4 Einführung in die Programmiersprache Basic (QBasic)

Beim Kauf eines Rechners wird meist ein Basic-Interpreter mitgeliefert. Es gibt zahlreiche Dialekte von Basic. Wir werden uns hier auf die wichtigsten Anweisungen von ***QBasic*** beschränken.

4.1 Grundelemente der Sprache

4.1.1 Der Zeichensatz (Alphabet)

Externe Daten der Sprache werden aus folgenden ***Zeichen*** zusammengesetzt:

alphanumerische Zeichen	**a – z A – Z 0 – 9**	englisches Alphabet
Sonderzeichen	**! = + -* / ^ ( ) % # $! [ ] , . ' " ; : & ? < > \ @ _**	␣ . . . Leerzeichen

4.1.2 Konstanten

Zeichenkettenkonstanten (Strings)extern Folge von Zeichen, in " eingeschlossen
Beispiel "Hallo Welt!", "" (leere Zeichenkettenkonstante)
intern ASCII-Code

Numerische Konstanten

ganze Zahlen (Integer)
extern in üblicher Schreibweise mit oder ohne Vorzeichen
Beispiel 0 1 -123 123456
intern je nach Größe als ganze Zahl (2 Bytes) oder lange ganze Zahl (4 Bytes; das letzte Beispiel)

Festkommakonstanten
extern in üblicher Schreibweise mit oder ohne Vorzeichen und genau einem ***Dezimalpunkt***
Beispiel -123.45 +.0001 1. 3.141592 3.1415926
intern je nach Größe als rationale Zahl einfacher (4 Bytes) oder doppelter Genauigkeit (8 Bytes; das letzte Beispiel)

Gleitkommakonstanten

extern in halblogarithmischer Darstellung

$$\textit{Mantisse}\ \begin{matrix}E\\D\end{matrix}\ \textit{Exponent} = \textit{Mantisse} * 10^{\textit{Exponent}}$$

Mantisse . . . Festkommakonstante
Exponent . . . ganze Zahl

Beispiel -651.234 E-7 ≙ - 651,243*10^{-7} = 0,000 065 123 4

intern als rationale Zahl mit
E . . . einfache Genauigkeit (4 Bytes)
D . . . doppelte Genauigkeit (8 Bytes)

4.1.3 Variablen

Einfache Variablen

Variablen dienen als Platzhalter für Daten im Speicher. Ihre Grundbestandteile sind

Namen	Identifikation ⇒ symbolische Speicherplatzadresse
Typ	Interpretationsvorschrift der Bitfolge, Byteanzahl
Wert	Konstante, die durch die Bitfolge dargestellt wird

Variablenname

Ein ***Variablenname*** ist eine beliebige Folge alphanumerischer Zeichen, mit einem Buchstaben beginnend, mit einem Typenklassifikationszeichen endend, bestehend aus maximal 40 Zeichen, keine Basic-Schlüsselworte (**CLS**, **REM**, ...). Man wählt "sprechende Variablennamen", d.h. Namen, die sich selbst kommentieren.

Variablentyp

Dieser wird durch Typenklassifikationszeichen als letztes Zeichen des Variablennamen festgelegt:

Zeichenkette	**$**
ganze Zahl	**%**
lange ganze Zahl	**&**
rationale Zahl einfacher Genauigkeit	**!**
rationale Zahl doppelter Genauigkeit	**#**

Lässt man das Typklassifikationszeichen weg, so wird immer "**!**" als Typ angenommen.

Beispiel

Name$, **Alter%**, **Gehalt!**, **PI#**, **Einwohnerzahl&**; **Wert** ≙ **Wert!**

Variablenvereinbarung

Variablen werden durch ihr erstes Auftreten vereinbart, d.h. deklariert (Speicher wird bereitgestellt.) und definiert (Ein Wert wird zugewiesen, falls nicht anders angegeben, ist dieser **0**.).

strukturierte Variablen

Mehrere Daten werden durch einen Namen im Speicher identifiziert. Solche strukturierten Variablen haben zwar einen Namen und einen Typ, aber keinen Wert. Wir behandeln hier nur eine Sorte strukturierter Variablen.

Felder

Ein ***Feld*** ist die Zusammenfassung von Daten gleichen Typs (Vektoren, Matrizen). Die Daten bezeichnet man als ***Komponenten*** des Feldes.

Feldname: Variable (Integerliste)

Integerliste Folge von positiven ganzen Konstanten durch Kommas getrennt.

Der ***Feldtyp*** ergibt sich aus dem Typklassifikationszeichen des Feldnamen, d.h. alle Komponenten sind von diesem Typ.

Beispiel Vec%(11), **A#(1, 3)**

Feldvereinbarung **DIM** ***Feldname***
REDIM ***Feldname***

Felder werden durch den Befehl **DIM** bzw. **REDIM** vereinbart, d.h. deklariert (Speicher wird bereitgestellt.) und definiert (**0** wird jeder Feldkomponenten als Wert zugewiesen.).

Soll ein Feldname innerhalb eines Programms mehrfach vereinbart werden, so muss der Befehl **REDIM** verwendet werden. **DIM** gestattet nur die einmalige Feldvereinbarung ein und des selben Feldes.

Beispiel

Ein ***Vektor*** ist ein lineares Feld, dessen Komponenten vom gleichen Typ sind:

DIM Vec%(11)

Vektor , welcher 12 ganze Zahlen zusammenfasst.

Komponenten: **Vec% (0), Vec% (1), Vec% (2), . . . , Vec% (11*)***

Eine ***Matrix*** ist ein nicht lineares Feld, die Komponenten sind ebenfalls vom gleichen Typ:

REDIM A#(1, 3)

Matrix , mit 8 rationale Zahlen doppelter Länge.

Komponenten: **A# (0, 0), A# (0, 1), A# (0, 2), A# (0, 3)**
A# (1, 0), A# (1, 1), A# (1, 2), A# (1, 3)

Löschen aller vereinbarten Variablen

CLEAR Alle Variablen, auch Feldvariablen, werden aus dem Speicher gelöscht. Der Programmspeicher bleibt dabei unverändert.

Übersicht über die Datendarstellung in Basic und deren Grenzen:

Typ	Bezeichnung	MIN	MAX	Byteanzahl
	Variablennamen	1 Zeichen	40 Zeichen	Zeichenanzahl + 1
$	Zeichenfolgen	0 Zeichen	32 767 Zeichen	Zeichenanzahl + 1
%	ganze Zahlen	- 32 768	32 767	2
&	lange ganze Zahlen	- 2 147 483 648	2 147 483 647	4
!	rationale Zahlen einfacher Genauigkeit positive Zahlen negative Zahlen	 2.802597 E-45 -3.40 823 E+38	 3.402823 E+38 -2.802597 E-45	4
#	rationale Zahlen doppelter Genauigkeit positive Zahlen negative Zahlen	 -4.940656458412465D-324 -1.79769313486231D+308	 1.79769313486231D+308 -4.940656458412465D-324	8

4.1.4 Ausdrücke

Je nach dem Ergebnistyp werden drei Ausdrucksarten unterschieden:

numerische Ausdrücke	Ergebnis:	numerische Konstante
Zeichenkettenausdrücke		Zeichenkette
logische Ausdrücke (Bedingung)		0 ≙ falsch; sonst (1) ≙ wahr

Konstanten und Variablen zählen zu den Ausdrücken. Ansonsten werden Ausdrücke wie üblich mit Operanden (Teilausdrücke), Operatoren (Operationszeichen) und runden Klammern **( )** gebildet. Dabei sind die folgenden Operatoren, hier in ihrer Prioritätsreihenfolge angegeben, möglich:

Arithmetische Operatoren		**Beispiel**	**Bedeutung**
^	Potenz	$x^{\wedge} y$	x^y
-	Vorzeichen	$x--y$	$x-(-y)$
* /	Multiplikation, Division	$x*x/y$	$\frac{x^2}{y}$
\	ganzzahlige Division	5 \ 3	⇒ 1
MOD	Rest der ganzzahligen Division	5 MOD 3	⇒ 2
+ -	Addition, Subtraktion	$x+y-z$	$(x+y)-z$

Zeichenkettenoperatoren			
+	Verketten	"ANNE"+"MARIE"	⇒ "ANNEMARIE"

logische Operatoren			
Vergleiche			
< >	kleiner, größer,		
<= >=	kleiner gleich, größer gleich	$x <= y$	$x \le y$
= <>	gleich, ungleich	$x <> y$	$x \ne y$
Boolsche Operatoren			
NOT	Negation	NOT x	nicht
AND	Konjunktion	x AND y	und
OR	Alternative (Inklusiv-Oder)	x OR y	oder
XOR	Exklusiv-Oder	x XOR y	entweder, oder
EQV	Äquivalenz	x EQV y	genau dann wenn
IMP	Implikation	x IMP y	wenn, so

Hier die Wertetabelle der neu hinzugekommenen logischen Operatoren:

x	***y***	**NOT *x***	**AND**	**OR**	**XOR**	**EQV**	**IMP**
0	0	1	0	0	0	1	1
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	0
1	1	0	1	1	0	1	1

Beispiel

NOT (x < 2) AND (x < 5) ≙ (NOT (x < 2)) AND (x < 5) ≙ (x ≥ 2) AND (x < 5)
Diese Bedingung ist für x = 2, 3, 4 wahr (≙ 1).

NOT (D < 0) IMP (((D > 0) IMP (x1 <> x2)) OR ((D = 0) IMP (x1 = x2)))
Diese Aussage ist immer wahr für die Lösung quadratischer Gleichungen, wobei D die Diskriminante und x1, x2 die beiden Lösungen der Gleichung sein sollen.

4.1.5 Funktionen

Funktionsaufrufe sind ebenfalls Ausdrücke, können somit als Operanden in Ausdrücken Verwendung finden.

Funktionsaufruf ***Funktionsname*** **(** ***Ausdrucksliste*** **)**
Ausdrucksliste Liste von Ausdrücken, durch Komma getrennt.
Die Werte der Ausdrücke in der Ausdrucksliste werden der Funktion als Argumente übergeben, d.h. in die entsprechende Parametervariablen der Funktion kopiert. Danach wird die Funktion mit diesen Parameterwerten berechnet.

Hier soll tabellarisch auf einige wichtige ***Standardfunktionen*** eingegangen werden. Diese Funktionen, bis auf die letzte, haben nur einen Parameter, dieser wurde mit *x* bezeichnet.

Funktionsname	***Parametertyp***	***Ergebnistyp***	***Bedeutung***
SQR	rational, ≥ 0	rational	$\sqrt[2]{x}$
EXP	rational, ≤ 88.02969	rational	e^x
LOG	rational, > 0	rational	ln *x*
SIN	Winkel (Bogenmaβ)	rational	sin *x*
COS	Winkel (Bogenmaβ)	rational	cos *x*
TAN	Winkel (Bogenmaβ)	rational	tan *x*
ATN	rational	Winkel (Bogenmaβ)	arctan *x*
ABS	rational	rational	$\lvert x \rvert$
SNG	rational	ganze Zahl	Vorzeichen (+1, 0, -1) von *x*
FIX	rational	ganze Zahl	Abschneiden von Nachkommastellen
INT	rational	ganze Zahl	liefert die größte ganze Zahl, die kleiner oder gleich *x* ist

CINT	rational	ganze Zahl	Runden auf ganze Zahlen
CLNG	rational	lange ganze Zahl	Runden auf lange ganze Zahlen
LEN	Zeichenkette	ganze Zahl	Bestimmen der Länge der Zeichenkette
RND		rational im Intervall $[0,1)$	Erzeugen einer Zufallszahl

Beispiele

INT(RND * 6 +1)
Der Zufallsgenerator erzeugt eine zufällige Zahl im Intervall $[0,1)$. Die Multiplikation mit Sechs und die Addition mit Eins transformiert diese Zahl in das Intervall $[1,6)$. Anschieβend wird der ganze Teil der Zahl ermittelt, eine Zahl aus der Menge $\{1,2,3,4,5,6\}$.

CLNG(PI! * r! * r! * 100) / 100
Der Flächeninhalt des Kreises mit dem Radius **r!** wird als einfache rationale Zahl auf zwei Stellen hinter dem Komma berechnet.

4.2 Anweisungen in Basic

Der Aufbau eines Basic-Programms und die wichtigsten Basic-Anweisungen sollen jetzt erklärt werden, weitere befinden sich in der Hilfe (Taste ↑+ **F1**, zurück **Esc**).

***Basic-Programm*:**	**Folge von *Programmzeilen***
Programmzeile:	***Anweisung* : *Anweisung* :** ... ↵
Anweisung:	***Einfache Anweisungen***
	Schleifenanweisungen
	Auswahlanweisungen
	Unterprogrammanwesungen

Basic hat zwei Modi, zwischen diesen kann man mit der Taste **F6** umschalten:

- ***Direktmodus***: Jede Anweisung (***Befehl***) wird unabgespeichert sofort vom Interpreter übersetzt und ausgeführt, ***Taschenrechnermodus***.
- ***Programmiermodus***: Alle Anweisungen werden in einem Programm abgespeichert, dessen zeilenweise Abarbeitung erst durch ein entsprechendes Kommando (Taste ↑+ **F5**) gestartet wird.

4.2.1 Einfache Anweisungen

Kommentaranweisung **REM *Kommentar*** oder **' *Kommentar***
Kommentar beliebiger Text

Programmende **END**

Bildschirm löschen **CLS**

Wertzuweisung ***Variable = Ausdruck***
Der Wert des Ausdrucks wird berechnet und der Variablen zugewiesen, d.h. in deren Speicher entsprechend ihrem Typ geschrieben.
Beachte: = ist hier ein Zuweisungsoperator, kein Gleichheitszeichen!

Beispiele
a% = 3 : a% = a% + 1 Speicher: **a%** = 4

z% = INT(RND * 6) +1 **z%** $\in \{1,2,3,4,5,6\}$

A! = CLNG(PI! * r! * r! * 100) / 100
Der Flächeninhalt des Kreises mit dem Radius **r!** wird als einfache rationale Zahl auf zwei Stellen hinter dem Komma in **A!** abgespeichert.

Vorname$ = "Anne" : Nachname$ = "Kaiser"
Diese beiden Variablen haben Zeichenketten als Werte.

Eingabeanweisung **INPUT "*Bildschirmtext*" , *Variable***
Das Programm hält an, schreibt den Text an den Bildschirm und erwartet eine Tastatureingabe. Der über Tastatur eingegebene Wert wird unter der angegebenen Variablen entsprechend deren Typ abgespeichert.

Beispiel **INPUT "Wie ist Dein Vorname? " , Vorname$**

Ausgabeanweisung **PRINT *Printliste***
Die ***Printliste*** ist eine durch Kommas oder/und Semikolon getrennte Liste von Ausdrücken. Die Werte der Ausdrücke werden berechnet und anschließend ausgegeben. Durch das Semikolon wird die Ausgabe hintereinander ausgeführt, das Komma erzeugt einen Tabulatorsprung zwischen den Werten.

Beispiel **INPUT "Geben Sie eine Zahl ein: ", z!** Eingabe: 5
PRINT z! ; "*" ; z! ; "=" ; z! * z! Ausgabe: 5 * 5 = 25

DmToEuro.bas
```
REM Euroumrechnung DM  ->  Euro
REM M. Meiler
REM 26.11.2001

CLS                                 'Bildschirm loeschen
DmToEuro! = .511292                  'Umrechnungsfaktor
```

```
REM DM-Eingabe
  INPUT "DM: ", Dm!

REM Euro-Berechnung
  Euro! = Dm! * DmToEuro!
  EuroRund! = CLNG(Euro! * 100) / 100            'Runden

REM Euro-Ausgabe
  PRINT "Eur: "; EuroRund!
END
```

In diesem Programm spiegelt sich sehr schön das ***EVA-Prinzip*** wieder. Nach der ***Eingabe*** der Information erfolgt die ***Verarbeitung*** dieser und schließlich die ***Ausgabe*** des Ergebnisses. Sie werden feststellen, dass Sie dieses Prinzip in den meisten Programmen wiederfinden.

Zufallsgenerator

Um zufällige Zahlen zu erzeugen, gibt es die spezielle Funktion **RND**. Diese Funktion benötigt keine Argumente und stellt eine Zufallszahl her. Die zufällig erzeugte Zahlenfolge wird bei jedem erneuten Start reproduziert. Um eine sich ständig ändernde Zahlenfolge zu bekommen, ist es notwendig, ***vor dem ersten Gebrauch*** der Funktion **RND,** und nur einmal am Anfang des Programms, den Befehl **RANDOMIZE TIMER** zu setzen. Dieser Befehl erzeugt in Abhängigkeit der aktuellen Uhrzeit eine sich somit ständig ändernde Startzahl für die Zahlenfolge.

RANDIMIZE TIMER

	Zufallszahlenbereich
z! = RND	$0 \le z! < 1$
z! = RND * 6	$0 \le z! < 6$
z! = RND * 6 + 1	$1 \le z! < 7$
z% = INT(RND * 6) + 1	$z\% \in \{1,2,3,4,5,6\}$

Wurf_1_6.bas

```
REM 1 - 6 - Wuerfel
REM M. Meiler
REM 10.12.2001

CLS                                  'Bildschirm loeschen
RANDOMIZE TIMER                'Zufallsgenerator starten
Wurf% = INT( RND * 6) + 1                       'wuerfeln
PRINT "Gewuerfelt wurde: ", Wurf%

END
```

4.2.2 Schleifenanweisungen

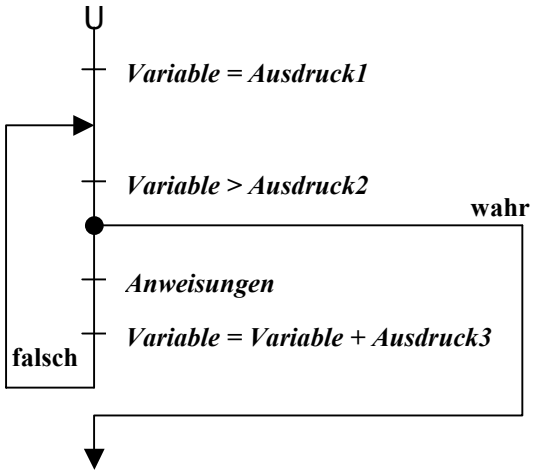
Um einen Würfel zu programmieren, der eine beliebige Anzahl von Würfen ausführt, ohne dass man das Programm ständig von neuem starten muss, benötigt man eine Möglichkeit, Befehle zu wiederholen.

FOR-Anweisung

Ist die Anzahl der Wiederholungen von Anfang an bekannt, so eignet sich die FOR-Anweisung.

FOR ***Variable*** **=** ***Ausdruck1*** **TO** ***Ausdruck2*** **STEP** ***Ausdruck3***
  Anweisungen
NEXT ***Variable***

Programmablaufplan (PAP)



Variable	Laufvariable, deren Wert verändert sich während des Schleifendurchlaufes.
Ausdruck1	Startwert der Laufvariable
TO ***Ausdruck2***	Abbruchwert der Laufvariable
STEP ***Ausdruck3***	Schrittweite. Sie ist der Wert, um den sich die Laufvariable während der einzelnen Schleifendurchläufe erhöht. Lässt man diese Angabe weg, so wird die Schrittweite auf **1** gesetzt.

Bei jedem Durchlauf wird der Wert der Laufvariablen, der mit dem Startwert beginnt, um die Schrittweite erhöht. Die Anweisungen werden so lange wiederholt, bis die Laufvariable den Endwert überschritten hat.

Beispiel

```
FOR i% = 1 to 10
  PRINT i% ; "*"; i% ; "=" ; i% * i%
NEXT i%
```

Protokolliert man die Speicherveränderungen und die Ausgaben, die durch diese Schleifenanweisung initiiert werden, so erhält man:

Programmprotokoll	i%	Ausgabe
	1	1 * 1 = 1
	2	2 * 2 = 4
	3	3 * 3 = 9
	4	4 * 4 = 16
	5	5 * 5 = 25
	6	6 * 6 = 36
	7	7 * 7 = 49
	8	8 * 8 = 64
	9	9 * 9 = 81
	10	10 * 10 = 100

ForSum.bas

```
REM Summe mit FOR-Schleife
REM M. Meiler
REM 10.12.2001

CLS

INPUT "n= ", n%

Summe% = 0
FOR i% = 1 TO n%
  Summe% = Summe% + i%
NEXT i%
PRINT "Die Summe der"; n%; "Zahlen ist"; Summe%; "."

END
```

LottoTip.bas

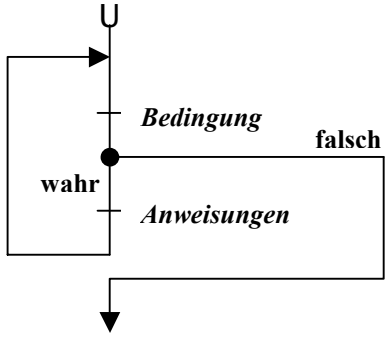
```
REM Tip: 6 aus 49
REM M. Meiler
REM 10.12.2001

CLS                                  'Bildschirm loeschen
RANDOMIZE TIMER                 'Zufallsgenerator starten

FOR i% = 1 TO 6               'sechs Schleifendurchlaeufe
  Wurf% = INT( RND * 49) + 1                    'wuerfeln
  PRINT "Wurf "; i% ; ": " ; Wurf%
NEXT i%

END
```

Das Programm hat noch den Nachteil, dass Zahlen mehrfach gewürfelt werden können. In einem Lottotipp müssen diese aber paarweise verschieden sein!

WHILE-Anweisung

Bei dieser Anweisung muss man ein Abbruchkriterium als Bedingung (logischer Ausdruck) angeben. Dabei kann es Anfängern leicht passieren, dass sie unendliche Schleifen programmieren.. Sollte das eingetroffen sei, so bricht man die Ausführung mit der Tastenkombination Strg + Pause. Anschließend ist dann aber das Setzen des Programms auf dessen Anfang (**Start/Neustart)** notwendig. Das Programm kann wieder normal gestartet werden.

WHILE ***Bedingung***
Anweisungen
WEND



Die Anweisungen werden ausgeführt, solange die Bedingung wahr ist.

Beispiel

```
i% = 1
WHILE i% <= 10
  PRINT i% ; "*"; i% ; "=" ; i% * i%
  i% = i% + 1
WEND
```

Das Programm erzeugt genau die gleichen Speicherplatzänderungen und Ausgaben wie das der FOR-Schleife. Deshalb wird hier auf ein Protokoll verzichtet.

WhileSum.bas

```
REM Summe mit WHILE-Schleife
REM M. Meiler
REM 10.12.2001

CLS

INPUT "n= ", n%

Summe% = 0: i% = 1
WHILE i% <= n%
  Summe% = Summe% + i%
  i% = i% + 1
WEND

PRINT "Die Summe der"; n%; "Zahlen ist"; Summe%; "."

END
```

Lotto1.bas

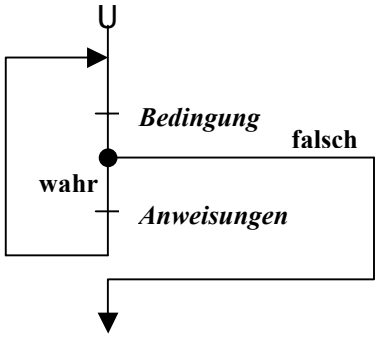
```
REM Lotto: 6 aus 49
REM M. Meiler
REM 10.12.2001

CLS                                  'Bildschirm loeschen
RANDOMIZE TIMER                'Zufallsgenerator starten

Weiter$ = "j"
                     'beliebig viele Schleifendurchlaeufe
WHILE Weiter$ = "j"
  Wurf% = INT( RND * 49) + 1                  'wuerfeln
  PRINT "Wurf :"; Wurf%
  INPUT "Weiter (j/n)? ", Weiter$
WEND

END
```

Jetzt kann man solange würfeln, bis alle 6 Zahlen verschieden sind. Leider wird dies nicht vom Programm gesteuert, sondern vom Nutzer. Im Abschnitt Komplexbeispiele wird auch dieses Problem noch behoben.

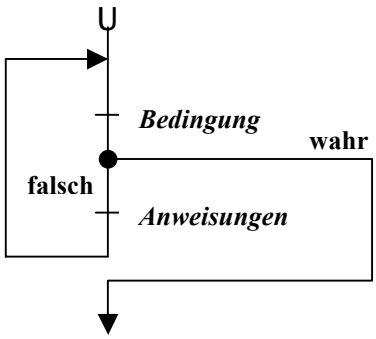
DO-Anweisungen

Es gibt noch eine weitere Gruppe von Schleifenanweisungen, die vier DO-Anweisungen. Diese sollen hier ergänzend kurz erwähnt werden.

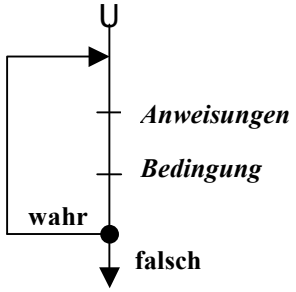
DO WHILE ***Bedingung***
Anweisungen
LOOP



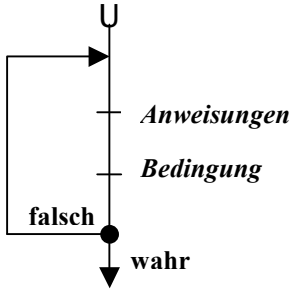
DO UNTIL ***Bedingung***
Anweisungen
LOOP



DO
Anweisungen
LOOP WHILE ***Bedingung***



DO
Anweisungen
LOOP UNTIL ***Bedingung***



Noch einmal Lotto! Worin besteht der Unterschied?

Lotto2.bas
```
REM Lotto: 6 aus 49
REM M. Meiler
REM 10.12.2001

CLS                                  'Bildschirm loeschen
RANDOMIZE TIMER                'Zufallsgenerator starten

DO                  'beliebig viele Schleifendurchlaeufe
  Wurf% = INT( RND * 49) + 1                   'wuerfeln
  PRINT "Wurf :"; Wurf%
  INPUT "Weiter (j/n)? ", Weiter$
LOOP WHILE Weiter$ = "j"

END
```

Die Bedingung **`Weiter$ = "j"`** wird erst nach dem ersten Wurf abgefragt. Damit braucht die Variable **`Weiter$`** keinen Anfangswert.

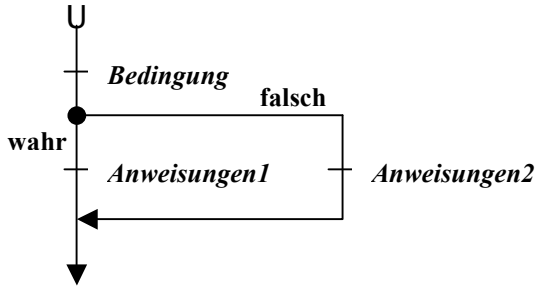
4.2.3 Auswahlanweisungen

Die Auswahlanweisungen gestatten Fallunterscheidungen, die es ermöglichen, nur einen Teil der Anweisungen abzuarbeiten.

IF-Anweisungen

Soll ein Programmteil nicht immer abgearbeitet werden, so reguliert man das über eine Bedingung.

IF ***Bedingung*** **THEN**
  Anweisungen1
ELSE
  Anweisungen2
END IF



Ist die ***Bedingung*** wahr, so werden die ***Anweisungen1*** ausgeführt,. sonst die ***Anweisungen2***. Der **ELSE**-Zweig muss nicht vorhanden sein.

Beispiel

```
INPUT "Zahl1 = ", Z1%
INPUT "Zahl2 = ", Z2%
PRINT "Addition ... a; Subtraktion ... s!"

INPUT "Waehle aus: ", Op$

IF Op$ = "a" THEN
  PRINT Z1% ; " + " ; Z2% ; " = " ; Z1% + Z2%
ELSE
  PRINT Z1% ; " - " ; Z2% ; " = " ; Z1% - Z2%
END IF
```

Es gibt noch eine weitere Varianten der IF-Anweisung aus den Basic-Anfängen, auf die hier nicht eingegangen werden soll.

Jetzt kann ein Eurorechner programmiert werden, der die Umrechnung in beiden Richtungen gestattet:

Euro.bas

```
REM Euro-Rechner Euro  <->  DM
REM M. Meiler
REM 10.10.2001

CLS                                  'Bildschirm loeschen
EuroToDm! = 1.95583                   'Umrechnungsfaktor
DmToEuro! = .511292                   'Umrechnungsfaktor

DO
  PRINT "Euro -> DM    ...  1"
  PRINT "DM   -> Euro  ...  2"
  INPUT "Waehle aus! ", Wahl%

  IF Wahl% = 1 THEN
    REM Euro-Eingabe
    INPUT "Eur: ", Euro!

    REM DM-Berechnung
    Dm! = Euro! * EuroToDm!
    DmRund! = CLNG(Dm! * 100) / 100             'Runden
```

```
      REM DM-Ausgabe
      PRINT "DM: "; DmRund!

    ELSE
      REM DM-Eingabe
      INPUT "DM: ", Dm!

      REM Euro-Berechnung
      Euro! = Dm! * DmToEuro!
      EuroRund! = CLNG(Euro! * 100) / 100          'Runden

      REM Euro-Ausgabe
      PRINT "Eur: "; EuroRund!

    END IF

    INPUT "Weiter (j/n)? ", Weiter$
  LOOP WHILE Weiter$ = "j"

  END
```

SELECT-CASE-Anweisung

Diese Anweisung gestattet Fallunterscheidungen mit mehr als zwei Fällen.

SELECT CASE ***Ausdruck***

CASE ***Ausdrucksliste1***
Anweisungen1
CASE ***Ausdrucksliste2***
Anweisungen2

. . .

CASE ***AusdrucklisteN***
AnweisungenN
CASE ELSE
Anweisungen

END SELECT

Ausdrucksliste	***Ausdruck , Ausdruck , . . .***
	Ausdruck **TO** ***Ausdruck***
	IS ***Vergleichsoperator Ausdruck***

Punkte.bas

```
REM Leistungsauswertung
REM M. Meiler
REM 10.12.2002
CLS

REM Auswertung Punktergebnis            'Ergebnis total
INPUT "Gesamtpunktzahl: ", Gesamt%
INPUT "Erreichte Punktzahl: ", Punkte%

PRINT "Du hast von insgesamt"; Gesamt%; "Punkten";

SELECT CASE Punkte%
  CASE IS > 1
    PRINT Punkte%; "Punkte ";
  CASE 1
    PRINT " einen Punkt ";
  CASE IS <= 0
    PRINT " keinen Punkt ";
END SELECT

PRINT "erreicht!"

REM Auswertung Note
Prozent% = Punkte% * 100 / Gesamt%       'Ergebnis in %
PRINT "Deine Note ist eine ";

SELECT CASE Prozent%
  CASE IS < 20
    PRINT "6!"
    PRINT "Schade, Du musst noch sehr viel üben!"
  CASE 20 TO 39
    PRINT "5!"
    PRINT "Üben, üben, üben!";
    PRINT "Es ist noch kein Meister vom Himmel";
    PRINT "gefallen!"
  CASE 40 TO 59
    PRINT "4!"
    PRINT "Das kann noch besser werden!"
  CASE 60 TO 79
    PRINT "3!"
    PRINT "Zufriedenstellend, es geht noch besser!"
  CASE 80 TO 94
    PRINT "2!"
    PRINT "Gut! Weiter so!"
  CASE IS >= 95
    PRINT "1!"
    PRINT "Prima, ich habe mich sehr gefreut! ";
    PRINT " Weiter so!"
END SELECT
END
```

4.2.4 Komplexbeispiele

Zunächst fassen wir die Summenbildungen s.o. in einem Programm zusammen. Beim Testen des Programms stellt man fest, das unter Verwendung der beiden Schleifenarten das Programm ab der natürlichen Zahl 256 einen Überlauf liefert. Die Gauß-Formel[1] bricht bereits bei der natürlichen Zahl 181 ab. Erklären Sie sich diese Tatsache anhand Ihrer Kenntnisse über die Datentypen in Basic! Wie könnte man das Programm verbessern? Bis zu welchen Zahlen läuft das geänderte Programm korrekt? Kann man ähnliche Erfolge auch durch geeignete Umformung der Gauß-Formel erreichen?

Sum.bas

```
REM Summe mit FOR-, WHILE- Schleife und nach Gauss
REM M. Meiler
REM 10.12.2001

CLS

DO                                ‘ DO_LOOP_WHILE-Schleife
  REM Meunue
  PRINT "Summe mit FOR-Schleife   ... 1"
  PRINT "Summe mit WHILE-Schleife ... 2"
  PRINT "Summe nach Gauss         ... 3"
  INPUT "Waehle aus ", Wahl%

                             ‘ IF_THEN_ELSE_END_IF-Auswahl
  IF Wahl% < 1 OR Wahl% > 3 THEN
    PRINT "fehlerhafte Eingabe!"
  ELSE
    REM Eingabe der natuerlichen Zahl
    INPUT "n= ", n%
  END IF


  REM Summe mit FOR-Schleife
  IF Wahl% = 1 THEN
    PRINT "FOR-Summe:                 ";
    Summe% = 0
    FOR i% = 1 TO n%                ‘ FOR_NEXT-Schleife
      Summe% = Summe% + i%
    NEXT i%
    PRINT "Die Summe der"; n%; "Zahlen ist";
    PRINT Summe%; "."
  END IF
```

[1] Carl-Friedrich Gauß (1777-1855) lieferte die Formel $\sum_{i=1}^{n} i = n*(n+1)/2$ als neunjähriger in der Anfangerschule dem erstaunten Lehrer Büttner, als er als Strafarbeit die Zahlen 1 bis 100 addieren sollte.

```
    REM Summe mit WHILE-Schleife
    IF Wahl% = 2 THEN
      PRINT "WHILE-Summe:              ";
      Summe% = 0: i% = 1
      WHILE i% <= n%                ' WHILE_WEND-Schleife
        Summe% = Summe% + i%
        i% = i% + 1
      WEND
      PRINT "Die Summe der"; n%; "Zahlen ist";
      PRINT Summe%; "."
    END IF


    REM Summe nach Gauss
    IF Wahl% = 3 THEN
      PRINT "Summe mit Gauss - Formel: ";
      PRINT "Die Summe der"; n%; "Zahlen ist";
      PRINT n% * (n% + 1) / 2; "."
    END IF

    INPUT "Noch einmal (j/n)? ", Weiter$
  LOOP WHILE Weiter$ = "j"
  END
```

Als nächstes wird ein Würfelspiel programmiert. Zwei Spieler würfeln abwechselnd. I Die gewürfelten Augenzahlen (1 bis 6) des jeweiligen Spielers werden addiert. Gewonnen hat der Spieler, welcher nach einer vorher festgelegten Anzahl von Würfen die größte Augensumme erhalten hat. Denkbar wäre aber auch, dass die Spieler einen anderen Würfel verwenden und sich das sogar vorher durch eine Abfrage auswählen dürfen. Wie wäre dann das Programm zu verändern?

Programmablauf:

- Die Spieler werden nach ihrem Namen gefragt und im Verlauf des Spieles mit ihrem Namen angesprochen.
- Die Anzahl der Runden wird abgefragt.
- Jede der Runden wird wie folgt behandelt:
  - Der erste Spieler stoppt den Zufallsgenerator mittels Tastendruck und ermittelt somit eine Zufallszahl zwischen 1 und 6. Die Zahl wird auf seine bereits gewürfelte Augenzahl addiert.
  - Der zweite Spieler stoppt den Zufallsgenerator mittels Tastendruck und ermittelt somit eine Zufallszahl zwischen 1 und 6. Die Zahl wird auf seine bereits gewürfelte Augenzahl addiert.
- Der Sieger wird ermittelt.

Das Spiel kann beliebig oft wiederholt werden.

Wuerfel.bas

```
REM Wuerfel-Spiel fuer zwei Spieler
REM M. Meiler
REM 17.12.2001
```

```
RANDOMIZE TIMER

CLS
INPUT "Wie ist Dein Vorname? ", Spieler1$
INPUT "Wie ist der Name des Mitspielers? ", Spieler2$

DO
  CLS
  INPUT "Wie viele Runden wollt Ihr spielen? ", Runden%

  REM Wurfelspiel
  Augenzahl1% = 0: Augenzahl2% = 0

  FOR i% = 1 TO Runden%
    CLS : PRINT "Runde: "; i%

    REM Spieler 1
    PRINT
    PRINT Spieler1$; ": ";
    INPUT "Halt den Wuerfel an (ENTER)! ", Halt$
    Wurf% = INT( RND * 6) + 1
    FOR j% = 1 TO Wurf%
      PRINT " *": BEEP: SLEEP 1
    NEXT j%
    PRINT "Du hast eine"; Wurf%; "gewuerfelt!"
    Augenzahl1% = Augenzahl1% + Wurf%

    REM Spieler 2
    PRINT
    PRINT Spieler2$; ": ";
    INPUT "Halt den Wuerfel an (ENTER)! ", Halt$
    Wurf% = INT( RND * 6) + 1
    FOR j% = 1 TO Wurf%
      PRINT " *": BEEP: SLEEP 1
    NEXT j%
    PRINT "Du hast eine"; Wurf%; "gewuerfelt!"
    Augenzahl2% = Augenzahl2% + Wurf%

    REM Augenzahl
    PRINT
    PRINT "Augenzahl von "; Spieler1$; ": ";
    PRINT Augenzahl1%
    PRINT "Augenzahl von "; Spieler2$; ": ";
    PRINT Augenzahl2%
    SLEEP 3

  NEXT i%

  REM Auswertung
  PRINT
```

```
    IF Augenzahl1% > Augenzahl2% THEN
      PRINT "Sieger ist "; Spieler1$; "!"
    END IF

    IF Augenzahl1% < Augenzahl2% THEN
      PRINT "Sieger ist "; Spieler2$; "!"
    END IF

    IF Augenzahl1% = Augenzahl2% THEN
      PRINT "Beide sind Sieger !"
    END IF

    INPUT "Noch ein Spiel (j/n)? ", Weiter$
  LOOP WHILE Weiter$ = "j"

  REM Abschlussbild"
  PRINT
  PRINT "     //"
  PRINT "   _____"
  PRINT "  / _ _ \"
  PRINT "  | o o |"
  PRINT "  |  L  |"
  PRINT "  | \_/ |"
  PRINT "  \_____/"
  PRINT
  PRINT "   ENDE"
  END
```

Jetzt kommt das versprochene Lottospiel, welches kontrolliert, ob eine Zahl bereits gewürfelt wurde. Falls dies der Fall ist, wird noch einmal gewürfelt. Als Ergebnis bekommt man einen Lottotipp, der sogar noch sortiert ausgegeben wird. Dieses Programm kann also zufällige Lottotipps ausgeben. Ob es Gewinntipps sind, kann aber kein Programm, so gut es auch programmiert wurde, garantieren.
Zum Speichern der Lottozahlen wurde ein Feld verwendet. In diesem Feld werden dann auch die Lottozahlen sortiert. Wie funktioniert der hier verwendete Sortieralgorithmus?

Lotto.bas

```
  REM 6 aus 49, beliebig viele Tipps
  REM Ausgabe sortiert
  REM M. Meiler
  REM 10.12.2001
  CLS                                 'Bildschirm loeschen
  RANDOMIZE TIMER                'Zufallsgenerator starten
  DIM Lottozahlen%(6)               'Feld fuer Lottozahlen

  DO
    REM je Tipp sechs Schleifendurchlaeufe
    FOR i% = 1 TO 6
      Wurf% = INT( RND * 49) + 1                'wuerfeln
```

```
      Lottozahlen%(i%) = Wurf%   'als Lottozahl speichern

      REM Zahl schon einmal gewuerfelt?
      FOR j% = 1 TO i% - 1
        IF Lottozahlen%(j%) = Wurf% THEN
          i% = i% - 1
        END IF
      NEXT j%
    NEXT i%

    REM Lottozahlen ordnen
    FOR i% = 1 TO 6
      FOR j% = i% + 1 TO 6
        IF Lottozahlen%(i%) > Lottozahlen%(j%) THEN
          Temp% = Lottozahlen%(i%)
          Lottozahlen%(i%) = Lottozahlen%(j%)
          Lottozahlen%(j%) = Temp%
        END IF
      NEXT j%
    NEXT i%

    REM Lottozahlenausgabe
    FOR i% = 1 TO 6
      PRINT "Zahl "; i%; ": "; Lottozahlen%(i%)
    NEXT i%

    INPUT "Noch einen Tipp (j/n)? ", Weiter$
  LOOP WHILE Weiter$ = "j"
  END
```

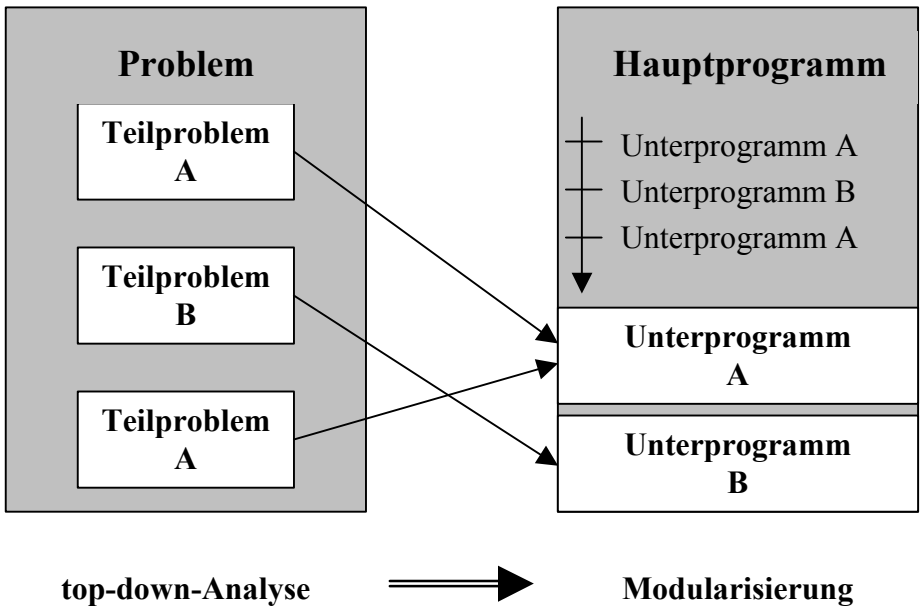
Abschließend wird ein weiteres Programm zur Berechnung des Mittelwertes eingegebener Größen unter Verwendung von Feldern vorgestellt. Im Unterschied zum Lotto muss das Feld mit **REDIM** vereinbart werden, da sich die Länge des Feldes während der Ausführung verändern kann.

Mittelwert.bas

```
  REM Mittelwertberechnung
  REM M. Meiler
  REM 17.12.2001

  CLS                                  'Bildschirm loeschen

  DO
    INPUT "Anzahl der Werte: ", N%

    IF N% = 0 THEN
      PRINT "Fehler: Mittelwert von 0 Zahlen!": END
    END IF

    REDIM Vec!(N%)                        'Feld fuer Werte
```

```
  REM Eingabe der Werte
  PRINT "Eingabe der Werte: "
  FOR i% = 1 TO N%
    PRINT "Vec!("; i%; ")=";  '; .. kein Zeilenvorschub
    INPUT " ", Vec!(i%)    'Eingabe auf derselben Zeile
  NEXT i%

  REM Mittelwertberechnung
  Mittelwert! = 0
  FOR i% = 1 TO N%
      Mittelwert! = Mittelwert! + Vec!(i%)
  NEXT i%
  Mittelwert! = Mittelwert! / N%

  REM Ausgabe der eingegebenen Werte
  REM und des Mittelwertes
  PRINT "Der Mittelwert der"; N%; "Zahlen";
  FOR i% = 1 TO N%
    PRINT Vec!(i%);
  NEXT i%
  PRINT "betraegt: "; Mittelwert!

  INPUT "Noch ein Mittelwert (j/n)? ", Weiter$
LOOP WHILE Weiter$ = "j"

END
```

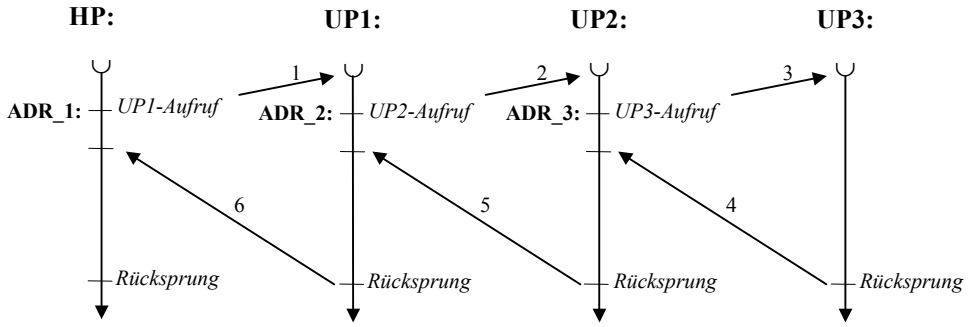
4.2.5 Unterprogrammtechnik



Mittels **top-down-Analyse** zerlegt man ein komplexes Problem solange in kleinere Teilprobleme bis diese überschaubar werden. Diese Teilprobleme kann man zunächst als **Unterprogramme** realisieren und auch einzeln testen, ehe man sie zu einem Ganzen zusammenfasst, um schließlich das komplexe Problem zu lösen.

⇒ **Unterprogramme dienen der Strukturierung eines Programms,**
1. **der Ausgliederung wiederkehrender Berechnungen und**
2. **der Zerlegung komplexer Probleme in kleinere.**

UP-Aufruf:



Zur Realisierung der Unterprogrammtechnik in einer Programmiersprache sind somit drei Sprachelemente notwendig:

1. ***Unterprogrammvereinbarung (Deklaration und Definition)***
2. ***Unterprogrammrücksprung***
3. ***Unterprogrammaufruf***

Unterprogrammvereinbarung (Deklaration und Definition)

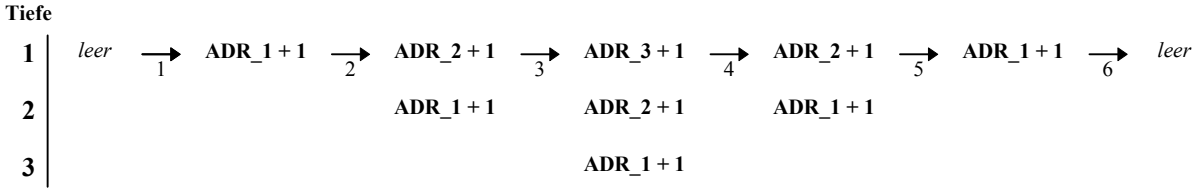
***Marke*:**
Anweisungen
RETURN

Ein Unterprogramm beginnt immer mit einer ***Marke*** und einem Doppelpunkt ***:***. Es folgen die ***Anweisungen***, die zum Unterprogramm gehören und abschließend der Rücksprungbefehl **RETURN**.
Die Marke dient als symbolische Adresse des Unterprogramms im Programmspeicher und ist damit der Einstiegspunkt des Unterprogramms. Hat man mehrere Unterprogramme im Programm, so müssen deren Marken paarweise verschieden sein.

Unterprogrammrücksprung **RETURN**

Die Rücksprungadressen werden in einem Unterprogrammkeller verwaltet.

UP-Keller:



Unterprogrammaufruf **GOSUB** ***Marke***

Ein Sprung in ein Unterprogramm erfolgt mit dem Befehl **GOSUB** und der Angabe der Marke des Unterprogramms.

Verzweigender Unterprogrammaufruf

ON ***Ausdruck%*** **GOSUB** ***Markenliste***

Ausdruck% ist ein Ausdruck, dessen Wert eine ganze Zahl liefert.

Der Wert des Ausdrucks ***Ausdruck%*** wird berechnet und das Unterprogramm angesprungen, dessen Marke in der ***Markenliste*** an entsprechender Position steht.

Beispiel EURO-Rechner mit Unterprogrammen.

EuroUP.bas

```
REM Euro-Rechner Euro  <->  DM
REM M. Meiler
REM 10.10.2001

REM Hauptprogramm

DO
  CLS                                'Bildschirm loeschen

  PRINT "Euro -> DM    ...  1"            'Auswahlmenue
  PRINT "DM   -> Euro  ...  2"
  INPUT "Waehle aus! ", Wahl%                   'Auswahl
                               'Sprung ins Unterprogramm
  ON Wahl% GOSUB EuroToDm, DmToEuro

  INPUT "Weiter j/n? ", Weiter$
LOOP WHILE Weiter$ = "j"
END

REM Unterprogramm Umrechnung Euro  ->  DM
EuroToDm:
  EuroToDm! = 1.95583                 'Umrechnungsfaktor

  REM Euro-Eingabe
```

```
    INPUT "Eur: ", Euro!

    REM DM-Berechnung
    Dm! = Euro! * EuroToDm!
    DmRund! = CLNG(Dm! * 100) / 100              'Runden

    REM DM-Ausgabe
    PRINT "DM: "; DmRund!
  RETURN                         'Unterprogrammruecksprung

  REM Unterprogramm Umrechnung DM  ->  Euro
  DmToEuro:
    DmToEuro! = .511292               'Umrechnungsfaktor

    REM DM-Eingabe
    INPUT "DM: ", Dm!

    REM Euro-Berechnung
    Euro! = Dm! * DmToEuro!
    EuroRund! = CLNG(Euro! * 100) / 100          'Runden

    REM Euro-Ausgabe
    PRINT "Eur: "; EuroRund!
  RETURN                         'Unterprogrammruecksprung
```

4.3 Elemente der Dialoggestaltung

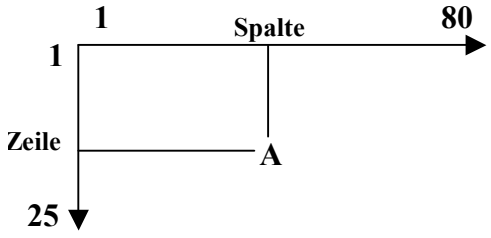
Einige Elemente der Dialoggestaltung, wie **CLS**, **PRINT** und **INPUT,** wurden schon besprochen.

Pausenbefehl **SLEEP** ***Sekundenanzahl***

akustisches Signal **BEEP**

Positionierung der Schreibmarke

LOCATE ***Zeile%*** **,** ***Spalte%***



Eme.bas

```
  REM Einmaleins mit Zeilen und Spaltensteuerung
  REM Monika Meiler
  REM 17.01.2000

  CLS

  FOR i% = 1 TO 10
```

```
    FOR j% = 1 TO 10

      REM Spaltensteuerung
      s% = 0: k% = i% * j%
      WHILE k% <> 0
        k% = INT(k% / 10): s% = s% + 1
      WEND

      REM Ausgabe
      LOCATE 2 * i% + 1, j% * 5 - s%: PRINT i% * j%
      SLEEP 1

    NEXT j%
    BEEP: LOCATE 25, 35: INPUT "Weiter mit ENTER ", e$
  NEXT i%

  END
```

Setzen von Bildschirmfarben

COLOR ***Vordergrundfarbe%*** **,** ***Hintergrundfarbe%***

Zur Bildschirmgestaltung gehört auch, dass man die Farben des Bildschirmhintergrunds und der Schrift verändern kann. Der Befehl **COLOR** setzt für die nachfolgenden Bildschirmausgaben die gewünschten Farben. Möchte man den ganzen Bildschirm in den Farben haben, so muss man ***nach*** der **COLOR** - Anweisung den Bildschirm löschen, **CLS**.
Die Farben werden als natürliche Zahlen codiert. Für den Vordergrund können alle Farben verwendet werden, für den Hintergrund nur die Farben 0 – 7:

Farbencode	Farbe	Farbencode	Farbe
0	schwarz	**8**	grau
1	blau	**9**	hellblau
2	grün	**10**	hellgrün
3	cyanblau	**11**	hellcyan
4	rot	**12**	hellrot
5	magenta	**13**	pink
6	braun	**14**	gelb
7	hellgrau	**15**	weiβ

Beispiel

COLOR 1, 7 : CLS ⇒ blaue Schrift auf hellgrauem Hintergrund

Mit einem Trainingsprogramm soll das Kopfrechnen in den vier Grundrechenarten geübt werden. Dabei soll der Schüler die Rechenoperation und den Zahlenbereich auswählen können. In Abhängigkeit von der Grundrechenart wird dann das entsprechende Unterprogramm ausgewählt. Zur Auswertung wurde das schon einmal besprochene Programm ***Punkte.bas***, hier aber als Unterprogramm, verwendet.

Rechnen.bas

```
REM Rechenuebung
REM M. Meiler
REM 10.12.2002

COLOR 7, 1: CLS


REM Hauptprogramm
DO
  rc% = 0                              'richtig gerechnet

  PRINT : PRINT "Rechenuebung - WAEHLE AUS"
  PRINT
  PRINT "Moechtest Du addieren ? .........1"
  PRINT "Moechtest Du subtrahieren ? .....2"
  PRINT "Moechtest Du multiplizieren ? ...3"
  PRINT "Moechtest Du dividieren ? .......4"
  PRINT "ENDE ............. ..............5"
  PRINT : INPUT "Waehle aus : ", w%

  IF w% < 5 THEN
    PRINT
    PRINT "Bis zu welcher Zahl kannst du rechnen? ";
    INPUT "Gib sie ein : ", z%
    PRINT
    INPUT "Wie viele Aufgaben? ", ac%

    SELECT CASE w%
      CASE 1: GOSUB add
      CASE 2: GOSUB subtr
      CASE 3: GOSUB mult
      CASE 4: GOSUB div
    END SELECT

    GOSUB note
  END IF
LOOP UNTIL w% = 5

PRINT "Ende"
SLEEP 1

END


REM Unterprogramme


REM Addition
add:
```

```
    COLOR 1, 7: CLS
    PRINT : PRINT "Additionsuebung": PRINT
    z% = z% + 1
    FOR i% = 1 TO ac%
      a1% = INT(RND * z%): d% = z% - a1%
      a2% = INT(RND * d%)
      PRINT a1%; "+"; a2%; : INPUT ; "= ", e%
      s% = a1% + a2%
      COLOR 4
      IF e% = s% THEN
        LOCATE , 25: PRINT "Richtig"
        rc% = rc% + 1
      ELSE
        LOCATE , 25: PRINT "Falsch";
        PRINT a1%; "+"; a2%; "="; s%
      END IF
      PRINT : COLOR 1
    NEXT i%
    LOCATE , 25: INPUT "weiter mit ENTER", Weiter$
  RETURN


  REM Subtraktion
  subtr:
    PRINT "sub ..."
  RETURN


  REM Multiplikation
  mult:
    PRINT "mult ..."
  RETURN


  REM Division
  div:
    PRINT "div ..."
  RETURN


  REM Auswertung
  note:

    REM Auswertung Punktergebnis          'Ergebnis total
    COLOR 7, 1: CLS
    PRINT
    PRINT "Du hast von insgesamt"; ac%; "Punkten";

    SELECT CASE rc%
      CASE IS > 1
        PRINT rc%; "Punkte ";
```

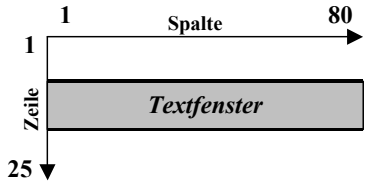
```
      CASE 1
        PRINT " einen Punkt ";
      CASE IS <= 0
        PRINT " keinen Punkt ";
    END SELECT

    PRINT "erreicht!"

    REM Auswertung Note
    Prozent% = rc% * 100 / ac%              'Ergebnis in %
    PRINT : PRINT "Deine Note ist eine ";

    SELECT CASE Prozent%
      CASE IS < 20
        PRINT "6!"
        PRINT "Schade, Du musst noch sehr viel Ueben! "
      CASE 20 TO 39
        PRINT "5!"
        PRINT "Ueben, ueben, ueben! ";
        PRINT "Es ist noch kein Meister vom Himmel ... !"
      CASE 40 TO 59
        PRINT "4!"
        PRINT "Das kann noch besser werden!"
      CASE 60 TO 79
        PRINT "3!"
        PRINT "Zufriedenstellend, es geht noch besser!"
      CASE 80 TO 94
        PRINT "2!"
        PRINT "Gut! Weiter so!"
      CASE IS >= 95
        PRINT "1!"
        PRINT "Prima, ich habe mich sehr gefreut! ";
        PRINT "Weiter so!"
      END SELECT

  RETURN
```

4.4 Grafik

Grafische Elemente in der Programmierung kann man nur im Grafikmodus darstellen. Dieser ist abhängig von der Grafikkarte. In der Hilfe (Taste ↑ + **F1**, zurück **Esc**) sind unter **SCREEN 0 ... 15** die einzelnen Bildschirmmodi erläutert.
Standardmäβig wird im Textmodus (**SCREEN 0**) gearbeitet. Für VGA-Karten steht auβerdem der Grafikmodus (**SCREEN 9**) zur Verfügung. Im Grafikformat hat man 640 * 350 Pixel und im Textformat 80 * 25 Zeichen mit der Zeichenfeldgröβe 8 * 14 Pixel.
Es ist möglich, einen Teil des Bildschirms als Textfenster und einen als Grafikfenster zu definieren.

Textfenster

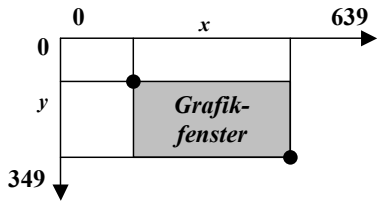
VIEW PRINT ***erste_Zeile%*** **TO** ***letzte_Zeile%***
Die erste Zeilenummer und die letzte Zeilennummer des Textfensters werden angegeben.



VIEW PRINT
Der ganze Bildschirm ist Textfenster.

Grafikfenster

VIEW (***x1%*** **,** ***y1%*** **) - (** ***x2%*** **,** ***y2%*** **) ,** ***Fensterfarbe%*** **,** ***Randfarbe%***
Die linke obere Ecke **(** ***x1%*** **,** ***y1%*** **)** und die rechte untere Ecke **(** ***x2%*** **,** ***y2%*** **)** des Grafikfensters werden angegeben, außerdem kann man noch die Hintergrunds- und die Randfarbe definieren.



VIEW
Der ganze Bildschirm ist Grafikfenster.

Grafikelemente (Punkte, Linien, Kreise, Rechtecke, ...) werden durch spezielle Anweisungen erzeugt. In dem folgenden Beispiel wird ein Grafikfenster definiert und mit dem Befehl `LINE (x1%, y1%)-(x2%, y2%), 0` Strecken gezeichnet, dabei wird der Startpunkt, der Endpunkt und die Linienfarbe angegeben.

Beispiel

Nicolaus.bas

```
REM * * Nicolaus * * (HAUS VOM NICOLAUS)
REM M. Meiler
REM 07.01.2002

SCREEN 9

DO
          'Loeschen des Text und des Grafikbildschirms
  CLS 1: CLS 2
  VIEW (200, 1)-(638, 332), 3, 1      'Grafikbildschirm

REM Eckpunkte zeichnen
  LOCATE 20, 28: PRINT "A": LOCATE 20, 30: PRINT "*"
  LOCATE 20, 52: PRINT "B": LOCATE 20, 50: PRINT "*"
  LOCATE 10, 28: PRINT "C": LOCATE 10, 30: PRINT "*"
  LOCATE 10, 52: PRINT "D": LOCATE 10, 50: PRINT "*"
  LOCATE 5, 38: PRINT "E": LOCATE 5, 40: PRINT "*"

REM Startpunkt festlegen
  INPUT "Beginn bei Punkt: ", B$
```

```
    IF B$ = "A" THEN x1% = 36: y1% = 272
    IF B$ = "B" THEN x1% = 196: y1% = 272
    IF B$ = "C" THEN x1% = 36: y1% = 132
    IF B$ = "D" THEN x1% = 196: y1% = 132
    IF B$ = "E" THEN x1% = 116: y1% = 62

  REM Haus zeichnen
    FOR k% = 1 TO 8
      INPUT "Weiter zu Punkt: ", w$
      IF w$ = "A" THEN x2% = 36: y2% = 272
      IF w$ = "B" THEN x2% = 196: y2% = 272
      IF w$ = "C" THEN x2% = 36: y2% = 132
      IF w$ = "D" THEN x2% = 196: y2% = 132
      IF w$ = "E" THEN x2% = 116: y2% = 62
      LINE (x1%, y1%)-(x2%, y2%), 0
      x1% = x2%: y1% = y2%
    NEXT k%

    PRINT "Ende"
    PRINT
    INPUT "Haus vom Nicolaus noch einmal(j/n)? ", Weiter$
  LOOP WHILE Weiter$ = "j" OR Weiter$ = "J"

  SCREEN 0: CLS
  END
```