QBasic in 12h

Von Kai Schnieder
Überarbeitung und PDF-Konvertierung von Thomas Antoni - www.antonis.de

Zweite Auflage, 1.12.2002 - 30.5.2003

Inhaltsverzeichnis:

Vorwort:
In dem QBasic Lernkurs *„QBasic in 12h“* lernen Sie innerhalb 12 Stunden schnell und bequem die wichtigsten QBasic Befehle. Nach einem halben Tag werden Sie sich bereits zu einem fortgeschrittenen Programmierer weiterentwickelt haben! Im Folgenden erfahren Sie, ...

- was QBasic ist,
- was man mit QBasic alles machen kann,
- wo die Grenzen von QBasic liegen
- und wie sie die wichtigsten QBasic-Befehle anwenden.

Dabei halten wir uns nicht lange mit trockener Theorie auf. Programmieren soll ja schließlich Spaß bringen. Wir steigen sofort in die Praxis ein, und Sie machen sich anhand vieler kleiner Beispielprogramme und Übungsaufgaben spielerisch mit den Lerninhalten vertraut. Die Lösungen zu den Übungsaufgaben finden Sie in Anhang B. Alle Beispielprogramme und Lösungen sind im Download-Paket von "QBasic in 12h" als .BAS-Quellsprachedateien enthalten. Das komplette Download-Paket steht auf www.qbasic.de in der QBasic-Tutorial-Rubrik zum Herunterladen bereit.

Was ist neu in der zweiten Auflage?
Den ersten Kapiteln wurden FAQs hinzugefügt. Außerdem wurden viele Dinge geändert, ergänzt und verbessert.

Über den Autor
Der 13-jährige Kai Schnieder (N-Fighter) ist der Autor zahlreicher Kurse und Programme rund um den PC. Um

ihm Fragen zu stellen, schreiben Sie an KaiSchnieder@gmx.de . Er wird sich bemühen, diese zu beantworten.

Stunde 1: Was ist QBasic?
Bevor man Programme in einer Programmiersprache schreibt, sollte man die Programmiersprache genauer kennen.

QBasic ist eine für DOS entwickelte Programmiersprache. Sie ist sehr einfach und hervorragend für Programmier-Anfänger geeignet. Kleine Projekte lassen sich mit QBasic erstaunlich schnell umsetzen. Aber sehen Sie am Besten selber – in den folgenden 60 Minuten lernen Sie ihr erstes eigenes Programm zu schreiben.

Woher bekomme ich QBasic?

Bevor sie aber mit QBasic programmieren können, benötigen Sie die QBasic-Entwicklungsumgebung, bestehend aus einen Text-Editor und dem Interpreter. Im Editor tippen Sie ihre QBasic-Befehle ein. Der Interpreter sorgt für die Abarbeitung dieser Befehle. Später können Sie Ihre Programmme auch zu eigenständigen EXE-Programmen kompilieren, wozu Sie den "echten" Compiler QuickBasic 4.5 benötigen; mehr dazu in Anhang A. Doch vorerst wollen wir beim Interpreter QBasic 1.1 bleiben. Der reicht für die Beispiele in diesem Kurs vollkommen aus.

QBasic gibt es kostenlos im Internet, z.B. auf www.qbasic.de unter "Download - Compiler". Auch auf den älteren Installations-CD's von Microsoft Windows war QBasic dabei. QBasic besteht aus den beiden Dateien QBASIC.EXE und QBASIC.HLP.

Hinweis: Ein Interpreter ist ein Programm, welches ihre QBasic-Befehle aus der Entwicklungsumgebung heraus sofort ausführt - sie werden Schritt für Schritt in Maschinensprache übersetzt und ausgeführt. Anders als der Compiler erstellt der Interpreter keine direkt ausführbaren EXE-Dateien.

Hinweis: Die Maschinensprache ist die Sprache ihres Computers. Sie besteht aus Einsen und Nullen.

QBasic bedienen
Ich gehe jetzt davon aus, dass Sie eine Version von QBasic besitzen. Besonders empfehlen kann ich QBasic 1.1, und da besonders die deutsche Version. Da QBasic ein DOS-Programm ist, hat es manchmal gewisse "Macken" unter dem Betriebssystem Microsoft Windows, welches auf fast jedem Heim-PC installiert ist. Unter Windows 2000 und XP kann es am ehesten Probleme geben. Aber was soll man machen? Alle Probleme sind irgendwie umschiffbar. Es geht, und das ist die Hauptsache.

Sollten Sie Ihr QBasic nicht mit der Maus bedienen können, geht das auch sehr bequem über die Tastatur. Mit der ALT-Taste aktivieren Sie das Menü, mit den Pfeil-Tasten können Sie sich durch alle Menüs und Untermenüs hangeln.

Hinweis: MS-DOS ist ein Betriebssystem, welches man als Vorgänger von Microsoft-Windows bezeichnen kann. Die ersten Windows-Versionen wurden über MS-DOS gestartet. Ab Windows NT/XP ist MS-DOS nur noch simuliert und wird von Windows nicht mehr gebraucht.

Das erste QBasic-Programm
Tippen Sie nun den folgenden Code in ihrer QBasic - Entwicklungsumgebung ein:

Programm 1:

```
CLS
PRINT "Hello World!"
SLEEP 2
```

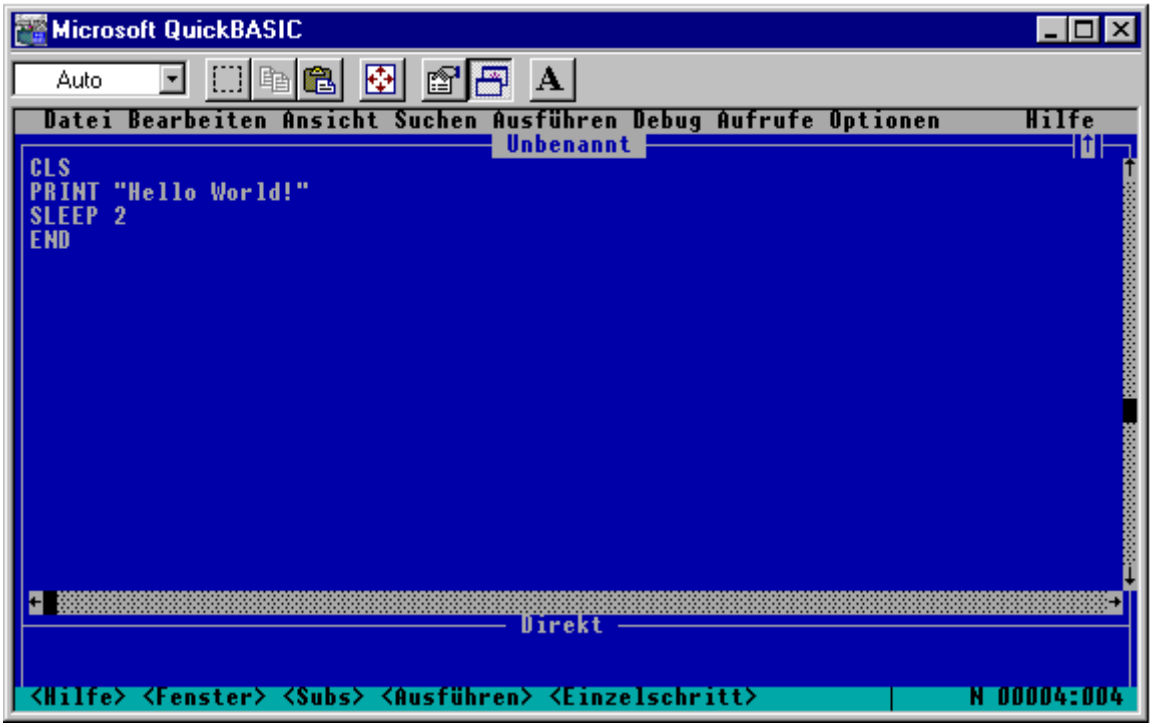
```
END
```

Ausgabe:

```
---

Hello World!

---
```



(so sollte es Aussehen)

Erklärungen:
Der erste Befehl (cls) erstellt einen leeren Bildschirm.
Mit dem zweiten Befehl (print) wird der Text "Hello World!" ausgegeben.
Der dritte Befehl (sleep 2) hält das Programm für zwei Sekunden an.
Der vierte Befehl schließlich beendet das Programm (end).

Sie sehen: Mit ein paar Englisch-Kenntnissen ist dieses Programm leicht zu verstehen. Sollten Sie über solche Kenntnisse nicht verfügen, müssen Sie sich leider so durchbeißen ;-).

Das Programm können Sie nun über "Ausführen | Start" oder mit der Taste F5 ausführen. Ihr Interpreter übersetzt es nun Zeile für Zeile in Aktionen und führt es aus.

Hinweis: QBasic schreibt alle ihre Befehle - solange sie zum QBasic-Wortschatz gehören - automatisch groß. Tippen Sie zum Beispiel "cls" ein, wird "cls" nach dem Verlassen der Zeile zu "CLS". Damit erhalten Sie immer sofort eine Rückmeldung darüber, ob Sie einen Befehl richtig geschrieben haben - für Einsteiger eine höchst praktische Sache!

Hinweis: Sie können mehrere Qasic-Befehle können auch in einer Zeile schreiben, wenn Sie diese durch einen Doppelpunkt voneinander trennen (Beispiel: *CLS: A = 5: PRINT A*). Außerdem müssen Sie auch nicht stur am Rand stehen. Rücken Sie ihre QBasic-Befehle so ein, wie es ihnen passt.

Damit ist die erste Lern-Stunde fast beendet. Im Folgenden noch ein paar kleine Aufgaben:

Aufgaben:

a) Schreiben Sie ein Programm, welches die Worte "Ich bin ein QBasic-Programmierer" ausgibt.
b) Schreiben Sie ein Programm, welches von dem Sleep-Befehl drei Sekunden lang angehalten wird.

FAQ
QBasic wird von Microsoft nicht mehr weitergeführt – warum?
Um diese Frage zu beantworten sollten Sie wissen, dass Microsoft MS-DOS nicht mehr unterstützt und Windows-basierte Compiler für Visual-C++ und VisualBasic anbietet. VisualBasic ist praktisch die Aufsteiger-Sprache für QBasic-Programmierer. Von daher gibt es schon eine Art Upgrade für die beliebte Basic-Sprache.

Stunde 2: Kommentare, Variablen und Strings
Nachdem Sie nun ihre ersten QBasic-Programme schreiben können, werde ich ihnen beibringen, wie Sie ihren Quellcode kommentieren.

Hinweis: Kommentare sind erklärende Worte oder Sätze des Programmierers im Quellcode seines Programms. Bei der Ausführung des Programms merkt der User nichts von den Kommentaren. Ein kleines Beispiel dazu:

Programm 2:

```
'dies ist die erste Art für Kommentare, man kann sie auch
CLS 'direkt hinter einen Befehl setzen
REM Dies ist die zweite Art für Kommentare
PRINT "Kommentare..."
SLEEP
END
```

Ausgabe:

Kommentare...

Wie sie jetzt sehen, haben die Kommentare keinerlei Wirkung auf ihr Programm bzw. auf die Ausgabe des Programms. Ich will ihnen jedoch nicht verheimlichen, dass zu viele Kommentare ein größeres Programm etwas bremsen können. Kurz möchte ich noch sagen, wann und wie Sie Kommentare verwenden sollten und dieses Thema damit abschließen.

- Nutzen Sie Kommentare, um Ihr Programm besser nachvollziebar zu machen,
- erläutern Sie per Kommentar Ihre Programmier-Tricks, die Sie viel Schweiß gekostet haben, besonders sorgfältig,
- entschuldigen Sie sich nicht mit Kommentaren für ihren unordentlichen Quellcode.

Variablen
Im Folgenden werde ich ihnen die Arbeit mit Variablen nahe legen. Eine Variable ist eine Speicherzelle im RAM, die über einen Namen ansprechbar ist, den Sie selber wählen können und in der sich ein Zahlenwert

speichern lässt.
Hierzu gehe ich gleich in die Praxis über:

Programm 3:

```
CLS
LET a = 1 'erstellt die Variable a und weist ihr den Wert 1 zu
PRINT "Wert von a: "; a 'gibt die Variable a aus
PRINT                                'Leerzeile...
INPUT a 'schreibt eine Anwendereingabe in die Variable a
PRINT
PRINT "Neuer Wert von a: "; a    'gibt a erneut aus
SLEEP   'Warte auf Tastendruck (ohne bestimmte Wartezeit)
```

Ausgabe:

Wert von a: 1

? 10

Neuer Wert von a: 10

Nun analysieren wir das Ganze. Ich habe die neuen Befehle im Quellcode zwar schon kurz erklärt, aber jetzt noch einmal ausführlich:

LET a = 1 – erstellt eine Variable mit dem Namen a und weist ihr den Wert 1 zu. Anstatt a könnte auch *Zahl* oder *Wert* verwendet werden. Benutzen Sie für ihre Programme einfach aussagekräftige Variablen-Namen. Das LET ist ein Überbleibsel aus älteren BASIC-Sprachen. Sie können es unbesorgt weglassen und stattdessen *a = 1* schreiben.
PRINT – erstellt eine Leerzeile
PRINT "Wert von a: "; a diese Zeile ist ein wenig schwer zu erklären. Ich könnte sie vielleicht mit einem Merksatz erklären: *Alles was zwischen den Rufzeichen steht, wird so ausgegeben, wie es ist. Alles was bei dem Print-Befehl nicht zwischen den Rufzeichen steht, wird als Variable ausgegeben. Sollten Sie einer Variable keinen Wert zuweisen, ist diese automatisch 0.*
INPUT a – mit *INPUT* können Sie den Programm-User auffordern, einen Text oder einen Zahlenwert einzugeben. Die Eingabe wird in die angegebene Variable, in diesem Falle *a* eingetragen. Diese lässt sich jederzeit wieder aufrufen.

Die QBasic-Online-Hilfe
QBasic hat eine hervorragende Online-Hilfe, die glatt ein Handbuch mit vielen 100 Seiten ersetzt. Die Hilfe arbeitet "kontext-sensitiv": Wenn Sie mit der rechten Maustaste auf ein Befehls-Schlüsselwort klicken, z.B. auf BEEP, erhalten Sie eine detaillierte Hilfe zu genau diesem Befehl. Statt der rechten Maustaste können Sie auch die F1-Taste verwenden. Über den Menüpunkt <Hilfe - Index> erhalten Sie eine alfabetisch sortierte Liste sämtlicher QBasic-Befehle; ein Doppelklick auf einen Befehl öffnet dessen Hilfetext. Mit Esc verlassen Sie die Online-Hilfe. Machen Sie ausgiebig Gebrauch von der mit vielen kleinen Programmbeispielen gespickten Online-Hilfe, um Ihre Kenntnisse zu vertiefen.

Aufgaben:

a) Schreiben Sie ein Programm, in welchem der Variablen *Variable* der Wert 10 zugewiesen wird. Danach soll der User in die Variable schreiben können. Geben Sie den neuen Wert erneut aus.

Strings

Versuchen Sie einmal ihren Namen in einer Variable zu speichern. Dies ist mit Ihrem bisher erworbenem Wissen unmöglich. Um Zeichen zu speichern, benötigen Sie Strings. Eine Stringvariable kennzeichnet man in deren Namen mit einem nachgestellten Dollar-Zeichen (*$*).

Programm 4:

```
Name$ = "N-Fighter"
Alter = 13
CLS
PRINT
PRINT "Name : "; Name$
PRINT
PRINT "Alter: "; Alter
PRINT
SLEEP 2
END
```

Ausgabe:

Name: N-Fighter

Alter: 13

Die Stringvariable in diesem Programm ist "Name$". Ihr wird der Text "N-Fighter" zugewiesen. Eine String-Variable wird prinzipiell genauso behandelt wie eine numerische Variable.
Durch ein Plus-Zeichen "+" können Texte aneinandergehängt werden.

Beispiele für die Verwendung von Stringvariablen:

- Input "Gib Deinen Namen ein: "; Name$
  ' Anwendereingabe in die String-Variable *Name$*
  ' kombiniert mit einem Aufforderungstext
- t$ = "Anna"
- u$ = t$ + "Anna" + " Egon" + " Karl" ' Texte aneinanderhängen

Sie können die Namen von numerischen Variablen und String-Variablen auch wie folgt auf den Bildschirm ausgeben:

PRINT name$;"ist "; alter; "Jahre alt."

Wie ich in meinem Merksatz bereits gesagt habe: die Namen von numerischen Variablen und String-Variablen werden nicht in Rufzeichen geschrieben bzw. nicht in die Rufzeichen ihres Textes (Wie z.B. "Hello World").

Mathematisches
Variablen können addiert, subtrahiert etc. werden:

```
A = A + 1
A = A – 1
A = A * 1   ' Multiplikation
A = A / 1   ' Division
```

```
A = A ^ 2   ' Quadrieren
A = SQR(2)   ' Wurzel aus 2 ("Square Root")
A = SIN (45 * 2 * 3.142 / 360)   ' sin (45°) mit Vorgabe des Winkels im Bogenmaß

A = A + A
A = A – A
A = A * A
A = A / A
A = A ^ B   ' A hoch B
A = SQR(A)   ' Wurzel aus A

PRINT A ;"+"; A ;"="; A+A
```

Experimentieren Sie noch ein wenig mit Variablen & Co. herum.

Aufgaben:

b) Schreiben Sie ein Programm, bei dem der User seinen Namen und sein Alter angegeben kann, welches dann wieder ausgegeben wird.

FAQ

Welchen Wert hat eine Variable, wenn ihr kein Wert zugewiesen wird?
Wenn man eine numerische Variable erstellt und ihr keinen Wert zuweist, hat sie den Wert 0; Stringvariablen sind auf den Leerstring "" vorbesetzt.

Wenn ich keine Text-Zeichen in numerischen Variablen speichern kann, kann ich dann auch keine Zahlen in String-Variablen speichern?
Sie können in Strings auch Zahlenwerte speichern. Diese werden allerdings in Textform abgelegt. Probieren Sie mal das folgende Programm:

```
INPUT "gib eine Zahl ein: "; t$
PRINT t$
```

Stunde 3: Textfarbe und Hintergrundfarbe

Ebenfalls können Sie als Programmierer die Schriftfarbe in der Bildschirmanzeige verändern:

```
COLOR 2
```

Dieser Befehl ändert die Schriftfarbe ihres Programms auf die Farbe grün.

Nun ein Programm, bei dem alle möglichen Farben in QBasic ausgegeben werden:

Programm 5:

```
CLS
COLOR 0, 15
PRINT "Dies ist die Vordergrundfarbe 0 (Schwarz)."
COLOR 1
PRINT "Dies ist die Vordergrundfarbe 1 (Blau)."
COLOR 2
PRINT "Dies ist die Vordergrundfarbe 2 (Gruen)."
COLOR 3
PRINT "Dies ist die Vordergrundfarbe 3 (Helltuerkisblau)."
COLOR 4
PRINT "Dies ist die Vordergrundfarbe 4 (Rot)."
COLOR 5
```

```
PRINT "Dies ist die Vordergrundfarbe 5 (Violett)."
COLOR 6
PRINT "Dies ist die Vordergrundfarbe 6 (Braun)."
COLOR 7, 0
PRINT "Dies ist die Vordergrundfarbe 7 (Hellgrau)."
COLOR 8
PRINT "Dies ist die Vordergrundfarbe 8 (Grau)."
COLOR 9
PRINT "Dies ist die Vordergrundfarbe 9 (Hellblau)."
COLOR 10
PRINT "Dies ist die Vordergrundfarbe 10 (Hellgruen)."
COLOR 11
PRINT "Dies ist die Vordergrundfarbe 11 (Sehr helles Tuerkisblau)."
COLOR 12
PRINT "Dies ist die Vordergrundfarbe 12 (Hellrot)."
COLOR 13
PRINT "Dies ist die Vordergrundfarbe 13 (Rosa)."
COLOR 14
PRINT "Dies ist die Vordergrundfarbe 14 (Gelb)."
COLOR 15
PRINT "Dies ist die Vordergrundfarbe 15 (Weiss)."
SLEEP
END
```

Ausgabe: Schreiben Sie dieses Programm doch selber einmal, und starten Sie es, um den Ausgabebildschirm zu betrachten.

In diesem Beispiel sehen Sie alle möglichen Farben im Text-Modus. Die hier nicht dargestellten Farbcodes 16-31 erzeugen noch einmal dieselben Farben, jodoch als blinkenden Text. Das funktioniert jedoch nur unter reinem DOS, nicht unter Windows.

Hinweis: Der Text-Modus ist eine Bildschirmauflösung, bei der man nur Text-Zeichen verwenden kann. Jede Verwendung von Grafiken in diesem Modus ist nicht möglich.

Nun können Sie die Schriftfarbe ihres Programms verändern. Doch wissen Sie immer noch nicht, wie Sie die Hintergrundfarbe beeinflussen. Schauen Sie mal im obigen Programm auf die Zeilen *COLOR 0, 15* und *COLOR 7, 0*. Da wird der Schrifthintergrund für die nachfolgenden Bildschirmausgaben durch ", 15"auf weiß bzw. durch ", 0" auf schwarz gesetzt. Dazu nun ein weiteres Beispiel:

```
COLOR 14, 1
CLS
PRINT "Gelb auf Blau"
```

Die "1" ändert die Hintergrundfarbe auf Blau. Das *CLS* sorgt dafür, dass der gesamte Bildschirm mit der Hintergrundfarbe eingefärbt wird.

Hinweis: Die Farbcodes für die Hintergrund- und die Schriftfarben sind bis auf kleinere Ausnahmen identisch. Sie können alle 15 Standard-Farben als Hintergrundfarbe verwenden, wobei die Codes 8-15 allerdings dieselben Farben erzeugen wie die Codes 0-7.

Und was passiert, wenn Sie das *CLS* weglassen? Das sehen Sie, wenn Sie die folgende Befehlsfolge ausprobieren:

```
COLOR 14,4
PRINT "Gelb auf Rot"
```

Der *COLOR*-Befehl ändert die Hintergrundfarbe nur für die tatsächlich angezeigten Schriftzeichen. Der restliche Bildschirm behält die ursprüngliche Hintergrundfarbe bei. Es sei denn nach dem *COLOR*-Befehl folgt ein *CLS*-Befehl, der den ganzen Bildschirm bleibend einfärbt. Experimentieren Sie einfach ein wenig mit

diesem Befehl. Ein Tipp von mir: Wählen Sie eine auch bei abgedunkeltem Bildschirm gut lesbare Kombination von Schriftfarbe und Hintergrundfarbe. Nicht alle Anwender reißen den Helligkeitsregler ihres Monitors grundsätzlich voll auf.

FAQ
Kann ich einer Variable einen Wert zuweisen und diese Variable anstatt einer Zahl hinter den Color-Befehl setzen?
Ja, das können Sie. QBasic wird keine Fehlermeldung erzeugen (es sei denn, der Wert der Variable gehört nicht zu den Farbcodes von QBasic ;-)).

Stunde 4: Labels und Verzweigungen
Ein Label ist eine Sprungmarke. Man kann mit dem GOTO-Befehl zu jeder beliebigen Stelle im Programm springen, welche mit eine Sprungmarke gekennzeichnet ist. GOTO-Befehle sind jedoch – was ich ihnen nicht verheimlichen möchten – schon früh in die Kritik der Programmierer geraten. Da man mit GOTOs an jede beliebige Stelle des Programms springen kann, wird der Quellcode ihres Programms bald sehr unübersichtlich, da man die verschiedenen Sprungmarken immer erst suchen muss. Ein Quellcode mit vielen GOTO-Befehlen wird daher auch Spagetti-Code genannt.
Genug der Theorie. Nun ein Beispiel:

Programm 6:

```
CLS
PRINT "Wohin wollen Sie springen? "
PRINT
PRINT " (1) Sprungmarke 1"
PRINT " (2) Sprungmarke 2"
PRINT " (3) Beenden"
INPUT men
IF men = 1 THEN GOTO 10
IF men = 2 THEN GOTO Textmarke
IF men = 3 THEN END ELSE BEEP: END

10 CLS
   PRINT "Sprungmarke in Form einer Zahl."
   SLEEP
   END

Textmarke:
CLS
PRINT "Sprungmarke mit dem Namen »Testmarke«“
SLEEP
END
```

Ausgabe:

```
---
Wohin wollen Sie springen?

 (1)       Sprungmarke 1
 (2)       Sprungmarke 2
 (3)       Beenden
? 1
Sprungmarke in Form einer Zahl.

---
```

In diesem Programm waren eine ganze Menge neuer Befehle. Der erste neue Befehl war

IF men = 1 THEN GOTO 10

Dieser Befehl prüft die Variable *men*.
Wenn *men* = 1 ist, gehe zu 10.
10 ist die Sprungmarke.
Dasselbe geschieht mit *IF men = 2 THEN GOTO sprungmarke*.
Die Sprungmarke *Sprungmarke* endet mit einem Doppelpunkt.

Hinweis: Ein QBasic-Befehlswort ("Schlüsselwort") darf nicht als Sprungmarke oder als Variablennname verwendet werden. Auch Umlaute wie ä, ö oder ü dürfen in Sprungmarken und Variablennamen nicht vorkommen. Außerdem dürfen Sprungmarken sich nicht wiederholen.

Neu ist ebenfalls die Zeile

IF men = 3 THEN END ELSE BEEP: END

Wenn die Variable *men* = 3 ist, wird das Programm beendet. Wenn *men* etwas außer 1,2 oder 3 ist, gibt der Computer einen Piepston aus (BEEP). Danach wird das Programm ebenfalls beendet.
Der Else-Befehl muss immer hinter der letzten If-Anweisung stehen.

Aufgabe:

a) Schreiben Sie ein kleines Quiz

Dies dürfte zu dem Thema *Sprungmarken und Verzweigungen* fürs Erste reichen. Eine Sache wäre aber noch: Wenn Sie eine Stringvariable auf ihren Inhalt testen möchten, müssen Sie dies wie folgt tun:

IF name$ = "N-Fighter" THEN PRINT "Toller Name!"

Der String muss wieder zwischen Rufzeichen stehen.

Es gibt übrigens auch **Mehrfachverzweigungen**, die Sie mit dem Befehl ***SELECT CASE variable*** realisieren können. Dabei hängt das Sprungziel vom Inhalt der *variable* ab. Diese Verzweigungsart wollen wir aber hier nicht weiter vertiefen. Wenn Sie mehr darüber wissen wollen, geben Sie einfach in der QBasic-Entwicklungsumgebung "select" ein und starten sofort anschließend die kontextsensitive Online-Hilfe mit der F1-Taste.

FAQ
Wieso hat man Spagetti-Code, wenn man mit Sprungmarken arbeitet?
Dies lässt sich dadurch erklären, dass die Sprungmarken im ganzen Programm verteilt sind. Mal springt das Programm zu einer, und mal zu einer anderen Stelle. Und beim Ändern und Erweitern von Befehlsfolgen muss man sich immer darüber im Klaren sein, von welchen anderen Programmstellen aus mit *GOTO* dorthin gesprungen wird. Dadurch verliert man oft schnell den Überblick.

Stunde 5: Schleifen
Eine Schleife ist eine Folge von Befehlen, welche so oft durchlaufen wird, bis die Abbruchbedingung erfüllt ist. Hier einmal ein Beispiel:

Programm 7:

```
CLS
INPUT "Bitte geben Sie eine Zahl ein: ", anzahl
```

```
DO
x = x + 1
PRINT "Hallo ("; x; ")"
LOOP UNTIL x = anzahl
SLEEP
END
```

Ausgabe:

Bitte geben Sie eine Zahl ein: 6
Hallo (1)
Hallo (2)
Hallo (3)
Hallo (4)
Hallo (5)
Hallo (6)

In diesem Programm gibt es schon wieder eine Menge neuer Befehle. Zuerst erhält der User die Aufforderung, eine Zahl einzugeben, die in die Variable *anzahl* eingetragen wird. Vor der Eingabe steht der Aufforderungs-Text *Bitte geben Sie eine Zahl ein*. An dieser Stelle funktioniert der Input-Befehl wie der Print-Befehl,nur dass mit *PRINT* keine Eingaben erfolgen können.

Dann beginnt die **Do...Loop Until-Schleife**.
Sie wird so oft durchlaufen, bis *x* den Wert von *anzahl* hat. Probieren Sie mit diesem Programm einfach selber ein wenig.

Eine weitere Schleife ist die **For...Next-Schleife**.
Wie Programm 8 zeigt, funktioniert sie ganz ähnlich wie die **Do...Loop...Until-Schleife**.

Programm 8:

```
CLS
INPUT "Bitte geben Sie eine Zahl ein: ", anzahl
FOR x = 1 to anzahl
PRINT "Hallo ("; x; ")"
NEXT
SLEEP
END
```

Ausgabe:

Bitte geben Sie eine Zahl ein: 6
Hallo (1)
Hallo (2)
Hallo (3)
Hallo (4)
Hallo (5)
Hallo (6)

Diese Schleife ist für dieses Beispiel um einiges einfacher zu verstehen.

Sie wird so lange durchlaufen, bis *x* dieselbe Größe wie *anzahl* hat
(*FOR x = 1 to anzahl*). Die Schleife endet mit *next.*

In QBasic gibt es noch eine dritte Schleife, die **While...Wend-Schleife.**
Sie funktioniert ähnlich wie die **For...Next-Schleife**.
Ich möchte sie aber an dieser Stelle nicht weiter besprechen.

FAQ
Ich habe noch nie Schleifen gebraucht, was soll das Ganze?
Anfänger bevorzugen GOTO. Schleifen werden von ihnen wenig verwendet. Doch später kommt man kaum noch um sie herum. Warten Sie einfach ab und denken Sie an meine Worte.

Stunde 6: Anzeigen und Eingaben lokalisieren
Sie haben in QBasic die Möglichkeit, Bildschirmanzeigen an beliebige Stellen auf dem Bildschirm zu platzieren. Dieses geschieht mit Hilfe des LOCATE-Befehls wie folgt:

```
CLS
LOCATE 3, 2
PRINT "Zeile 3, Spalte 2"
SLEEP
END
```

Der Befehl *Locate* gibt die Zeile und die Spalte an.
Bringen wir nun unser altes Wissen mit hinein:

Programm 9:

```
CLS
FOR a = 1 to 10
LOCATE a, a
PRINT "Zeile "; a ; ", Spalte "; a; "
NEXT
SLEEP
END
```

Ausgabe:

```
Zeile 1, Spalte 1
 Zeile 2, Spalte 2
  Zeile 3, Spalte 3
   Zeile 4, Spalte 4
    Zeile 5, Spalte 5
     Zeile 6, Spalte 6
      Zeile 7, Spalte 7
       Zeile 8, Spalte 8
        Zeile 9, Spalte 9
         Zeile 10, Spalte 10
```

Die Schleife wird so oft durchlaufen, bis der Wert der Variablen *a* = 10 ist. Der Locate-Befehl bekommt immer

den jeweiligen Wert von a sowohl als Zeilen- als auch als Spaltennummer übergeben.
Wenn die Schleife also zwei Mal durchlaufen wurde, ist der Wert von *a* = 2 und die Anzeige erfolgt in Zeile 2, Spalte 2.

Spielen Sie noch ein wenig mit dem Locate-Befehl herum.

Damit endet diese Stunde schon. Sie haben jetzt einen Viertel-Tag mit QBasic gearbeitet. Testen sie noch einmal alle Befehle, bevor es mit der Stunde sieben losgeht.

Stunde 7: Grafikbefehle
Sie können in QBasic ihre eigenen Grafiken erstellen. Dazu müssen Sie zuerst in einen Grafik-Modus schalten. Das geht mit dem **SCREEN-Befehl**. Bisher haben wir es ausschließlich mit dem Textbildschirm SCREEN 0 zu tun gehabt, den QBasic standardmäßig beim Programmstart aktiviert. Hierbei hat man keinen direkten Zugriff auf die einzelnen Bildschirmpunkte, sondern kann nur Textzeichen zur Anzeige bringen.

Für Grafikanzeigen verwenden QBasic-Programmierer am häufigsten *SCREEN 9* und *SCREEN 12*.

Mit dem *PSET-Befehl* können Sie Punkte zeichnen. Beispiel:

```
SCREEN 9
PSET (20, 20), 15
SLEEP
END
```

Es wird ein Punkt wird an die X-Koordinate 20 und die Y-Koordinate 20 gezeichnet. Die Farbe ist hellweiß (15).

Screen 9 hat übrigens eine Auflösung von 640 * 350 und Screen 12 von 640 * 480 Bildschirmpunkten. Der Koordinatenursprung befindet sich in der linken oberen Bildschirmecke

Ein weiterer QBasic-Grafikbefehl ist der *CIRCLE*. Damit können Sie Kreise zeichnen.

```
SCREEN 9
CIRCLE (20, 20), 15, 15
SLEEP
END
```

Hierdurch wird der Mittelpunkt des Kreises an der X-Koordinate 20 und der Y-Koordinate 20 gezeichnet. Der Radius beträgt 15 Bildschirm-Punkte. Die Farbe ist hellweiß (15). Nun wollen wir diesen Kreis innen mit einer Farbe ausfüllen. Dazu verwendet man den *Paint-Befehl* :

```
SCREEN 9
CIRCLE (20, 20), 15, 15
PAINT (20, 20), 15
SLEEP
END
```

Auch der *LINE-Befehl* wird häufig verwendet, um Linien zu zeichnen:

```
SCREEN 9
LINE (20,20) – (20, 30)
SLEEP
END
```

Die Linie wird zwischen die X-Koordinate (20) und die Y-Koordinate (20) bis hin zur zweiten X-Koordinate (20) und Y-Koordinate (30) gezeichnet. In diesem Falle ist die Linie senkrecht.

Hinweis: Auch wenn die QBasic-Grafiken auf ihrem Bildschirm gut aussehen, vielleicht sehen Sie auf anderen Bildschirmen weniger gut aus. Vermeiden Sie also Grafiken in QBasic oder testen Sie sie auf mehren Bildschirmen.

In den Text-Modus kann man wieder mit *SCREEN 0* schalten.

Aufgaben:

a) Schreiben Sie ein Programm, bei dem alle Grafik-Befehle, die Sie nun kennen, vorkommen.

Stunde 8: Zufallszahlen erzeugen
In QBasic haben Sie die Möglichkeit Zufallszahlen zu erzeugen.
Hierzu muss mit *RANDOMIZE TIMER* der Zufallsgenerator aktiviert werden.

Eine Zufalls-Zahl zwischen 1 und 10 erstellt man wie folgt:

```
RANDOMIZE TIMER
zufall = INT(RND * 10) + 1
PRINT zufall
```

Nachdem der Zufallsgenerator aktiviert wurde, erzeugt RND eine Zufallszahl zwischen 0 und 0.9999999. Das "INT(RND * 10)" bildet hieraus eine ganze Zahl zwischen 0 und 9. Das "+1" bewirkt, dass der Variablen *zufall* ein ganzzahliger Zufallswert zwischen 1 und 10 zugewiesen wird.

Nutzen wir dieses Beispiel einmal für ein größeres Programm:

Programm 10:

```
SCREEN 12
CLS
RANDOMIZE TIMER
FOR a = 1 TO 400
variable1 = INT(RND * 640) + 1
variable2 = INT(RND * 480) + 1
PSET (variable1, variable2), 15
NEXT
SLEEP
END
```

(Die Ausgabe ist eine grafische und zufällige Ausgabe, testen Sie sie doch bitte selber.)

Zeile 1 aktiviert den Grafikmodus (640*480 Punkte). Nun beginnt eine *For…Next-Schleife.*
Wenn die Variable a von 1 den Wert 400 erreicht hat, endet das Programm. Variable1 hat einen Wert zwischen 1 und 640, Variable2 zwischen 1 und 480.
Nun werden die Variablen dem *Pset-Befehl* als X- und als Y-Koordinate zugewiesen.
Schließlich endet das Programm. Es ist übrigens ein Sternenhimmel mit 400 nach dem Zufallspinzip über den Bildschirm verteilten Bildpunkten.

Aufgabe:

a) Schreiben Sie unter Verwendung des Zufallsgenerators ein kleines Zahlenraten. Verzweifeln Sie jedoch nicht, wenn Sie es nicht schaffen. Schauen Sie sich dann einfach schon jetzt die Lösung in Anhang B an. Aber versuchen Sie es zumindest.

Damit endet diese Stunde. Studieren Sie den Inhalt dieses Kapitels gegebenenfalls noch mal, denn Zufallszahlen braucht man sehr häufig.

Stunde 9: Töne
In QBasic können Sie über den PC-Speaker ihre eigenen Töne ausgeben, ein kleiner Vorgeschmack war der Beep-Befehl.

In QBasic gibt es zwei Befehle zum Ausgeben von Tönen. Den *Play*- und den *Sound-Befehl*. Wir verwenden den *Play-Befehl*.
Für die Töne der Tonleiter gelten in QBasic die gewohnten Buchstaben, nur der Ton "h" wird davon abweichend durch den Buchstaben "b" dargestellt:

```
PLAY "cdefgab"     ' Tonleiter c-d-e-f-g-a-h
```

Nun lässt sich in QBasic nicht nur die Tonleiter abspielen, sondern auch die Länge der Töne und die Oktave angeben:

```
PLAY "O0 L8 cdefgab"     ' Alle Töne der Oktave 0 mit der Länge 8 spielen
```

Die Oktave 4 ist die Standard-Oktave. Die Oktaven gehen von 0 (tiefste) bis 8 (höchste). Die Standard-Länge der Töne ist 8. L50 ist der längste Ton, L1 der Kürzeste.
Natürlich müssen nicht alle Töne die gleiche Länge haben:

```
PLAY "o0 a10 o1 a10 o2 a10 o3 a10 o4 a10"
```

Auch wenn man es nicht für möglich hält: Man kann sogar eine Oper mit QBasic zu Gehör bringen:

Programm 11:

```
REM Der erste Satz aus Beethovens fünfter Symphonie
REM (Du weisst schon - das beruehmte TaTaTaTaaaa):
PLAY "T180 o2 P2 P8 L8 GGG L2 E- P24 P8 L8 FFF L2 D"
```

Ausgabe: (Tonbasierend)

Testen Sie den Play-Befehl auch eigenständig ein wenig.

Aufgabe:

a) Erstellen Sie ein Programm, das "Alle meine Entchen abspielt".

Stunde 10: SUBs und FUNCTIONs
Eine SUB ist eine "Subroutine", also eine Art Unterprogramm, in das man häufig benötigte Befehlssequenzen sehr elegant auslagern kann. Man erstellt SUBs wie folgt:

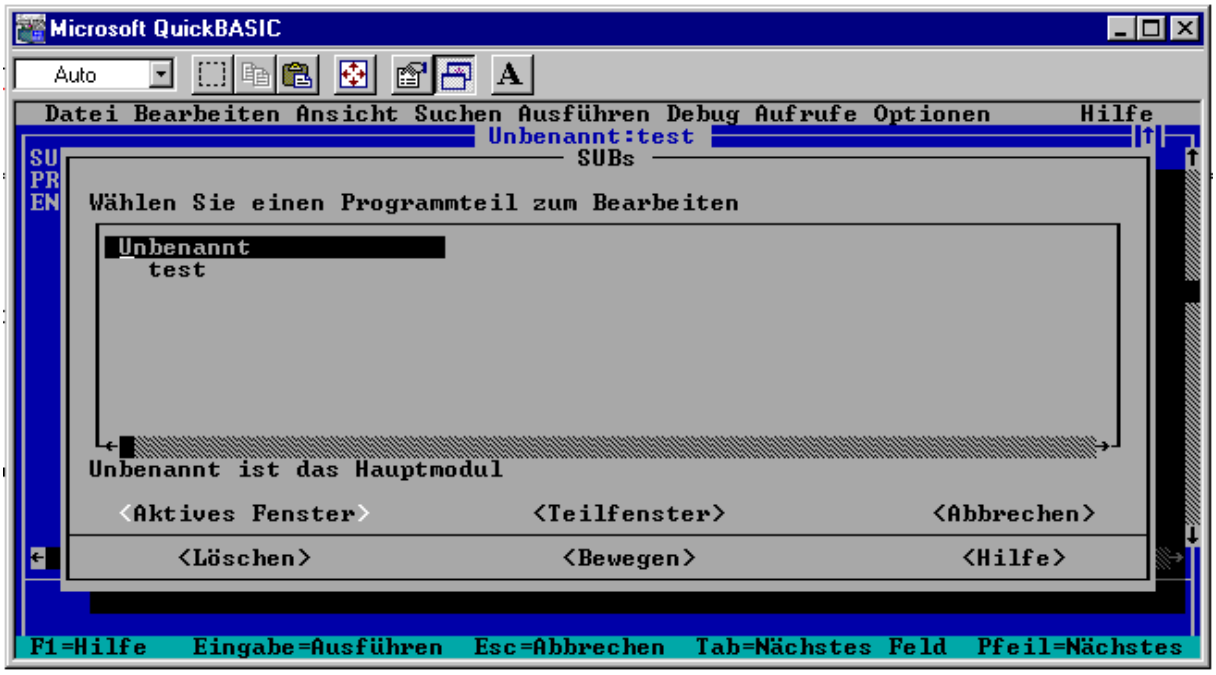
```
SUB test
```

Nun öffnet QBasic den folgenden Bildschirm:

```
SUB test
```

END SUB

Tragen Sie nun Ihre Befehle ein. Zu ihren Hauptprogramm gelangen Sie wieder mit der Taste F2:



(Die *Sub Test* wurde erstellt, nun gehe ich mit F2 wieder in das bisher noch "unbenannte" *Hauptprogramm*).

Wenn Sie nun ein solchen Unterprogramm erstellt haben, können Sie es im *Hauptprogramm* mit *CALL subname* aufrufen.

Aufgabe:

a) Erstellen Sie ein Programm mit einer SUB, welche Sie aufrufen. Die Lösung finden Sie wie gewohnt in Anhang B.

Hinweis 1: Sie können auch SUBs innerhalb von SUBs erstellen. Diese SUBs nennt man auch *verschachtelte SUBs.*
Hinweis 2: In einer nachfolgenden Klammer können Sie vom Hauptprogramm aus Variablen an die SUB übergeben - siehe das Beispiel 10a) in Anhang B

FUNCTIONs
FUNCTIONs sind SUBs, die einen Wert an das aufrufende Programm zurückliefern. Das folgende Programm zeigt ein Beispiel für eine Funktion "Quadrat", die den übergebenen Zahlenwert "i" quadriert und zurückliefert. Mit Hilfe dieser Funktion werden die Quadrate der Zahlen 1 bis 10 berechnet und auf dem Bildschirm angezeigt.

Programm 12:

```
'  Hauptprogramm
FOR i = 1 TO 10
  PRINT Quadrat(i)
NEXT

'Funktion "Quadrat" zum Quadrieren
FUNCTION Quadrat (i)
a = i ^ 2          'a = i hoch 2
Quadrat = a      'diesen Wert gibt die Funktion ans Hauptprogramm zurück
```

```
END FUNCTION
```

Stunde 11: DOS-Befehle, Ascii-Codes, "Schnell-Menüs" und Sonstiges
In diesem Kapitel möchte ich kurz Befehle ansprechen, welche in die anderen Kapitel nicht so recht hineingepasst haben.

DOS-Befehle
In QBasic können Sie auch DOS-Befehle aufrufen. Dies geschieht über den SHELL-Befehl.

Beispiele:
SHELL "CLS"
Löscht den Bildschirm.
SHELL "DIR"
Zeigt den Inhalt des aktuellen Ordners an.

ASCII-Codes
ASCII-Codes sind Zeichen-Codes, mit denen man die auf der Tastatur und dem Bildschirm erscheinenden Schriftzeichen intern im Computer ablegt. Je Zeichen werden dafür 8 Bits = 1 Byte benutzt. In QBasic kann man die ASCII-Zeichen über CHR$(ascii-code) aufrufen. Sie können die ASCII-Codes der einzelnen Zeichen in der QBasic-Online-Hilfe unter "Inhalt | ASCI-Zeichencodes" nachschlagen

Der Backslash "\" hat z.B. den ASCII-Code 92. Sie können ihn aufrufen, indem Sie die Alt-Taste gedrückt halten und währenddessen die Zahlen 9 und 2 auf dem rechten Nummern-Tastenfeld eintippen.
In QBasic kann man den Backslash und andere Zeichen wie folgt aufrufen:

```
PRINT CHR$(92)     - zeigt den Backslash an
PRINT CHR$(1)      - zeigt einen Smiley an
PRINT CHR$(2)      - zeigt einen weiteren Smiley an
```

Das folgende Programm zeigt zu einem ASCII-Code (*zahla*) das dazugehörige Textzeichen an:

Programm 13:

```
DO
   DO
      INPUT "Bitte geben Sie einen ASCII-Code ein (1...255): ", zahla
   LOOP WHILE zahla < 1 OR zahla > 255
   PRINT "Dem ASCII-Code ist das folgende Textzeichen zugeordnet: "; CHR$(zahla)
   PRINT 'Leerzeile
   PRINT "Wiederholen...[Beliebige Taste]     Abbruch...[Esc]"
   DO: taste$ = INKEY$: LOOP WHILE taste$ = ""
   ' Warten auf Tastenbetätigung
   IF taste$ = CHR$(27) THEN END ' Abbruch mit Esc (ASCII-Code 27)
   PRINT
LOOP 'Wiederholen mit beliebiger Taste
```

Der User muss eine Zahl eingeben. Wenn Sie unter eins liegt (< 1) oder über 255 (OR zahla > 255), d.h. wenn es sich nicht um einen gültigen ASCII-Code handel, wird der User aufgefordert die Zahl erneut einzutippen (obere DO...LOOP-Schleife). Anschließend wird das dem eingegebenen ASCII-Code zugeordnete Textzeichen angezeigt.

Nun kommt eine für viele Programme sehr nützliche Funktion, nämlich ein **Wiederholen-Abbruch-Dialog**: Die einzeilige DO...LOOP-Schleife wartet auf eine beliebige Tastenbetätigung. Der ASCII-Code der betätigten Taste wird in *taste$* hinterlegt. Hierfür greifen wir etwas vor und verwenden den *INKEY$*-Befehl, der eigentlich erst im folgenden Abschnitt durchgenommen wird. Handelt es sich um die Esc-Taste *CHR$(27)*, die den ASCII-Code 27 hat, dann wird das Programm mit *END* abgebrochen. Ansonsten erfolgt ein Neudurchlauf durch die äußerste DO...LOOP-Schleife.

Dieses Listing zeigt übrigens, wie Sie durch Einrückungen der Schleifen die Lesbarkeit Ihres Programm erheblich verbessern können.

Schnellmenüs:
Bisher kennen Sie nur die Eingabemethode mit INPUT, bei der der User die Enter- oder Eingabetaste drücken muss bevor seine Eingabe akzeptiert wird. Das ist etwas lästig, wenn nur ein einziges Zeichen eingegeben werden soll, z.B. bei Menü-Dialogen. Mit Hilfe des INKEY$-Befehls kann man das lästige Enter vermeiden. wie das folgende Programmbeispiel zeigt:

Programm 14:

```
10
PRINT "(1) Neustart"
PRINT "(2) Ende"
DO
20
Taste$ = INKEY$
LOOP WHILE Taste$ = ""
IF Taste$ = "1" THEN GOTO 10
IF Taste$ = "2" THEN END ELSE GOTO 20
```

INKEY$* liefert das der gedrückten Taste zugeordnete ACII-Zeichen und trägt es in die String-Variable *Taste$ ein. INKEY liefert den den leeren String "" zurück, solange keine Taste betätigt wird. *Taste$* wird in einer **Do...Loop-Schleife** abgefragt. Die Schleife wird so lange wiederholt, bis der User eine Taste drückt. Der Rest dürfte ihnen bekannt sein.

Feinheiten des Print-Befehls
Bisher kennen Sie den *Print-Befehl* nur so, dass er eine ganze Zeile beansprucht und zur nächsten Zeile fortschaltet. Dieser Zeilenvorschub kann jedoch durch ein Semikolon ab Ende der Rufzeichen unterdrückt werden:

```
PRINT "Zeile 1";
PRINT "Immer noch Zeile 1  ?! "
```

Stunde 12: Ihre weiteren Schritte
Nun haben Sie sich bereits 11 Stunden mit QBasic beschäftigt - fast einen halben Tag.
Sie haben bereits viel über QBasic gelernt. Sie kennen nun eine ganze Menge Befehle und sind mittlerweile schon ein recht guter QBasic-Programmierer.

Das sollten Sie tun, um sich in Sachen QBasic zu perfektionieren:

- Schreiben Sie alle Beispiel-Programme dieses Kurses selber erneut.
- Schließen Sie sich einem QBasic-Forum an, am ehesten zu empfehlen ist das Forum unter www.QBasic.de/forum. Dort finden Sie eine lebendige, freundliche und sehr hilfsbereite Community von QBasic-Fans.
- Lesen Sie noch mehr Tutorials, ein Einsteigerkurs wie dieser kann ihnen nicht alles erläutern. Sehr empfehlenswert sind das ausführliche deutsche QBasic-Tutorial "Adok's Way to QBasic" und das "QBasic-Kochbuch", eine komlette, nach Funktionen gegliederte Befehlsreferenz. Diese beiden Tutorials stehen auf www.qbasic.de in der QBasic-Tutorial-Rubrik zum Herunterladen bereit.
- Bei fast allen Einsteigerfragen hilft Ihnen die "QB-MonsterFAQ" weiter, die über www.qbasic.de erreichbar ist. Sie werden feststellen, dass es kaum eine Frage gibt, die nicht irgendjemand vor Ihnen bereits gestellt hat und deren Antwort nicht bereits in diese gigantische FAQ eingefügt wurde (FAQ = Frequently Asked Questions = Sammlung häufig gestellter Fragen und der dazugehörigen Antworten).

- Schreiben Sie eigene Programme und Spiele und experimentieren Sie selber mit den bereits gelernten QBasic-Befehlen herum.
- Kaufen Sie sich Bücher über QBasic. Es gibt noch einige im Buchhandel. Und bei eBay finden Sie eine Riesenauswahl guter QBasic-Bücher, die Sie günstig ersteigern können.
- Vielleicht kennen Sie ja auch einen anderen QBasic-Programmierer. Tauschen Sie sich mit ihm aus.

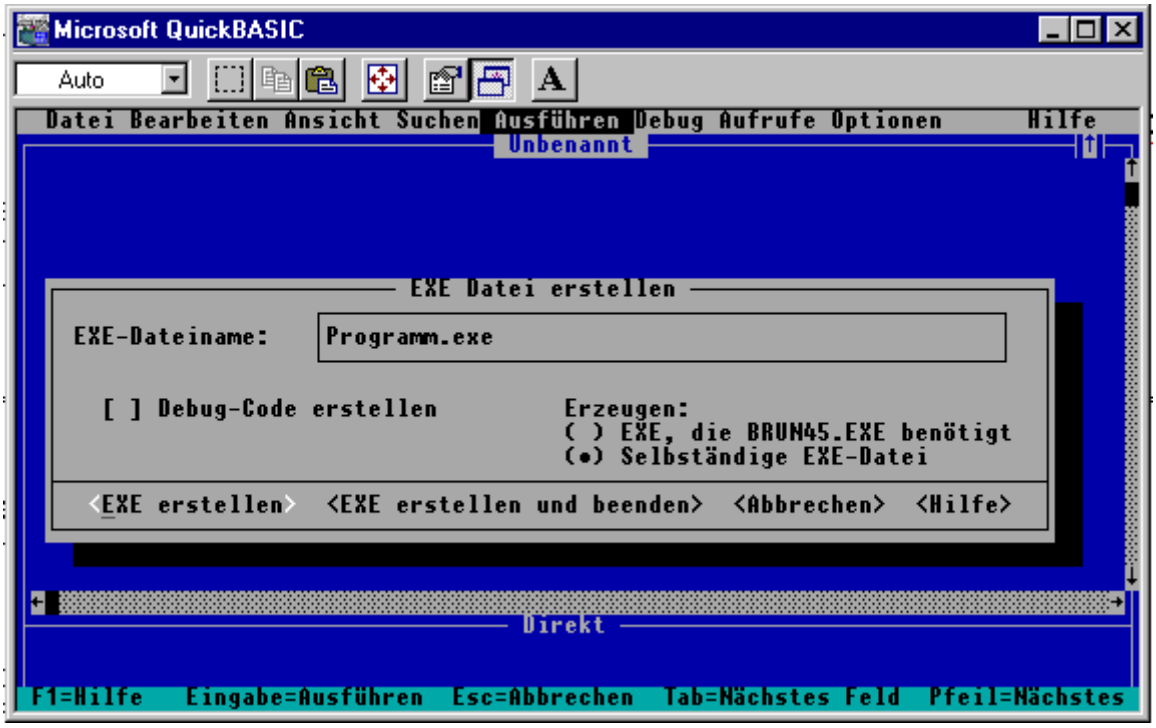
Was hat QBasic noch zu bieten? Wenn Sie weiterlernen, werden Sie u.a. die folgenden fortgeschrittenen QBasic-Funktionen entdecken, die wir in diesem Einsteigerkurs der Einfachheit halber bewusst weggelassen haben:

- Numerische Datentypen (Gleitpunkt und Festpunkt) und Konstanten
- Mehrdimensionale Datenfelder
- Wartezeiten erzeugen, Datum und Uhrzeit
- Dateibearbeitung - Dateien schreiben und lesen
- Joystick, serielle und parallele Schnittstelle ansprechen

Die nächste Programmiersprache
Sie können zwar eine Zeit lang weiter in QBasic programmieren erstellen, aber um ein professioneller Programmierer zu werden, sollten Sie später auch Programmiersprachen wie Delphi, Java, VisualBasic C++ oder C# erlernen.

Anhang A: So erstellen Sie eine Exe-Datei
Um aus Ihren BAS-Dateien direkt ausführbare EXE-Dateien zu erzeugen, benötigen Sie einen "echten Compiler", am besten QuickBasic 4.5. Dieser Compiler steht z.B. auf www.qbasic.de unter "Download Compiler" zum Herunterladen bereit. Gehen Sie wie folgt vor, um eine EXE-Datei zu erstellen:

- Aktivieren Sie das Menü,
- klicken Sie auf *Ausführen*,
- klicken Sie auf *Exe Datei erstellen...*,
- wählen Sie *Selbständige EXE-Datei* und
- bestätigen Sie mit *EXE erstellen*.



Anhang B: Lösungen
Hier finden Sie die Lösungen zu den verschieden Aufgaben, welche Sie während des Kurses lösen sollten:

1a) Schreiben Sie ein Programm, welches die Worte "Ich bin ein QBasic-Programmierer" ausgibt.

```
PRINT "Ich bin ein QBasic-Programmierer"
```

1b) Schreiben Sie ein Programm, welches von dem Sleep-Befehl drei Sekunden lang angehalten wird.

```
PRINT "Programm..."
SLEEP 3
```

2a) Schreiben Sie ein Programm, in welchem der Variablen *Variable* der Wert 10 zugewiesen wird. Danach soll der User den Wert der Variablen verändern können. Geben Sie den neuen Wert ebenfalls aus.

```
Variable = 10
PRINT Variable
INPUT Variable
PRINT Variable
```

2b) Schreiben Sie ein Programm, bei dem der User seinen Namen und sein Alter angegeben kann, welches dann wieder ausgegeben wird.

```
INPUT "Wie heisst du "; name$
INPUT "wie alt bist DU?"; alter
PRINT "Du heisst "; name$; " und bist"; alter; "Jahre alt"
```

4a) Schreiben Sie ein kleines Quiz

```
10
PRINT "Dieser Kurs heißt ?"
PRINT " (1) QBasic in 12h"
PRINT " (2) Anders
INPUT frage1
IF frage1 = 1 THEN
   PRINT "Richtig! "
ELSE IF frage1 = 2 THEN PRINT "Falsch! " ELSE BEEP: GOTO 10
END IF
END
```

7a) Schreiben Sie ein Programm, bei dem alle Grafik-Befehle, die Sie nun kennen, vorkommen.

```
SCREEN 9
PSET (10, 10), 15
LINE (20, 20) - (20, 30)
CIRCLE (60, 60) , 15, 15
PAINT (60, 60), 15
```

8a) Schreiben Sie unter Verwendung des Zufallsgenerators ein kleines Zahlenraten.

```
RANDOMIZE TIMER
Zufall = INT(RND * 20) + 1     'Zufallszahl 1...20
```

```
10
PRINT "Welche Zahl vermuten Sie (1...20) ? "
INPUT frage
IF frage < Zufall THEN PRINT "Zu niedrig! ": SLEEP: GOTO 10
IF frage > Zufall THEN PRINT "Zu hoch! ": SLEEP: GOTO 10
PRINT "Gewonnen!!!"
SLEEP
END
```

9a) Schreiben Sie ein Programm, das "Alle meine Entchen" abspielt.

```
'Alle meine Entchen abspielen
PLAY "MFT160O3L8cdefL4ggL8aaaaL2gL8aaaaL2gL8ffffL4eeL8ggggL4c"
```

10a) Erstellen Sie ein Programm mit einer SUB, welches sie aufrufen. Dieses Beispiel zeigt, wie man an eine SUB zwei Variablen, nämlich t$ und i, als Parameter übergeben kann

Hauptprogramm:

```
t$ = "Hello World"
FOR i = 1 TO 10
CALL test(t$, i)
NEXT i
SLEEP
```

Unterprogramm:

```
SUB test(t$, i)
PRINT t$; i; ". Ausgabe"
END SUB
```

Copyright © 2002-2003 by Kai Schnieder (KaiSchnieder@gmx.de) und Thomas Antoni (www.antonis.de)