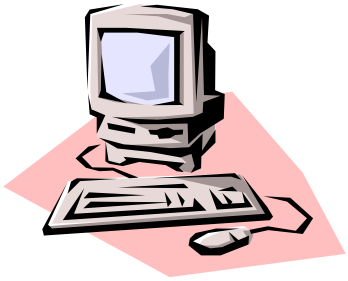
Einführung in

Quick BASIC 4.5



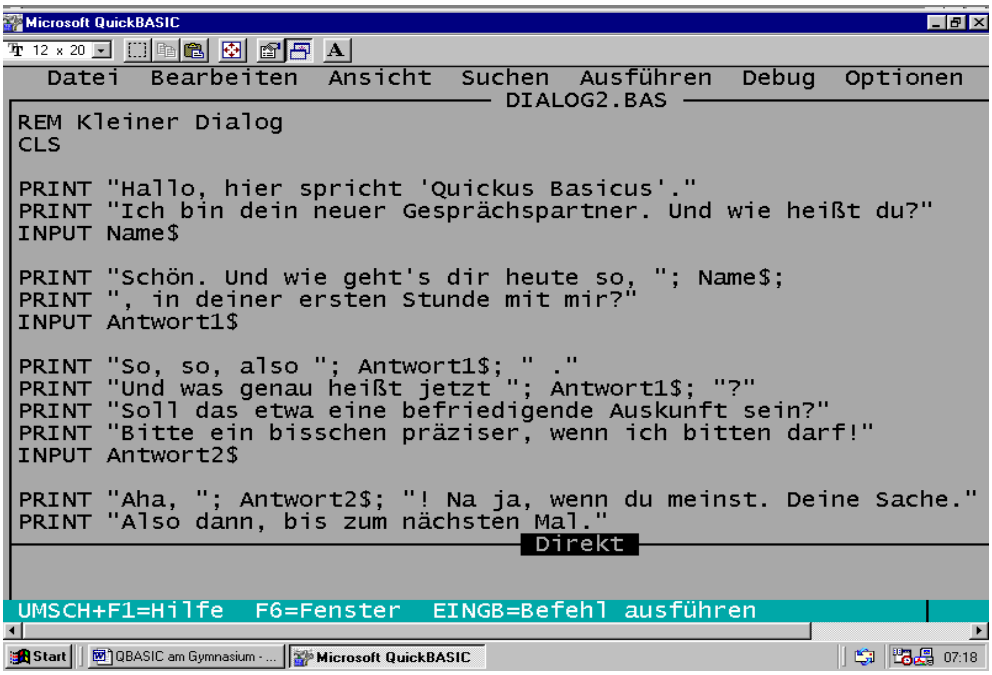
Inhalt:

0. Einführung:
   - einfaches Programmbeispiel
   - komplexeres Programmbeispel

1. Einfache Mathematik
2. Kommentare
3. Bildschirmausgaben
4. Entscheidungen mit IF ... THEN
5. Zahlenraten
6. Schleifen
7. Schleifen mit FOR-NEXT
8. Weitere Schleifen mit FOR-NEXT
9. Schleifen mit WHILE-WEND
10. Schleifen mit DO-LOOP
11. Subroutinen (Unterprogramme)
12. Subs und Functions
13. Erzeugen von Subs und Functions
14. Verwenden von programminternen Daten
15. Ausdrücke
16. Strings (Zeichenketten)
17. Verändern von Zeichenketten
18. Weitere Zeichenkettenfunktionen
19. Spezielle Zeichenkettenfunktionen
20. Anwendung mehrerer Zeichenkettenfunktionen
21. Rechnen mit Zeichenketten
22. Zahlvariablen
23. Der Bildschirm im Textmodus
24. Bildschirmausgaben im Textmodus
25. Bildschirmausgaben mit Tabulatoren
26. Bewegungen auf dem Bildschirm
27. Bildschirmausgaben im Grafikmodus
28. Komplexe Figuren im Grafikmodus
29. Das Achsenkreuz
30. Vereinfachtes Setzen von Punkten im Achsenkreuz
31. Zeichnen von Linien im Achsenkreuz
32. Zeichnen von Rechtecken
33. Zeichnen von Kreisen und Ellipsen
34. Grafikbeispiel
35. Programmsteuerung mit SELECT...CASE
36. Töne mit qBASIC - der SOUND-Befehl
37. Töne mit qBASIC - der PLAY-Befehl
38. Grafische Figuren - der DRAW-Befehl
39. Figuren zeichnen und löschen
40. Bewegte Figuren

0. Einführung

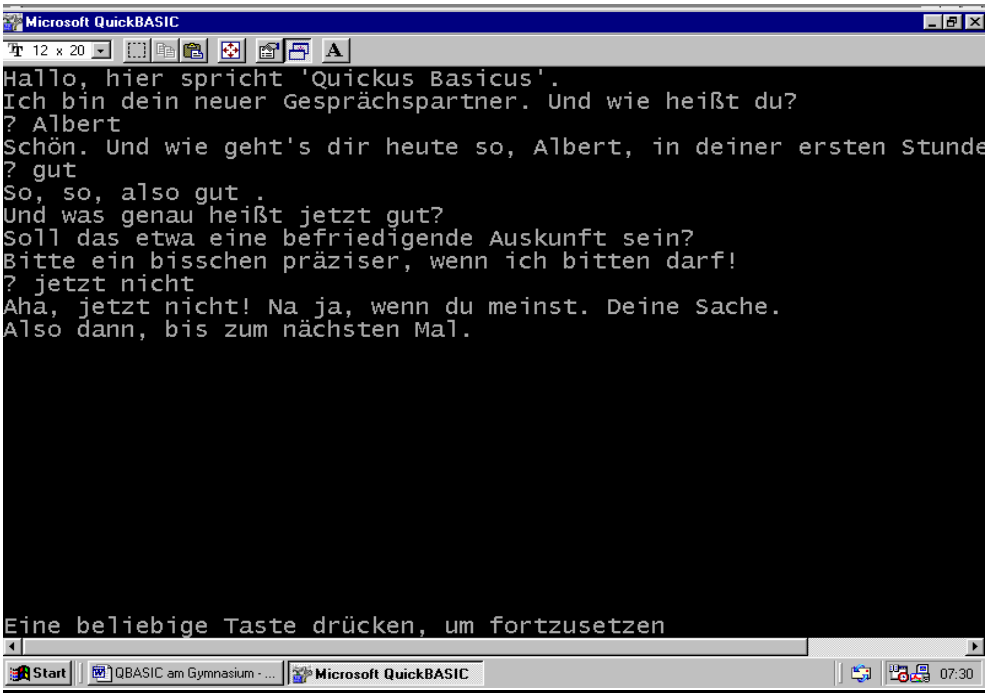
a. Einfaches Programmbeispiel



Eingabe möglicher Antworten:

- Antwort a: Albert
- Antwort b: gut
- Antwort c: jetzt nicht

Resultat auf Ausgabebildschirm



b. komplexeres Programmbeispiel

```
REM farbige Rechtecke

RANDOMIZE TIMER
SCREEN 12
CLS

FOR rechteck = 1 TO 20

LET breite = INT(RND * 80) + 20
LET hoehe = INT(RND * 80) + 20

LET x1 = INT(RND * (640 - breite))
LET y1 = INT(RND * (480 - hoehe))
LET farbe = INT(RND * 15) + 1
LINE (x1, y1)-(x1 + breite, y1 + hoehe), farbe, B

NEXT rechteck
a$ = ""

WHILE a$ = ""
a$ = INKEY$
WEND

PRINT
```

Resultat:

Nach dem Startbefehl „F 5“ (oder „run“) erscheinen zwölf Rechtecke auf dem Bildschirm, die bei wiederholtem Programmdurchauf jeweils neu positioniert sind.

Ihre verschiedenen Breiten und Höhen werden im Rahmen der definierten Größenangaben ebenfalls immer neu bestimmt,

wie auch die Erzeugung ihrer verschieden Farben dem Zufallsgenerator überlassen bleibt.

Aufgabe: Bitte ausprobieren!

1. Einfache Arithmetik

1. Starte qBASIC und tippe in das Programmfenster die folgenden drei Zeilen ein:

```
LET a=10
LET b=15
PRINT a+b
```

Starte das Programm mit Ausführen - Start oder Shift+F5. Das Programm reserviert zunächst zwei Variablen (a und b), weist diesen Variablen zwei Werte zu (LET-Befehl) und gibt schließlich auf dem Bildschirm die Summe der zwei Variablen (25) aus.

2. Wechsle in das Direktfenster (Fenster mit der Maus anklicken oder Taste F6). Tippe dort den Direktbefehl

```
PRINT a, b
```

ein. Wie Du siehst (es werden die Zahlen 10 und 15 ausgegeben), sind die Variablen noch da, obwohl das Programm fertig ist.

3. Bewege die Schreibmarke an das Ende der Zeile LET b=15. Erzeuge eine Leerzeile, indem Du dort die Eingabetaste drückst. Schreibe in diese leere Zeile den Befehl LET b=5. Starte das Programm erneut und beobachte die Bildschirmanzeige. Die Summe beträgt dieses Mal 15. Der Variablen b wird zunächst der Wert 15 zugewiesen. Dieser Wert wird jedoch sofort wieder zu 5 geändert, so dass letzten Endes die Zahlen 10 und 5 addiert werden (Summe: 15). Lösche jetzt die Zeile LET b=15.

Als Ergebnis bleibt, dass man die Werte der einzelnen Variablen jederzeit während des Programmlaufs ändern kann.

4. Füge an das Ende des Programmes die folgenden Zeilen an:

```
PRINT a-b
PRINT a*b
PRINT a/b
PRINT a^b
```

Nachdem Du das Programm gestartet hast, werden nacheinander die Summe (+), die Differenz (-), das Produkt (*), der Quotient (/) und die Potenz(^) ausgegeben.

Um andere Berechnungen durchführen zu können, muss jedes Mal der Programmtext geändert werden. Wähle für a und b die Werte 18 und 2 bzw. 36 und 3 und starte jeweils das Programm. Lösche dann das Programm mit Datei - Neu.

5. Brüche werden in qBASIC mit Hilfe des Divisionszeichens (/) eingegeben. Dabei sind Zähler und Nenner einzuklammern. Tippe dazu das folgende Programm ein. Versuche vorab, das Ergebnis im Kopf zu bestimmen und prüfe das Ergebnis indem Du das Programm startest.

```
PRINT 2+(6+18)/(15-3)*4-5
```

2. Kommentare

1. Starte qBASIC, lösche ein eventuell vorhandenes Programm und tippe in das Programmfenster die folgenden drei Zeilen ein:

```
REM Temperaturumrechnung
CLS
PRINT "Grad F", "Grad C"
PRINT
INPUT "Eingabe Grad F",f
PRINT f, (f-32)*5/9
```

Das Programm dient zur Umrechnung von Fahrenheit-Graden in Celsius-Grade. Die Zeile mit dem REM-Befehl wird vom Computer völlig ignoriert. REM kommt von engl. remark oder reminder (Bemerkung oder Hinweis) und ist allein dazu da, dich an das zu erinnern, was das Programm leistet. Der CLS-Befehl (Clear Screen) löscht den Bildschirm. Der INPUT-Befehl gibt auf dem Bildschirm den Text "Eingabe Grad F: " aus und wartet, bis du mit Hilfe der Tastatur eine Zahleneingabe (Abschluss mit ENTER) vorgenommen hast. Dabei wird der Variablen f der eingegebene Wert (z.B. 82.4) zugewiesen. Beachte, dass Kommazahlen mit einem Dezimalpunkt eingegeben werden. Die letzte Zeile gibt schließlich die eingegebene Fahrenheit- und die berechnete Celsius-Temperatur auf dem Bildschirm aus.

2. Zur Berechnung mehrerer Werte kann man eine sogenannte Endlos-Schleife verwenden. Ändere dazu das Programm wie folgt ab:

```
REM Temperaturumrechnung F -> C
CLS
PRINT "Grad F", "Grad C"
PRINT
Hierher:
INPUT "Eingabe Grad F: ",f
LET c=(f-32)*5/9
PRINT f, c
GOTO Hierher
```

Die Zeile Hierher: ist eine sogenannte Sprungmarke (wichtig der Doppelpunkt am Ende). Trifft der Computer auf einen GOTO-Befehl, so prüft er die nachfolgende Angabe. Handelt es sich dabei um eine ihm bekannte Sprungmarke, so setzt er seine Berechnungen nicht wie sonst in der folgenden Zeile fort, sondern direkt nach der Sprungmarke. Dort steht aber der INPUT-Befehl, was eine erneute Temperatureingabe erfordert, dann folgt die Temperatur-Umrechnung. Schließlich trifft der Computer wiederum auf die GOTO-Zeile, springt wieder zur INPUT-Zeile, usw. Dieses Programm läuft solange, bis der Computer ausgeschaltet wird. Ein reguläres Programmende ist nicht möglich. Zum gewaltsamen Programm-Abbruch kann man aber auch die Taste Strg+C oder die Taste Untbr verwenden.

3. Ändere das Programm so ab, dass nicht die Fahrenheit-Temperatur eingegeben wird, sondern die Celsius-Temperatur. Speichere das Programm in deinem Datenverzeichnis C:\INFO\AG unter dem Namen CELSIUS.BAS.

3. Bildschirmausgaben

1. Starte qBASIC, lösche ein eventuell vorhandenes Programm und tippe in das Programmfenster die folgenden drei Zeilen ein:

```
REM Hallo
CLS
INPUT "Wie heißt Du? ",vorname$
PRINT "Hallo "; vorname$; "!"
```

Starte das Programm, tippe auf die Frage Deinen Vornamen ein und beobachte die Bildschirmausgabe. Ändere dann die Strichpunkte in der letzten Zeile in Kommas ab und vergleiche mit der vorigen Bildschirmausgabe.

Statt Zahlen können beim INPUT-Befehl auch Texte (Zeichenketten) eingegeben werden. Voraussetzung: es wird eine entsprechende Variable, eine Zeichenketten- oder String-Variable verwendet (beachte das Dollarzeichen als Abschluss des Variablennamens).

Strichpunkte im PRINT-Befehl bewirken, dass die Bildschirmausgaben lückenlos aneinandergefügt werden bis die Zeile voll ist. Kommas bewirken, dass die nächste Ausgabe eine Spalte weiter erfolgt.
Diese Zeichen können auch in verschiedenen Zeilen verwendet werden, was folgendes Beispiel zeigt.

```
REM Reiten
CLS
PRINT "Blu"
PRINT "mento"
PRINT "pferde"
```

Starte das Programm und beobachte die Bildschirmausgabe. Setze dann an das Ende der ersten beiden PRINT-Zeilen ein Komma (-> Neustart) und schließlich einen Strichpunkt.

2. Wie bei einer Schreibmaschine können auch Tabulatoren verwendet werden. Da der Bildschirm in 25 Zeilen und 80 Spalten eingeteilt ist, gibt es 80 Tabulatorpositionen:

```
REM Tabulator
CLS
PRINT TAB(5); "<- Das ist Spalte 5"
PRINT TAB(10); "<- Das ist Spalte 10"
PRINT TAB(20); "<- Das ist Spalte 20"
PRINT TAB(5); "Hallo"; TAB(20); "Hey"; TAB(35); "Grüß Gott"
PRINT TAB(5); 1; TAB(20); 2; TAB(35); 3
```

Die TAB-Funktion setzt die Schreibmarke in eine bestimmte Bildschirmspalte. Die folgende Bildschirmausgabe erfolgt dann genau an dieser Stelle.

3. Schreibe mit Hilfe der TAB-Funktion ein Programm, welches Deinen Stundenplan ausgibt.

```
REM Stundenplan
CLS
PRINT TAB(5); "MO"; TAB(10); "DI"; TAB(15); "MI";      usw.
```

Speichere das fertige und lauffähige Programm unter dem Namen PLAN.BAS in Deinem Datenverzeichnis C:\INFO\AG ab.

4. Entscheidungen mit IF ... THEN

Achtung: Auch Skript 05 „Einfache Algorithmen mit QUICK BASIC“ beachten!

1. Jedes der bisherigen Programme wird vom Computer linear abgearbeitet, d.h. von Zeile zu Zeile (auch bei Sprungmarken). In der Praxis wird jedoch oftmals verlangt, dass der Computer Entscheidungen treffen kann. Die dafür notwendige Anweisung hat die Form IF... THEN....

```
REM Zahlenraten
INPUT "Ratezahl: ",a : CLS
Weiter:
INPUT "Welche Zahl habe ich mir gemerkt? ", b
IF b=a THEN GOTO Hurra
IF b<a THEN PRINT "Zu klein, noch einmal"
IF b>a THEN PRINT "Zu groß, noch einmal"
GOTO Weiter
Hurra:
PRINT "Du bist super!" : PRINT "Die Zahl war ",a,"."
```

Ein Mitspieler gibt zunächst die Ratezahl ein (keine Dezimalzahlen!). Daraufhin wird der Bildschirm sofort gelöscht. Das Raten kann beginnen. Nach Eingabe einer Zahl beginnt das Programm die Ratezahl mit der Eingabe zu vergleichen:
•Wenn die Zahlen gleich sind (IF b=a), dann gehe zur Marke Hurra (THEN GOTO Hurra), gib die Meldung aus und beende das Programm. •Wenn die Eingabe kleiner ist als die Ratezahl (IF b<a), dann gib eine entsprechende Meldung aus (THEN PRINT .) und setze den Programmablauf an der Stelle Weiter: fort (GOTO Weiter). •Wenn die Eingabe größer ist als die Ratezahl (IF b>a), dann gib eine entsprechende Meldung aus (THEN PRINT .) und setze den Programmablauf an der Stelle Weiter: fort (GOTO Weiter).
Speichere das Programm unter dem Namen RATEN1.BAS in Deinem Datenverzeichnis.

2.Nachteil des Programms ist es, dass man einen Mitspieler braucht, der die Ratezahl eingibt. Diesen Teil kann auch der Computer übernehmen. Mit Hilfe eines Zufallsgenerators können Zufallszahlen erzeugt werden.

```
REM Zufall
RANDOMIZE TIMER : LET zaehler=1
Schleife:
PRINT TAB(5); RND; TAB(30); INT(6*RND+1)
LET zaehler=zaehler+1
IF zaehler<500 THEN GOTO Schleife
PRINT "Fertig"
```

Der Befehl RANDOMIZE TIMER setzt den Zufallsgenerator auf einen bestimmten Startpunkt (abhängig von der Einschaltzeit des Computers). In zwei Spalten werden dann Zufallszahlen ausgegeben. In Spalte 5 stehen Zahlen zwischen 0 und 0,999999999999 (mit RND), in Spalte 30 stehen Zahlen von 1 bis 6 (mit INT(6*RND+1). Sollen Lottozahlen (1 bis 49) erzeugt werden, wählt man INT(49*RND+1).
Ändere damit das Programm Zahlenraten (Zahlen zwischen 1 und 100) und speichere das Programm unter RATEN2.BAS ab.

5. Zahlenraten

1. Bei dieser Programmversion sind nur maximal 20 Rateversuche möglich. Öffne die Datei RATEN2.BAS und ändere das Programm so ab, dass es wie folgt aussieht. Speichere dann das Programm unter dem Namen RATEN3.BAS ab.

```
REM verbessertes Zahlenraten
CLS
PRINT "Zahlenraten"
PRINT "-----------"
PRINT
PRINT "Ich denke mir eine Zahl zwischen 1 und 1000, die es zu"
PRINT "erraten gilt."
PRINT
RANDOMIZE TIMER
LET zaehler = 0
LET cz = INT(RND(1) * 1000) + 1
Wiederhole:
LET zaehler = zaehler + 1
PRINT zaehler; ".Zahl: ";
INPUT "", rz
IF rz < cz THEN PRINT "zu klein"
IF rz > cz THEN PRINT "zu groß"
PRINT
IF zaehler < 20 and rz <> cz THEN GOTO Wiederhole
IF rz=cz THEN
PRINT "Gewonnen ..."
ELSE
PRINT "Verloren ..."
END IF
```

Das o. a. Programm wird beendet, wenn man die Zahl errät oder wenn man bereits 20 Zahlen eingegeben hat.

2. Die IF... THEN...-Anweisung kann um die ELSE-Anweisung erweitert werden. Die englischen Begriffe, die in BASIC für diese Möglichkeit verwendet werden, lassen sich auch leicht ins Deutsche übersetzen:

WENN die im folgenden formulierte Bedingung erfüllt ist,
DANN führe die hier stehende Anweisung aus,
SONST führe die hier stehende Anweisung aus.

Beachte: Erstreckt sich die IF...THEN...-Anweisung oder die IF...THEN...ELSE-Anweisung über mehrere Zeilen, so muss die Anweisung mit END IF abgeschlossen werden.
Bsp.:

```
INPUT zahl
IF zahl<10 THEN
PRINT "Hallo"
PRINT "Die Zahl ist kleiner als 10."
ELSE
PRINT "Hi"
PRINT "Die Zahl ist größer als 9."
END IF
```

6. Schleifen

Im folgenden sollen fünf Zahlen eingegeben und addiert werden. Um dieses Vorhaben zu realisieren, gibt es verschiedene Methoden.

1. Starte qBASIC bzw. lösche ein eventuell vorhandenes Programm. Die erste Methode sieht wie folgt aus:

```
REM Addition 1
CLS
LET gesamt=0
INPUT "Zahl: ", a
LET gesamt=gesamt+ a
INPUT "Zahl: ", a
LET gesamt=gesamt+a
INPUT "Zahl: ", a
LET gesamt=gesamt+a
INPUT "Zahl: ", a
LET gesamt=gesamt+a
INPUT "Zahl: ", a
LET gesamt=gesamt+a
PRINT gesamt
```

Da einige Programmzeilen wiederholt auftreten, können diese mit Bearbeiten - Kopieren und Bearbeiten - Einfügen erzeugt werden und müssen nicht jedes Mal neu getippt werden. Starte das Programm, und addiere damit die Zahlen 23, 56, 78, 99 und 204. Die Summe ist: ..

2. Trotz der Erleichterung durch Kopieren / Einfügen kann das die richtige Programmierpraxis nicht sein. Bei fünf Zahlen mag das ja noch angehen, bei hundert oder tausend Zahlen jedoch nicht mehr. Es gibt einen viel besseren Weg. Man stellt eine Variable auf, die bis 5 zählt und hält das Programm dann an:

```
REM Addition 2
CLS
LET gesamt=0
LET zaehler=1
Marke:
PRINT "Zahl"; zaehler;": ";
INPUT a
LET gesamt=gesamt+a
LET zaehler=zaehler+1
IF zaehler<=5 THEN GOTO Marke
PRINT gesamt
```

Addiere mit diesem Programm die selben Zahlen wie zuvor und vergleiche die Summen. Die Anzahl der zu addierenden Zahlen lässt sich mit dieser Methode sehr leicht ändern, indem man

IF zaehler <= 5 ...

ändert zu

IF zaehler <= 100 ... .

Diese Zählweise ist so praktisch, dass es zwei eigene Befehle gibt: den FOR-Befehl und den NEXT-
Befehl. Sie werden stets gemeinsam verwendet.

7. Schleifen mit FOR ... NEXT

Mit Schleifen können Befehle oder ganze Programmteile so oft wiederholt werden, wie dies notwendig erscheint. Schleifen zählen zu den wichtigsten Programmierhilfsmitteln, die es gibt.

Je nach Aufgabenstellung verwendet man die unterschiedlichsten Schleifen-Konstruktionen; eine der einfachsten ist die FOR...NEXT-Schleife. Die FOR...NEXT-Schleife kann immer dann eingesetzt werden, wenn die Anzahl der Wiederholungen schon im voraus feststeht.

1. Ändere das vorige Programm wie folgt ab. Dieses Programm leistet genau dasselbe wie das Programm Addition 2.

```
REM Addition 3
CLS
LET gesamt=0
FOR zaehler=1 TO 5
PRINT "Zahl";zaehler;": ";
INPUT a
LET gesamt=gesamt+a
NEXT zaehler
PRINT gesamt
```

In der FOR-Zeile wird der Anfangs- und der Endwert für die Zählvariable gesetzt. In der NEXT-Zeile wird die Zählvariable automatisch um 1 erhöht, wird geprüft, ob der Endwert schon erreicht wurde und falls nicht, wird der Programmteil zwischen FOR und NEXT erneut abgearbeitet.

2. Soll die Zählvariable um einen anderen als den voreingestellten Wert 1 erhöht werden, verwendet man den STEP-Zusatz. Was bewirkt das folgende Programm?

```
REM Schleife 1
CLS
FOR zaehler=1 TO 20 STEP 2
PRINT zaehler;", ";
NEXT zaehler
PRINT
```

Verwende anschließend auch andere STEP-Werte wie 3, 4, 8 oder 1.5. Beachte, dass auch Kommazahlen als STEP-Werte verwendet werden können und dass der Endwert durch die Zählvariable nicht genau getroffen werden muss.

8. Weitere Schleifen mit FOR ... NEXT

1.Ändere das Programm 'Schleife 1' so ab, dass es das folgende Aussehen hat.

```
REM Schleife 2
CLS
FOR zaehler=10 TO 1
  PRINT zaehler;", ";
NEXT zaehler
PRINT
```

Warum funktioniert das Programm nicht?

2.Füge an Zeile 3 des vorigen Programmes den Zusatz STEP -1 an und starte das Programm erneut.

```
REM Schleife 3
CLS
FOR zaehler=10 TO 1 STEP -1
  PRINT zaehler;", ";
NEXT zaehler
PRINT
```

Das Programm gibt jetzt die Zahlen von 1 bis 10 in umgekehrter Reihenfolge korrekt aus.

3. Betrachte das folgende Programm und versuche vorherzusagen, was das Programm leistet. Wie sieht die Bildschirmausgabe aus?

```
REM Domino
CLS
FOR m=0 TO 6
  FOR n=0 TO m
    PRINT m;":";n;" ";
  NEXT n
  PRINT
NEXT m
PRINT
```

Das Programm besteht aus zwei Schleifen. Schleife 1 verwendet die Zählvariable m, Schleife 2 die Zählvariable n. Beide Schleifen sind ineinander verschachtelt. Beachte, dass sich die n-Schleife völlig innerhalb der m-Schleife befindet.

9. Schleifen mit WHILE-WEND

Eine sehr einfache Schleifenkonstruktion ist auch die WHILE...WEND-Schleife. Die WHILE...WEND-Schleife kann eingesetzt werden, wenn die Anzahl der Wiederholungen nicht schon im voraus feststeht.

1.Die folgende WHILE...WEND-Schleife bewirkt das gleiche, wie eine FOR...NEXT-Schleife.

```
REM WW1
CLS
zaehler = 1
Ende = 10
WHILE zaehler <= Ende
  PRINT zaehler; " ";
  LET zaehler = zaehler + 1
WEND
PRINT
```

2.Die folgende WHILE...WEND-Schleife liest durch den Befehl READ aus den DATA-Zeilen kurze Texte ein. Die Anzahl der einzulesenden Wörter ist nicht im voraus festgelegt, sie kann jederzeit verändert werden (prüfe dies!). Wichtig ist, dass als letztes Wort ein leeres Wort ("") eingelesen wird, weil nur dann die Schleife beendet wird.

```
REM WW2
CLS
RESTORE TextDaten
LET text$ = "*"
WHILE text$ <> ""
  READ text$
  PRINT text$; " ";
WEND
PRINT
TextDaten:
DATA "Ich","und","Du,","Müller's","Esel","der","bist","Du!"
DATA ""
```

3.Mit dem Befehl RESTORE kann der Lesezeiger für den READ-Befehl auf eine beliebige Programmstelle umgelenkt werden.

Ändere im obigen Programm die Zeile mit RESTORE zu RESTORE Sender und füge an das Programm die folgenden Zeilen an:

```
Sender:
DATA ARD,ZDF,S3,RTL,RTL2,SAT 1,PRO 7,Kabelkanal,Premiere
DATA ""
```

10. Schleifen mit DO ... LOOP

Der mächtigste Schleifen-Typ ist die DO...LOOP-Schleife. Bei DO...LOOP-Schleifen kann die Abbruchbedingung zu Beginn der Schleife (z.B. DO WHILE a<4 oder DO UNTIL a$="Ende") oder am Ende der Schleife (z.B. LOOP WHILE a<7 oder LOOP UNTIL a$<>"Ende") geprüft werden. Eine DO...LOOP-Schleife kann auch im Inneren der Schleife verlassen werden, z.B. durch IF a=9 THEN EXIT DO.

1.Das folgende Programm verwendet den Grafikmodus (SCREEN 12). Der Bildschirm ist dabei in 640 mal 480 Bildpunkte (Pixel) aufgeteilt. Die obere linke Ecke hat die Koordinaten 0,0; die untere rechte Ecke die Koordinaten 639,479.
Ein Punkt kann gesetzt werden mit Hilfe des Befehls PSET (x,y),farbe.

```
REM Stars
CLS
SCREEN 12
LINE (0, 0)-(639, 459), 15, B
RANDOMIZE TIMER
sterne = 0
maxSterne=5000
DO
  sterne = sterne + 1
  x = 2 + RND * 636
  y = 2 + RND * 456
  farbe = 1 + RND * 15
  PSET (x, y), farbe
LOOP UNTIL sterne > maxSterne
END
```

2.Das folgende Programm zeigt alle vier Möglichkeiten der DO...LOOP-Schleife.

```
REM DL 1
CLS
SCREEN 12
x = 1
DO
  PSET (x, 0), 1 : x = x + 1
LOOP WHILE x < 639
INPUT nix$
x = 1
DO
  PSET (x, 459), 4 : x = x + 1
LOOP UNTIL x > 638
INPUT nix$
y = 1
DO WHILE y < 458
  PSET (0, y), 2 : y = y + 1
LOOP
INPUT nix$
y = 1
DO UNTIL y > 457
  PSET (639, y), 3 : y = y + 1
LOOP
```

11. Subroutinen (Unterprogramme)

Manchmal haben verschiedene Teile eines Programms ganz ähnliche Aufgaben zu bewältigen. Normalerweise müßte man dann dieselben Zeilen zweimal oder noch öfter eingeben. Das ist aber nicht notwendig. Die Zeilen können einmal in einer sogenannten Subroutine (oder auch Unterprogramm) eingegeben werden und dann überall im Programm verwendet werden, ohne dass sie ein zweites Mal getippt werden müssen.
Dazu benutzt man z.B. die Anweisungen GOSUB (GO to SUB routine = gehe zum Unterprogramm) und RETURN.

1. Das folgende Beispiel besteht aus einem Hauptprogramm und einem Unterprogramm, das vom Hauptprogramm aus mehrmals aufgerufen wird. Wichtig ist der END-Befehl, der das Hauptprogramm beendet und verhindert, dass der Computer ohne GOSUB-Aufruf das Unterprogramm ausführt. Beachte, dass der Name des Unterprogramms (Linie) mit einem Doppelpunkt endet.

```
REM Sub1
Hauptprogramm:
CLS
zaehler=0
PRINT "Ich und du,"
GOSUB Linie
PRINT "Müllers Kuh,"
GOSUB Linie
PRINT "Müllers Esel,"
GOSUB Linie
PRINT "das bist du."
GOSUB Linie
PRINT "Das Unterprogramm wurde ";zaehler;"-mal aufgerufen"
END

Linie:
zaehler=zaehler+1
PRINT "----------"; zaehler
RETURN
```

2.Das Beispiel Tausche1 vertauscht zwei Variablen z1 und z2.

```
REM Tausche1
CLS
LET z1=4
LET z2=6
IF z1>z2 THEN GOSUB Tausche
LET z1=8
LET z2=5
IF z1>z2 THEN GOSUB Tausche
END

Tausche:
PRINT "Ich tausche ";z1; " und "; z2
z=z1:z1=z2:z2=z
RETURN
```

12. SUBs und FUNCTIONs

In qBASIC gibt es noch eine weitere Möglichkeit, Unterprogramme zu verwenden. Man unterscheidet zwischen Subs und Functions. Subs und Functions erscheinen nicht im eigentlichen Programmlisting, für beide wird ein eigenes Fenster erzeugt. Das Anlegen einer Sub oder einer Function erfolgt über das Menü Bearbeiten - Neue Sub ... oder Bearbeiten - Neue Function ...
Daraufhin öffnet sich eine Dialogbox, in der der Name der Sub bzw. Function gewählt werden kann. Schließlich wird zum Sub- bzw. Function-Fenster umgeschaltet. Der Rahmen für eine neue Sub

```
SUB Name_der_Sub
END SUB
```

bzw. für eine neue Function

```
FUNCTION Name_der_Function
END FUNCTION
```

wird vorgegeben. Zwischen SUB und END SUB bzw. FUNCTION und END FUNCTION wird dann der Programmcode eingegeben. Zwischen den Subs, Functions und dem Hauptmodul des Programms kann mit Ansicht - SUBs ... oder der F2-Taste umgeschaltet werden.

1.Das Beispiel ggT1 berechnet mit einer Function den größten gemeinsamen Teiler (ggT) zweier Zahlen. Außerdem werden mittels der Sub Tausche zwei Variablen der Größe nach sortiert. Wähle zuerst Datei - Neu, um ein neues Programm zu beginnen. Gib dann das Hauptmodul des Programmes ein.

```
REM ggT1
CLS
PRINT "ggT"
PRINT "--------------------"
INPUT "Zahl 1: ",z1
INPUT "Zahl 2: ",z2
PRINT "--------------------"
IF z1>z2 THEN Tausche z1,z2
PRINT "ggT("; z1; ","; z2 ; ")=";
PRINT ggt(z1,z2)
PRINT
END
```

Speichere das Programm an dieser Stelle unter dem Namen GGT1.BAS in Deinem Datenverzeichnis ab.

Wird das Programm an dieser Stelle mit <Shift> F5 gestartet, so meldet das Programm einen Fehler, weil sowohl die Sub als auch die Function noch nicht definiert sind. Für qBASIC sind "Tausche" und "ggT" (noch) unbekannte Befehle oder Variablen.

13. Erzeugen von SUBs und FUNCTIONs

1. Öffne gegebenenfalls die Datei GGT1.BAS. Wähle im Menü Bearbeiten den Punkt Neue Sub .... In der folgenden Dialogbox trägst du bei Name den Text "Tausche" ein. Das Programm schaltet um zum Modul Tausche.
Gib dann den folgenden Programmcode ein:

```
SUB Tausche (x,y)
  merke=x
  x=y
  y=merke
END SUB
```

Wähle im Menü Bearbeiten den Punkt Neue Function .... In der folgenden Dialogbox trägst du bei Name den Text "ggT" ein. Das Programm schaltet um zum Modul ggT.
Gib dann den folgenden Programmcode ein:

```
FUNCTION ggT (x,y)
  DO
    rest = INT(y / x)
    w = rest * x
    rest = y - w
    result = x
    IF rest < 1 THEN EXIT DO
    y = x
    x = rest
  LOOP
  ggT=result
END FUNCTION
```

Speichere das Programm an dieser Stelle unter dem Namen GGT2.BAS in Deinem Datenverzeichnis ab. Das Programm kann jetzt mit <Shift> F5 gestartet werden.

2.Schreibe ein Programm zur Berechnung des kleinsten gemeinsamen Vielfachen (kgV) zweier Zahlen. Das kgV der Zahlen a und b wird berechnet, indem man die Zahlen a und b multipliziert und das Ergebnis durch den ggT der Zahlen a und b dividiert.
Hinweise:
•Öffne gegebenenfalls die Datei GGT2.BAS und speichere sie unter dem Namen KGV1.BAS ab.
•Wähle im Menü Bearbeiten den Punkt Neue Function .... In der folgenden Dialogbox trägst du bei Name den Text "kgV" ein. Das Programm schaltet um zum Modul kgV.
•Das Modul kgV benötigt als Parameter zwei Variablen a und b. Der Modulkopf sieht also wie folgt aus:

```
FUNCTION kgV(a,b)
```

•Die Berechnung erfolgt nach:

```
kgV = a * b / ggT(a,b)
```

•Ändere die übrigen Teile des Hauptmoduls und speichere das Programm.

14. Verwenden von programminternen Daten

In einigen vorherigen Programmen sind Daten direkt mit dem INPUT-Befehl in den Computer eingegeben worden. Manchmal wäre das sehr mühsam, vor allem dann, wenn bei jedem Programmlauf viele Daten sich wiederholen. Mit den Befehlen READ, DATA und RESTORE kann man viel Zeit sparen.

1.Starte gegebenenfalls qBASIC und gib das folgende Programm ein.

```
REM Read1
CLS
READ a,b,c
PRINT a,b,c
END
DATA 10,20,30
```

Auf den READ-Befehl folgen eine Reihe von Variablen, die durch Kommas getrennt sind. Der Computer holt sich die Werte für diese Variablen aus der oder den DATA-Zeile(n).

2.Mehrere Variablen-Werte lassen sich mit Hilfe von Schleifen recht einfach einlesen, wie das folgende Beispiel zeigt.

```
REM Read2
CLS
FOR n=1 TO 6
  READ a
  PRINT a,
NEXT n
END
DATA 2,4,6,8,10,12
```

3.Auch Zeichenkettenvariablen (Strings) können mit der READ- Anweisung gelesen werden.

```
REM Read3
CLS
READ d$
PRINT "Das Datum ist "; d$
END
DATA "1. Juni 1982"
```

4.Die erste READ-Anweisung liest normalerweise den Wert der ersten DATA-Zeile im Programm ein. Sollen Werte in einer bestimmte DATA-Zeile gelesen werden, kann man den RESTORE-Befehl verwenden.

```
REM Read4
CLS
RESTORE Anfang
READ d$
DATA "großartig"
PRINT "Du bist als Programmierer "; d$
END
Anfang:
DATA "erbärmlich"
```

15. Ausdrücke

Der Computer kann die vier Rechenarten +, -, * und / ausführen und den Wert einer Variablen finden, wenn man ihm den Namen mitteilt. Eine Kombination wie LET Steuer=Betrag*15/100 wird Ausdruck genannt. Ein Ausdruck ist also eine verkürzte Art, den Computer anzuweisen, verschiedene Berechnungen der Reihe nach auszuführen. Der Computer holt sich in obigem Beispiel den Wert der Variablen 'Betrag', multipliziert ihn mit 15, dividiert das Produkt durch 100 und weist das Ergebnis der Variablen 'Steuer' zu.

Jede Rechenoperation hat eine bestimmte Priorität: * und / haben die Priorität 8, + und - die Priorität 6. Also gilt auch hier: Punkt- vor Strichrechnung. Diese Reihenfolge beim Rechnen kann nur durch die Verwendung von Klammern umgangen werden.

1.Starte gegebenenfalls qBASIC und gib die folgende Programmzeile ein.

```
PRINT 1827.4 * 0.14
```

Notiere das Ergebnis: ...

2.Berechne mit Hilfe des Computers die folgenden Ausdrücke:

Ausdruck	Ergebnis
367,256 + 6892,1	
10638,1 - 6781,349	
17,94 · 23,6	
4096 : 0,16	
36·(3,4 - 1,2)	

3.Prüfe durch einen PRINT-Befehl (z.B. PRINT x), welche der folgenden Variablennamen in Ausdrücken zulässig sind:

- DerNameistsolangdaßichihnnichthinschreibenkann
- jetzt sind wir sechs
- JETZtsinDwiRsEchS
- 2001undeineNacht
- 3_Bären
- M*A*S*H
- Müller-Remscheid
- Ich_bin_doof

16. Strings (Zeichenketten)

Ursprünglich war der Computer nur zum Rechnen gedacht (to compute = rechnen). Nach und nach wollte man auch Texte erfassen, d.h. es genügte nicht, dass der Computer nur mit Zahlen (im Binärsystem) umgehen konnte, sondern ein Computer mußte auch Buchstaben, Sonderzeichen, usw. darstellen und verarbeiten können.

Ein String besteht aus einer beliebigen Anzahl verschiedener Ziffern, Buchstaben oder Sonderzeichen. qBASIC erlaubt allerdings nur Zeichenketten bis zu einer Länge von 255 Zeichen. Zeichenketten können auf dem Bildschirm ausgegeben werden (PRINT "Hallo Dödel") oder einer Stringvariablen zugewiesen werden (LET datum$="1.1.1991"). Eine Stringvariable muss durch ein Dollar-Zeichen am Ende des Variablennamens gekennzeichnet werden. Mit Stringvariablen kann nicht gerechnet werden, sie können jedoch auf verschiedene Arten verändert werden.

1.Starte gegebenenfalls qBASIC und gib die folgenden Programmzeilen ein. Starte das Programm und achte auf das Resultat.

```
REM Strings1
CLS
LET zahl1$="2"
LET zahl2$="4"
PRINT zahl1$+zahl2$
```

2.Gib die folgenden Programmzeilen ein. Starte das Programm und achte auf das Resultat.

```
REM Strings2
CLS
LET a$="sachs"
LET b$="angel"
LET c$=b$+a$+"en"
```

3.Gib die folgenden Programmzeilen ein. Starte das Programm und achte auf das Resultat.

```
REM Strings3
CLS
LET a$="Haben Sie schon 'Vom Winde verweht' gelesen?"
LET b$=LEFT$(a$,5)
LET c$=RIGHT$(a$,8)
LET d$=MID$(a$,7)
LET e$=MID$(a$,11,5)
LET a$=e$+" "+c$
PRINT a$
PRINT b$
PRINT c$
PRINT d$
PRINT e$
```

17. Verändern von Zeichenketten

In nahezu jeder BASIC-Version gibt es die Möglichkeit, zwei oder mehrere Zeichenketten miteinander zu verknüpfen. Sehr oft wird hierfür das mathematische Pluszeichen (+), manchmal aber auch das kaufmännische Und-Zeichen(&) verwendet. In qBASIC wird das Pluszeichen verwendet (siehe Beispiele Strings1 und Strings2).

Die Länge einer Zeichenkette kann man mit der Zeichenkettenfunktion LEN bestimmen.

1.Starte gegebenenfalls qBASIC und gib die folgenden Programmzeilen ein. Starte das Programm.

```
REM Strings4
CLS
LET x$="Realschule"
PRINT "Das Wort "; x$; "besteht aus "; LEN(x$); "Zeichen."
```

Im Beispiel Strings3 treten drei Zeichenkettenfunktionen auf, die es erlauben, Teile einer Zeichenkette abzutrennen.

RIGHT$(x$,n)	trennt n Zeichen der Zeichenkette x$ von rechts ab
LEFT$(x$,n)	trennt n Zeichen der Zeichenkette x$ von links ab
MID$(x$,m,n)	trennt n Zeichen der Zeichenkette x$ ab dem m. Zeichen ab
MID$(x$,m)	trennt alle Zeichen der Zeichenkette x$ nach dem m. Zeichen ab

2.Betrachte das folgende Programm. Versuche anhand des Listings herauszufinden, was das Programm bewirkt. Überprüfe diese Vermutung mit Hilfe des Programms.

```
REM Strings5
CLS
LET x$="Realschule"
FOR n=1 TO LEN(x$)
  PRINT LEFT$(x$,n), MID$(x$,n,1), RIGHT$(x$,n)
NEXT n
```

3.Mit Hilfe der Zeichenkettenfunktionen lassen sich auch einige kleine Spielereien durchführen wie das folgende Beispiel zeigt.

```
REM Strings6
CLS
LET x$="ene mene meck und du bist weg"
LET y$=""
FOR n=LEN(x$) TO 1 STEP -1
  LET y$=y$+MID$(x$,n,1)
NEXT n
PRINT "Wer kann das lesen?"
PRINT y$
```

18. Weitere Zeichenkettenfunktionen

Der Zeichensatz des Computers besteht aus 256 Zeichen, die jedoch nicht alle darstellbar sind, da einige Sonderfunktionen haben. Mit Hilfe der Tastatur können nicht alle dieser 256 Zeichen eingegeben werden (warum wohl?). Deshalb gibt es in qBASIC die Funktion CHR$(n), mit der sich (fast) alle Zeichen auf dem Bildschirm darstellen lassen.

1.Das folgende Programm gibt alle wichtigen Zeichen des Zeichensatzes aus.

```
REM Strings7
CLS
PRINT "Die wichtigsten Zeichen des Computers:"
FOR n=32 TO 255
  PRINT CHR$(n);
NEXT n
```

2.Prüfe auch, wie die Zeichen 1 bis 6, 11 und 12, 21, 24 bis 27 aussehen. Erweitere dazu das Programm Strings7 um die entsprechenden Zeilen. Was bewirkt die Zeile PRINT CHR$(7)?

Soll eine Zeichenkette aus lauter gleichen Zeichen bestehen, so verwendet man dazu die Funktion STRING$(n,x$).

3.Das folgende Programm erzeugt einen Rahmen um einen Text. Verwende für x$ verschiedene Namen und starte jeweils das Programm.

```
REM Strings8
CLS
LET x$="Winfried Furrer"
LET n=LEN(x$)
PRINT CHR$(201); STRING$(n+2,CHR$(205)); CHR$(187)
PRINT CHR$(186); " "; x$; " "; CHR$(186)
PRINT CHR$(200); STRING$(n+2,CHR$(205)); CHR$(188)
PRINT
```

Soll der Rahmen anstelle einer Doppellinie aus einer einfachen Linie bestehen, dann sind die im obigen Programm verwendeten Zeichen wie folgt zu ersetzen:

201 ==> 218, 205 ==> 196, 187 ==> 191, 186 ==> 179, 200 ==> 192, 188 ==>217

4.Ändere das Programm Strings8 mit den obenstehenden Angaben so ab, dass ein einfacher Rahmen entsteht.

5.Die Zeichen 196 bzw. 205 können zum Unterstreichen verwendet werden.

```
REM Strings9
CLS
LET x$="Herr Meier"
PRINT x$
PRINT STRING$(LEN(x$),CHR$(196))
```

19. Spezielle Zeichenkettenfunktionen

Zur Umwandlung von Klein- in Großbuchstaben verwendet man die Funktion UCASE$(x$), im umgekehrten Fall die Funktion LCASE$(x$).

1.Das folgende Programm wandelt eine beliebige Zeichenkette in Großbuchstaben um.

```
REM Strings10
CLS
LET x$="Reden ist Silber, Schweigen ist Gold."
PRINT x$
FOR n=1 TO LEN(x$)
  z$=MID$(x$,n,1)
  z$=UCASE$(z$)
  MID$(x$,n,1)=z$
NEXT n
PRINT x$
```

2.Ändere das obige Programm so ab, dass die Zeichenkette nur aus Kleinbuchstaben besteht.

3.Wähle als Zeichenkette den Text "Ältere Menschen ärgern sich öfters, dass Ihnen keiner über die Straße hilft.". Überprüfe, ob die beiden Funktionen auch bei dieser Zeichenkette korrekt arbeiten.

Der schwierigste Zeichenkettenfunktion ist die INSTR-Funktion. Mit ihr kann man feststellen, ob eine Zeichenkette (such$) in einer anderen Zeichenkette (gesamtkette$) vorkommt. Wird such$ in gesamtkette$ gefunden, so gibt c die Position an, wird such$ nicht gefunden hat c den Wert 0.
Syntax: c=INSTR (gesamtkette$,such$)

4.Das folgende Programm demonstriert die Verwendung der INSTR-Funktion. Starte das Programm mehrfach und prüfe verschiedene Buchstaben (klein und groß), und auch solche, die in der Zeichenkette nicht vorkommen.

```
REM Strings11
CLS
LET x$="Reden ist Silber, Schweigen ist Gold."
INPUT "Buchstabe: ", y$
c=INSTR(x$,y$)
PRINT "Der Buckstabe "; y$; " kommt in "; x$;
IF c>0 THEN
  PRINT " an Position "; c;
ELSE
  PRINT " nicht ";
ENDIF
PRINT " vor."
```

20. Anwendung mehrerer Zeichenkettenfunktionen

Die Kombination der bisher vorgestellten Zeichenkettenfunktionen stellt ein mächtiges Werkzeug dar. Damit lassen sich nahezu alle Veränderungen einer beliebigen Zeichenkette vornehmen.

1.Das folgende Programm erwartet die Eingabe eines Vokals. In der Zeichenkette x$ werden dann alle anderen Vokale durch den ausgewählten ersetzt. Der Variablen r$ wird das Zeichen 13 (r$=CHR$(13)) zugeordnet. Dieses Zeichen bewirkt in einer Zeichenkette, dass bei ihrer Darstellung auf dem Bildschirm eine neue Zeile angefangen wird.

```
REM Strings12
CLS
PRINT "Drei Chinesen"
PRINT "-------------"
INPUT "Vokal:", y$
r$ = CHR$(13)
x$ = "Drei Chinesen mit dem Kontrabaß" + r$
x$ = x$ + "saßen auf der Straße und erzählten sich was" + r$
x$ = x$ + "da kam die Polizei, ja was ist denn das" + r$
x$ = x$ + "Drei Chinesen mit dem Kontrabaß."
PRINT x$
PRINT
PRINT "Mit dem Vokal "; y$; ":"
FOR n = 1 TO LEN(x$)
  b$ = MID$(x$, n, 1)
  c = INSTR("aeiouäöü", b$)
  IF c > 0 THEN MID$(x$, n, 1) = LCASE$(y$)
  b$ = MID$(x$, n, 1)
  c = INSTR("AEIOUÄÖÜ", b$)
  IF c > 0 THEN MID$(x$, n, 1) = UCASE$(y$)
NEXT n
PRINT x$
```

2.Sehr wichtig ist auch die Abfrage der Tastatur, um festzustellen, welche Taste der Anwender gedrückt hat.

```
REM Tastaturabfrage
CLS
PRINT "Drücke verschiedene Tasten auf der Tastatur."
PRINT "Das Programm wird beendet durch ESC."
PRINT STRING$(79, "-")
PRINT
DO
  t$ = INKEY$
  IF t$ <> "" THEN
    PRINT t$; ", Länge der Zeichenkette: "; LEN(t$);
    PRINT ", Codes: ";
    FOR n = 1 TO LEN(t$)
      PRINT ASC(MID$(t$, n, 1)); " ";
    NEXT n
    PRINT
  END IF
LOOP UNTIL t$ = CHR$(27)
```

21. Rechnen mit Zeichenketten

Rechnen kann der Computer eigentlich nur mit ganzen Zahlen bzw. Kommazahlen. Deshalb muss man auch entsprechende Variablen verwenden.

1.Das folgende Programm berechnet zwei Ausdrücke.

```
REM Rechnen1
CLS
LET zahl1=4
LET zahl2=7
PRINT "Die Summe der Zahlen "; zahl1; " und "; zahl2;
PRINT " ist "; zahl1 + zahl2
LET zahl3=4.7
LET zahl4=7.8
PRINT "Das Produkt der Zahlen "; zahl3; " und "; zahl4;
PRINT " ist "; zahl3 * zahl4
```

2.Im folgenden Programm werden Zeichenkettenvariablen verwendet.

```
REM Rechnen2
CLS
LET zahl1$="4"
LET zahl2$="7"
PRINT "Die Summe der Zahlen "; zahl1$; " und "; zahl2$;
PRINT " ist "; zahl1$ + zahl2$
```

Das Programm liefert nicht das korrekte Ergebnis, weil nur die beiden Zeichenketten verknüpft werden. Versuche auch statt der Summe das Produkt zu berechnen.

Um mit Zeichenketten rechnen zu können sind verschiedene Umwandlungen nötig. Eine große Rolle spielen dabei die Funktionen VAL und STR$. Die VAL-Funktion wandelt eine Zeichenkette in eine Zahlvariable um, die STR$-Funktion eine Zahlvariable in eine Zeichenkette.

3.Im folgenden Programm werden Zeichenkettenvariablen verwendet. Die Variablen zahl1$ und zahl2$ sind Zeichenkettenvariablen (Endung $), die Variable summe ist eine Zahlvariable. Zur Umwandlung der Zeichenkettenvariablen in Zahlvariablen wird die VAL-Funktion verwendet.

```
REM Rechnen3
CLS
LET zahl1$="4"
LET zahl2$="7"
LET summe=VAL(zahl1$)+VAL(zahl2$)
PRINT "Die Summe der Zahlen "; zahl1$; " und "; zahl2$;
PRINT " ist "; summe
PRINT "Das Ergebnis als Zeichenkette: "; STR$(summe)
```

22. Zahlvariablen

Je nach Größe und Art der Zahlen, die verwendet werden sollen, benutzt man unterschiedliche Typen von Zahlvariablen. Diese Typen unterscheiden sich darin, dass sie unterschiedlich

a.viel Speicherplatz verbrauchen b.große Zahlen darstellen können c.schnell verarbeitet werden können.

Typ	Endung	Wertebereich	Beispiel
Integer	%	-32768...+32767	i%
Long	&	-2147483648...+214748647	l&
Single	!	positive: 3,4·1038 ...2,8·10-45	
		negative: -2,8·10-45 ...-3,4·1038	s!
Double	#	positive: 1,8·10308 ...4,9·10-324	
		negative: -4,9·10-324 ...-1,8·10308	d#

In den meisten Fällen werden Integer- und Single-Variablen verwendet. Mit Integer-Variablen (Ganzzahl-Variablen) kann der Rechner am schnellsten rechnen. Deshalb sollte man wo immer es geht diesen Variablentyp verwenden und die Variable deshalb entsprechend kennzeichnen (Endung %).
Beispiele: LET a% = 367
LET b% = -2
LET c% = 12867

Muß man auf Komma-Zahlen oder sehr große Ganzzahlen zurückgreifen wird meist der Single-Typ verwendet (Endung !).
Beispiele: LET a! = 3.67
LET b! = -2.0
LET c! = 1286.7

Wird eine Variable ohne Endung benutzt, so weist ihr der Computer den Typ Single zu.
Beispiele: LET a = 367
LET b = -2
LET c = 12867

23. Der Bildschirm im Textmodus

Der Bildschirm eines PCs kann unter qBASIC in verschiedenen Modi betrieben werden. Man unterscheidet den Text- und den Grafikmodus. Jeder dieser Bildschirmmodi kann mit dem entsprechenden Monitor (Hercules, Monochrom, EGA oder VGA) in verschiedenen Bildschirm-Auflösungen verwendet werden.

Im Standard-Textmodus (ist beim Aufruf von qBASIC eingestellt) sind 25 Zeilen zu je 80 Zeichen darstellbar. Zur Einstellung des gewünschten Textmodus verwendet man den Befehl SCREEN und zur Einstellung der Anzahl der Zeilen und Spalten den Befehl WIDTH.

1.Das folgende Programm stellt den Modus 0 ein und beschreibt alle möglichen Zeilen (bis auf die letzte) mit zufälligen Zeichen. Starte das Programm mehrfach und gib für die Anzahl der Zeilen und Spalten unterschiedliche Werte ein. Beobachte die Bildschirmausgaben.

```
REM Textmodus1
SCREEN 0
RANDOMIZE TIMER
INPUT "Anzahl der Zeilen (25, 43 oder 50): ",zeilen
INPUT "Anzahl der Spalten (40 oder 80): ",spalten
WIDTH spalten,zeilen
CLS
FOR z=1 TO zeilen-1
  FOR s=1 TO spalten
    PRINT CHR$(65+INT(26*RND));
  NEXT s
NEXT z
```

Abhängig vom verwendeten Bildschirmmodus können verschiedene Farben verwendet werden (meist 16 Farben mit den Farbnummern 0 bis 15). Der Befehl zur Einstellung der Farben (Vordergrund- und Hintergrundfarbe) heißt COLOR.

2.Ergänze obiges Programm wie folgt und starte das Programm mehrfach. Die Vordergrundfarben (0 bis 15) werden per Zufall gewählt, die Hintergrundfarbe (0 bis 7) kann in Zeile 10 verändert werden.

```
REM Textmodus2
SCREEN 0
RANDOMIZE TIMER
INPUT "Anzahl der Zeilen (25, 43 oder 50): ",zeilen
INPUT "Anzahl der Spalten (40 oder 80): ",spalten
WIDTH spalten,zeilen
CLS
FOR z=1 TO zeilen-1
  FOR s=1 TO spalten
    hintergrundfarbe=0
    vordergrundfarbe=INT(15*RND)
    COLOR vordergrundfarbe, hintergrundfarbe
    PRINT CHR$(65+INT(26*RND));
  NEXT s
NEXT z
```

24. Bildschirmausgaben im Textmodus

In jedem Textverarbeitungsprogramm kann die Schreibmarke mit Hilfe der Cursortasten (Pfeiltasten) an eine beliebige Stelle des Bildschirms gebracht werden. Der anschließend eingegebene Text erscheint exakt an dieser Stelle. In qBASIC kann die Schreibmarke ebenfalls positioniert werden. Dazu dient der Befehl LOCATE.

1.Das folgende Programm gibt verschiedene Texte an verschiedenen Stellen des Bildschirms aus. Beachte, dass mehrere Befehle in einer Zeile durch einen Doppelpunkt voneinander getrennt werden.

```
REM Textmodus3
SCREEN 0
WIDTH 80,25
CLS
LOCATE 10,1: PRINT "Zeile 10, Spalte 1"
LOCATE 5,40: PRINT "Zeile 5, Spalte 40"
LOCATE 15,20: PRINT "Zeile 15, Spalte 20"
```

2.Das folgende Programm verwendet verschiedene Farben und zufällig ausgewählte Bildschirmpositionen.

```
REM Textmodus4
SCREEN 0
WIDTH 80,25
RANDOMIZE TIMER
CLS
FOR i = 1 TO 20
  farbe = INT(14 * RND) + 1
  COLOR farbe
  zeile = INT(24 * RND) + 1
  spalte = INT(50 * RND + 1)
  text$ = "." + CHR$(27)
  text$ = text$ + " Zeile " + STR$(zeile)
  text$ = text$ + ", Spalte " + STR$(spalte)
  LOCATE zeile, spalte: PRINT text$;
NEXT i
```

3.Das folgende Programm stellt verschiedene Texte zentriert, d.h. in der Bildschirmmitte, dar. Dazu wird das Unterprogramm Center aufgerufen.

```
REM Textmodus5
SCREEN 0
WIDTH 80,25
CLS
zeile=4: text$="Hallo Computer-Freak": GOSUB Center
zeile=6: text$="Der gesamte Text dieser Seite": GOSUB Center
zeile=7: text$="wird zentriert": GOSUB Center
zeile=8: text$="ausgegeben.": GOSUB Center
END
Center:
  LOCATE zeile, (80 - LEN(text$)) / 2: PRINT text$
RETURN
```

25. Bildschirmausgaben mit Tabulator

Der Bildschirm ist standardmäßig in verschiedene Tabulatorspalten eingeteilt. Um Texte in einer Zeile genauer per Tabulator ausrichten zu können, benutzt man die TAB-Funktion (nicht zu verwechseln mit der TAB-Taste!).

1.Das folgende Programm verwendet die voreingestellten Tabulatorspalten des PRINT-Befehls (beachte das Komma am Ende des PRINT-Befehls).

```
REM Textmodus6
CLS
FOR i=1 TO 10
  PRINT i*5,
NEXT i
```

2.Im folgenden Programm wird das kleine Einmaleins dargestellt (10 Zeilen und 10 Spalten). Bei einer Bildschirmbreite von 80 Spalten muss also in jeder 8. Spalte ein Tabulator gesetzt werden.

```
REM Textmodus7
CLS
PRINT "Das kleine Einmaleins"
PRINT "----------------------"
PRINT
FOR i=1 TO 10
  FOR j=1 TO 10
   PRINT TAB(j*8-7); i*j;
  NEXT j
  PRINT
NEXT i
```

3.Das folgende Programm ist eine kleine Spielerei mit der TAB-Funktion. Die Geschwindigkeit der Bildschirmanzeige lässt sich mit der FOR.NEXT-Schleife in Zeile 15 (Wert 1000) regulieren. Beendet wird das Programm durch einen beliebigen Tastendruck.

```
REM Textmodus8
CLS
LET spalte=1
LET farbe=1
LET zuwachs=1
DO
  a$=INKEY$
  COLOR farbe
  PRINT TAB(spalte);"Informatik-AG";
  LET farbe=farbe+1
  IF farbe>15 THEN LET farbe=1
  LET spalte=spalte+zuwachs
  IF spalte>66 THEN LET zuwachs=-1
  IF spalte<2 THEN LET zuwachs=1
  FOR pause=1 TO 1000
  NEXT pause
LOOP UNTIL a$<>""
```

26. Bewegungen auf dem Bildschirm

Um ein Objekt auf dem Bildschirm zu bewegen, muss das Objekt zunächst einmal auf dem Bildschirm dargestellt werden. Nach einer kurzen Pause wird das Objekt gelöscht, an die neue Position gebracht und wieder dargestellt. Dieser Zyklus wird im folgenden wiederholt.

1.Das folgende Programm bewegt ein Objekt über den Bildschirm und kann als Grundlage für ein kleines Spiel verwendet werden.

```
REM Bewegung1
SCREEN 0
WIDTH 40
CLS
COLOR 15
PRINT TAB(7); "Bewegung auf dem Bildschirm"
LOCATE 2, 1
PRINT CHR$(201); STRING$(38, CHR$(205)); CHR$(187);
FOR i = 3 TO 23
  LOCATE i, 1
  PRINT CHR$(186); STRING$(38, CHR$(32)); CHR$(186);
NEXT i
LOCATE 24, 1
PRINT CHR$(200); STRING$(38, CHR$(205)); CHR$(188);
x = 20: y = 13
dx = 1: dy = 1
DO
  a$ = INKEY$
  COLOR 14
  LOCATE y, x: PRINT CHR$(2);
  FOR pause = 1 TO 500
  NEXT pause
  COLOR 14: LOCATE y, x: PRINT CHR$(32);
  x = x + dx
  y = y + dy
  IF x > 38 THEN dx = -1
  IF x < 3 THEN dx = 1
  IF y < 4 THEN dy = 1
  IF y > 22 THEN dy = -1
LOOP UNTIL a$ <> ""
WIDTH 80
```

Die Auflösung des Bildschirms wird zunächst auf 40 Spalten eingestellt, der Bildschirm selbst gelöscht. In weißer Schrift (COLOR 15) wird die Überschrift und der Rahmen gezeichnet. Die Koordinaten des bewegten Objektes sind x und y, die Veränderung der Objekt-Position wird mit den Variablen dx und dy vorgenommen. Das Objekt selbst, CHR$(2), wird gelb (COLOR 14) gezeichnet. Nach einer kleinen Pause (Wert=500) wird das Objekt durch CHR$(32) überschrieben, also gelöscht. Daraufhin wird die neue Position berechnet und geprüft, ob das Objekt auf den Rahmen trifft. Je nach Ausfall des Tests wird die Bewegungsrichtung verändert. Solange keine Taste gedrückt wird, werden die letzten Vorgänge ständig wiederholt. Unter Umständen ist eine Anpassung des Pausenwertes erforderlich.

27. Bildschirmausgaben im Grafikmodus

Im Grafikmodus ist der Bildschirm viel feiner in Zeilen und Spalten aufgeteilt. Wichtige Auflösungen sind z.B. 320 Spalten / 200 Zeilen, 640 Spalten / 200 Zeilen, 640 Spalten / 350 Zeilen oder 640 Spalten / 480 Zeilen. Die linke obere Ecke hat die Koordinaten 0/0, die rechte untere Ecke z.B. 639/479.

Zur Einstellung des Grafikmodus dient ebenfalls der SCREEN-Befehl. Linien werden mit dem LINE-Befehl, Punkte mit dem PSET-Befehl gezeichnet. An jedem Schnittpunkt einer Zeile mit einer Spalte kann ein Punkt in einer bestimmten Farbe gesetzt werden. Ein solcher Punkt heißt Pixel.

1.Das folgende Programm zeichnet bei SCREEN 8 (640*200, 16 Farben) einen gelben Rahmen.

```
REM Grafikmodus1
SCREEN 8
farbe = 14
LINE (0, 0)-(639, 0), farbe
LINE -(639, 199), farbe
LINE -(0, 199), farbe
LINE -(0, 0), farbe
LOCATE 13, 34: PRINT "Ende -> Taste"
DO
  a$ = INKEY$
LOOP UNTIL a$ <> ""
```

2.Das folgende Programm zeichnet bei SCREEN 9 (640*350, 16 Farben) senkrechte und waagrechte Linien.

```
REM Grafikmodus2
SCREEN 9
CLS
farbe = 12
FOR x = 0 TO 639 STEP 16
  FOR y = 0 TO 349 STEP 16
    LINE (x, 0)-(x, 349), farbe
    LINE (0, y)-(639, y), farbe
  NEXT y
NEXT x
LINE (0, 0)-(639, 349), 15, B
LOCATE 13, 34: PRINT "Ende -> Taste"
DO
  a$ = INKEY$
LOOP UNTIL a$ <> ""
```

28. Komplexe Figuren im Grafikmodus

Die folgenden zwei Programme verwenden die mathematischen Funktionen Sinus und Cosinus, um die Koordinaten der Pixel zu berechnen.

1.Das folgende Programm zeichnet bei SCREEN 12 (640*480, 16 Farben) verschieden gefärbte Kurven.

```
REM Grafikmodus3
SCREEN 12: CLS
FOR j = 0 TO 9
  farbe = 15 - j
  FOR i = 0 TO 510
    x = 40+i
    y = (220+210*SIN(i*i/10000)+j/2)*(1-j*j/100)
    IF i = 0 THEN
      PSET (x, y), farbe
    ELSE
      LINE -(x, y), farbe
    END IF
  NEXT i
NEXT j
LOCATE 30, 1: PRINT "Ende -> Taste";
DO
  a$ = INKEY$
LOOP UNTIL a$ <> ""
```

2.Das folgende Programm zeichnet bei SCREEN 12 (640*480, 16 Farben) eine endlose Figur.

```
REM Grafikmodus4
SCREEN 12:CLS
LET i = 0
LOCATE 30, 1: PRINT "Ende -> Taste";
DO
  a$ = INKEY$
  x = 319 + (75 + 125 * COS(i / 41)) * COS(i / 40)
  y = 240 + (125 + 75 * SIN(i / 39)) * SIN(i / 40)
  IF i = 0 THEN
    PSET (x, y), 14
  ELSE
    LINE -(x, y), 14
  END IF
  i = i + 1
LOOP UNTIL a$ <> ""
```

3.Das folgende Programm zeichnet bei SCREEN 13 (320*200, 256 Farben) viele senkrechte Linien.

```
REM Grafikmodus5
SCREEN 13: CLS
FOR i=1 TO 255
  LINE (32+i,0)-(32+i,199),i
```

```
NEXT i
```

29. Das Achsenkreuz

Der Grafikbildschirm im Modus 12 besteht aus 640 Spalten und 480 Zeilen. An jedem Kreuzungspunkt einer Spalte mit einer Zeile kann ein Punkt in einer von 16 Farben gesetzt werden (=Pixel).

1.Das folgende Programm zeichnet ein beschriftetes Achsenkreuz in der Farbe gelb im Bereich von
-5 >= x >= 5 und -4 >= y >= 4.

```
REM Achsenkreuz
SCREEN 12
CLS
farbe = 14 : REM Farbe gelb

REM x-Achse
LINE (0, 232)-(639, 232), farbe
LINE (639, 232)-(635, 228), farbe
LINE (639, 232)-(635, 236), farbe
FOR i = -8 TO 8
  LINE (320 + i * 40, 230)-(320 + i * 40, 234), farbe
NEXT i
FOR i = -7 TO 7
  IF i <> 0 THEN
    LOCATE 16, (i + 8) * 5: PRINT i
  END IF
NEXT i

REM y-Achse
LINE (320, 0)-(320, 479), farbe
LINE (320, 0)-(316, 4), farbe
LINE (320, 0)-(324, 3), farbe
FOR i = -5 TO 5
  LINE (318, 232 + i * 48)-(322, 232 + i * 48), farbe
NEXT i
FOR i = -4 TO 4
  IF i <> 0 THEN
    LOCATE 30 - (i + 5) * 3, 42: PRINT i
  END IF
NEXT i
```

2.Eine Einheit in Richtung der x-Achse umfasst 40 Pixel, die y-Achse selbst wird bei x=320 gezeichnet. In Richtung der y-Achse werden für eine Einheit 48 Pixel verwendet, die x-Achse wird bei y=232 gezeichnet.
Füge an das obige Programm die Zeile

```
PSET (320+(-2)*40,232-(+3)*48),farbe
```

an. Welche Koordinaten (in Einheiten des Achsenkreuzes) hat der gezeichnete Punkt? Zeichne auch die Punkte P(4/3), Q(-2/-1), R(3/-1) und S(-4/2). Verwende dazu verschiedene Farben (von Farb-Nummer 1 bis 15). Füge dazu vier neue Zeilen an das Programm an und ersetze die Variable farbe durch die entsprechende Farbnummer.

30. Vereinfachtes Setzen von Punkten im Achsenkreuz

Um das Setzen von Punkten im Programm 'Achsenkreuz' zu vereinfachen, kann man eine entsprechende Subroutine schreiben.

1.Entferne im Programm 'Achsenkreuz' zunächst alle selbst angefügten Zeilen und ergänze dann das Programm um die Zeile Plot 1,-3,farbe.
Bewege die Schreibmarke unter einen Buchstaben des Wortes Plot und erzeuge mit Bearbeiten - Neue Sub einen Rahmen für die neue Subroutine. Der Bildschirm hat daraufhin zunächst folgendes Aussehen:

```
SUB Plot

END SUB
```

Ergänze dann dieses Unterprogramm wie folgt:

```
SUB Plot (x,y,f)
  PSET (320+x*40,232-y*48),f
END SUB
```

Diesem Unterprogramm wird beim Aufruf eine x- und eine y-Koordinate sowie eine Farbe übergeben. Der PSET-Befehl berechnet dann für das durch das Hauptprogramm gezeichnete Achsenkreuz die Pixelkoordinaten und zeichnet den gewünschten Punkt in der entsprechenden Farbe.

2.Zeichne mit Hilfe des Unterprogrammes Plot die Punkte P(4/3), Q(-2/-1), R(3/-1) und S(-4/2). Verwende dazu verschiedene Farben (von Farb-Nummer 1 bis 15). Füge dazu die vier entsprechenden neuen Zeilen an das Hauptprogramm an.

3.Die gezeichneten Punkte auf dem Bildschirm sind sehr klein. Um die Punkte etwas besser erkennen zu können, kann man statt eines Punktes vier Punkte zeichnen.
Wechsle mit Ansicht - Subs zum Unterprogramm Plot und ergänze das Unterprogramm wie folgt:

```
SUB Plot (x,y,f)
  PSET (320+x*40,232-y*48),f
  PSET (321+x*40,232-y*48),f
  PSET (320+x*40,233-y*48),f
  PSET (321+x*40,233-y*48),f
END SUB
```

Wechsle zurück zum Modul des Hauptprogrammes und starte das Programm. Beobachte die veränderte Bildschirmausgabe.

31. Zeichnen von Linien im Achsenkreuz

Um zwei Punkte im Achsenkreuz durch eine Linie zu verbinden werden die Koordinaten des Anfangs- und des Endpunktes benötigt.

1.Das Programm 'Achsenkreuz' wird um eine weiter Subroutine erweitert. Entferne dazu zunächst sämtliche Plot-Befehle am Ende des Hauptprogramms. Füge dann die Zeile

```
Connect -2,-3,3,1,15
```

an das Hauptprogramm an. Die ersten beiden Zahlenangaben sind die Koordinaten des Anfangspunktes, dann folgen die zwei Zahlenangaben für den Endpunkt. Die letzte Zahlenangabe gibt die zu verwendende Farbe an.

Bewege die Schreibmarke unter einen Buchstaben des Wortes Draw und erzeuge mit Bearbeiten - Neue Sub einen Rahmen für die neue Subroutine. Der Bildschirm hat daraufhin zunächst folgendes Aussehen:

```
SUB Connect

END SUB
```

Ergänze dann dieses Unterprogramm wie folgt:

```
SUB Connect (x1,y1,x2,y2,f)
  LINE (320+x1*40,232-y1*48)-(320+x2*40,232-y2*48),f
END SUB
```

2.Alle bisher erzeugten Unterprogramme funktionieren nur einwandfrei, wenn die verwendeten Koordinaten im Bereich -5<Bild>x<Bild>5 und -4<Bild>y<Bild>4 liegen. Werden andere Koordinaten verwendet, so wird 'außerhalb' des Bildschirms gezeichnet. Überprüfe folgende Zeichenoperationen:

```
Plot -8,2,3
Plot 2,-16,3
Connect -8,2,-6,4,3
Connect -8,2,6,4,3
Connect -3,-2,4,8,3
```

3.Fehlerhafte Eingaben des Benutzers können in den Unterprogrammen abgefangen werden. Füge dazu in den Subroutinen vor den PSET- bzw. LINE-Befehlen die folgenden Zeilen ein:

```
Plot:   IF ABS(x)>5 THEN BEEP : EXIT SUB
        IF ABS(y)>4 THEN BEEP : EXIT SUB
Lines:  IF ABS(x1)>5 OR ABS(x2)>5 THEN BEEP : EXIT SUB
        IF ABS(y1)>4 OR ABS(y2)>4 THEN BEEP : EXIT SUB
```

Starte dann das Hauptprogramm mit den fehlerhaften Programmzeilen erneut und beobachte die Bildschirmausgabe. Ein akustisches Signal weist auf falsche Koordinaten hin.

32. Zeichnen von Rechtecken

Mit qBASIC lassen sich durch entsprechende Befehle sehr einfach Rechtecke, Quadrate, Kreise und Ovale zeichnen. Bei den eckigen Figuren kann gewählt werden, ob sie leer oder gefüllt dargestellt werden soll.

1.Das folgende Programm zeichnet bei SCREEN 12 (640*480, 16 Farben) 20 verschiedene Rechtecke an zufälligen Bildschirmpositionen.

```
REM Rechteck1
RANDOMIZE TIMER
SCREEN 12
CLS
FOR rechteck = 1 TO 20
  LET breite = INT(RND * 80) + 20
  LET hoehe = INT(RND * 80) + 20
  LET x1 = INT(RND * (640 - breite))
  LET y1 = INT(RND * (480 - hoehe))
  LET farbe = INT(RND * 15) + 1
  LINE (x1, y1)-(x1 + breite, y1 + hoehe), farbe, B
NEXT rechteck
a$ = ""
WHILE a$ = ""
  a$ = INKEY$
END
```

2.Ändere das obige Programm wie folgt ab:

```
  :
  LET y1 = INT(RND * (480 - hoehe))
  LET farbe = INT(RND * 15) + 1
  LET gefuellt = INT(RND *2)
  IF gefuellt = 0 THEN
    LINE (x1, y1)-(x1 + breite, y1 + hoehe), farbe, B
  ELSE
    LINE (x1, y1)-(x1 + breite, y1 + hoehe), farbe, BF
  END IF
NEXT rechteck
:
```

Das Programm erzeugt jetzt abhängig vom Wert der Variablen gefuellt (0 oder 1) ungefuellte oder gefuellte Rechtecke.

3.Versuche anhand des Programm-Listings folgende Frage zu beantworten.
a. Welches ist die minimale bzw. maximale Breite/Höhe eines Rechtecks?

b. Welches sind für jeden der 4 Fälle die minimalen bzw. maximalen Koordinaten der linken oberen Ecke des Rechtecks?

c. Welche Farbnummer ist die kleinste bzw. die größte?

33. Zeichnen von Kreisen und Ellipsen

1.Das folgende Programm zeichnet bei SCREEN 12 (640*480, 16 Farben) 40 verschiedene Kreise an zufälligen Bildschirmpositionen.

```
REM Kreise1
RANDOMIZE TIMER
SCREEN 12
CLS
RANDOMIZE TIMER
FOR kreis = 1 TO 40
  LET radius = INT(RND * 80) + 20
  LET x = radius + INT(RND * (640 - 2 * radius))
  LET y = radius + INT(RND * (480 - 2 * radius))
  LET farbe = INT(RND * 15) + 1
  CIRCLE (x, y), radius, farbe
NEXT kreis
a$ = ""
WHILE a$ = ""
  a$ = INKEY$
WEND
```

2.Ändere das obige Programm wie folgt ab:

```
  :
  LET y = radius + INT(RND * (480 - 2 * radius))
  LET farbe = INT(RND * 15) + 1
  CIRCLE (x, y), radius, farbe, , , .5
NEXT kreis
a$ = ""
WHILE a$ = ""
  a$ = INKEY$
WEND
:
```

Der CIRCLE-Befehl wird im obigen Beispiel mit 7 Parametern aufgerufen. x und y sind die Koordinaten des Mittelpunktes, radius ist der Kreisradius, farbe ist die Farbe des Randes. Danach folgen 2 Parameter, die nicht angegeben (ausgelassen werden) und der letzte Parameter (0,5) steht für das Verhältnis der Radien in x bzw. y-Richtung (0,5 bedeutet, dass der y-Radius halb so groß ist wie der x-Radius).

3.Ändere das obige Programm so ab, dass 99 Luftballons gezeichnet werden. Wähle dazu als letzten Parameter 1,6 und zeichne am tiefsten Punkt der Ellipse beginnend eine 70 Pixel lange senkrechte Linie (Farbe: 15=leuchtend weiß).

4.Wie muss das Programm abgeändert werden, dass alle Ballons vollständig auf dem Bildschirm gezeichnet werden können (einschließlich Schnur)?

34. Grafikbeispiel

1.Gib das folgende Programm ein und starte es.

```
REM 007
SCREEN 12 : CLS
LINE (0, 479)-(639, 440), 15, BF
Figur1 240, 400, 6, 2 : Figur1 420, 390, 6, 2
Figur1 320, 440, 6, 2 : Figur1 110, 440, 6, 2
Figur1 550, 440, 6, 2
LOCATE 4, 28
DO
  READ a$ : PRINT CHR$(VAL(a$));
LOOP UNTIL a$ = ""
DATA 70,82,79,72,69,32,87,69,73,72,78,65,67,72,84,69,78,32,49,57,57,54,""

DIM sx(60), sy(60), ss(60)
FOR i = 0 TO 60
  sx(i) = INT(RND * 640) : sy(i) = INT(RND * 440)
  ss(i) = POINT(sx(i), sy(i))
  PSET (sx(i), sy(i)), 15
NEXT i

i = 0 : a$ = ""
WHILE a$ = ""
  a$ = INKEY$
  PSET (sx(i), sy(i)), ss(i)
  sy(i) = sy(i) + 3
  s = ss(i) : ss(i) = POINT(sx(i), sy(i))
  PSET (sx(i), sy(i)), 15
  IF s = 0 AND ss(i) <> 0 THEN
    PSET (sx(i), sy(i) - 3), 15
  END IF
  IF sy(i) > 440 THEN
    sx(i) = INT(RND * 640) : sy(i) = INT(RND * 440)
    ss(i) = POINT(sx(i), sy(i))
    PSET (sx(i), sy(i)), 15
  END IF
  i = i + 1 : IF i > 60 THEN i = 0
WEND
```

Das Programm ist nur lauffähig, wenn ...
auch die folgende Subroutine eingegeben wurde:

```
SUB Figur1 (x, y, f1, f2)
  LINE (x - 20, y)-(x + 20, y - 80), f1, BF
  FOR i = 8 TO 0 STEP -8
    LINE (x - 100, y - 80)-(x + 60, y - 80), f2 + i
    LINE -(x + 100, y - 80), f2 + i
    LINE -(x + 45, y - 130), f2 + i
    LINE -(x + 85, y - 130), f2 + i
    LINE -(x + 30, y - 180), f2 + i
    LINE -(x + 70, y - 180), f2 + i
    LINE -(x + 15, y - 230), f2 + i
    LINE -(x + 55, y - 230), f2 + i
    LINE -(x +  0, y - 280), f2 + i
    LINE -(x - 55, y - 230), f2 + i
```

```
    LINE -(x - 15, y - 230), f2 + i
    LINE -(x - 70, y - 180), f2 + i
    LINE -(x - 30, y - 180), f2 + i
    LINE -(x - 85, y - 130), f2 + i
    LINE -(x - 45, y - 130), f2 + i
    LINE -(x - 100, y - 80), f2 + i
    PAINT (x, y - 270), f2 + i, f2 + i
  NEXT i
END SUB
```

35. Programmsteuerung mit CASE ... SELECT

Sind mehrere Entscheidungen zu treffen, so können IF...THEN-Abfragen auch kombiniert werden. Solche Kombinationen sind jedoch sehr unübersichtlich und schwer nachzuvollziehen (linkes Beispiel). Kürzer und sehr viel übersichtlicher erscheint das rechte Beispiel, welches die SELECT...CASE-Struktur verwendet.

```
REM Antibeispiel
CLS
INPUT "Deine Zahl: ",zahl
IF zahl<4 THEN
  PRINT "Hallo"
ELSE
  IF zahl>8 THEN
    PRINT "Guten Morgen"
  ELSE
    IF zahl=6 THEN
      PRINT "Hi"
    ELSE
      PRINT "Grüß Gott"
    ENDIF
  ENDIF
ENDIF
```

```
REM SELECT...CASE
CLS
INPUT "Deine Zahl: ",zahl
SELECT CASE zahl
CASE 0 TO 3
PRINT "Hallo"
  CASE 4 TO 5, 7 TO 8
    PRINT "Grüß Gott"
CASE 6
  PRINT "Hi"
CASE ELSE
PRINT "Guten Morgen"
  END SELECT
```

1.Gib das rechte Programm ein und starte es mehrmals. Gib verschiedene Zahlen ein und beobachte die jeweilige Bildschirmausgabe.

2.Das folgende Programm verwendet ebenso die SELECT...CASE-Struktur zur Programmsteuerung. Wird eine bestimmte Buchstabentaste gedrückt, so wird ausgegeben, ob ein Klein- oder ein Großbuchstabe gedrückt wurde.

```
REM SELECT..CASE1
CLS
PRINT "Drücke bitte eine beliebige Taste..."
PRINT "(ESC = Programmende)"
DO
  DO
    taste$ = INKEY$
  LOOP UNTIL taste$ <> ""
  SELECT CASE taste$
    CASE "a" TO "z"
      LOCATE 4, 1
      PRINT "Kleinbuchstabe "; taste$; "       "
    CASE "A" TO "Z"
```

```
      LOCATE 4, 1
      PRINT "Großbuchstabe "; taste$; "      "
    CASE ELSE
      LOCATE 4, 1
      PRINT "keine Buchstabentaste"
  END SELECT
LOOP UNTIL taste$ = CHR$(27)
```

3.Erweitere das obige Programm so, dass auch die Zahlentasten (1...0) erkannt werden (Ausgabe Zahlentaste).

4.Die Umlaute (ä, ö, ü, Ä, Ö, Ü) und das ß werden nicht als Buchstabentasten erkannt. Erweitere das Programm um die dazu notwendigen Programmzeilen.

36. Töne mit qBASIC - der SOUND-Befehl

Zur Ausgabe von Tönen über den eingebauten PC-Lautsprecher dient z.B. der SOUND-Befehl. Dieser Befehl erwartet zwei Parameter: die Frequenz des Tones und seine Dauer.

1.Gib die folgenden Programme ein und starte sie.

```
REM Sound1
CLS
PRINT "Ein springender Ball ..."
ton1 = 246
ton2 = 32767
FOR zaehler = 60 TO 1 STEP -2
  SOUND ton1 - zaehler / 2, zaehler / 20
  SOUND ton2, zaehler / 15
NEXT zaehler

REM Sound2
CLS
PRINT "Etwas fällt ..."
ton1 = 2000
ton2 = 550
pause = 500
FOR zaehler = ton1 TO ton2 STEP -10
  SOUND zaehler, pause / zaehler
NEXT zaehler

REM Sound3
CLS
PRINT "Alarm ..."
ton1 = 987
ton2 = 329
FOR zaehler = 1 TO 20
  SOUND ton1, 5
  SOUND ton2, 5
NEXT zaehler

REM Sound4
CLS
```

```
PRINT "Sirene ..."
ton = 780
bereich = 650
FOR zaehler1 = 1 TO 6
  FOR zaehler2 = bereich TO -bereich STEP -4
    SOUND ton - ABS(zaehler2), .3
    zaehler2 = zaehler2 - 2 / bereich
  NEXT zaehler2
NEXT zaehler1
REM Sound5
CLS
PRINT "Noch ein Beispiel ..."
FOR zaehler = 440 TO 1000 STEP 5
  SOUND zaehler, zaehler / 1000
NEXT zaehler
```

2.Versuche, mit dem SOUND-Befehl eigene Geräusche zu erzeugen. Beachte, dass die Frequenz zwischen 37 und 32767 liegen muss und die Tondauer zwischen 0 und 65535.

37. Töne mit qBASIC - der PLAY-Befehl

Auch mit dem PLAY-Befehl können in qBASIC Töne erzeugt werden. Die Anwendung dieses Befehles ist jedoch ungleich komplizierter. Als Parameter erwartet der PLAY-Befehl eine Zeichenkette wie z.B. "AFG".

1.Gib das folgende Programm ein und starte es.

```
REM Play1
tonlaenge = "L2"
oktave$ = "O1"
noten$ = "CDEFGAB"
PLAY tonlaenge$
PLAY oktave$
PLAY noten$
```

Die Tonlänge kann im Bereich von L1 bis L64 variiert werden. L1 bedeutet ganze Noten, L2 halbe Noten, L4 viertel Noten, usw.
Die Oktave wird mit O0 bis O6 gewählt. Mit dem Größer-als-Zeichen (>) kann die Oktave erhöht werden, mit dem Kleiner-als-Zeichen (<) wird sie erniedrigt.
Als Noten pro Oktave sind erlaubt: C, D, E, F, G, A, B.

2.Teste auch die folgenden Programme.

```
REM Play2
lvb$ = "O2 P2 P8 L8 GGG L2 E- P24 P8 L8 FFF L2 D"
PLAY lvb$
```

Pausen können mit dem Buchstaben P eingeleitet werden. Mögliche Werte sind P1 bis P64. Soll eine Note um einen Halbtonschritt erhöht oder erniedrigt werden, hängt man an den Buchstaben der Note ein Pluszeichen (+) oder ein Minuszeichen (-) an. Anstelle des Pluszeichens kann auch das Doppelkreuz (#) verwendet werden.

```
REM Play3
```

```
lied$ = "O1 L4 CD L8 ED L4 CCD L8 ED L4 CEF L2 G L4 EF L2 G"

PLAY lied$
```

3.Zum Abschluss noch ein etwas umfangreicheres Musikbeispiel:

```
REM Play4
n$ = ""
n$ = n$ + "o2 t80 l4 > C L8 C L16 < B > C < D >"
n$ = n$ + "C < BA L8 > C L16 C < A L4 > C L8 C L16 < B> C"
n$ = n$ + "< AGEF L2 G L16 GFEDEFGA L4 G"
n$ = n$ + "G L16 GABAGFEDED L8 C L16 CD L8 EF"
n$ = n$ + "L4 D L2 G L8 P8 L4 > D L8 DC L16 < BAB > CD"
n$ = n$ + "C L8 < B L16 B > C < BA > C < BAF L8 F L16 FF L8 FA"
n$ = n$ + "L16 > C < ABG L8 F L16 FF L8 FA L16 BG"
n$ = n$ + "AF L8 D L16 DC L4 D L8 D L16 DD L8 DF"
n$ = n$ + "L16 AFGE L8 D L16 DC L4 D L8 D"
n$ = n$ + "L16 DC L8 D L16 EF L2 G L16 GFED L3 C"
PLAY n$
```

Der Tempo-Befehl in Zeile 3 (T80) legt fest, wie schnell das Stück gespielt wird. Versuche auch andere Werte (T32 bis T255).

38. Grafische Figuren - der DRAW-Befehl

Neben den bereits erwähnten Grafikbefehlen (PSET, LINE, CIRCLE) gibt es noch einen weiteren Grafikbefehl, den DRAW-Befehl. Diesem Befehl wird ähnlich wie beim PLAY-Befehl eine Zeichenkette übergeben, z.B. DRAW "U10".

Innerhalb der Zeichenkette können folgende Zeichenbefehle verwendet werden:
U10: zeichnet eine Linie um 10 Punkte nach oben
D10: zeichnet eine Linie um 10 Punkte nach unten
R10: zeichnet eine Linie um 10 Punkte nach rechts
L10: zeichnet eine Linie um 10 Punkte nach links
E10: zeichnet eine Linie um 10 Punkte nach rechts oben
F10: zeichnet eine Linie um 10 Punkte nach rechts unten
G10: zeichnet eine Linie um 10 Punkte nach links unten
H10: zeichnet eine Linie um 10 Punkte nach links oben

Wird einem der obigen Zeichenbefehle der Buchstabe B vorangestellt, so wird keine Linie in der aktuellen Farbe gezeichnet, sondern der Zeichenstift wird nur in die entsprechende Richtung bewegt.
Um den Zeichenstift an eine bestimmte Position zu setzen, verwendet man den Move-Befehl (M320,200 bewegt den Zeichenstift an die Position 320,200).

Weitere Zeichenbefehle:
C4:

P2,3:
verwendet die Farbe 4
füllt an der Stiftposition mit der Farbe 2 bis zum Rand mit der Farbe 3
1.Gib das folgende Programm ein und starte es:

```
REM Draw1
SCREEN 12
CLS
DRAW "BM320,240"
DRAW "C15 G50"
DRAW "U100 E50 F50 D100 L100 E100 L100 F100"
```

2.Erweitere das Programm wie folgt. Versuche in Gedanken die Bewegungen des Zeichenstiftes nachzuvollziehen.

```
REM Draw2
SCREEN 12
CLS
DRAW "BM320,240"
DRAW "C14 G50"
DRAW "U100 E50 F50 D100 L100 E100 L100 F100"
DRAW "BU110 BL50"
DRAW "P4,14"
```

39. Figuren zeichnen und löschen

Nahezu alle Spielprogramme laufen in einem der zahlreichen Grafik-Modi des Computers. Als Standard-Modus wählen wir SCREEN 12. Dabei ist der Bildschirm in 640 Spalten und 480 Zeilen eingeteilt. Die Koordinate links oben ist (0/0), die Koordinate rechts unten (639,479).

1.Das folgende Programm ist die Basis, um eine grafische Figur über den Bildschirm zu bewegen. Es zeigt, wie man eine Figur (hier: eine Rakete) zeichnen und danach auch wieder löschen kann.

```
REM Rakete1
SCREEN 12
x = 320
y = 150
ZeichneRakete x, y
DO
  a$ = INKEY$
LOOP UNTIL a$ = CHR$(27)
LoescheRakete x, y
DO
  a$ = INKEY$
LOOP UNTIL a$ = CHR$(27)
SCREEN 0
END
```

Das Hauptprogramm stellt den Bildschirmmodus ein, legt die Position der Rakete fest (Position 320,150), zeichnet die Rakete und wartet dann auf das Betätigen der Esc-Taste. Anschließend wird die Rakete gelöscht (durch erneutes Zeichnen mit der Farbe schwarz). Bei erneutem Betätigen der Esc-Taste wird das Programm beendet.

Die Programm-Teile zum Zeichnen und Löschen der Rakete sind ausgelagert in Unterprogramme (Subs). Ohne diese Subs ist das Programm nicht lauffähig. Beiden Unterprogrammen werden Parameter übergeben (x,y). Diese stellen die jeweilige Position der Rakete auf dem Bildschirm dar. Beachte, dass die beiden Unterprogramme nahezu identisch sind, so dass der Code für das zweite Unterprogramm durch Bearbeiten Kopieren - Bearbeiten Einfügen erzeugt werden kann und anschließend angepaßt werden kann.

```
SUB LoescheRakete (x, y)
  DRAW "BM" + STR$(x) + "," + STR$(y)
  DRAW "C0 G20 D50 G4 D20 R10 U10 E4 R10"
  DRAW "R10 F4 D10 R10 U20 H4 U50 H20"
  DRAW "BD10 P0,0"
END SUB

SUB ZeichneRakete (x, y)
  DRAW "BM" + STR$(x) + "," + STR$(y)
  DRAW "C14 G20 D50 G4 D20 R10 U10 E4 R10"
  DRAW "R10 F4 D10 R10 U20 H4 U50 H20"
  DRAW "BD10 P4,14"
END SUB
```

Die Rakete wird mit der Farbe 14 (gelb) gezeichnet und mit der Farbe 4 (rot) gefüllt. Beim Löschen wird sowohl für den Rand als auch das Innere der Rakete die Farbe schwarz verwendet.

40. Bewegte Figuren

1.Erweitere das Programm Rakete1 wie folgt und starte es.

```
REM Rakete2
SCREEN 12
x = 320
y = 200
ZeichneRakete x, y
DO
  DO
    a$ = INKEY$
  LOOP UNTIL a$ <> ""
  IF LEN(a$) > 1 THEN a$ = MID$(a$, 2, 1)
  LoescheRakete x, y
  SELECT CASE ASC(a$)
    CASE 80   'nach unten
      y = y + 4
    CASE 72   'nach oben
      y = y - 4
    CASE 75   'nach links
      x = x - 4
    CASE 77   'nach rechts
      x = x + 4
  END SELECT
  ZeichneRakete x, y
LOOP UNTIL a$ = CHR$(27)
LoescheRakete x, y
SCREEN 0
END
```

Das Programm stellt zuerst die Rakete dar, wartet auf das Drücken einer gültigen Taste, löscht dann die Rakete, verändert je nach Richtungstaste die Koordinaten x und y der Rakete und zeichnet dann die Rakete neu. Wird die äußere DO...LOOP -Schleife durch Drücken der Esc-Taste verlassen, dann wird die Rakete gelöscht und das Programm beendet.

2.Das Programm Rakete2 hat den Nachteil, dass die Rakete außerhalb des Bildschirmbereichs bewegt werden kann. Um zu verhindern, dass die Rakete den Bildschirm-Rahmen verlassen kann, werden nach dem END SELECT-Befehl an die folgenden Zeilen eingefügt.

```
IF x < 24 THEN x = 615
IF x > 615 THEN x = 24
IF y < 0 THEN y = 0
IF y > 385 THEN y = 385
```

3.Füge nach dem SCREEN-Befehl die folgende Programm-Zeile ein und ergänze das Programm um das Unterprogramm ZeichneSterne.

```
SCREEN 12
ZeichneSterne 3000
:


SUB ZeichneSterne (n)
  FOR i = 1 TO n
    PSET (RND * 640, RND * 480), RND * 13 + 2
  NEXT i
END SUB
```

*** * ***