

Schleifen

Ein wichtiges Element der von MuPAD zur Verfügung gestellten Programmiersprache sind sogenannte Schleifen. Eine Schleife ermöglicht die wiederholte Ausführung von Befehlen.

Die FOR-Schleife

Die folgende Eingabezeile bedeutet:

Für i von 1 bis 4 tue folgendes: „Schreibe den Wert von i auf den Bildschirm.“ **Ende** der Schleife

- **for i from 1 to 4 do print(i); end_for;**

Da die Laufvariable i in ganzen Schritten hochgezählt wird, müssten die Zahlen 1, 2, 3 und 4 auf dem Bildschirm erscheinen, weil i bei 1 startet, dann den Wert 2, dann den Wert 3, dann den Wert 4 erhält und nach dem 4. Durchlauf die Schleife beendet wird.

- **for i from 1 to 5 do x:=i^2; end_for;**

Hier wird nur 25 ausgegeben, weil die einzelnen Berechnungen der Schleife nicht auf den Bildschirm geschrieben werden, auch wenn wie hier geschehen, die Befehle mit Semikolon abgeschlossen wurden. Nur der letzte Wert von x erscheint. Will man alle Werte, so muss man die Ausgabe in jedem Schleifendurchlauf durch den *print*-Befehl erzwingen:

- **for i from 1 to 5 do x:=i^2; print(x); end_for;**

Die folgende Schleife zählt rückwärts:

- **for i from 10 downto 3 do print(i); end_for;**

Will man nicht in Einerschritten hoch- bzw. runterzählen, so verwende man *step*-Befehl:

- **for i from 3 to 20 step 2 do print(i); end_for;** liefert die ungeraden Zahlen von 3 bis 20
- **for i from 100 downto 11 step 5 do print(i); end_for;**

Dabei bricht, wie hier zu sehen, die Schleife auch ab, wenn der angegebene Endwert nicht direkt errechnet, sondern über- bzw. unterschritten wird.

Man kann auch Listen angeben, aus welcher die Laufvariable Werte annehmen soll:

- **for i in [5, 11, -14, 1.3, y, -z^3] do print(i^2); end_for;** Quadrate der Elemente

Die Laufvariable der *FOR*-Schleife wird also in festgelegten Schritten von einem Anfangswert hoch- bzw. runtergezählt, bis ein angegebener Endwert über- bzw. unterschritten wird; oder die Laufvariable durchläuft alle Elemente einer Liste.

Eine flexiblere Schleife ist die *REPEAT*-Schleife, bei der in jedem Schritt die Laufvariable in beliebiger Weise abgeändert werden kann:

Die REPEAT-Schleife

- **i:=2: repeat i:=i^2; print(i) until i>100 end_repeat;** beachten Sie den “:” nach **i:=2:**

Diese Schreibweise ist etwas unübersichtlich. Nutzen Sie *SHIFT* + *ENTER*, um die Befehle untereinander zu schreiben, ohne die einzelnen Befehle sofort auszuführen.

- **i:=2:
repeat
 i:=i^2; print(i)
until i>100 end_repeat;** Mit der *TAB*-Taste (links neben Q) können sie Befehle einrücken.

Schließen Sie nun mit *ENTER* ab, so werden die Befehle zusammengefasst ausgeführt. (Ausgabe: 4, 16, 256)

Was bedeutet diese Anweisung nun? Im Vorfeld wird die Laufvariable auf 2 gesetzt, damit MuPAD den Anfangswert der Laufvariablen kennt. Der Doppelpunkt bedeutet, dass das Ergebnis dieses Befehls nicht auf dem Bildschirm erscheint. Ein Semikolon würde die zusätzliche Ausgabe von 2 zur Folge haben. Alles, was zwischen *repeat* und *end_repeat* steht, wird solange wiederholt, bis die Abbruchbedingung erfüllt ist. Für dieses Beispiel heißt das: Wiederhole die Befehle **i:=i^2;** und **print(i)**, bis die Laufvariable i größer als 100 ist. Im ersten Durchlauf ist i am Anfang 2, wird dann aber sofort durch **i:=i^2** auf 4 gesetzt (Das neue i ergibt sich aus dem alten i, indem man das alte i quadriert.). Die 4 wird nun durch **print(i)** ausgegeben. Da 4 noch nicht größer als 100 ist, wird die Schleife erneut durchlaufen. Die Variable i erhält also den Wert $4^2 = 16$; dieser Wert wird ausgegeben. Da 16 immer noch nicht größer als 100 ist, wird die Schleife noch mal durchlaufen; i erhält den Wert $16^2 = 256$, dieser Wert wird ausgegeben. Da nun aber 256 größer als 100 ist und damit die hinter **until** angegebene Abbruchbedingung erfüllt ist, wird die Schleife verlassen.

Will man neben den neuen i-Werten auch noch die alten bei jedem Schleifendurchlauf ausgegeben haben, so könnte man einen weiteren *print*-Befehl einfügen:

- **i:=2:
repeat
 print(i); i:=i^2; print(i)
until i>100 end_repeat;**

Man erhält die Ausgabe 2, 4, 4, 16, 16, 256. Die erste, dritte und fünfte Zahl sind jeweils die alten Werte.

Das sieht nicht so schön aus! Versuchen wir es etwas anders:

- **neuerWert:=2:**
repeat
 alterWert := neuerWert; neuerWert := alterWert^2; print(alterWert, neuerWert)
 until neuerWert > 100 end_repeat;

Hier werden jeweils die alten Werte und neuen Werte ausgegeben. Die Ausgabe der beiden Werte, getrennt durch ein Komma, ermöglicht **print(alterWert, neuerWert)**.

Am Anfang ist **neuerWert** auf 2 gesetzt. Beim 1. Durchlauf wird zuerst **alterWert** auf 2 gesetzt, dann ergibt sich der neue Wert durch Quadrieren des alten Wertes, also **neuerWert := 2² = 4**. Da 4 kleiner als 100 ist, wird die Schleife erneut durchlaufen. Der alte Wert wird durch den neuen Wert ersetzt, also **alterWert := 4**, der neue Wert entsteht durch Quadrieren des alten Wertes, also **neuerWert := 4² = 16**, usw.

Versuchen wir, wie weiter oben mit der **FOR**-Schleife, alle ungeraden Zahlen von 3 bis 20 auszugeben:

- **i:=1:**
repeat
 i := i + 2; print(i)
 until i >= 19 end_repeat;

Beachten Sie, dass die Schleife erst durchlaufen wird und dann die Abbruchbedingung geprüft wird. Geben Sie statt 19 den Endwert 20 an, so wird auch noch die Zahl 21 ausgegeben.

Alternativ kann man mit dem Startwert $i = 3$ beginnen. Dann muss aber der alte i -Wert ausgegeben werden.

- **i:=3:**
repeat
 print(i); i := i + 2
 until i >= 19 end_repeat;

Die **REPEAT**-Schleife wiederholt also die angegebenen Befehle solange, bis die Abbruchbedingung erfüllt ist. Die Abbruchbedingung wird am Ende geprüft. Die **WHILE**-Schleife prüft am Anfang eine Bedingung und die Schleife wird nur durchlaufen, wenn die Bedingung erfüllt ist:

Die While-Schleife

- **n := 2:**
while n < 100 do
 a := n; n := a^2; print(a, n)
 end_while;

Dies bedeutet: Solange die Bedingung $n < 100$ erfüllt ist, wiederhole die Befehle. Vergleichen Sie mit der weiter oben aufgeführten **REPEAT**-Schleife. Die Variablen heißen nun „a“ und „n“, statt „alterWert“ und „neuerWert“.

Syntax der Schleifen im Vergleich

- **for** Variable **from** Startwert **to** Endwert **step** Schrittweite **do**
 Anweisungen
 end_for;

Von Startwert bis zum Endwert führe die Anweisungen aus. Erhöhe (erniedrige) dabei die Laufvariable um die Schrittweite.

- **Laufvariable :=** Startwert;
repeat
 Anweisungen
 until Abbruchbedingung **end_repeat;**

Wiederhole die Anweisungen solange, bis die Abbruchbedingung erfüllt ist.

(Die Schleife wird mindestens einmal durchlaufen, da die Abbruchbedingung am Ende geprüft wird.)

- **Laufvariable :=** Startwert;
while Bedingung **do**
 Anweisungen
 end_while;

Solange die Bedingung erfüllt ist, führe die Anweisungen aus. (Die Bedingung wird am Anfang geprüft.)

Aufgaben:

1. Geben Sie die Folgenglieder 5, 11, 17, 23, 29, ..., 89 aus. Versuchen Sie dies mit allen drei Schleifen.
2. Geben Sie die Folgenglieder 89, 83, 77, 71, 65, ... 5 aus. Versuchen Sie dies mit allen drei Schleifen.
3. Berechnen Sie die Summe der Zahlen 4, 5, 6, ..., 47. Versuchen Sie dies mit allen drei Schleifen.
4. Berechnen Sie die Summe aller ungeraden Zahlen von 3 bis 29. Versuchen Sie dies mit allen drei Schleifen.

Speichern Sie unter **Schleifen.txt** ab.