCATE 18, 20: PRINT "Dieser Monat hat nur 30 Tage!"
      FOR Zaehler& = 1 TO 1000000: NEXT Zaehler&
      Test% = 0
   ELSEIF Tag% > 29 AND Monat% = 2 THEN
      LOCATE 18, 20: PRINT "Der Februar hat maximal 29 Tage!"
      FOR Zaehler& = 1 TO 1000000: NEXT Zaehler&
      Test% = 0
   ELSEIF Tag% = 29 AND Monat% = 2 AND SchaltJahr% = 0 THEN
      LOCATE 18, 20: PRINT "Dieses Jahr ist kein Schaltjahr!"
      FOR Zaehler& = 1 TO 1000000: NEXT Zaehler&
      Test% = 0
   ELSE Test% = 1
   END IF
```

```
END SUB

SUB Ausgabe (WochenTag%)
   COLOR 11
   LOCATE 1, 13: PRINT "  Es werden die Tage (die Wochen + zusätzlichen Tage)"
   LOCATE 2, 13: PRINT " berechnet, die seit Beginn unserer Zeitrechnung ver-"
   LOCATE 3, 13: PRINT " gangen sind. Dabei wird angenommen, dass - würde man"
   LOCATE 4, 13: PRINT " nach unserem Kalender zurückrechnen - der 1. 1. 0 ein"
   LOCATE 5, 13: PRINT "     Sonntag (und das Jahr 0 kein Schaltjahr) sei."
   COLOR 12: LOCATE 15, 35
   IF WochenTag% = 0 THEN PRINT "Sonntag"
   IF WochenTag% = 1 THEN PRINT "Montag"
   IF WochenTag% = 2 THEN PRINT "Dienstag"
   IF WochenTag% = 3 THEN PRINT "Mittwoch"
   IF WochenTag% = 4 THEN PRINT "Donnerstag"
   IF WochenTag% = 5 THEN PRINT "Freitag"
   IF WochenTag% = 6 OR WochenTag% = -1 THEN PRINT "Samstag"
   FOR Zaehler& = 1 TO 50: SOUND 50 + RND * 2000, .2: NEXT Zaehler&
END SUB
```

Programmiersprachen

MENSCH	SOFTWARE	HARDWARE
Mensch	**Programm**	**Maschine**
	Programme helfen dem Menschen, sich die Maschine nutzbar zu machen. Sie sollen möglichst „benutzerfreundlich“ sein: menügesteuert, deutsch, mit „Hilfen“ versehen, fehlerlos, „absturzsicher“, ...	
Sprache	**Programmiersprache**	**Maschinensprache**
Der Mensch versteht deutsch, englisch, ...	**Programmiersprachen** sind „Übersetzungs-programme“: sie übersetzen die Befehle des Menschen in die Sprache des Computers. Z. B.: FORTRAN, COBOL, ALGOL, **BASIC**, **LOGO**, **PASCAL**, FORTH, C, PL/1	Der Computer „versteht“ eigentlich nur **Maschinensprache:** Das sieht so aus: 0100 1101 1101 1011