

QBasic - Programmierkurs von Ludger Mählmann

QBasic ist ein Dialekt der Programmiersprache BASIC (= Beginners All purpose Symbolic Instruction Code = Schneller Anfänger - Allzweck - symbolischer - Anweisungscode). BASIC ist ohne Zweifel nicht nur die am weitesten verbreitete, sondern auch die am leichtesten erlernbare Sprache für die Programmierung von Computern.

Der QBasic Interpreter ist auf den DOS - basierten Betriebssystemen (MS Dos 5.0 bis Windows 9x) als Beilage vorhanden. Ein Interpreter ist ein Programm, welches deine QBasic-Befehle sofort ausführt- sie werden Schritt für Schritt übersetzt und ausgeführt.

I N H A L T

- **Lektion 1: erste Schritte**
 - Rechnen mit QBasic
 - Formatierte Bildschirmausgaben
- **Lektion 2: Schleifen**
 - GOTO
 - IF ... THEN
 - FOR ... NEXT
 - DO ... LOOP
- **Lektion 3: Unterprogramme**
 - interne Unterprogramme (GOSUB)
 - externe Unterprogramme (SUB und FUNCTION)
- **Lektion 4: Strings**
 - Zeichenketten
 - Zeichenkettenfunktionen
- **Lektion 5: Grafik**
 - Textmodus
 - Grafikmodus
 - Rechteck
 - Kreis
- **Lektion 6: Achsenkreuz**
 - Achsenkreuz
 - Punkte
 - Geraden
- **Lektion 7: Musik**
 - Klänge erzeugen (SOUND)
 - Klänge spielen (PLAY)
- **Übersicht der Befehle**
- **Übungsaufgaben**

• QBasic - erste Schritte

Rechnen mit QBasic

Rufe QBasic auf und. schreibe in den Editor dein erstes Programm, den folgenden Dreizeiler:

```
LET a=105
LET b=5
PRINT a+b
```

Starte das Programm mit 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 (110) aus. Füge an das Ende des Programmes die folgenden Zeilen an:

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

Speichere das Programm unter **rechnen1.bas** in deinen Ordner. 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 jedesmal 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.

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
```

Eingaben

Starte QBasic neu, tippe in das Programmfenster die folgenden Zeilen ein und speichere unter **power1.bas**:

```
REM PS-Rechner
CLS
PRINT "Kilowatt", "PS"
PRINT
INPUT "Eingabe Kilowatt",k
LET p=k*1.36
PRINT k, p
```

Das Programm dient zur Umrechnung von Kilowatt in PS. 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 k der eingegebene Wert (z.B. 82.4) zugewiesen. Beachte, daß Kommazahlen mit einem Dezimalpunkt eingegeben werden. Die LET - Zeile weist der Variablen p den berechneten Wert zu. PRINT gibt schließlich die eingegebene Kilowattangabe und den berechneten PS - Wert auf dem Bildschirm aus.

Formatierte Bildschirmausgaben

Gebe folgende vier Zeilen ein und speichere unter **name.bas**:

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

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
```

Speichere unter **tabulat.bas**. Die TAB-Funktion setzt die Schreibmarke in eine bestimmte Bildschirmspalte. Die folgende Bildschirmausgabe erfolgt dann genau an dieser Stelle. Weiter mit [Lektion2](#)

Aufgaben:

1. Schreibe ein Programm, welches bei Eingabe von zwei Zahlen a und b folgendes ausgibt: Summe, Differenz, Produkt, Quotient. Speichere das Programm unter **rechnen2.bas**
2. Schreibe ein Programm, welches bei Eingabe der Kantenlängen a, b, c eines Quaders die Oberfläche, das Volumen und die Raumdiagonale des Quaders ausgibt. Nenne das Programm **quader.bas** Übrigens brauchst du Für die Raumdiagonale folgende Wurzelfunktion: $\text{sqr}(x)$.

3. Ändere das Programm **power1.bas** (PS - Rechner) so ab, dass nicht die Kilowattzahl eingegeben wird, sondern die PS - Zahl. Speichere das Programm unter dem Namen **power2.bas**
4. 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 Programm unter dem Namen **plan.bas**

• QBasic - Schleifen

GOTO Endlosschleife

Zur Berechnung mehrerer Werte kann man eine sogenannte Endlos-Schleife (GOTO) verwenden. Ändere dazu das Programm **power2.bas** aus Lektion 1 wie folgt ab und speichere unter **power3.bas**:

```
REM
CLS
PRINT "PS", "Kilowatt"
PRINT
Hierher:
INPUT "Eingabe PS",p
LET k=p*0.735
PRINT p, k
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, der eine erneute Eingabe fordert, dann folgt die Umrechnung von PS in Kilowatt. 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 Tastenkombination Strg+C verwenden.

Für ein reguläres Ende braucht der Computer Bedingungen, unter denen er entscheiden kann, wann Schluss ist.

IF - THEN Entscheidung

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....

Tippe dazu folgendes Programm ein und speichere das Programm unter dem Namen **raten1.bas**

```
REM Zahlenraten
CLS
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).

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 (**zufall.bas**).

```
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).

Das folgende Programm ermöglicht nur maximal 20 Rateversuche. Tippe das Programm ein und 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 obige Programm wird beendet, wenn man die Zahl errät oder wenn man bereits

20 Zahlen eingegeben hat.

Die IF... THEN...-Anweisung kann um die ELSE-Anweisung erweitert werden. Die englischsprachigen 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 hierstehende Anweisung aus,
SONST führe die hierstehende 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.

FOR ... NEXT - Schleife

Starte QBasic und schreibe folgendes einfaches Programm (**addi1.bas**), welches fünf eingegebene Zahl aufaddiert und dann die Summe ausgibt:

```
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 jedesmal neu getippt werden. Starte das Programm, und addiere damit die Zahlen 23, 56, 78, 99 und 204. Die Summe ist:

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 *zaehler* auf, die bis 5 zählt und dann das Programm über eine IF THEN Schleife anhält (**addi2.bas**):

```
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. Die FOR...NEXT-Schleife kann immer dann eingesetzt werden, wenn die Anzahl der Wiederholungen schon im voraus feststeht.

Ändere das vorige Programm wie folgt ab. Dieses Programm (**addi3.bas**) 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.

Soll die Zählvariable um einen anderen als den voreingestellten Wert 1 erhöht werden, verwendet man den STEP-Zusatz. Was bewirkt das folgende Programm (**folge1.bas**)?

```
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.

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?

Füge an Zeile 3 des vorigen Programmes den Zusatz STEP -1 an und starte das Programm erneut. Speichere unter **folge2.bas**


```
REM Schleife 2
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.

DO ... LOOP ... UNTIL - Schleife

Diese Schleife rahmt einen Satz von Befehlen ein, welcher so oft durchlaufen wird, bis die Aussage wahr ist – also True ergibt. Hier einmal ein Beispiel (**hallo.bas**):

```
REM Hallo
CLS
INPUT "Bitte geben Sie eine Zahl ein: ", anzahl
DO
x = x + 1
PRINT "Hallo (; x; )"
LOOP UNTIL x = anzahl
END
```

Dieses Programm gibt das Wörtchen "Hallo" mit in Klammern geschriebener Zählvariable aus. Ist die Zählvariable so groß wie die eingegebene Zahl, endet das Programm.

Im folgenden Beispielprogramm (**sterne.bas**) werden erste Grafikbefehle verwendet: im Grafikmodus SCREEN 12 wird der Bildschirm 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.

LINE (a,b)-(x,y),5, B zieht ein Rechteck zwischen den Punkten (a,b) und (x,y) mit der Farbe 5.

```
REM Sterne
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
```

Weiter gehts mit Unterprogrammen in [Lektion3!](#)

Aufgaben

1. Ändere das Programm Zahlenraten (***raten1.bas***), indem du Zufallszahlen zwischen 1 und 100 berechnen läßt, und speichere das Programm unter ***raten2.bas***
 2. Schreibe ein Programm, welches 6 Lottozahlen aus 49 Zahlen ausgibt. Speichere unter ***lotto.bas***
 3. Erstelle ein Programm mit einer verschachtelten FOR NEXT Schleife, welches das kleine Einmaleins auf dem Bildschirm darstellt Speichere unter ***einmal.bas***. Verwende bei der Bildschirmausgabe die Tabulatorfunktion!
-

• QBasic - Unterprogramme, Subs und Functions

Interne 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**.

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. Tippe das Programm ein und speichere unter **sub1.bas**

```
REM Sub1
Hauptprogramm:
CLS
zaehler=0
PRINT "Drei Chinesen mit dem Kontrabaß,"
GOSUB Linie
PRINT "saßen auf der Straße und erzählten sich
was,"
GOSUB Linie
PRINT "da kam die Polizei, na was ist denn dass?,"
GOSUB Linie
PRINT "Drei Chinesen mit dem Kontrabaß."
GOSUB Linie
PRINT "Das Unterprogramm wurde ";zaehler;"-mal
aufgerufen"
END

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

Das folgende Programm gibt nach Eingabe von Namen und Noten ein Minizeugnis aus. Die Eingabe erfolgt im Hauptprogramm, die Zuordnung der Schriftnoten im Unterprogramm *noten*. Probiere das Programm aus und speichere unter **zeugnis1.bas**

```
REM Zeugnis schreiben
CLS
INPUT "Name des Schülers: ", schueler$
INPUT "Deutschnote? ", deu
INPUT "Englischnote? ", eng
INPUT "Mathenote? " , math
CLS
PRINT TAB(15); "Zeugnis"
PRINT
PRINT "Name des Schülers: "; schueler$
PRINT
```

```

PRINT "Deutsch ";
zens = deu
GOSUB noten
PRINT "Englisch ";
zens = eng
GOSUB noten
PRINT "Mathe ";
zens = math
GOSUB noten
END

noten:
IF zens = 1 THEN PRINT TAB(20); "sehr gut"
IF zens = 2 THEN PRINT TAB(20); "gut"
IF zens = 3 THEN PRINT TAB(20); "befriedigend"
IF zens = 4 THEN PRINT TAB(20); "ausreichend"
IF zens = 5 THEN PRINT TAB(20); "mangelhaft"
IF zens = 6 THEN PRINT TAB(20); "ungenügend"
RETURN

```

Externe Unterprogramme (Sub und Function)

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

```

analog 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. Es ist auch möglich, innerhalb von Subs und Functions weitere Subs zu schreiben. Zwischen den Subs, Functions und dem Hauptmodul des Programms kann mit Ansicht - SUBs ... oder der **F2-Taste** umgeschaltet werden.

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.

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 gestartet werden.

Weiter gehts mit [Lektion 4!](#)

Aufgaben

1. Das Programm Zeugnis soll ein vollständiges Zeugnis in zwei Spalten ausdrucken. Ergänze das Programm **zeugnis1.bas** mit den Nebenfächern und speichere unter **zeugnis2.bas**
2. Die Post befördert Briefe nach Gewicht. Schreibe einen Portorechner, der bei Eingabe des Gewichts das Porto ausgibt. Schreibe dazu folgende Zuordnungen ähnlich dem Programm Zeugnis in ein Unterprogramm:
- 3.

- ist das Gewicht < 20g kostet der Brief 55 Cent
- ist das Gewicht < 50g kostet der Brief 1 Euro
- ist das Gewicht < 500g kostet der Brief 1,44 Euro
- ist der Brief < 1000g kostet der Brief 2,20 Euro
- ist der Brief > 1000g, spricht man von einem Paket!

Speichere das Programm unter **porto.bas** und überlege: macht hier ein Unterprogramm Sinn?

4. 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 **kgv.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.

• QBasic - Strings

Zeichenketten (Strings)

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.

Starte gegebenenfalls QBasic und gib die folgenden Programmzeilen ein. Speichere das Programm (**strings1.bas**), starte das Programm und achte auf das Resultat.

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

Das Programm addiert nicht die Zahlen 2 und 4! Es ordnet den Stringvariablen zahl1\$ und zahl2\$ die Zeichen 2 bzw. 4 zu und gibt die Ziffern hintereinander aus.

Gib nun die folgenden Programmzeilen (**strings2.bas**) ein. Starte das Programm und achte auf das Resultat.

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

Die Ausgabe erlaubt eine Kombination von Stringvariablen und Zeichenketten, mit dem Pluszeichen verknüpft. Achtung: eine Mischung von Strings und Zahlenvariablen verknüpft man nicht mit einem Plus. Will man sie in einer Zeile darstellen, verwendet man das Semikolon.

Gib die folgenden Programmzeilen (**strings3.bas**) ein. Starte das Programm und achte auf das Resultat.

```
REM strings3
CLS
INPUT "Nachname: " , nname$
INPUT "Vorname: " , vname$
INPUT "Alter: " , x
GOSUB einstuftung
PRINT "Hallo, " + vname$ + " " + nname$ " , du bist  
"; x ; " Jahre alt "  
PRINT "und damit " + c$
```

```
END
```

```
einstufung:
IF x <= 12 THEN c$ = "ein Kind."
IF x > 12 AND x <= 19 THEN c$ = "ein Teenie!"
IF x > 19 AND x <= 29 THEN c$ = "ein Tweenie!"
IF x > 29 THEN c$ = "eine alte Socke!"
RETURN
```

Zeichenkettenfunktionen

Mit der Funktion ASC(Zeichen\$) kann Basic den ASCII - Code eines Zeichens ausgeben. Schreibe das folgende Programm und speichere unter **ascii1.bas**

```
REM ascii1
CLS
INPUT "Gib1 ein beliebiges Zeichen ein! ",
zeichen$
PRINT
PRINT "Du hast das Zeichen " + zeichen$ + "
eingegeben. "
PRINT "Sein ASCII-Code lautet";
PRINT ASC(zeichen$);
PRINT "."
```

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.

Das folgende Programm (**ascii2.bas**) gibt alle wichtigen Zeichen des Zeichensatzes aus.

```
REM ascii2
CLS
PRINT "Die wichtigsten Zeichen des Computers:"
FOR n=32 TO 255
PRINT CHR$(n);
NEXT n
```

Prüfe auch, wie die Zeichen 1 bis 6, 24 bis 27 aussehen. Erweitere dazu das Programm ascii2 um die entsprechenden Zeilen. Weiter mit [Lektion 5](#)!

Aufgaben

1. Schreibe mit Stringvariablen und Variablen das Programm **strings4.bas**, welches dazu auffordert, deine Adresse einzugeben (Name, Vorname, Straße (Stringvariablen), Hausnummer, Postleitzahl (Zahlvariablen)). Schreibe ein SUB, welche für fünf Postleitzahlen Städte zuordnet. Das Programm soll die vollständige Adresse wieder ausgeben.
2. Schreibe das Programm **ascii3.bas**, welches das Alphabet mit em entsprechenden ASCII - Zeichen ausgibt. Ein kleiner Tipp: Der Buchstabe A hat das ASCII - Zeichen 65!

• QBasic - Grafik

Der Bildschirm eines PCs kann unter QBasic in verschiedenen Modi betrieben werden. Man unterscheidet den Text- und den Grafikmodus.

Textmodus

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`.

Das folgende Programm ***tmodus1.bas*** stellt den Modus 0 ein und beschreibt alle möglichen Zeilen (bis auf die letzte) mit zufälligen Zeichen des Alphabets. Starte das Programm mehrfach und gib für die Anzahl der Zeilen und Spalten unterschiedliche Werte ein. Beobachte die Bildschirmausgaben.

```
REM tmodus1
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). Mit dem Befehl `COLOR` (Vordergrundfarbe, Hintergrundfarbe) können die Farben für Zeichen (Vordergrundfarbe) und Zeichenhintergrund (Hintergrundfarbe) bestimmt werden. Folgendes Programm (***tmodus2.bas***) gibt die Zahlen von 0 bis 1000 in Fünferschritten mit gewünschter Farbe aus:

```
REM tmodus2
SCREEN = 0
CLS
INPUT "Farbe der Zeichen: ", vordergrundfarbe
INPUT "Hintergrundfarbe: ", hintergrundfarbe
FOR zaehler = 0 TO 1000 STEP 5
COLOR vordergrundfarbe, hintergrundfarbe
PRINT zaehler;
NEXT zaehler
```

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`.

Das folgende Programm (***tmodus4.bas***) gibt verschiedene Texte an verschiedenen Stellen des Bildschirms aus. Beachte, daß mehrere Befehle in einer Zeile durch

einen Doppelpunkt voneinander getrennt werden.

```
REM tmodus4
SCREEN 0
WIDTH 80,25
CLS
LOCATE 10,1: PRINT "Zeile 10, Spalte 1"
LOCATE 15,20: PRINT "Zeile 15, Spalte 20"
```

Das folgende Programm (**tmodus5.bas**) stellt verschiedene Texte zentriert, d.h. in der Bildschirmmitte, dar. Dazu wird das Unterprogramm Center aufgerufen. Der Befehl LEN gibt dabei die Anzahl des Strings text\$ inklusive Leerzeichen wieder.

```
REM tmodus5
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
```

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.

Das folgende Programm (**gmodus1.bas**) zeichnet bei SCREEN 8 (640*200, 16 Farben) einen gelben Rahmen.

```
REM gmodus1
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$ <> ""
```

Das folgende Programm (**gmodus3.bas**) zeichnet bei SCREEN 12 (640*480, 16 Farben) eine Sinuskurve:

```
REM gmodus3
SCREEN 12: CLS
farbe = 12
FOR x= 0 TO 639
y = 220+210*SIN(x*x/10000)
NEXT x
```

Rechteck

Das folgende Programm zeichnet bei SCREEN 12 (640*480, 16 Farben) ein Rechteck. Die linke obere Ecke hat die Koordinate (20,40), die rechte untere Ecke die Koordinate (500,400). Das Rechteck ist blau (9) und gefüllt (BF). Speichere unter **rechteck1.bas**

```
REM rechteck1
SCREEN 12
CLS
LINE (20, 40) - (500, 400), 9, BF
```

Das folgende Programm (**rechteck3.bas**) zeichnet bei SCREEN 12 (640*480, 16 Farben) 20 verschiedene Rechtecke an zufälligen Bildschirmpositionen.

```
REM rechteck3
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
```

Kreis

Mit dem CIRCLE Befehl kann man an einer gewünschten Bildschirmposition einen Kreis mit einem beliebigen Radius und einer beliebigen Farbe zeichnen. Das folgende Programm (**kreis1.bas**) zeichnet bei SCREEN 12 (640*480, 16 Farben) nach Eingabe von Farbe und Radius (in Pixeln) zentriert auf dem Bildschirm einen Kreis!

```
REM kreis1
SCREEN 12
CLS
INPUT "Radius (0 - 240): ", radius
INPUT "Farbe (0-15): " , farbe
CIRCLE (320,240), radius, farbe
```

Das Programm **kreis3.bas** erzeugt 40 verschiedene Kreise mit beliebigen Farben an zufälligen Bildschirmpositionen.

```
REM kreis3
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
```

Ändere die vorletzte Zeile des Programms **kreis3.bas** wie folgt ab und speichere unter **kreis4.bas**:

```
:
CIRCLE (x, y), radius, farbe, , , .5
PAINT (x,y), farbe
NEXT kreis
```

Der CIRCLE-Befehl wird im obigen Beispiel mit 7 Parametern aufgerufen. x und y sind die Koordinaten des Mittelpunktes, radius ist der Kreistradius, 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). Mit dem PAINT - BEFEHL kann man einen Bereich ausfüllen. Die Füllfarbe wird hier zufällig erzeugt.

Weiter mit Lektion 6!

Aufgaben

1. Ergänze das Programm **tmodus1.bas** wie folgt: Nach Eingabe der Spalten- und Zeilenzahl sowie der Hintergrundfarbe (Zahl von 1 bis 10) soll das Programm das Alphabet mit gewünschtem Hintergrund in zufälliger Reihenfolge und mit zufälligen Farben ausgeben. Verwende dazu den Befehl COLOR, wobei die Vordergrundfarbe mit dem Befehl INT(15*RND) erzeugt wird. Speichere unter **tmodus3.bas**
2. Schreibe ein Programm **tmodus6.bas**, welches bei Eingabe der Spalten- und Zeilenzahl (siehe tmodus1) den Text aus tmodus5 sowohl in Zeilen als auch in Spalten zentriert wiedergibt. Wähle dazu eine rote Schrift (12) und einen grauen Texthintergrund (15)
3. Schreibe das Programm **gmodus2.bas**, welches bei SCREEN 9 (640*350, 16 Farben) ein Kästchenpapier erzeugt. Ein Tipp: Ändere gmodus1.bas um, erzeuge mit zwei FOR NEXT Schleifen waagerechte und senkrechte Linien im Abstand von 16 Pixeln. Die Abbruchbedingung übernimm aus gmodus1.bas
4. Experimentiere mit den mathematischen Funktionen ähnlich gmodus3. Versuche cos, tan und sqr darzustellen!
5. Schreibe das Programm **rechteck2.bas**, welches zur Eingabe der Seitenlängen auffordert. Flächeninhalt und Umfang sollen ausgegeben werden. Dann soll das Rechteck zentriert auf dem Bildschirm gezeichnet werden.
6. Schreibe das Programm **kreis2.bas**, welches zur Eingabe des Radius auffordert. Flächeninhalt, Umfang und Kreis sollen ausgegeben werden.

7. Ändere das obige Programm ***kreis4.bas*** so ab, daß 99 Luftballons gezeichnet werden. Wähle dazu als letzten Parameter 1,6. Speichere unter ***kreis5.bas***
-

• QBasic - Geometrie

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).

Das folgende Programm zeichnet ein beschriftetes Achsenkreuz in der Farbe gelb im Bereich von -5 bis 5 (X - Wert) und -4 bis 4 (y - Wert). Speichere unter **akreuz.bas**

```

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

```

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.

Punkte im Achsenkreuz

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.

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

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. Speichere unter **akreuz1.bas**

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.

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 (**akreuz3.bas**):

```
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.

Geraden im Achsenkreuz

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

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

```
Connect x1,y1,x2,y2,f
```

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
```

Weiter mit [Lektion 7](#)

Aufgaben

1. Ändere das Programm **akreuz1.bas**, dass es bei Eingabe von x und y den entsprechenden Punkt darstellt. Speichere unter **akreuz2.bas**
2. Schreibe ein Programm **akreuz4.bas**, welches bei Eingabe von zwei Koordinaten die entsprechende Linie zeichnet. Darüber hinaus soll das Programm die Steigung a, den Schnittpunkt mit der y Achse (b) und die Funktionsgleichung ausgeben. Dazu die folgenden Hinweise:

- $a = (y_2 - y_1) / (x_2 - x_1)$
- $b = (y_2 - y_1)$
- Funktionsgleichung: $y = ax + b$

• QBasic - Musik

Zur Ausgabe von Tönen über den eingebauten PC-Lautsprecher (leider nicht über Soundkarte) gibt es in QBasic zwei Befehle: SOUND und PLAY. Während SOUND die Töne über eine eingegebene Frequenz ausgibt, kann man mit PLAY Tonhöhe und Länge über die Tonleiter ausgeben.

SOUND - Befehl

Der SOUND - Befehl erwartet zwei Zahlen/ Variablen. Die erste Zahl gibt die Tonfrequenz an, (440 Hz für den Kammerton A), die zweite Zahl beschreibt die Tondauer. Beachte, daß die Frequenz (erste Zahl) zwischen 37 und 32767 liegen muß und die Tondauer (zweite Zahl) zwischen 0 und 65535. Gib die folgenden Programme ein, speichere sie unter **sound1.bas**, **sound2.bas** und **sound3.bas** und starte sie. Experimentiere dabei mit der Tonfrequenz und der Tondauer.

```
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 "Sirenengeheul ..."
FOR zaehler = 440 TO 1000 STEP 5
SOUND zaehler, zaehler / 1000
NEXT zaehler
```

PLAY - Befehl

Mit dem PLAY Befehl läßt sich einfach die Tonleiter spielen. Bei der Eingabe des Programms (**leiter1.bas**) achte darauf, dass die englische Tonleiter kein H kennt.

Schreibe stattdessen B.

```
REM tonleiter1
PLAY "CDEFGABC"
```

Du wirst gehört haben, dass dein Programm kein hohes C spielt. Der Computer braucht einen Befehl für eine neue Oktave: ox für $1 \leq x \leq 6$ legt die Oktave fest. Ändere das Programm wie folgt um (**leiter2.bas**)

```
REM tonleiter2
PLAY "o3CDEFGABo4C"
```

Das Eingeben der gesamten Klaviatur wäre somit kein Problem mehr. Mit weiteren Befehlen kannst du sogar Tonlänge, Pausen etc. bestimmen. Hier eine Übersicht:

o3	legt die Oktave fest (0 - 6), o5 ist die 3. Oktave
L4	legt die Tonlänge fest (1 - 64), L1 ist eine ganze Note, L2 eine Halbe, ...
P2	legt die Pausen fest (1-64)
ML	spielt die Töne gebunden (legato)
MN	spielt die Töne normal
MS	spielt die Töne Staccato (abgehakt)
CDEFGAB	legt die Tonhöhe fest (B = H!)
T80	Tempobefehl (32 - 255)

Kannst du folgende Melodie (**melodie1.bas**) erraten?

```
PLAY "MS o3 L4 eefggfedccde L3 e L6 d L2 d"
```

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. Am besten ist du experimentiert einfach ein bißchen, und schon gelingen (falls man das Talent hat) die ersten Eigenkompositionen.

Das folgende Programm ermöglicht es, über den Nummernblock Klavier zu spielen. Du kannst mit der Taste q das Programm abbrechen. (**klavier.bas**)

```
REM Klavier1
CLS
PRINT " Klavierstunde"
taste$ = ""
WHILE taste$ <> "q"
taste$ = INKEY$
IF taste$ = "1" THEN PLAY "o3 L8 C"
IF taste$ = "2" THEN PLAY "o3 L8 D"
IF taste$ = "3" THEN PLAY "o3 L8 E"
...
WEND
```

Aufgaben

1. Ändere das Programm **sound1.bas** so um, dass eine Alarmanlage zu hören ist. (**alarm.bas**) Nimm für den ersten Ton 440 Herz, für den zweiten 640 Herz. Für die Tonlänge reicht der Wert 5. Experimentiere mit den Parametern!
 2. Ergänze die Melodie "Freude schöner Götterfunken" im Programm Melodie1 und speichere unter **melodie2.bas**
 3. Schreibe dir deinen eigenen Handy - Klingelton: Suche dir aus dem Musikbuch eine Melodie deiner Wahl und gebe sie als Programm ein! (**melodie3.bas**)
 4. Erweitere das Programm Klavier, so dass du mindestens 10 Töne spielen kannst und speichere unter **klavier.bas**
-

• QBasic - Befehle

Lektion 1: erste Schritte

LET a=104	weist der Variablen a den Wert 104 zu
PRINT "Hallo"	Schreibt die Zeichenkette Hallo in die Zeile. Anstatt Hallo kann auch eine Variable oder Zahl stehen.
CLS	Clear Screen = Bildschirminhalt löschen, sollte am Anfang von jedem Textmodusprogramm stehen
INPUT "Zahl?", x	zeigt wie PRINT den Text Zahl an, wartet danach aber auf die Eingabe des Benutzers. Diese Eingabe wird dann unter der nebenstehenden Variablen x gespeichert
TAB (x)	setzt die Bildschirmausgabe in die Bildschirmspalte x
SQR (x)	Square = Quadratwurzel, berechnet die Wurzel der Zahl x

Lektion 2: Schleifen

GOTO ... Sprungmarke:	durchläuft Programm von GOTO bis Sprungmarke. Achtung! Ohne IF Anweisung läuft das Programm endlos weiter (hängt sich auf)
IF a=b THEN ...	wenn a=b gilt, dann tue ...
IF a<b THEN ... ELSE ...ENDIF	wenn a<b gilt, dann tue ... sonst (a>=b) tue ...
FOR x=1 TO 10 ... NEXT x	für x = 1 bis 10 tue ...
DO ... LOOP UNTIL x<2	die Schleife wird so lange ausgeführt, bis x<2 ist
RANDOMIZE TIMER	initialisiert den Zufallsgenerator
RND	liefert eine Zahl zwischen 0 und 1
INT(RND*x)+1	erzeugt eine ganzzahlige Zufallszahl zwischen 1 und x
STEP	in der FOR Next Schleife
SCREEN (12)	schaltet den den 640 x 480 Pixel Grafikmodus um bei 16 Farben
PSET (x,y), 5	setzt auf der Bildschirmkoordinate einen Punkt mit der Farbe 5
LINE (a,b) - (x,y), 4, B	zieht ein Rechteck (B) mit der Farbe 4 von links oben (a,b) nach rechts unten (x,y). Schreibt man BF anstatt B, wird das Rechteck ausgefüllt.
LINE (a,b) - (x,y), 4	zeichnet nur eine Linie mit der Farbe 4 (rot)

Lektion 3: Unterprogramme

GOSUB klein	ruft das interne Unterprogramm <i>klein</i> auf
klein:	klein: leitet das Unterprogramm <i>klein</i> ein, RETURN weist zum Hauptprogramm zurück
...	
RETURN	
SUB Name	hiermit beginnt ein externes Unterprogramm, <i>Name</i> muss im Hauptprogramm vorkommen
END SUB	Ende des Unterprogramms

FUNCTION Name	Beginn des externen Unterprogramms <i>Name</i> , <i>Name</i> muss im Hauptprogramm vorkommen
END FUNCTION	Ende des externen Unterprogramms
INT (x/y)	gibt die Ganzzahl des Quotienten x/y wieder!

Lektion 4: Strings

a	Variable, Platzhalter für eine Zahl
a\$	Stringvariable, Platzhalter für ein(e) Zeichen(kette)
ASC(a\$)	gibt den ASCII - Zahlenwert für das Zeichen heraus, welches sich hinter der Variablen a\$ verbirgt
CHR\$(n)	gibt das ASCII - Zeichen für den Zahlenwert n heraus (1<n<255)

Lektion 5: Grafik

SCREEN	verschiedene Bildschirmmodi von 0 bis 12
WIDTH spalte, zeile	gibt die Größe des Bildschirms an!
COLOR vordergrundfarbe, hintergrundfarbe	färbt Zeichen und Zeichenhintergrund in der angegebenen Farbe (je nach Modus meist von 1 - 16)
LOCATE x,y	positioniert die folgenden zeichen oder Zahlen auf der Koordinate x, y
LEN (a\$)	gibt die Länge der Stringvariable inklusive Leerzeichen aus
SIN (x)	berechnet den Sinus von x
COS (x)	berechnet den Cosinus von x
TAN (x)	berechnet den Tangens von x
SQR (x)	berechnet die Quadratwurzel von x
CIRCLE (x,y), radius, farbe	zeichnet einen Kreis um den Mittelpunkt (x,y)
PAINT (x,y), füllfarbe, randfarbe	füllt einen Kreis um den Punkt (x,y) aus

Lektion 7: Musik

SOUND a, b	spielt einen Ton mit der Frequenz a und der Tonlänge b
PLAY " T80 o3 L4 G P4"	spielt im Tempo 80 auf der dritten Oktave eine viertel Note G und eine viertel pause

• QBasic - Übungsaufgaben

1. Der Umfang und der Flächeninhalt eines Rechtecks (Dreiecks, Trapez', ...) soll bei Eingabe entsprechender Größen berechnet werden. (**rechteck.bas**)
2. Von einem Würfel sollen Oberfläche, Volumen, Summe der Kantenlängen und Raumdiagonale berechnet werden. (**wuerfel.bas**)
3. Schreibe ein Programm, welches bei Eingabe der Lebensjahre das Lebensalter in Tagen ausgibt. Beachte dabei die Schaltjahre! (**jahr.bas**)
4. Schreibe ein Programm, bei dem der Computer
 - von 1 bis 12 zählt, d.h. die Zahlen von 1 bis 12 ausgibt (**zaehler1.bas**)
 - rückwärts von 10 bis 0 zählt (**zaehler2.bas**)
5. Es sollen alle Quadratzahlen von einem Anfangswert A bis zu einem Endwert E bestimmt werden und in eine Wertetabelle ausgegeben werden. (**qzahl.bas**)
 - Verwende die FOR - NEXT - Schleife
 - Eine Zählvariable x wird auf den Anfangswert a gesetzt und bis zum Endwert automatisch um 1 erhöht
6. Erstelle ein Programm, welches nach n Jahren ein Endkapital einschließlich Zinseszinsen berechnet und in Tabellenform (Jahr für Jahr) ausgibt. Anfangskapital, Zinssatz und Laufzeit müssen eingegeben werden. Speichere unter **zinsen.bas**
 - Verwende die For - next Schleife.
 - Berechne in der Schleife zunächst die Zinsen, dann das Endkapital (Zinsen + Kapital) und schließlich das neue Anfangskapital
7. Ein Großhändler gibt ab der Summe von 1000 Euro einen Rabatt von 10%, sonst nur 3%. Abhängig von der Summe S sind Rabatt R und Endwert E zu bestimmen. Die Berechnung ist für die Summe von zwei Kaufbeiträgen B1 und B2 durchzuführen. (**rabatt.bas**)
 - Verwende die IF - THEN Verzweigung
 - Die Beträge B1 und B2 müssen eingelesen werden, der Rabatt wird nach einer Fallunterscheidung berechnet und mit dem Endwert ausgegeben.
8. Weil Lehrer Pille gerne Strafarbeiten aufgibt, wie "schreibe 200 mal: 'Ich darf nicht ...', hat sich Rudi für seine Mitschüler einen Trick ausgedacht. Erschreibe ein Programm, bei dem er einen 200 mal zu schreibenden Satz nur einmal einzugeben braucht. (**straze.bas**)
 - Verwende eine Stringvariable, zum Beispiel satz\$
 - Löse das Problem mit der FOR - NEXT - Schleife
9. Programmiere einen Kraftstoffrechner, der bei der Wahl des Kraftstoffes (Benzin oder Diesel), der Eingabe der Füllmenge den zu zahlenden Betrag ausgibt. (**benzin.bas**)