Lektion 4: Schleifen | **Anwendung: Informationen von der Bank**

Diese Anwendung verwendet Schleifen, um so viele Daten, wie notwendig, zu sammeln. Als Erweiterung werden die Daten auf Gültigkeit geprüft. Mit **If**-Anweisungen wird eine geeignete Meldung ausgegeben.

Lernziele:

- Zähler- und Sammelanweisungen verwenden
- Eine Programmschleife zur Eingabe einer unbestimmten Anzahl von Daten verwenden
- Mit einem 'Schalter' eine Schleife abbrechen

Hinweis: Dieses Beispiel zeigt, dass Software-Entwicklung rasch komplex werden kann. Der Begriff ‚Schachtelung' bezieht sich darauf, eine Kontrollstruktur in eine andere zu verpacken. Wie unten gezeigt, weiß' das BS, zu welcher Anweisung jedes End gehört.

Geschachtelte Strukturen

- **Verschachteln** ist die Programmiertechnik, bei der eine Kontrollstruktur innerhalb einer anderen auftritt.
- Der Ausdruck kommt vom Verstauen einer Schachtel in einer anderen um Platz zu sparen.
- Ein Programmierer setzt Schleifen innerhalb von Schleifen, **If**s in Schleifen und Schleifen in **If**s, um nach Erfordernissen des Programms komplexere Aufgaben zu erfüllen.
- Es ist wichtig, eine *komplette* Struktur *ganz* innerhalb eines anderen Blocks zu setzen, um Fehler zu vermeiden.
- Das Programm rechts zeigt eine **While** Schleife und eine **If** Konstruktion innerhalb einer anderen **While**-Schleife.
- Beachte das mehrfache Auftreten von **EndWhile**. Der Computer "weiß", zu welchem **While** jedes **EndWhile** gehört.
- Die Einrückungen sind eine optische Unterstützung und helfen, die Programmlogik zu verdeutlichen.

```
Define perfqu()=
Prgm
Local a,s
a:=1
While a>0
  Request "Gib eine Zahl ein:",a
  While a<0
    Disp "keine negative Zahl!"
    Request "Gib eine positive Zahl ein:",a
  EndWhile
  s:=√(a·1.)
  If s=int(s) Then
    Disp "Ist eine Quadratzahl!"
  Else
    Disp "Ist keine Quadratzahl!"
  EndIf
  Disp "Die Quadratwurzel ist",s
EndWhile
EndPrgm
```

Zuerst wird eine Schleife eingerichtet, die so lange Eingaben erwartet bis 0 eingegeben wird. Dann wird überprüft ob a<0. In diesem Fall wird die Meldung „Gib eine positive Zahl ein:" gezeigt und um eine andere Eingabe gefragt. Wenn a>0, wird die Quadratwurzel berechnet und in einer weiteren **If**-Abfrage werden die **Disp**-Anweisungen ausgeführt.

Zusammenfassung der drei Schleifen:

For(*Variable*, *Start*, *Ende***)**	**While** <*Bedingung*>	**Loop**
		If <*Bedingung*> **: Exit**
EndFor	**EndWhile**	**EndLoop**

For wird verwendet für eine Zählung oder für die Verarbeitung einer arithmetischen Folge von Werten.

While verwendet man, wenn man die Schleife komplett überspringen können will.

Loop wählt man, wenn die Schleife zumindest einmal durchlaufen werden soll. Sie muss eine **Exit**-Anweisung

enthalten, üblicherweise als Teil einer **If**-Abfrage.

Hinweis: Die einzige ‘wichtige’ Schleife ist die While-Schleife. Sie kann die gleichen Aufgaben wahrnehmen wie die beiden anderen. Verwende keine GoTo-Anweisungen, um eine Schleife zu verlassen oder sonst wo in einem Programm, Das ist schlechter Programmierstil.
Im Steuerungsmenu gibt es noch weitere Anweisungen, die in diesen Lektionen nicht behandelt werden. Im TI-Nspire Reference Guide findet man entsprechende Informationen.

Die Variable a wird unter der Wurzel mit 1 multipliziert. Damit wird erreicht, dass die Ausgabe in der numerischen und CAS-Ausgabe gleich ist. Der Dezimalpunkt in einem Ausdruck erzwingt eine Ausgabe als Dezimalzahl.

Lektion 4 Anwendung: Programm “bankinfo”

Ein Bankkunde hat bei einer Bank mehrere Konten. Die Bank verlangt, dass der Durchschnitt der Guthaben auf allen Konten mindestens $1000 beträgt, anderenfalls wird eine Verwaltungsgebühr fällig.

Liegt der Durchschnitt zwischen $1000 und $1250 benachrichtigt die Bank den Kunden mit einer Information über die drohende Gebühr.

Wenn der Durchschnitt aber über $1250 liegt, dann bedankt sich die Bank für den guten Durchschnittswert.

Wir wollen ein Programm schreiben, das den Kunden über seine Kontostände informiert. Er gibt seine Kontostände ein und das Programm zählt die Einlagen, berechnet deren Gesamtwert, sowie deren Durchschnitt und gibt dann dem Kunden eine entsprechende Information. Diese können z.B. lauten: „Verwaltungsgebühr fällig!“. „Verwaltungsgebühr könnte bald anfallen!” und schlicht “VIELEN DANK!”.

Wir können zwei Methoden zur Eingabe einer Unbekannten Anzahl von Werten einsetzen:

- Methode 1: Frage zuerst nach der Zahl der Konten und verende dann eine **For**-Schleife zur Eingabe
- Methode 2: Frage nach den Beträgen und gib einen “Schalter” wie z.B. -999 als Anzeige dafür an, dass die Eingabe beendet ist. Bei dieser Methode setzt man eine **While**-Schleife ein.

Bei beiden Methoden rechnen wir die *Gesamtsumme* aller Werte laufend mit.

In Methode 2 müssen wir auch die *Anzahl* der Beträge mitzählen, sonst können wir den *Durchschnitt* nicht als Quotient aus *Gesamtsumme* und *Anzahl* berechnen. Die folgenden Informationen können hilfreich sein.

Dein Programm sollte ausgeben: 1. Die Anzahl der eingegebenen Beträge, 2. den Durchschnitt aller Beträge und 3. eine Meldung, die sich auf diesen Durchschnitt bezieht.

Liegt der Durchschnitt: unter $1000: „Verwaltungsgebühr fällig!“

zwischen $1000 und $1250: „Verwaltungsgebühr könnte bald anfallen!“

… über $1250: „VIELEN DANK!“

Hinweis: Im folgenden Absatz werden zwei verwandte Programmiertechniken besprochen: Zähler und Akkumulator. Beachte, dass die Variable links vom Zuweisungsoperator (:=) die gleiche Variable ist wie rechts von ihm, dass sie aber je nach der ausgeführten Prozedur unterschiedliche Werte darstellen.

Zählen und Aufsummieren (Akkumulieren)

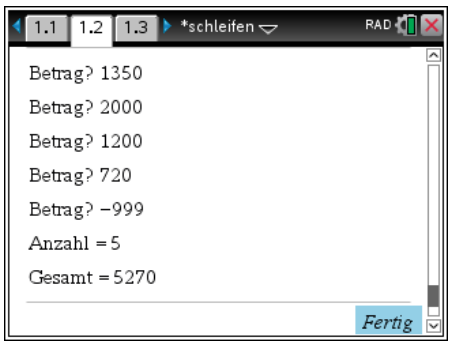
Eine Anweisung wie **c:=c+1** nennt man „Zähler“, da die Variable **c** bei jeder Ausführung um 1 erhöht wird.

Eine Anweisung wie **t:=t+n** wird „Akkumulator“ genannt, weil die Werte von **n** laufend aufsummiert (akkumuliert) werden. **n** wird zu **t** addiert und die entstehende Summe wieder als **t** abgespeichert. Am Ende der Abarbeitung der Schleife enthält die Variable **t** die Summe aller **n**-Werte.

Hier folgt ein Beispiel, das einen Zähler, einen Akkumulator (*gesamt*) und einen Schalter (-999), um die Anzahl der eingegebenen Beträge zu zählen.

Prgm	Erklärung
Local zaehler,betrag,gesamt	Initialisierung der Variablen
gesamt:=0	
zaehler:=1	
Request "Betrag?",betrag	Ersten Betrag einlesen
While betrag≠−999	So lange bis -999 eingegeben wird
** zaehler:=zaehler+1**	Zähler wird um 1 erhöht
** gesamt:=gesamt+betrag**	Betrag wird zur Summe addiert
** Request "Betrag?",betrag**	Um nächsten Betrag wird gefragt
EndWhile	
Disp "Anzahl =",zaehler	
Disp "Gesamt =",gesamt	
endPrgm	

Beispiel eines Programmlaufs



Die **While**-Schleife zählt und akkumuliert so lange bis -999 als Betrag eingegeben wird. Dann wird die Schleife beendet und die Resultate werden angezeigt.

Hinweis: Weise auf die beiden **Request**-Anweisungen im Programm oben hin. Die erste fragt nach dem ersten Betrag, während die zweite alle übrigen Beträge verlangt. Diese zweite befindet sich am *Ende* der Schleife und sorgt für den Rücksprung zum **While**, wo die Eingabe erneut geprüft wird.

Erweiterung

Versichere dich im Rahmen der Eingaberoutine, dass ein sinnvoller Betrag (größer 0) eingegeben wird und sieh für den Fall einer falschen Eingabe eine entsprechende Meldung an den Kunden vor.

Hinweis: Die Erweiterung erfordert eine weitere Schleife um die Eingabe, um sicher zu stellen, dass sie sinnvoll ist. Der Lösungsvorschlag unten bricht in diesem Fall einfach das Programm ab. Die Schüler finden da sicher eine bessere Lösung!

Hier ist ein erster Lösungsvorschlag:

```
Define bankinfo()=
Prgm
Local betrag,zaehler,gesamt,durchschn
betrag:=0 : zaehler:=0 : gesamt:=0
Request "Geben Sie einen Betrag ein:",betrag
While betrag≠−999
  If betrag<0 Then
    Disp "KEINE NEGATIVEN BETRÄGE, BITTE!"
    Stop
  Else
     zaehler:=zaehler+1
     gesamt:=gesamt+betrag
  EndIf
  Request "Nächster Betrag, bitte (oder -999):",betrag
EndWhile
durchschn:=((gesamt*1.)/(zaehler))
Disp "ANZAHL der Beträge:",zaehler
Disp "GESAMTSUMME auf allen Konten:",gesamt
Disp "DURCHSCHNITT der Beträge:",durchschn
If durchschn<1000 Then
  Disp "Verwaltungsgebühr fällig!"
ElseIf durchschn<1250 Then
  Disp "Verwaltungsgebühr könnte bald anfallen!"
Else
  Disp "VIELEN DANK!"
EndIf
EndPrgm
```

Wenn du ein Programm testest, dann versuche auch Ausnahmefälle, wie -999 als erste Eingabe und auch die Eingabe von negative Werten an beliebigen Stellen.

Wenn das Programm eine Fehlermeldung anzeigt, dann hat der Programmierer nicht alle möglichen Fälle in Betracht gezogen. Dieses Programm wird einen Fehler anzeigen, wenn -999 als erste Eingabe erfolgt, da es dann zu einer Division 0 durch 0 kommt, und das ist nicht definiert.

Daher sollte die Berechnung des Durchschnitts mit einer Abfrage verknüpft werden, ob zumindest ein Betrag eingegeben wurde. Das könnte so aussehen:

```
If zaehler>0 Then
  durchschn:=((gesamt*1.)/(zaehler))
Else
  Disp "KEIN ETRAG EINGEGEBEN!"
  Stop
EndIf
```

Die hier verwendete **Stop**- Anweisung lässt das Programm *sofort* abbrechen. Sie kann überall im Programm eingefügt werden, so dass die Durchführung abgebrochen wird und die Meldung *Fertig* erscheint.