

SelfQB - ein QBasic-Crashkurs

Version 2.0, (c) Thomas Antoni und Frank Neumann (Triton), 2001 - 18.2.2004

Inhalt

1. Was ist SelfQB?.....	Seite 1
2. Was ist QBasic?.....	Seite 2
3. Wie bekomme ich QBasic auf meinen Computer?.....	Seite 3
4. Mein erstes Programm.....	Seite 4
5. Farbcodes.....	Seite 6
6. Variablen und Datentypen.....	Seite 6
7. Verzweigungen und Anwendereingaben.....	Seite 7
8. Schleifen und Abfrage von Einzeltasten-Betätigungen.....	Seite 11
9. Die QBasic-Onlinehilfe.....	Seite 13
10. Wartezeiten erzeugen.....	Seite 13
11. Mathe-Funktionen und FOR...NEXT-Schleifen.....	Seite 14
12. Zufallszahlen.....	Seite 16
13. Text bearbeiten.....	Seite 17
14. Subroutinen.....	Seite 18
15. Funktionen.....	Seite 20
16. Grafik.....	Seite 20
17. Sound.....	Seite 22
18. Felder.....	Seite 24
19. Arbeiten mit Dateien.....	Seite 25
20. Mehrfachverzweigungen mit SELECT CASE.....	Seite 27
21. EXE-Dateien erstellen.....	Seite 30
22. Tipps und Tricks.....	Seite 30
23. Wie steige ich noch tiefer in QBasic ein?.....	Seite 32
24. Liste der Beispielprogramme.....	Seite 32

1. Was ist SelfQB?

SelfQB ist ein QBasic- Selbstlernkurs, der Anfängern einen blitzschnellen Einstieg in QBasic innerhalb von 2 bis 3 Stunden ermöglicht. Du brauchst keinerlei Vorkenntnisse in der Datenverarbeitung und im Programmieren. Der Kurs führt Dich ohne unnötigen theoretischen Ballast anhand leicht nachvollziehbarer Beispiele bequem in die wichtigsten Aspekte der QBasic- Programmierung ein. Für die ganz Eiligen unter euch reicht es aus, zunächst nur die Kapitel 1 bis 11 durchzuarbeiten, was in ca. einer Stunde möglich sein sollte.

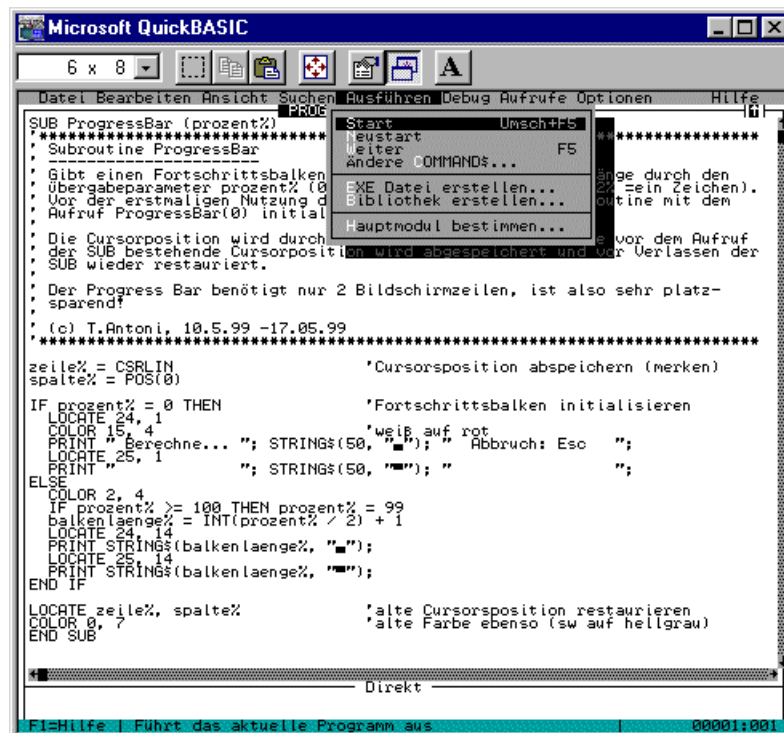
Die 20 Beispielprogramme sind als direkt startbare BAS-Dateien im Verzeichnis PROGS/ des Download-Pakets enthalten.

Diesen Kurs gibt es exklusiv nur auf www.qbasic.de und www.silizium-net.de sowie in der QB-MonsterFAQ. Eine Veröffentlichung auf anderen Webseiten ist nur mit Erlaubnis der Autoren gestattet.

2. Was ist QBasic?

QBasic ist eine Programmiersprache für MS-DOS, die aber problemlos auch unter allen Windows-Versionen lauffähig ist. QuickBasic ist der ältere und größere Bruder von

QBasic und dazu 100% kompatibel. QuickBasic hat mehr Befehle als QBasic und beinhaltet einen zusätzlichen Compiler zum Erstellen von EXE-Dateien. QBasic hingegen ist ein so genannter Interpreter und kann erstellte Programme nur mit Hilfe der Entwicklungsumgebung Zeile für Zeile ablaufen lassen. QBasic wurde mit den MS-DOS Versionen 5.0 - 7.0 sowie Windows 95 und 98 kostenlos mitgeliefert. QB ist die Abkürzung für diese beiden Programmiersprachen. QB erfreut sich auch über 10 Jahre nach Erscheinen der letzten offiziellen Version von 1991 immer noch großer Beliebtheit. Im Internet findest Du hunderte von Webseiten zu QB mit Unmengen von Beispielprogrammen, Tutorials und Foren.



Screenshot meiner QuickBasic 4.5 Entwicklungsumgebung, individuell konfiguriert mit 50 Zeilen und weißem Hintergrund

Doch was heißt eigentlich "BASIC"? BASIC ist eine Abkürzung für: "Beginner's All-purpose Symbolical Instruction Code". Was übersetzt etwa "symbolische Allzweck-Programmiersprache für Anfänger" heißt. Wobei diese Bezeichnung nicht mehr ganz zutrifft, denn auch Profis benutzen heutzutage verschiedene BASIC-Sprachen.

Die Geschichte von BASIC beginnt 1960. Die Herren Kemeny und Kurtz stellen die erste Version vor. Bis heute ist BASIC berüchtigt als Sprache mit der Programme entstehen, die schwer lesbar und voller Fehler (Bugs) sind, eine schlechte Bedienung enthalten, sehr langsam sind usw. Diese Vorurteile stammen alle aus der Zeit der Homecomputer. Und die waren bekanntlich ziemlich dürrtig ausgestattet. 40 KB RAM und schnecken- langsame Prozessoren gab es da! Das ist natürlich kein Vergleich zu den Ressourcen heutiger PCs. Die Programmierer waren damals also gezwungen durch Tricks Anweisungen zu sparen. Das führte schon bald zum sogenannten "Spagetticode" der häufig so extrem unübersichtlich war, dass die Programmierer ihn selber kaum noch verstanden. Und gerade dieser "Spagetticode" ist auch der Grund dafür, dass BASIC-

Dialekte auch heute noch bei vielen Informatik- Professoren indiskutabel sind. Und gerade weil BASIC fast jeder hatte, gab es auch mehr BASIC-Spagetti :). Wäre z.B Pascal die vorherrschende Sprache geworden, hätte es genauso Pascal-Spagetti gegeben. Auch die Bedienung von BASIC selber war früher sehr schlecht.

Das alles hat sich aber mit Erscheinen der Microsoft-BASIC- Varianten QuickBasic und QBasic entscheidend geändert. Nun steht eine sehr leistungsfähige Entwicklungsumgebung zur Verfügung. Ein weiteres Argument gegen BASIC war der geringe Sprachumfang. Das kann man QBasic/QB aber nicht mehr anlasten. Dinge wie Prozeduren, Unterprogramme, Einbindung von Assembler und C, Hardware- Programmierung und Ähnliches sind nun selbstverständlich. Bei etwas Disziplin kann man sauber, übersichtlich und ohne Spagetticode programmieren. Du siehst also: BASIC ist ein modernes Entwicklungssystem und beliebter denn je. Die alten Vorurteile gegen BASIC treffen alle nicht mehr zu. Vor allem ist BASIC eine Sprache mit Zukunft, denn es gibt immer mehr und leistungsfähigere Dialekte. Besonders erwähnenswert sind hier Blitz Basic, PureBasic, DarkBasic, PowerBASIC und VisualBasic.

Heute wird QBasic hauptsächlich von Programmier- Einsteigern und im IT- Unterricht der Schulen verwendet. Stelle bitte nicht all zu hohe Ansprüche daran, und versuche nicht, ein Spiel wie Quake 3 oder einen Internet-Browser damit zu programmieren. QBasic ist nach wie vor die ideale Sprache um die Grundlagen des Programmierens zu lernen. Alles, was Du mit QBasic lernst, kannst Du beim Aufstieg zu einem anderen BASIC- Dialekt oder zu einer ganz anderen Programmiersprache sehr gut brauchen.

3. Wie bekomme ich QBasic auf meinen Computer?

Die QBasic-Entwicklungsumgebung besteht aus den beiden Dateien QBASIC.EXE und QBASIC.HLP, die insgesamt nur 335 KB groß sind. Du kannst QBasic auf vielen Webseiten herunterladen, z.B. auf www.qbasic.de unter [Download | Compiler].

Kopiere die Dateien auf Deine Festplatte, z.B. in den Ordner "C:\QBASIC\". Bei der Gelegenheit solltest Du auch gleich einen Unterordner für Deine selbst erstellten QBasic- Programme anlegen, z.B. "C:\QBASIC\PROGS\".

Jetzt noch schnell eine Verknüpfung auf dem Desktop zu QBASIC.EXE eingerichtet und fertig ist die Installation.

Wenn Du willst, kannst Du im Windows-Eigenschaften-Dialog einige Optimierungen durchführen. Klicke dazu mit der rechten Maustaste auf QBasic.exe. Dort solltest Du die folgenden Anpassungen machen:

- Bei Bedarf die Schriftart anpassen, um eine optimale Lesbarkeit zu erzielen
- Über [Bildschirm | Fenster] den Ablauf in einem Teilfenster wählen
- Die "Leerlaufaktivität" auf einen mittleren Wert setzen
- Bei Windows 2000/XP "Ausführen als Symbol" aktivieren und unter "Sonstiges" den Punkt "Hintergrund...immer vorübergehend aussetzen" deaktivieren
- Bei Windows 2000/XP die "DOS-kompatible Timer-Emulation" aktivieren

4. Mein erstes Programm

Fast alle Kurse übers Programmieren beginnen mit einem Programm, das ein simples "Hallo Welt" auf den Bildschirm ausgibt. So wollen wir auch hier verfahren. Starte die QBasic-Entwicklungsumgebung QBASIC.EXE und tippe dort folgende Zeile ein:

```
print "Hallo Welt"
```

Wenn Du die erste Zeile mit der Eingabetaste abschließt, siehst Du, dass das QBasic-Befehlsschlüsselwort "print" sich automatisch in ein groß geschriebenes "PRINT" verwandelt. Diese automatische "Syntaxkontrolle" ist gerade für Einsteiger eine sehr angenehme Eigenschaft von QBasic. Du siehst jederzeit sofort, ob Du die Befehle richtig hingeschrieben hast. Wenn ein Befehl sich nicht in Großbuchstaben verwandelt, muss irgend etwas faul sein.

Wähle jetzt den Menüpunkt [Ausführen | Start] und schwuppdiewupp führt QBasic das Programm aus: Auf dem Bildschirm erscheint der Text "Hallo Welt". Gratulation: Du hast soeben Dein allererstes QBasic-Programm erfolgreich geschrieben und ausgeführt. Diesen Tag solltest Du Dir ganz dick im Kalender ankreuzen. Heute ist der Beginn Deiner Programmierer-Laufbahn :)

Um die genaue Bedeutung des Befehls PRINT... wollen wir uns erstmal nicht kümmern. Das wird später noch im Detail erklärt.

Mit einer beliebigen Tastenbetätigung kehrst Du zur Entwicklungsumgebung zurück. Da ich Dich für sehr gebildet halte ;-), denke ich, dass wir gleich mit etwas Schwierigerem weitermachen können. Zum Beispiel mit dem folgendem kleinen Progrämmchen:

```
' *****  
' ERSTPROG.BAS - erstes kleines Programm fuer SelfQB  
' =====  
' (c) by Triton 2001/2002 *g*  
' *****  
CLS 'Bildschirm loeschen  
texta$ = "Ich sitze hier am Computer und..."  
textb$ = "...schreibe Programme, die keinen Sinn machen."  
textc$ = "Doch sie sind cool - sie sind ja auch von mir!"  
'  
COLOR 1, 15 'blaue Schrift (=1) auf weissem Grund (=15)  
LOCATE 8, 20 'Ausgabe ab Zeile 8, Spalte 20  
PRINT texta$ 'Inhalt von texta$ anzeigen  
'  
SLEEP 2 '2 sec warten  
LOCATE 10, 20  
COLOR 4, 6 'rote Schrift auf braunem Grund  
PRINT textb$  
'  
SLEEP 3  
COLOR 15, 1 'weisse Schrift auf blauem Grund  
PRINT 'Leerzeile anzeigen  
LOCATE 12, 20  
PRINT textc$  
SLEEP 5  
END
```

Wie Du zugeben musst, sind das sehr einfach zu verstehende Zeilen. Wenn Du willst, kannst Du Dir den hier angegebenen Code auch in den Windows-Editor kopieren [Makieren | STRG+C | Editor starten | STRG+V | als Datei namens ERSTPROG.BAS speichern] und

in QBasic ausprobieren - über "Datei | Öffnen" .

Wir können aus diesem Beispiel Folgendes lernen:

1. CLS löscht den Bildschirm
2. Je Zeile schreibt man einen QBasic-Befehl hin. Mehrzeilige Befehle sind nicht möglich.
3. Das Hochkomma ' leitet Kommentar ein, das sind Erläuterungen zum Programm, die der QBasic-Interpreter beim Abarbeiten des Programms einfach überliest. Statt des Hochkommas kann man auch REM verwenden, das aber am Zeilenanfang stehen muss.
4. Texta\$ ist eine Textvariable - die Fachleute sprechen von "Stringvariable" - und kann Textdaten speichern. Mit Texta\$ = "Ich sitze hier am Computer" wird der in Anführungszeichen stehende Text in die Stringvariable Texta\$ im RAM eingetragen. Die Namen von Stringvariablen müssen mit einem Dollarzeichen "\$" abgeschlossen werden. Mehr über Variablen erfährst Du in Kapitel 6.
5. COLOR setzt die Farbe des Textes, der mit PRINT angezeigt wird, und nach dem Komma die Farbe des Texthintergrundes. Die Bedeutung der hinter COLOR stehenden Farbcodes erfährst Du im nächsten Kapitel. Du kannst sie auch in der QBasic- Onlinehilfe nachschlagen, indem Du mit der rechten Maustaste auf den Befehl COLOR klickst. Dadurch öffnet sich die passende Infoseite der QBasic- Onlinehilfe. Klicke dann unten im Hilfe-Fenster auf "Farbattribute". Wie Du siehst, verfügt QBasic über eine vorzügliche Onlinehilfe. Mehr dazu in Kapitel 9. Mit Esc beendest Du die Hilfe.
6. "LOCATE Zeile, Spalte" setzt die Position des Cursors auf dem Bildschirm für die anschließend folgende Textanzeige ("locate" ist das englische Wort für "lokalisieren", "Position festlegen").
7. PRINT zeigt den nachfolgenden Text an der durch LOCATE festgelegten Stelle auf dem Bildschirm an. Der Text kann direkt in Ausführungszeichen angegeben werden, wie beim obigen "Hallo Welt" - Programm, oder er kann in einer Stringvariablen, z.B. texta\$, stehen. Einige Besonderheiten des PRINT- Befehls erfährst Du in Kapitel 13.
8. PRINT ohne nachfolgenden Text überspringt einfach eine Zeile, und auf dem Bildschirm erscheint eine Leerzeile, wie Du bei der letzten ausgegeben Zeile (vor textc\$) sehen kannst.
9. SLEEP versetzt den Computer für die hinter SLEEP angegebenen Anzahl von Sekunden in einen schläfrigen Wartezustand und fährt danach mit der Ausführung des Programms fort. In Kapitel 11 erfährst Du, wie Du genauere und kürzere Wartezeiten erzeugen kannst.
10. END beendet das Programm und übergibt die Kontrolle wieder an die QBasic- Entwicklungsumgebung. Man kann END meist auch weglassen. Für eine saubere Programmieretechnik solltest Du es Dir aber angewöhnen, durch ein END den Abschluss Deines Programms eindeutig zu kennzeichnen.

Du siehst: Bei vielen Befehlen kann man die Funktion schon aus dem Befehlsnamen ableiten, wenn man ein wenig der englischen Sprache mächtig ist. CLS bedeutet z.B. "Clear Screen".

Jeder Befehl hat eine sogenannte Syntax. Dieser Begriff bezeichnet ganz einfach den schematischen Grundaufbau eines Befehls. Wird die Syntax falsch angewendet, so funktioniert der Befehl entweder gar nicht oder nicht so wie er soll. Das kann man gut mit der Grammatik eines Satzes vergleichen: "Oder richtig ist Satz das hier?" *g*. Bei den meisten Syntaxfehlern, wie etwa "SLEEP texta\$", zeigt QBasic beim Beenden der Zeile sofort eine aussagekräftige Fehlermeldung an.

5. Farbcodes

Programmieren soll keine triste schwarz/weiße Sache bleiben. Lass uns lieber farbige Welten erschaffen. In der folgenden Tabelle findest Du die Codes der 16 in QBasic standardmäßig verfügbaren Farben:

0	= Schwarz	8	= Grau
1	= Dunkelblau	9	= Hellblau
2	= Dunkelgrün	10	= Hellgrün
3	= Dunkel-Türkis	11	= Helltürkis
4	= Dunkelrot	12	= Hellrot
5	= Dunkelviolett	13	= Hellrot - Rosa
6	= Dunkelbraun-Oliv	14	= Gelb
7	= Hellgrau	15	= Weiß

Nur die ersten 8 Farben (Codes 0...7) sind als Hintergrundfarben verwendbar. In den später behandelten Grafik-Bildschirmmodi gibt es noch viele weitere Farben.

Und in dem folgenden Programm wollen wir gleich ein wenig mit Farben herumspielen:

```
' *****
' COLOR-1.BAS = Einsatz von Farben im QBasic-Textmodus
' =====
' Thomas Antoni, 6.1.2004
' *****
COLOR 12, 1 'hellrote Schrift auf blauem Hintergrund
CLS 'Bildschirm loeschen d.h. in der
' Hintergrundfarbe blau einfärben
LOCATE 12, 30 'Cursor auf Zeile 12, Spalte 30
PRINT "Die Welt ist farbig!"
PRINT
LOCATE , 25 'Cursor auf Spalte 30
COLOR 14, 2 'gelbe Schrift auf gruenem Hintergrund
PRINT " Und hier gelb auf gruen :) "
COLOR 13 'braune Schrift
LOCATE 20, 26 'Cursor auf Zeile 20, Spalte 22
PRINT "Rosa passt zu gruen ~(ø;ø)~"
COLOR 15, 0 'normale Farben: weiss auf schwarz
```

Wie Du siehst, kann man den Kommentar hinter dem Hochkomma hervorragend dazu verwenden, das Programm und die verwendeten Tricks näher zu beschreiben. Die ersten zwei Zeilen zeigen übrigens, dass CLS den gesamten Bildschirm in der vorher aktivierten Hintergrundfarbe (hier 1=blau) einfärbt - eine sehr praktische Sache!

6. Variablen und Datentypen

Variablen sind Plätze im RAM, in denen Du Zahlen oder Textstücke abspeichern und beliebig oft im späteren Programmverlauf verwenden kannst. Du kannst auch den Inhalt der Variablen überschreiben, also z.B den Inhalt von a\$ durch den Befehl a\$ = "Test" verändern.

Es gibt mehrere Typen von Variablen. Im letzten Abschnitt hast Du die Stringvariablen kennengelernt, die durch ein an den Variablennamen angehängtes "\$" gekennzeichnet

werden und die Text enthalten. Dann gibt es die sogenannten Integer- Variablen, die ganze Zahlen von -32768 bis 32768 (2^{15}) beinhalten. Integer- Variablen werden durch das nachfolgende Prozentzeichen % gekennzeichnet - z.B. otto%. Dann gibt es die langen Ganzzahlen, die von - 2.147.483.647 bis 2.147.483.647 (2^{31}) reichen. Sie werden durch die Typbezeichnung & gekennzeichnet. Die nächst größere Kategorie sind die Komma- Variablen einfacher Genauigkeit (Typbezeichnung = !) und doppelter Genauigkeit (#). Diese reichen bis zu astronomischen Größen mit jeweils 8 bzw. 16 Nachkommastellen. Benutze immer den niedrigsten Variablentyp der für Deine Anwendung ausreicht, also bitte nicht immer Variablen doppelter Genauigkeit verwenden. Integer-Variablen lassen sich am schnellsten verarbeiten.

Und hier noch eine kleine Zusammenfassung der QBasic-Datentypen mit Beispielen für die Variablennamen:

Variable...	Datentyp.....	Beschreibung und Wertebereich.....
- anna%	: INTEGER,	Ganzzahl mit Vorzeichen (16 bit) -32768...32767
- otto&	: LONG,	lange Ganzzahl mit Vorzeichen (32 bit) -2147483648...2147483647
- egon!	: SINGLE,	einfach lange Gleitpunktzahl (32 Bit, 7 Stellen genau) $\pm 2,802597 \cdot 10^{-45} \dots \pm 3,402823 \cdot 10^{38}$
- egon	: SINGLE	(Das "!" ist bei SINGLE-Variablen weglassbar)
- paul#	: DOUBLE,	doppelt lange Gleitpunktzahl (64 Bit, 16 Stellen genau) $\pm 4,446590812571219 \cdot 10^{-323} \dots \pm 1,79769313486231 \cdot 10^{308}$
- duda\$: STRING,	Text-String (Zeichenkette, max. ca. 32767 Zeichen)

Die Kommazahlen werden in der IT-Sprache "Gleitpunktzahlen" genannt. Das Komma erscheint in QBasic - wie in England und USA üblich - als Dezimalpunkt. Ein normales Komma wird von QBasic als Fehler angemeckert.

Exponentialdarstellung von Gleitpunktzahlen

QBasic zeigt sehr kleine und sehr große Zahlen in einer Exponentialdarstellung an. Dabei wird der Zehner- Exponent "10^" bei SINGLE-Werten mit einem vorangestellten "E" und bei DOUBLE- Werten mit "D" angezeigt.

Beispiele:

```
a! = 2 ^ 24 'Einfach lange Gleitpunktzahl 2 hoch 24
PRINT a! 'Anzeige: 1.677722E+07 entspricht 1,677722 * 10^7
b# = 1 / 1234567890 'doppelt lange Gleitpunktzahl
PRINT b# 'Anzeige: 8.10000007371D-10 entspricht 8,10000007371 * 10^(-10)
```

Regeln für Variablennamen

Variablennamen dürfen bis zu 40 Zeichen enthalten und müssen mit einem Buchstaben beginnen. Gültige Zeichen sind A-Z, 0-9 und der Punkt (.). Umlaute, Sonderzeichen und QBasic-Schlüsselworte, wie etwa PRINT, sind als Variablennamen verboten.

7. Verzweigungen und Anwendereingaben

Obwohl wir mit den bisher kennengelernten Befehlen schon ein paar nette kleine Programme schreiben könnten, würden die Programme schnell langweilig und eintönig werden und immer nach "Schema F" ablaufen. Es muss also mehr Abwechslung in unsere Programme - zum Beispiel durch Verzweigungen.

Dies lernen wir im folgenden Abschnitt, ebenso, wie wir den Programmbenutzer in das

Geschehen eingreifen lassen können. Stellen wir uns vor, wir schreiben ein kleines Programm, das den Benutzer fragt, wie er heißt und wie das Wetter heute ist. Auf die letztere Frage soll entweder mit "gut", oder "schlecht" geantwortet werden. Das Programm soll die Benutzereingabe dann durch eine passende Bildschirmausgabe kommentieren. Ein solches Programm sieht etwa so aus:

```
' *****
' INPUTIF.BAS = Demo fuer die Befehle INPUT und IF...THEN
' =====
' Dieses QBasic-Programm demonstriert
' - Benutzer-Abfragen mit INPUT
' - Verzweigungen mit IF...THEN
'
' (c) Triton, 2002
' *****
CLS
INPUT "Wie ist Dein Name ?", name$
INPUT "Wie ist das Wetter heute? (gut), oder (schlecht) "; wetter$
'
IF wetter$ = "gut" THEN text$ = ", dann gehen Sie lieber raus an die Luft!"
'
IF wetter$ = "schlecht" THEN text$ = ", dann bleiben Sie lieber hier!"
PRINT "Ok, "; name$; text$
SLEEP
```

Aus diesem Programm können wir sogar noch mehr lernen als aus dem ersten. Im Folgenden eine Erläuterung der neuen Befehle:

1. Anwendereingaben abfragen mit INPUT

Der Text in den Anführungszeichen "" wird als Frage gestellt, und die Antwort in der Variable hinter dem "," gespeichert. Vielleicht ist Dir aufgefallen, dass bei der 2. Frage ein "?" erscheint, obwohl es gar nicht in der Frage steht. Das kommt vom Semikolon ";" hinter der Frage. Wenn Du das Semikolon durch ein Komma ersetzt, wie bei der ersten Frage, dann erscheint kein Fragezeichen.

2. Verzweigung mit IF...THEN

Ganz offensichtlich bewirkt hier die IF...THEN - Anweisung eine Unterscheidung der Handlung. So wird bei der Eingabe von gutem Wetter ein anderer Text ausgegeben als bei schlechtem Wetter.

3. Erweitertes PRINT

PRINT wird hier anders verwendet als zuvor. Statt nur den Text in "" Zeichen auszugeben, wird auch der in den String-Variablen name\$ und text\$ hinterlegte Text 'rangehängt. Hätten wir übrigens statt einem ";" ein "," überall in die PRINT Anweisung gebracht, wären alle Textteile durch Tabulatorenweiten, also 14-er Zeichenschritte, voneinander getrennt! Mehr dazu in Kapitel 11.

4. SLEEP ohne Zeitangabe

Auch SLEEP wird hier anders verwendet als vorher. Da kein Zeitraum zum Warten angegeben wurde, wartet SLEEP einfach bis eine beliebige Taste gedrückt wird. Aber Vorsicht: SLEEP löscht den Tastaturpuffer nicht; das kann im nachfolgenden Programm zu unvorhergesehenen Effekten führen. In Kapitel 8 wirst Du eine professionellere Methode zum Warten auf einen Tastendruck kennenlernen, die diesen Mangel nicht hat.

5. IF...THEN- Abfrage

Die interessanteste Anweisung ist hier IF...THEN... Damit wollen wir uns jetzt

näher beschäftigen.

Diese Anweisung lässt sehr vielfältig benutzen. Ihre Syntax kommt fast der eines normalen Satzes gleich:

```
IF <Bedingung> THEN <Anweisungen> ELSE <Anweisungen>
```

bzw. auf Deutsch:

```
WENN dies erfüllt ist MACHE dasunddas ANSONSTEN diesesundjenes
```

Dieses doch recht einfache Prinzip kann in sehr vielen Varianten eingesetzt werden, sodass sich alle möglichen Dinge unterscheiden lassen. Es ist für einen Programmierer wichtig, das Prinzip dieser Anweisung zu verstehen. Beherrscht man diese, dann tut man sich nämlich bei vielen Programmieraufgaben erheblich leichter! Jetzt wollen wir zu einem weiteren Beispiel kommen, das die Möglichkeiten des IF Befehls gut demonstriert:

```
' *****
' IFTHEN.BAS = QBasic-Demo fuer IF...THEN-Abfragen
' =====
' (c) Triton, 2002
' *****
CLS
start:
INPUT "Welchen Wochentag haben wir heute (1=Montag...7=Sonntag): ", tag%
IF tag% < 1 OR tag% > 7 THEN
    PRINT "Unguelte Eingabe!"
    GOTO start
END IF
IF tag% > 0 AND tag% < 6 THEN
    PRINT "Werktag!"
ELSE
    PRINT "Wochenende!"
END IF
END
```

Hier gibt es wieder ein paar Neuigkeiten:

1. Mehrzeiliger IF...THEN-Befehl

Zum Einen bemerkt man, dass die THEN Anweisungen (also alles nach THEN) nicht in die gleiche Zeile geschrieben werden muss, sondern auch mehrere Zeilen lang sein kann. In solchen Fällen muss die IF...THEN -Anweisung durch eine extra Zeile mit END IF abgeschlossen werden.

Es gibt noch sehr viele weitere Verwendungsmöglichkeiten der IF Anweisung, die aber alle zu erklären würde zweifelsohne den Rahmen sprengen. Wenn Du Lust hast, kannst Du ja mal in der QBasic -Onlinehilfe unter "IF" stöbern. Gib dazu in der Entwicklungsumgebung einfach "if" ein und betätige dann die F1-Taste.

2. Unbedingte Sprünge mit GOTO

Außerdem entdeckt man GOTO mit einer nachfolgenden "Sprungmarke", in diesem Fall "start". GOTO veranlasst den Computer, den normalen Programmablauf zu verlassen und zu dem Befehl nach der Sprungmarke zu springen. Von dort macht er mit der Programmausführung weiter. Natürlich kann man mit GOTO auch in einen späteren Teil des Programms springen, der noch nicht regulär abgearbeitet wurde. Du solltest GOTO sparsam verwenden. Bei

wildem Hin- und Hergesprünge verlierst Du sonst schnell die Übersicht. Für die Namen von Sprungmarken gelten dieselben Regeln wie für Variablennamen; siehe Kapitel 6.

3. **LINE INPUT ermöglicht Eingaben, die Kommas enthalten**

Kommen wir zurück zu Abfragen, die den Benutzer betreffen. Wir hatten am Anfang bereits den Befehl INPUT verwendet. Wenn Du schon etwas mit ihm herumexperimentierst hast, hast Du vielleicht bemerkt, dass INPUT keine Kommas "," in den Antworten erlaubt. Das mag auf den ersten Blick zwar nicht schlimm erscheinen, kann Dir aber später große Probleme bereiten. Deshalb lernen wir jetzt noch eine Alternative kennen: LINE INPUT. Dieser Befehl unterscheidet sich von INPUT nur insofern, dass er auch "," in der Antwort toleriert. Die Syntax ist sonst die gleiche.

4. **Vergleichsoperatoren**

Nach IF erscheinen meist Vergleichoperationen. QBasic kennt die folgenden Vergleichsoperatoren:

```
IF a = b THEN      ' a gleich b
IF a < b THEN      ' a kleiner als b
IF a > b THEN      ' a größer als b
IF a <= b THEN     ' a kleiner oder gleich b
IF a >= b THEN     ' a größer oder gleich b
IF a <> b THEN     ' a ungleich b
```

5. **logische Verknüpfungen AND und OR**

Vergleichoperationen können durch die logischen Verknüpfungen AND und OR miteinander kombiniert werden z.B.:

```
IF a >= 10 AND a <= 20 THEN PRINT "a liegt zwischen 10 und 20"
IF a < 10 OR a > 20 THEN PRINT "a liegt außerhalb des Bereichs
10...20"
```

Das erste Beispiel ist eine UND-Verknüpfung. Der THEN-Zweig wird durchlaufen, wenn a größer oder gleich 10 UND kleiner oder gleich 20 ist.

Das zweite Beispiel beinhaltet eine ODER-Verknüpfung. Der THEN-Zweig wird durchlaufen, wenn a kleiner als 10 ODER größer als 20 ist.

8. Schleifen und Abfrage von Einzeltasten-Betätigungen

Ein weiterer Befehl, den man sehr gut gebrauchen kann, ist INKEY\$. INKEY\$ liest den Tastaturpuffer aus, also die letzte gedrückte Taste. Eine einfache Anwendung von INKEY\$ wäre zum Beispiel, eine Tastenabfrage in einer Schleife. Das kann man ganz einfach so machen:

```
' *****  
' TIME.BAS = Uhrzeitanzeige in QBasic  
' =====  
' Die aktuelle Uhrzeit wird solange  
' in Zeile 12, Spalte 35 angezeigt,  
' bis der Anwender die Esc-Taste  
' drückt (ASCII-Code 27).  
'  
' (c) Thomas Antoni, 7.1.2004  
' *****  
DO  
  LOCATE 12, 35: PRINT TIME$  
LOOP UNTIL INKEY$ = CHR$(27)  
END
```

Dies Programm demonstriert die folgenden neuen Funktionen:

1. **INKEY fragt eine Tastenbetätigung ab**
Hier wird abgefragt, ob die ESC Taste gedrückt wurde, die den ASCII-Code 27 hat. Die in der Klammer angegebenen Codes für die einzelnen Tasten kann man in der QB Hilfe nachlesen unter [Hilfe | Inhalt | ASCII Zeichencodes].
2. **Mehrere Befehle in einer Zeile**
Wie Du in der 2. Zeile siehst, kannst Du mehrere QBasic-Befehle auch ausnahmsweise in eine Zeile schreiben, wenn Du sie durch Doppelpunkte voneinander trennst. Mach bitte sparsamen Gebrauch davon. Es erhöht nicht gerade die Übersichtlichkeit, besonders wenn die Zeilen dadurch länger als ca. 60 Zeichen werden.
3. **Uhrzeit ermitteln mit TIME\$**
Die in QBasic eingebaute Systemfunktion TIME\$ ermittelt die aktuelle Uhrzeit und liefert diese als Textstring zurück. In der Schleife wird die Uhrzeit immer wieder neu in der Bildschirmzeile 12 ab Spalte 35 angezeigt.
4. **Schleife mit DO..LOOP UNTIL**
Der zwischen DO und LOOP... stehende Programteil wird solange durchlaufen, BIS (engl. "UNTIL") der Computer ein Betätigen der Esc-Taste erkennt. Diese Programmsequenz stellt eine so genannte "Schleife" dar.
5. **Weitere Schleifenkonstruktionen**
QBasic bietet sehr flexible Schleifenkonstruktionen. Es gibt z.B. die folgenden 5 verschiedenen Methoden, die Zahlen von 0 bis 100 anzuzeigen:

```

' *****
' LOOP.BAS = 5 Möglichkeiten, die Zahlen von
' ===== 0 bis 100 anzuzeigen
' (c) Thomas Antoni, 7.11.03 - 30.1.2004
' *****
'
'----- Zaehlschleife mit FOR...NEXT -----
'Bei jedem Durchlauf wird i% um 1 erhöht
FOR i% = 1 TO 100 'Zaehlschleife:
    PRINT i%
NEXT
'
'----- Fussgesteuerte DO...LOOP WHILE Schleife -----
SLEEP
COLOR 13
i% = 0
DO
    i% = i% + 1
    PRINT i%
LOOP WHILE i% < 100 'Ruecksprung nach DO, wenn die
                    'Fortsetzbedingung erfuehlt
'----- Fussgesteuerte DO...LOOP UNTIL Schleife -----
SLEEP
COLOR 14
i% = 0
DO
    i% = i% + 1
    PRINT i%
LOOP UNTIL i% = 100 'Ruecksprung nach DO solange bis
                    'die Abbruchbedingung erfuehlt ist
'----- Kopfgesteuerte DO UNTIL...LOOP Schleife -----
SLEEP
COLOR 8
i% = 0
DO UNTIL i% = 100 'Schleife durchlaufen bis die
    i% = i% + 1    'Abbruchbedingung erfuehlt ist
    PRINT i%
LOOP
'
'----- Kopfgesteuerte DO WHILE...LOOP Schleife -----
SLEEP
COLOR 10
i% = 0
DO WHILE i% < 100 'Schleife solange durchlaufen bis
    i% = i% + 1    'die Abbruchbedingung erfuehlt ist
    PRINT i%
LOOP

```

Die FOR...NEXT-Schleife wird in Kapitel 11 noch genauer erklärt. Die fußgesteuerten DO...LOOP-Schleifen unterscheiden sich von den kopfgesteuerten dadurch, dass der Schleifenkörper auf jeden Fall einmal durchlaufen wird bevor die Fortsetzbedingung abgeprüft wird.

Mit UNTIL ("BIS") oder WHILE ("WÄHREND", "SOLANGE WIE") gibt man an, ob die nachfolgende Bedingung als Abbruchbedingung oder als Fortsetzbedingung fungieren soll.

9. Die QBasic-Onlinehilfe

Wie Du schon gesehen hast, verfügt QBasic über eine hervorragende Onlinehilfe, die Du intensiv nutzen solltest. Sie ersetzt glatt ein viele 100 Seiten dickes Handbuch und enthält zu fast allen Befehlen aussagekräftige Beispielprogramme. Stöbere doch einfach mal ein bisschen darin herum.

Die QBasic-Online-Hilfe hat eine sehr komfortable Navigation. Ähnlich wie auf einer Webseite erhältst Du in jeder Situation Zusatzinformationen über eine Fülle von Text-Links, denen Du per Doppelklick mit der linken Maustaste oder - noch bequemer - per Einfachklick mit der rechten Maustaste folgen kannst.

Willst Du mehr zu einem bestimmten Befehl wissen, dann brauchst Du nur das Befehlsschlüsselwort in der Entwicklungsumgebung einzutippen und dann die F1- Taste zu betätigen. Oder Du klickst einfach mit der rechten Maustaste auf den fraglichen Befehl im Quelltext.

Wenn Du Dir einen Überblick darüber verschaffen willst, welche Befehle für eine bestimmte Problemstellung hilfreich sein könnten, dann wählst Du am besten [Hilfe | Inhalt | Schlüsselwort nach Kontext]. Daraufhin erscheint eine nette, nach Funktionen gegliederte Liste aller QBasic- Befehle. Klickst Du mit der rechten Maustaste auf einen der aufgelisteten Befehle, dann öffnet sich ein Hilfefenster mit detaillierten Informationen zu diesem Befehl.

Bei QuickBasic 4.5 ist die Onlinehilfe sogar noch wesentlich ausführlicher und auch etwas anders organisiert. Dort gelangst Du zu der nach Funktionen gegliederten Befehlsliste über [Hilfe | Themen | Schlüsselwortlisten | Doppelklick auf das gewünschte Thema].

10. Wartezeiten erzeugen

Den Befehl SLEEP haben wir schon kennengelernt. Mit "SLEEP AnzahlSec" kannst Du den Programmablauf für eine ganze Anzahl von Sekunden anhalten.

Genauere und kleinere Wartezeiten lassen sich mit dem TIMER-Befehl realisieren.

TIMER liefert den aktuellen Stand der Systemuhr zurück und zeigt die Anzahl der seit Mitternacht vergangenen Sekunden an. TIMER hat eine Auflösung von 18,2 "Takten" pro sec. D.h. TIMER wird alle 0,056 sec (= 56 ms) um den Wert 0,056 erhöht. Wartezeiten unter 0,056 sec sind daher mit TIMER nicht realisierbar.

Der Timer liefert Gleitpunktwerte vom Typ SINGLE zwischen 0.000 und 86400.000 sec (entspricht den 24 Stunden von 00:00:00h ... 23:59:59h). Bei der Realisierung von Stoppuhren und Countdown-Timern ist der Rücksprung vom Maximalwert 86400.000 auf 0.000 um Mitternacht zu berücksichtigen.

Das folgende Programmchen zeigt, wie Du Wartezeiten mit TIMER realisieren kannst.

```

' *****
' TIMER.BAS = Erzeugung von Wartezeiten
' =====
' Der Anwender gibt eine Wartezeit in sec ein.
' Anschliessend wird die TIMER-Funktion
' verwendet, um den Programmablauf fuer diese
' Zeit anzuhalten. Wenn die Zeit abgelaufen
' ist, ertoent ein Piepston.
'
' (c) Thomas Antoni, 12.11.2003
' *****
CLS
PRINT "Gib eine Wartezeit in [sec] ein; ";
PRINT "Nachkommastellen erlaubt....t = ";
INPUT t!
starttime! = TIMER                'Startzeit merken
'
DO
    PRINT TIMER - starttime!      'abgelaufene Zeit anzeigen
LOOP UNTIL TIMER > starttime! + t! 'Warteschleife
'
PRINT "Wartezeit ist abgelaufen"
BEEP
END

```

11. Mathe-Funktionen und FOR...NEXT-Schleifen

QBasic ist ideal geeignet für mathematische Berechnungen jeder Art. Die wichtigsten Mathe-Operationen sind:

```

x = a + b    'Addition
x = a - b    'Subtraktion
x = a * b    'Multiplikation
x = a / b    'Division
x = SQR(a)   'Quadratwurzel von a
x = a^b      'Exponentialfunktion x = a hoch b
x = a^(1/b)  'allgemeine Wurfelfunktion b-te Wurzel aus a
x = ABS(a)   'Betragsbildung "Absolutwert von a"
x = LOG(a)   'Natürlicher Logarithmus von a (zur Basis e)
x = SIN(a)   'Sinus (a im Bogenmaß; 360° entspricht 2 * Pi)
x = COS(a)   'Cosinus (a im Bogenmaß)
x = TAN(a)   'Tangens (a im Bogenmaß)
a = ATN(x)   'Arcus Tangens, Umkehrfunktion des Tangens, ergibt den
              'Winkel a, dessen Tangens = x ist, im Bogenmaß

```

QBasic bearbeitet Winkel grundsätzlich im Bogenmaß. Das Bogenmaß wird in der Mathematik oft auch mit der Einheit "Radian" oder "rad" bezeichnet. Ein Vollkreis von 360° entspricht im Bogenmaß einem Winkel von $2 \times \pi = 6,283$. Liegt der Winkel im Gradmaß vor, so muss er gemäß der folgenden Formel ins Bogenmaß umgerechnet werden:

Winkel im [Bogenmaß] = Winkel in [Grad] * 6,283 / 360

Die Arcus-Umkehrfunktionen des sin, cos und cot unterstützt QBasic nicht direkt. Sie sind aber leicht über einfache Formeln zu berechnen. Weitere Infos dazu findest Du in meinem QBasic-Kochbuch, das auf www.qbasic.de zum Herunterladen bereitsteht.

Und nun ein kleines Beispielprogramm, das in einer kleinen Tabelle für die Zahlen i von 1 bis 20 die Wurzel, den Quadratwert, die Potenz i hoch 2 und den Sinus von i (i im Gradmaß) angibt:

```

' *****
' MathDemo.bas = Kleines QBasic Mathe-Demoprogramm
' =====
' Fuer die Zahlen i=1 bis zwanzig werden folgende
' Berechnungen durchgefuehrt und die Ergebnisse
' angezeigt:
' - Wurzel aus i
' - i hoch 2
' - 2 hoch i
' - sin (i) ;i im Grad (2*Pi = 360 Grad)
'
' (c) Thomas Antoni, 24.10.2003 - 30.1.2004
' *****
'
CLS
Pi = 3.1416 ' Kreiskonstante
PRINT " i", "Wurzel(i)", "i^2", "2^i", "sin(i)"
PRINT
FOR i = 1 TO 20
    PRINT i, SQR(i), i ^ 2, 2 ^ i, SIN(i * 2 * Pi / 360)
    'der Klammerausdruck nach SIN rechnet den Winkel
    'i vom Gradmass (0..360 Grad ins Bogenmass (0..2*Pi) um
NEXT i
SLEEP
END

```

Zu den einzelnen Befehlen nun noch ein paar Informationen :

1. **FOR...NEXT-Schleife**

Dieses Programm enthält einen bisher noch nicht näher behandelten Schleifenbefehl, nämlich die FOR...NEXT-Schleife. Beginnend beim Anfangswert "1" wird nach jedem Schleifendurchlauf die Schleifenvariable i um "1" erhöht und die Schleife erneut durchlaufen. Das Programm verlässt die Schleife nachdem sie mit dem Endwert i=20 durchlaufen wurde. Anschließend wird die Verarbeitung mit dem Befehl SLEEP hinter der Schleife fortgesetzt.

Die FOR-Schleife eignet sich besonders gut, wenn man die genaue Anzahl der Schleifendurchgänge schon kennt.

Der FOR...NEXT-Befehl hat noch einige Spezialitäten. Die Beschreibung dazu kannst Du Dir in der QBasic-Onlinehilfe anzeigen lassen. Tippe dazu in der Entwicklungsumgebung einfach "for" ein und betätige dann die F1-Taste.

2. **Variablenamen ohne Typkennzeichen**

Die Variable "i" hat im Namen keinen angehängten Kennbuchstaben für den Datentyp. In solchen Fällen nimmt QBasic den Datentyp SINGLE an (einfach lange Gleitpunktzahl). Wenn Ganzzahlen für eine Aufgabe ausreichen, solltest immer ein Typenkennzeichen verwenden, z.B. % für INTEGER- Werte. Das macht Deine Programme wesentlich schneller.

3. **Besonderheiten des PRINT-Befehls**

Du hast bestimmt die wohlgeordnete Bildschirmanzeige des Programms MathDemo.bas bewundert. Soll der PRINT Befehl mehrere Textstücke anzeigen,

so werden diese normalerweise durch Semikolons voneinander getrennt, z.B.

```
t$ = "sel": PRINT "Wie"; t$ 'Anzeige erfolgt als "Wiesel"
```

Dabei erfolgt die Anzeige ohne trennende Leerzeichen, also im Beispiel als "Wiesel". Ersetzt Du das Semikolon durch ein Komma, so werden dagegen nach jedem Textstück die Spalten bis zum nächsten Tabulatorschritt mit Leerzeichen aufgefüllt. Ein Tabulatorschritt setzt den Ausgabe- Cursor auf den Beginn des nächsten 14er- Spaltenbereichs, also auf die 15., 29., 43., 57. oder 71. Spalte. Ersetze doch mal spaßeshalber in der obigen Befehlszeile das Semikolon durch ein Komma:

```
t$ = "sel": PRINT "Wie", t$ 'Anzeige erfolgt als "Wie_____sel"
```

Wie Du siehst, erfolgt die Ausgabe von "sel" ab der 15. statt der 4. Spalte. Die "tabulierte" Anzeige ist sehr nützlich für Anzeigen in Tabellenform, wie das obige Programm MathDemo.bas zeigt.

Es gibt noch eine weitere Besonderheit beim PRINT-Befehl: Normalerweise wird nach einer Bildschirmausgabe per PRINT- Befehl ein Zeilenvorschub eingefügt. Steht am Ende des PRINT-Befehls aber ein Semikolon, dann unterbleibt der Zeilenvorschub, und ein nachfolgender weiterer PRINT-Befehl zeigt die Zeichen bündig zur ersten Anzeige in derselben Zeile an.

Beispiel:

```
PRINT "DU ": PRINT "Esel" 'Anzeige in 2 Zeilen  
PRINT "DU";: PRINT "Esel" 'Anzeige in 1 Zeile
```

Der feine Unterschied liegt, wie gesagt, in dem Semikolon.

12. Zufallszahlen

QBasic hat einen netten eingebauten Zufallsgenerator. Du machst ihn mit dem Befehl RANDOMIZE TIMER scharf. Ein danach folgender RND-befehl liefert Dir dann eine Zufallszahl im Bereich 0 bis 0.999999999. RND steht für engl. "RaNDom number" = Zufallszahl.

Was ist aber nun, wenn Du eine ganzzahlige Zufallszahl zwischen 1 und n benötigst? Dann gehst Du einfach nach dem folgenden Rezept vor:

```
' *****  
' WUERFEL.BAS = Elektronischer Wuerfel  
' =====  
' Bei jeder Tastenbetaetigung wird eine Zufallszahl zwischen 1 und 6  
' erzeugt und angezeigt.  
' (c) Thomas Antoni, 28.10.2003 - 7.11.2003  
' *****  
CLS                                'Bildschirm loeschen  
DO  
    RANDOMIZE TIMER                'Zufallszahlengenerator initialisieren  
    Wuerfelzahl% = INT(RND * 6) + 1 'ganzzahlige Zufallszahl zwischen 1 u.6  
    PRINT Wuerfelzahl%  
DO  
    taste$ = INKEY$  
    LOOP WHILE taste$ = ""         'Warten auf Tastenbetaetigung  
    LOOP UNTIL taste$ = CHR$(27)   'Wiederholung bis Esc-Taste betaetigt
```


END

Dies Programm stellt einen "elektronischen Würfel" dar und liefert Zufallszahlen zwischen 1 und 6. Der durch RND erzeugte Zahlenwert 0...0.99999 wird mit 6 multipliziert, um einen Wert zwischen 0 und 5,999999 zu erhalten. Der INT-Befehl erzeugt daraus einen Integer-Wert (Ganzzahl) indem er die Nachkommastellen abschneidet. Der Wertebereich dieser Integerzahl ist 0 ...5. Um den Bereich um 1 auf 1...6 zu erhöhen, muss man noch eine "1" hinzuaddieren.

Je Tastendruck wird eine Zufallszahl angezeigt, solange bis der Anwender die Esc-Taste betätigt. Dafür sorgen die letzten 5 Zeilen. Diese solltest Du Dir genau durchlesen, denn sie bilden die Problemlösung für eine sehr häufige Aufgabenstellung: Eine Befehlsfolge soll bei jedem Tastendruck wiederholt und mit Esc abgebrochen werden.

Und wie erzeugt man eine ganzzahlige Zufallszahl zwischen min% und max%? Das geht nach der folgenden Formel:

```
RANDOMIZE TIMER 'Zufallsgenerator initialisieren
PRINT INT(RND * (max% - min% + 1)) + min%
'ganzzahlige Zufallszahl zwischen min% und max%
```

13. Text bearbeiten

QBasic bietet sehr leistungsfähige Befehle zum Bearbeiten von Text, die wir im folgenden Progrämmchen am Beispiel des Wortes "Wiesel" demonstrieren wollen:

```
'*****
' TEXTBEAR.BAS = Textbearbeitung mit QBasic - die wichtigsten Befehle
' =====
' Dies Programm demonstriert die wichtigsten QBasic-Befehle zum
' Manipulieren von Zeichenketten (Strings). Als Beispiel-String dient das
' Wort "Wiesel". Nach der Erkl  rung des jeweiligen Befehls folgt
' immer jeweils ein Anwendungsbeispiel.
'
' (c) Thomas Antoni, 7.1.2004 - 12.2.2004
'*****
CLS
t$ = "Wiesel" 'Die Befehle sind sowohl auf String-Konstanten (z.B. "Wiesel")
               'als auch auf String-Variablen (z.B. t$) anwendbar
'
'---- t$ = t1$ + t2$ -> Strings zusammenfuegen
c$ = "Wie" + "sel"
PRINT c$                'Ergebnis: "Wiesel"
'
'---- LEFT$(String$, n) -> Liefert n Zeichen links aus einem String zurueck
PRINT LEFT$("Wiesel", 5) 'Ergebnis: "Wiese"
'
'---- RIGHT$(String$, n) -> Liefert n Zeichen rechts aus einem String zurueck
PRINT RIGHT$(t$, 4)      ' Ergebnis: "Esel"
'
'---- MID$(String$, m, n) -> Liefert n Zeichen ab dem m-ten Zeichen zurueck
PRINT MID$("Wiesel", 3, 2) 'Ergebnis: "se"; das m-te Zeichen zaehlt mit
'
'---- MID$(String$, m, n) = Ersatzstring$ -> n Zeichen eines Strings ab
'---- Zeichenposition m durch einen Ersatzstring ersetzen; ideal zum Ersetzen
'---- von Textpassagen durch einen anderen Text!
a$ = "Wiesel"
MID$(a$, 3, 2) = "rb"      'MID$ steht hierbei links von e. Gleichheitszeichen
PRINT a$                  'Ergebnis: "Wirbel"; "es" wird durch "rb" ersetzt
'
'---- LTRIM$(String$) -> Entfernt fuehrende Leerzeichen aus einem String
```

```

PRINT LTRIM$("   Wiesel") 'Ergebnis: "Wiesel" ohne Leerzeichen
'
'---- RTRIM$(String$) -> Entfernt am Ende e.Strings stehende Leerzeichen
PRINT RTRIM$("Wiesel   ") 'Ergebnis: "Wiesel" ohne Leerzeichen
'
'---- INSTR(m, String$, Suchstring$) -> Sucht einen Suchstring in einem String
'----- ab dem m-ten Zeichen und meldet die Zeichenposition des ersten
'----- gefundenen Strings zurueck bzw. 0, wenn der Suchstring nicht gefunden
'----- wurde; ideal zum Suchen von Text!
PRINT INSTR(1, "Wiesel", "sel") 'Ergebnis: "4" = Zeichenposition von "sel"
'
'---- LCASE$ -> Gross- in Kleinbuchstaben umwandeln (nicht für Umlaute)
PRINT LCASE$("Wiesel") 'Ergebnis: "wiesel"
'
'---- UCASE$ -> Klein- in Grossbuchstaben umwandeln (nicht für Umlaute)
PRINT UCASE$("Wiesel") 'Ergebnis: "WIESEL"
'
'---- STR$ -> Numerischen Wert in String umwandeln
b$ = STR$(1 / 3)
PRINT "Ergebnis: "; b$ 'Ergebnis: .3333333
'
'---- VAL -> String in numerischen Wert umwandeln
PRINT VAL(".5") * 4 'Ergebnis: 2 (4 x 0,5)
'
'---- SPACE$(n) -> Liefert einen String mit n Leerzeichen
PRINT "Wie"; SPACE$(2); "sel" 'Ergebnis: "Wie__sel" mit 2 Leerzeichen
'
'---- STRING$(n, Text$) -> Liefert einen String, der n mal das erste Zeichen
'----- des angegebenen Textes enthaelt
PRINT STRING$(5, "Wiesel") 'Ergebnis: "WWWWW"
'
'---- LEN(String$) -> Liefert die Zeichenanzahl des angegebenen Strings
PRINT LEN("Wiesel") 'Ergebnis: 6
'
'---- ASC(String$) -> Liefert den ASCII-Code des ersten String-Zeichens
PRINT ASC("Wiesel") 'Ergebnis: 87 = ASCII-Code von "W"

```

Ich gehe auf die Befehle nicht näher ein, weil ich meine, dass die Beispiele im Programm "TEXTBEAR.BAS" für sich sprechen. Ich will nur noch erwähnen, dass in allen Befehlen als Text-Operand sowohl ein konstanter String-Wert in Anführungszeichen stehen kann, wie z.B. "Wiesel", als auch eine String-Variable, z.B. t\$.

14. Subroutinen

Häufig benötigte Programmteile kannst Du in eine Subroutine auslagern, die Du z.B. in einer Schleife immer wieder aufrufst. Subroutinen werden bei QBasic mit "SUB" abgekürzt und gelegentlich auch "Unterprogramme" oder "Prozeduren" genannt.

Tippe z.B. mal Folgendes in der QBasic-Entwicklungsumgebung ein:

```

CLS
FOR i% = 1 TO 20
CALL Quadrat(i%)
NEXT
SLEEP
END

```

Dies ist das Hauptprogramm, das 20 mal per CALL- Befehl die Subroutine "Quadrat" aufruft und an diese den "Parameter" i% übergibt.

Um jetzt die Subroutine "Quadrat" einzugeben, wählst Du in der Entwicklungsumgebung

den Menüpunkt [Bearbeiten | Neue SUB | Name: "Quadrat"]

In dem sich jetzt öffnenden Fenster gibst Du die folgenden Befehle ein:

```
SUB Quadrat (i%)
PRINT i% ^ 2
END SUB
```

Startest Du nun das fertige Programm mit F5 oder [Ausführen | Start], so erscheint auf dem Bildschirm eine Liste der Quadratzahlen von $1^2 = 1$ bis $20^2 = 400$. Wie Du siehst, ist die als Parameter übergebene Variable i% auch in der Subroutine zugreifbar. Die anderen im Hauptprogramm verwendeten Variablen sind dagegen in der Subroutine unbekannt.

Das Gesamtprogramm sieht dann so aus:

```
' *****
' SUBQ.BAS - Subroutine zur Quadratbildung
' =====
' (c) Thomas Antoni, 29.10.2003
' *****
DECLARE SUB Quadrat (i%) 'SUB deklarieren
'
CLS
FOR i% = 1 TO 20
    CALL Quadrat(i%) 'Subroutine aufrufen
NEXT
SLEEP
END
'
SUB Quadrat (i%) 'Subroutine dedefinieren
    PRINT i% ^ 2
END SUB 'Rueckkehr zum Hauptprogram
```

Der erste Befehl in dem obenstehenden Programm ist die so genannte "Deklaration" der Subroutine. Sie wird von QBasic automatisch eingefügt.

In der Entwicklungsumgebung wechselst Du zwischen dem Hauptprogramm und den SUBs über den Menüpunkt [Ansicht | SUBs...] oder die F2-Taste. Das ist eine sehr komfortable Sache, weil Du die einzelnen SUBs ganz bequem durchblättern und annavigieren kannst. Einmal ausgetestete Subroutinen kannst Du in Deinen verschiedensten Programmen immer wieder verwenden, und Du kannst auch auf eine Reihe von SUB- Sammlungen anderer Programmierer zurückgreifen. Einige gute SUB-Kollektionen für häufig vorkommende Programmierprobleme findest Du auf www.qbasic.de unter [Download | Sonstiges]. Scheue Dich nicht, die dort verfügbaren SUBs in eigene Programme einzubauen. Warum solltest Du das Rad zum x-ten Mal neu erfinden? Auch die Profi-Programmierer gehen so vor, um Zeit und Geld zu sparen.

Es lässt sich übrigens auch mehr als ein Übergabeparameter an eine Subroutine übergeben. Dann werden die einzelnen Parameter in der Klammer durch Kommas voneinander getrennt aufgeführt, z.B.

```
CALL Multiplikation (Ergebnis%, Faktor1%, Faktor2%)
```

Für die Namen von Subroutinen und Funktionen gelten dieselben Regeln wie für Variablenamen; siehe Kapitel 6.

15. Funktionen

Funktionen sind im Prinzip dasselbe wie Subroutinen. Sie können aber zusätzlich einen Zahlenwert oder einen String an das aufrufende Programm zurückliefern. Funktionen werden nicht durch CALL aufgerufen, sondern quasi anstelle einer Variablen hingeschrieben, z.B. rechts von einem Gleichheitszeichen in einer Wertzuweisung. Bei QBasic heißen die Funktionen FUNCTIONS.

Da die Funktion wie eine Variable benutzt wird, muss sie auch einen Datentyp haben, damit QBasic weiß, wie es den zurückgelieferten Wert interpretieren muss.

Im Folgenden findest ein Programmbeispiel mit einer FUNCTION "Quadrat%" vom Typ INTEGER, erkenntlich an dem Typkennzeichen "%". Es handelt sich dabei um eine Abwandlung des oben stehenden Programms SUBQ.BAS.

```
' *****
' FUNCQ.BAS - Funktion zur Quadratbildung
' =====
' (c) Thomas Antoni, 30.10.2003
' *****
DECLARE FUNCTION Quadrat% (i%) 'FUNCTION deklarieren
'
CLS
FOR i% = 1 TO 20
    PRINT Quadrat%(i%) 'FUNCTION aufrufen
NEXT
SLEEP
END
'
FUNCTION Quadrat% (i%) 'FUNCTION definieren
    Quadrat% = i% ^ 2 'Wertzuweisung an die Funktion
                    'Ruecksprung zum Hauptprogramm
END FUNCTION 'und i% uebergeben
```

In der QBasic-Entwicklungsumgebung "fühlt" sich eine Funktion genauso an wie eine Subroutine. Das Eingeben einer neuen Funktion erfolgt über [Bearbeiten | Neue FUNCTION...].

16. Grafik

Bisher haben wir nur den Textbildschirm 0 (SCREEN 0) kennengelernt, der bei QBasic voreingestellt ist und 25 bis 50 Textzeilen mit je max. 80 Zeichen unterstützt. Für Mathe- und Business- Programme ist das ganz OK. Wenn Deine Programme aber mehr Spaß vermitteln sollen und Du bunte Bilder und Spiele gestalten willst, dann kannst Du auch vielfältige Grafik- Effekte in QBasic programmieren.

Für Grafikanzeigen verwendest Du die Screenmodi (Bildschirmbetriebsarten) SCREEN 1 bis 13, die sich hinsichtlich der Auflösung und der Anzahl Farben unterscheiden. Mehr Infos findest Du in der QBasic-Hilfe unter [Index | SCREEN].

Am beliebtesten sind die Screenmodi 9, 12 und 15. Wir wollen hier den Bildschirmmodus SCREEN 9 verwenden, der eine VGA/EGA- Auflösung von 640 x 350 Pixeln bietet. Jeder Bildschirmpunkt ist dabei über seine x- und y- Koordinaten ansprechbar. Die linke obere Ecke hat die Koordinaten (x=0, y=0) und die rechte untere Ecke (x=639, y=199). Du

siehst also, die y-Achse zählt von oben nach unten und nicht wie aus der Schule gewohnt von unten nach oben.

Das folgende kleine Programm GRAFIK.BAS zeichnet ein Mondgesicht auf den Bildschirm und demonstriert dabei einige wichtige Grafikbefehle von QBasic:

```
' *****
' GRAFIK.BAS = Kleine Demo der wichtigsten Grafikbefehle
' =====
' Dieses QBasic-Programm zeigt ein Mondgesicht auf dem
' Bildschirm an und demonstriert damit die folgenden
' Grafikbefehle:
' - CIRCLE = Kreis zeichnen
' - PAINT  = Fläche ausfüllen
' - LINE   = Linie zeichnen
' - PSET   = Bildpunkt zeichnen
'
' (c) Thomas Antoni, 31.10.2003 - 8.1.2004
' *****
SCREEN 9 'VGA-Grafikbildschirm mit 640x350 Pixeln
COLOR 1 'Blauer Bildschirmhintergrund
CLS      'Bildschirm ruecksetzen und blau einfaerben
CIRCLE (320, 175), 150, 14 'Kopf malen
'Kreis um Mittelpunkt M=(320|175) mit Radius r=150 Pixel
'und der Randfarbe 14=gelb zeichnen
PAINT (320, 175), 4, 14 'Kopf rot fuellen
'Kreis ab Mittelpunkt mit der Farbe 4 (=rot) ausmalen
'bis die gelbe Randfarbe (14) erreicht wird
LINE (260, 130)-(290, 130), 14
'Linke Augenbraue als Linie von (260|130) nach (290|130)
'mit Linienfarbe gelb (14) zeichnen
LINE (350, 130)-(380, 130), 14
'rechte Augenbraue
LINE (320, 150)-(320, 200), 14
'Nase
LINE (270, 235)-(370, 235), 14
'Mund
PSET (275, 140), 14
'linkes Auge als Bildpunkt (275|140) mit der Farbe
'gelb (14) zeichnen
PSET (365, 140), 14
'rechtes Auge
SLEEP
END
```

Weil ich nun mal kein Künstler bin, ist das Gemälde ausgesprochen schlicht ausgefallen *lol*. Du wirst bestimmt coolere Dinge auf den Bildschirm zaubern.

Die einzelnen Grafikbefehle sind im Kommentar erläutert. Hier nochmal die Zusammenfassung:

1. **SCREEN n**
Bildschirmmodus n = 0 ... 13 aktivieren
2. **CIRCLE (x%, y%), r%, f%**
Kreis mit dem Mittelpunkt (x%|y%), dem Radius r% und der Randfarbe r% zeichnen (zu dem Farbcode f% siehe Kapitel 5)
3. **PAINT (x%, y%), f%, fr%**
Fläche ausgehend von dem Punkt (x%|y%) mit der Farbe f% ausfüllen, solange

bis die Randfarbe fr% erreicht wird

4. **LINE (x1%, y1%)-(x2%, y2%), f%**

Linie zeichnen zwischen den Punkten (x1%|y1%) und (x2%|y2%) in der Farbe f%

5. **PSET (x%, y%), f%**

Bildpunkt zeichnen an der Stelle (x%, y%) mit der Farbe f%

Die Kompletinfo zu diesen Befehlen findest Du wie immer in der QBasic- Onlinehilfe.

17. Sound

QBasic kennt 3 Befehle zur Erzeugung von Soundeffekten mit dem PC-Speaker:

- BEEP - erzeugt einen Piepston
- PLAY - spielt eine oder mehrere Noten der Tonleiter
- SOUND - erzeugt einen Ton wählbarer Frequenz und Länge

Zu BEEP brauchen wir nichts weiter zu sagen. Wir können uns also gleich den anderen beiden Befehlen zuwenden.

Der PLAY-Befehl

Wie der PLAY-Befehl funktioniert, demonstriert das folgende Beispielprogramm durch Abspielen von "Alle meine Entchen":

```
' *****
' ENTCHEN.BAS = "Alle meine Entchen" mit QBasic spielen
' =====
'
' (c) Thomas Antoni, 2.11.2003
' *****
PLAY "MFT16003L8cdefL4ggL8aaaaL2gL8aaaaL2gL8ffffL4eeL8ggggL4c"
SLEEP
END
```

Der PLAY-Befehl hat die Syntax `PLAY <Befehls-String$> .`

Der Befehls-String\$ kann die folgenden Bestandteile haben:

- M{F | B} - alle folgenden Noten im Vordergrund | Hintergrund abspielen.
Bei der Vordergrundbearbeitung hält das Programm an, bis der Play-Befehl abgearbeitet ist. Bei der Hintergrundbearbeitung wird während der Soundausgabe das nachfolgende Programm fortgesetzt
- {A|B|...|G|} - Note a, h, c, d, e, f oder g der Tonleiter in der aktuellen Oktave spielen
- O<n%> - aktuelle Oktave für die folgenden Noten festlegen (n=0...6)
- L<q%> - Länge der nachfolgenden Töne festlegen (q=1-64; Tonlänge = 1/q;
1 ist eine ganze Note; Vorbesetzung: q = 4 ==> 1/4 Note)
- P<q%> - Pausendauer zwischen den nachfolgenden Töne festlegen (q=1-64;
Pausendauer = 1/q; Vorbesetzung: q = 4 ==> 1/4 Note)
- T<q%> - Tempo der nachfolgenden Noten in Viertelnoten/min festlegen;
(q=32-255); Vorbesetzung: q= 128
- MS - alle nachfolgenden Noten in Staccato spielen (kurz und abgehackt,
nicht mit dem nächsten Ton verbunden)
- ML - alle nachfolgenden Noten in Legato spielen (lang und getragen,
mit der nächsten Note verbunden)
- MN - alle nachfolgenden Noten wieder normal spielen (nicht Staccato
oder Legato)

Der folgende Befehl zeigt einige der aufgelisteten Möglichkeiten und lässt einen Big-Ben-Schlag ertönen:

```

- PLAY          "MB ML T160 O1 L2 gdec P2 fedc"
  im Hinter- | | | | | | | |
  grund  --+ | | | | | | | |
  Legato -----+ | | | | | | | |
  Tempo 160 -----+ | | | | | | | |
  1.Oktave -----+ | | | | | | | |
                                +--- letzte 4 Noten
                                +----- 1/2 Notenlänge Pause
                                +----- erste 4 Noten
                                +----- Notenlänge: 1/2 Note

```

Der SOUND-Befehl

Der SOUND-Befehl hat die folgende Syntax

```
SOUND <FrequenzInHerz%>, <DauerInSystemtakten%>
```

Es handelt sich dabei um sehr einfache Art der Soundausgabe: Es wird ein Ton mit der angegebenen Frequenz der Dauer in Systemtakten von je 0,056 sec (= 56 ms) ausgegeben.

Beispiel 1 - 2000Hz-Ton für 336ms (6*56ms) spielen:

```
SOUND 2000, 6
```

Beispiel 2 - Motorengeräusch ausgeben bis eine beliebige Taste gedrückt wird:

```

DO
SOUND 192, 0.5 'Motorengeräusch
SOUND 188, 0.5
LOOP WHILE INKEY$ = ""

```

Beispiel 3 - Sirene

```

' *****
' SIRENE.BAS = Demo des QBasic-Befehls SOUND
' =====
' Dieses Programm erzeugt einen Sirenenklang
' (c) Winfried Furrer, 2000
' *****

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

```

Dies Beispiel zeigt, wie man den SOUND-Befehl in einer Schleife verwendet und dabei die Frequenz des Tons verändert. Mit dieser Methode lassen sich nette Soundeffekte erzielen.

Außerdem demonstriert das Programm noch eine Spezialität des FOR...NEXT-Befehls: STEP -4 bewirkt, dass das Programm die Schleife bei jedem Durchlauf mit einem um 4 reduzierten Wert von zaehler2 durchläuft.

18. Felder

Ein Feld (engl. "Array") ist eine Anordnung mehrerer Variablen gleichen Typs, die in einen zusammenhängenden Speicherbereich abgelegt werden und sich bequem in Schleifen schreiben und lesen lassen. Man kann sich ein Feld wie einen Schubladenschrank vorstellen, der in jeder einzelnen Schublade einen Platz für eine Variable bietet, die auch "Feldelement" genannt wird. Vorne auf den Schubladen stehen Nummern. Jedem Feldelement ist eine solche "Feldelement- Nummer", der so genannte "Index", zugeordnet.

Bevor man ein Feld benutzen kann, muss man es über den DIM-Befehl "dimensionieren". Mit

```
DIM Alter%(100)
```

wird z.B. ein Feld namens "Alter%" vom Typ INTEGER mit 101 Feldelementen (Indices 0 ... 100) dimensioniert, das etwa das Lebensalter von 101 Personen enthalten kann. Die Indices beginnen also normalerweise mit "0" und nicht mit "1".

Felder sind hervorragend geeignet zum Speichern und Bearbeiten von Datenbanken, Messdatenreihen, Tabellen, Vektoren, Matritzen und ähnlichen gegliederten Datensammlungen. Der Feldindex steht in einer auf den Feldnamen folgenden Klammer.

Einzelne Feldelemente lassen sich über ihren Index bequem und effizient auch in Schleifen bearbeiten, meist in einer FOR...NEXT-Schleife, weil die Anzahl der Bearbeitungsschritte ja oft bereits vorher festliegt. Die folgende Befehlssequenz zeigt beispielsweise alle 101 im Feld Alter%(100) gespeicherten Feldelemente Alter%(0) bis Alter%(100) auf dem Bildschirm an:

```
FOR i% = 0 TO 100
    PRINT Alter%(i%)
NEXT i%
```

Das folgende Programm FELD.BAS demonstriert den Umgang mit Feldern:

```
' *****
' FELD.BAS - Felder bearbeiten in QBasic
' =====
' Dies Programm demonstriert die Bearbeitung
' von Feldern in QBasic. Der Anwender wird
' aufgefordert, 3 Zahlen einzugeben. Diese werden
' im Feld z%(2) abgelegt. Anschliessend
' berechnet das Programm die Quadratwerte der
' Zahlen und zeigt sie an
'
' (c) Thomas Antoni, 4.11.2003 - 8.1.2004
' *****
DIM z%(2) 'Feld mit den 3 Feldelementen z%(0)...z%(2)
          'deklarieren bzw. "dimensionieren"
CLS
PRINT "Gib drei Zahlen von 1 bis 100 ein"
'
' *** Zahlen eingeben *****
FOR i% = 0 TO 2 'Schleife ueber alle Feldelemente
    PRINT "Gib die "; i%; ". Zahl ein: ";
    INPUT z%(i%)
NEXT
'
' *** Quadratzahlen berechnen und anzeigen *****
FOR i% = 0 TO 2
```



```

PRINT "z("; i%; ") ^2 = "; z%(i%) ^ 2
NEXT
SLEEP
END

```

Auch mehrdimensionale Felder sind möglich. Mit

```
DIM C#(9,4)
```

wird z.B. ein zweidimensionales Feld vom Typ DOUBLE (doppelt lange Gleitpunktzahl) deklariert (man sagt auch "dimensioniert"), das 10 Blöcke (Index 0...9) mit je 5 Feldelementen enthält (von 0 bis 4). Der Zugriff auf das 3. Element des 4. Blockes erfolgt so:

```
C# (2,3) = 86 .
```

Dieses zweidimensionale Feld kannst Du Dir quasi als eine Tabelle mit 10 Zeilen à 5 Spalten vorstellen. Mit dem obigen Befehl trägst Du den Wert 86 in Zeile 3, Spalte 4 ein.

19. Arbeiten mit Dateien

Mit QBasic ist es kein Problem, Dateien anzulegen, auszulesen und zu schreiben. Das folgende kleine Programmchen schreibt einen vom Anwender einzugebenden Text in die Datei C:\TMP.TXT und liest ihn wieder aus:

```

' *****
' DATEI.BAS = Dateibearbeitung mit QBasic
' =====
' Dieses Programm demonstriert, wie man einen
' kleinen Text in die Datei c:\tmp.txt
' hineinschreibt und wieder ausliest. Diese
' Datei muss nach Beendigung des Programms
' von Hand geloescht werden.
'
' (c) Thomas Antoni, 4.11.2003
' *****
'
' -- Datei schreiben --
CLS
OPEN "c:\tmp.txt" FOR OUTPUT AS #1
INPUT "Gib Deinen Namen ein "; name$
WRITE #1, name$
CLOSE #1
'
' --- Datei lesen ---
OPEN "c:\tmp.txt" FOR INPUT AS #1
INPUT #1, t$
CLOSE #1
PRINT
PRINT "Du hast Folgendes eingegeben: "; t$
SLEEP
END

```

Hierbei handelt es sich um eine "Sequentielle Datei". Mit jedem WRITE- Befehl wird eine neue Zeile angelegt, die mit einem INPUT-Befehl wieder auslesbar ist.

Die einzelnen Befehle wollen wir hier nun kurz erläutern:

1. **OPEN <Dateiname\$> FOR OUTPUT AS #<Dateinummer>**
Damit wird die Datei zum Schreiben geöffnet und ihr für die weitere Bearbeitung eine Dateinummer zwischen 1 und 255 zugewiesen. ACHTUNG: Existiert die Datei bereits, dann wird ihr Inhalt beim Öffnen komplett gelöscht!
2. **WRITE #<Dateinummer> , <Variable-1> [, Variable-2, ...]**
Dieser Befehl schreibt den Inhalt einer oder mehrerer Variablen in die Datei, nach der letzten Variablen gefolgt von einem Zeilenvorschub. Text wird immer in Anführungszeichen abgespeichert und numerische Werte in ASCII- Text- Form. In der Datei erscheinen die Variablen dann ebenfalls durch Kommas getrennt.
3. **CLOSE [#<Dateinummer>]**
Wenn Du das Schreiben oder Lesen beenden willst, musst Du die Datei jeweils ordnungsgemäß schließen. CLOSE ohne Dateinummer schließt alle offenen Dateien.
4. **OPEN <Dateiname\$> FOR INPUT AS #<Dateinummer>**
Dieser Befehl öffnet die Datei zum Lesen.
5. **INPUT #<Dateinummer>, <Variable-1> [, Variable-2, ...]**
Aus der Datei wird eine Zeile gelesen und die dort durch Kommas getrennten Bestandteile in die einzelnen Variablen eingetragen. Anmerkung: Zum Schreiben und Lesen von Text, der Kommas enthalten kann, verwendet man PRINT und LINE INPUT statt WRITE und INPUT; siehe QBasic- Kochbuch.

Das Lesen und Schreiben der Zeilen ist nur nacheinander ("sequentiell") möglich. Eine Zeile mitten in der Datei lässt sich nicht direkt auslesen. Das ist nur mit einem anderen Dateityp, der "Direktzugriffsdatei" (Random-Datei) möglich. Auf die Details und auf die verschiedenen Dateitypen wollen wir hier nicht weiter eingehen. Die Komplett- Info zur Dateibearbeitung mit QBasic erhältst Du in meinem QBasic- Kochbuch, das auf www.qbasic.de zum Herunterladen bereitsteht.

20. Mehrfachverzweigungen mit SELECT-CASE

Häufig ist bei einer Verzweigung der Wert einer einzigen Variablen dafür verantwortlich, welcher Zweig genommen werden soll. Anfänger lösen solche Programmierprobleme mit einer Folge von IF-Abfragen, etwa so:

```
IF Note% = 1 THEN ...
IF Note% = 2 THEN ...
IF Note% = 3 THEN ...
IF Note% = 4 THEN ...
```

Wesentlich effizienter und übersichtlicher ist aber die Verwendung des SELECT CASE Befehls. SELECT CASE heißt zu deutsch "wähle einen Fall aus". Es wird abhängig vom Wert der Selektor- Variablen eine Fallunterscheidung gemacht.

```
' *****
' CASE-1.BAS - Beispiel 1 fuer den SELECT CASE Befehl
' =====
' Es wird eine Schulnote als Zahl 1...6 abgefragt
' und in eine Textnote "sehr gut...ungenuegend"
' umgewandelt und angezeigt.
'
' von SM, 22.10.03
' *****
CLS
INPUT "Gib deine Note ein (1...6): ", Note%
SELECT CASE Note%
    CASE 1: PRINT "sehr gut"
    CASE 2: PRINT "gut"
    CASE 3: PRINT "befriedigend"
    CASE 4: PRINT "ausreichend"
    CASE 5: PRINT "mangelhaft"
    CASE 6
        PRINT "ungenuegend"
        PRINT "Die Versetzung ist gefaehrdet!!!"
    CASE ELSE: PRINT "Diese Note gibt es nicht."
END SELECT
```

Der Aufbau ist leicht zu verstehen. Hinter dem Befehl SELECT CASE steht der Name der Selektor- Variablen, die abgefragt wird, hier "Note%". Das Programm verzweigt dann zu demjenigen CASE-Zweig, bei dem hinter dem Schlüsselwort CASE der aktuelle Wert der Selektorvariablen aufgeführt ist. Wenn bei dem obenstehenden Programm beispielsweise die Variable Note% den Wert 4 hat, so springt das Programm zu CASE 4 und führt die dahinter bis zum nächsten CASE stehenden Befehle aus, in diesem Falle PRINT "ausreichend". Der ganze Block muß mit END SELECT abgeschlossen werden.

Sollte die Selektor-Variable einen Wert haben, für den kein CASE vorgesehen wurde, dann springt das Programm zum CASE-ELSE-Block. Wenn CASE ELSE fehlt, dann wird der gesamte SELECT-CASE-Block einfach übersprungen.

Wenn ein CASE-Block nur aus einem Befehl besteht, dann kannst Du ihn zusammen mit dem Befehl "CASE..." in eine Zeile schreiben - durch einen Doppelpunkt voeneinander getrennt. Besteht ein CASE-Block aus mehreren Befehlen, dann werden diese ab der Zeile unter dem CASE n hingeschrieben, wie Du bei CASE 6 sehen kannst.

Hier noch ein paar weitere Möglichkeiten der CASE-Anweisung:

```
CASE "j"
CASE "j", "J"
```

Wenn String-Variablen auf einen Text abgefragt werden, dann denke bitte an die Anführungsstriche. Der erste CASE Block wird durchlaufen, wenn in der Selektor-Variablen ein kleines j steht, der zweite CASE Block sowohl beim kleinen j als auch beim großen J. Mit dieser Methode kann man sehr elegante Tastenmenüs realisieren

```
CASE IS < 10
```

Bei Vergleichen muß das Schlüsselwort IS eingefügt werden. Diese Bedingung ist erfüllt, wenn die Selektor-Variable kleiner als 10 ist.

```
CASE 5 TO 10
```

Hier muß die Selektor-Variable einen Wert zwischen 5 und 10 besitzen.

Das folgende Programm zeigt einige der genannten Möglichkeiten:

```
' *****
' CASE-2.BAS - Beispiel 2 fuer den SELECT CASE Befehl
' =====
' Dies Beispiel zeigt einige spezielle Moeglichkeiten
' des SELECT CASE Befehls
'
' von Thomas Antoni, 6.1.2004 - 1.2.2004
' *****
CLS 'Bildschirm loeschen
INPUT "Gib eine Zahl zwischen 0 und 10 ein"; z
SELECT CASE z
  CASE 0: PRINT "Eingabewert ist 0"
  CASE 1 TO 8
    PRINT "Eingabewert liegt zwischen 1 und 8"
  CASE 9, 10: PRINT "Eingabewert ist 9 oder 10"
  CASE IS < 0
    PRINT "Falsche Eingabe. Eingabewert ist negativ!"
  CASE ELSE
    PRINT "Falsche Eingabe. Eingabewert zu gross!"
END SELECT
```

Und nun noch ein Beipielpogramm, das zeigt, wie Du mit CASE SELECT ein sehr schönes Tasten- basiertes Menü realisieren kannst. Als Selektor- Variable dient dabei das Textzeichen bzw. der ASCII-Code der betätigten Menüauswahl- Taste.

```
' *****
' CASE-3.BAS - Beispiel 3 fuer den SELECT CASE Befehl
' =====
' Dies Beispiel zeigt, wie man mit Hilfe des SELECT
' CASE Befehls ein Tastenmenue aufbauen kann.
'
' (c) Thomas Antoni, 9.1.2004 - 1.2.2004
' *****
DO
CLS
PRINT "Was willst Du tun?"
PRINT " (a)   = Schlafen"
PRINT " (b)   = Essen"
PRINT " (c)   = Fernsehen"
PRINT " (Esc) = Abbruch"
PRINT "Waehle die gewuenschte Aktion (Taste a, b, c oder Esc)"
DO: taste$ = INKEY$: LOOP WHILE taste$ = ""
'Warten auf die Betaetigung einer beliebigen Taste
```

```

PRINT
'
SELECT CASE taste$
  CASE "a": PRINT "Du hast Schlafen gewaehlt"
  CASE "b": PRINT "Du hast Essen gewaehlt"
  CASE "c": PRINT "Du hast Fernsehen gewaehlt"
  CASE CHR$(27) 'Esc betaetigt -> Programm beenden
    PRINT " ... und Tschuess!"
    SLEEP 2      '2 sec warten
    END          'Programm beenden bei Esc-Taste
  CASE ELSE
    PRINT "Fehler! Nur die Tasten a, b, c und Esc sind erlaubt!"
    PRINT "Wiederhole bitte die Eingabe"
END SELECT
PRINT
PRINT "... zurueck zum Hauptmenue mit beliebiger Taste";
DO: LOOP WHILE INKEY$ = ""
LOOP

```

Anstelle der Anzeige des gewählten Menüpunktes, z.B. mit PRINT "Du hast Essen gewählt", würde man normalerweise natürlich die zum Menüpunkt gehörenden Befehle oder den Aufruf einer entsprechenden Subroutine hinschreiben.

Dieses Menü ist wesentlich eleganter programmiert als die meisten Menüs, die ich in Anfänger- Programmen sonst so finde. Es hat die folgenden Vorteile:

- Durch die Verwendung von INKEY\$ statt INPUT ist nach Eintippen des Buchstabens keine zusätzliche Betätigung der Eingabetaste erforderlich. Du kannst statt der Buchstaben natürlich auch Ziffern verwenden. Das ist oft noch ergonomischer.
- Das ganze Menü steht in einer DO...LOOP- Schleife. Daher landet der Anwender nach Abarbeitung eines Menüpunktes wieder in der Menüauswahl und kann einen anderen Punkt wählen, ohne das Programm dafür verlassen und neustarten zu müssen.
- Eingabefehler werden abgefangen, und bei Betätigung einer falschen Taste erfolgt eine Fehlermeldung und ein Rücksprung zur Menüauswahl.
- Es wird auch ein Menüpunkt für den Programm-Abbruch per Esc-Taste angeboten. Viele Anfänger- Programme kranken leider daran, dass man sie nicht in jeder Situation mit Esc abbrechen kann. Das verärgert den Anwender und nimmt ihm die Lust, das Programm weiterzuverwenden.

Ich will Dir noch einen Trick verraten, der den meisten Anfängern völlig unbekannt ist: Statt der oberen DO...LOOP- Schleife zum Warten auf die Betätigung einer Menü-Auswahl taste kannst Du viel kürzer schreiben

```
taste$ = INPUT$(1) 'warten auf die Betaetigung einer beliebigen Taste
```

Dadurch wird ein Zeichen von der Tastatur in die Variable taste\$ eingelesen und das Programm hält dafür automatisch an.

21. EXE-Dateien erstellen

Wenn Du aus deiner .BAS-Datei ein selbständig ablauffähiges EXE-Programm erzeugen willst, benötigst Du einen QuickBasic-Compiler, am besten die mit einer deutschen Bedienungsoberfläche verfügbare Version QuickBasic 4.5, die 100% kompatibel zu QBasic ist.

Lade Dir QuickBasic 4.5 Compiler z.B. von www.qbasic.de herunter [Rubrik "QBasic | Download | Compiler"].

Gehe wie folgt vor, um ein BAS-Programm in ein lauffähiges EXE-Programm umzuwandeln :

- Starte den QuickBasic-Compiler QB.EXE
- Lade das BAS-Programm mit [Datei | Programm laden...]
- Kompiliere Dein BAS-Programm mit [Ausführen | EXE-Datei erstellen... | (.) selbständige EXE-Datei | EXE erstellen und beenden]
- Die .EXE-Datei wird jetzt erzeugt und im QB 4.5 Programmverzeichnis abgelegt
- Die gleichnamige .OBJ-Datei kann ohne Bedenken gelöscht werden

Eventuell musst Du vorher in der QB 4.5-Entwicklungsumgebung die Pfadnamen für "Include- und Bibliotheksdateien" im Menüpunkt [Optionen | Suchpfade festlegen] richtig angeben. Du kannst fürs Erste dort überall denjenigen Pfadnamen eingeben, unter dem QB.EXE selber hinterlegt ist.

Aufsteiger von QBasic sollten die folgenden Hinweise beachten, um die häufigsten Einsteigerfehler zu vermeiden:

Bei Programmen, die den CALL ABSOLUTE Befehle erhalten (z.B. einige Mausroutinen), muss die QuickLibrary QB.QLB mit eingebunden werden. Starte dazu QuickBasic mit einem der folgenden Aufrufe:

- QB /L
- QB /L meinprog.bas oder
- QB.exe /L qb.qlb /run meinprog.bas

Eventuell kannst Du das entsprechende Kommando mit dem Windows-Editor in eine so genannte Batchdatei eintragen, die Du statt QB.EXE startest. Diese Batchdatei ist eine reine Textdatei mit der Erweiterung .BAT. Nenne sie z.B. "qb45.bat".

22. Tipps und Tricks

Es gibt einige Tipps und Tricks die Dir das Programmieren in QBasic erleichtern. Eine kleine Auswahl haben wir hier aufgelistet. Unzählige weitere nützliche Tipps bietet die QB-MonsterFAQ auf www.qbasic.de .

QBasic mit 50 statt 25 Zeilen starten

Starte Quick Basic immer mit dem Parameter /h, um die Entwicklungsumgebung in einem doppelt großen Fenster zu genießen. Du kannst das entweder im Windows - Eigenschaftendialog von QBasic.exe eintragen oder Du erstellst Dir mit dem Windows-Editor eine Batch-Datei namens QBasic.bat folgenden Inhalts:

```
@echo off
cls
```

```
qbasic.exe /h  
cls
```

Kopiere diese Datei in das QBASIC\ -Verzeichnis und starte künftig diese Batchdatei statt QBasic.exe.

Auch Deine Programme selber können im Textmodus (SCREEN 0) ein 50 Zeilen großes Fenster nutzen, indem Du vorne im Programm Folgendes hinschreibst:

```
WIDTH 80, 50 'VGA-Auflösung mit 80 Spalten und 50 Zeilen  
COLOR 0, 7  'Schwarze Schrift auf hellgrauem Grund  
CLS         'Bildschirm löschen
```

Dabei bewirkt der COLOR-Befehl ein mehr Windows-likes Aussehen des Anzeigebildschirms.

Dein Programm ist abgestürzt? Kein Problem!

Oft kommt es vor, dass sich ein Programm "festfährt" und auf Grund eines Programmierfehlers in eine Unendlichschleife eintaucht. Es gibt 3 Möglichkeiten das Programm trotzdem noch zu beenden:

1. Du drückst [STRG + PAUSE], um zur QBasic-Entwicklungsumgebung zurückzukehren, ohne Informationen zu verlieren.
2. Wenn das Programm immer bei einer INPUT oder LINE INPUT Abfrage stehen bleibt, kannst Du es mit [STRG + C] abbrechen. Informationen gehen nicht verloren, da man zu QB zurückkehrt
3. Falls man sich im Vollbildmodus befindet, kann man mit [ALT]+[TAB] zum Windows- Desktop zurückkehren und das Programm auf Betriebssystemebene beenden. Nicht gespeicherte Veränderungen im Quellspracheprogramm gehen dabei allerdings verloren.

QB-Programme mit MS Word ausdrucken

QBasic-Quellspracheprogramme haben die Dateierweiterung "BAS" und liegen als "normale" Textdateien im DOS-ASCII-Code vor. Windows verwendet dagegen den so genannten ANSI- Zeichencode. Daher werden die Umlaute und einige Sonderzeichen Deiner BAS- Dateien von Windows- Textbearbeitungs- Programmen nicht korrekt angezeigt und ausgedruckt.

Alte MS-Word-Versionen konvertieren Textdateien, die im MS-DOS -ASCII- Zeichencode vorliegen, beim Öffnen automatisch in den Windows-ANSI-Code. Bei Word für Windows ab Version 97 ist dies leider nicht mehr der Fall. Daher werden die Umlaute und viele Sonderzeichen nicht richtig dargestellt und lassen sich nicht korrekt ausdrucken. Dies Problem ist jedoch leicht zu umgehen, wofür es zwei Lösungen gibt:

Lösung 1

Ändere die Dateierweiterung "BAS" in "ASC" (für "ASCII"). Dann konvertiert Word die Datei beim Öffnen automatisch in das Windows ANSI-Format.

Lösung 2

Der Konverter ist bereits installiert, Du musst ihn nur in Word unter [Extras | Optionen| Allgemein] mit dem Schalter "Konvertierung beim Öffnen bestätigen" aktivieren. Danach fragt Dich Word künftig beim Öffnen einer BAS oder TXT-Datei immer, in welchem Format genau diese vorliegt. "Nur Text", entspricht ANSI-Text, "MS-DOS-Text" entspricht dem bei BAS-Dateien vorliegenden ASCII-Text usw.

Jetzt kannst Du Deine BAS- Dateien korrekt ausdrucken. Du darfst sie natürlich nicht in Word abspeichern, wenn Du das ursprüngliche Format erhalten willst. Am Besten, Du erstellst für das Ausdrucken sicherheitshalber eine Kopie Deiner BAS-Datei.

23. Wie steige ich noch tiefer in QBasic ein?

Du hast jetzt einige der wichtigsten Aspekte von QBasic kennengelernt. Wir hoffen, es hat Dir Spaß und Appetit auf mehr gemacht. Zum Weiterlernen kann ich Dir die folgenden drei Tutorials empfehlen, die Du auf www.qbasic.de unter [QBasic | Tutorials] herunterladen kannst:

- **Adok's Way to QBasic** - Hervorragender großer deutscher QBasic-Kurs von Claus-Dieter Volko, Wien. Mit 51 Beispielprogrammen, 68 Seiten.
- **"Das QBasic-Kochbuch"** - eine Bauanleitung für QBasic-Programme für Fortgeschrittene mit kompletter QBasic-Befehlsreferenz, gegliedert nach Anwendungsgebieten, viele Beispiele, ca. 36 Seiten
- **"QB-MonsterFAQ"** - eine gigantische Sammlung mit über 500 der am häufigsten zu QBasic gestellten Fragen und deren Antworten. Mit einer sehr übersichtlichen Navigation. Da bleibt fast keine Deiner Fragen unbeantwortet. Ausgedruckt wäre das ein Buch mit weit über 1200 Seiten!

24. Liste der Beispielprogramme

- ERSTPROG.BAS - Erstes kleines Programm
- COLOR-1.BAS - Einsatz von Farben im QBasic-Textmodus
- INPUTIF.BAS - Demo für die Befehle INPUT und IF...THEN
- IFTHEN.BAS - Demo für IF...THEN-Abfragen
- TIME.BAS - Anzeige der Uhrzeit - eine kleine Digitaluhr
- LOOP.BAS - Schleifen - 5 Möglichkeiten, die Zahlen 0...100 anzuzeigen
- TIMER.BAS - Wartezeiten erzeugen
- MATHDEMO.BAS - Kleines Mathe-Demoprogramm
- WUERFEL.BAS - Elektronischer Wuerfel - Zufallszahlen erzeugen
- TEXTBEAR.BAS - Textbearbeitung - die wichtigsten Befehle
- SUBQ.BAS - Subroutine zur Quadratbildung
- FUNCQ.BAS - Funktion zur Quadratbildung
- GRAFIK.BAS - Die wichtigsten Grafikbefehle - eine kleine Demo
- ENTCHEN.BAS - "Alle meine Entchen" mit dem PLAY-Befehl spielen
- SIRENE.BAS - Sirenengeheul erzeugen mit dem SOUND-Befehl
- FELD.BAS - Felder bearbeiten in QBasic
- DATEI.BAS - Dateibearbeitung mit QBasic
- CASE-1.BAS - Einfaches Beispiel für Mehrfachverzweigungen mit SELECT CASE
- CASE-2.BAS - Spezialitäten des SELECT CASE Befehls
- CASE-3.BAS - Komfortables Tastenmenü mit SELECT CASE

© 9.2.2004 Thomas Antoni und Frank Neumann

www.qbasic.de und www.silizium-net.de - immer das Neueste zu QBasic und QuickBASIC