# 11 Schleifen abbrechen

**Lernziele** In diesem Abschnitt lernst du:

- Eine Schleife abzubrechen, bevor sie fertig ist.

**Einführung** Schleifen verwendest du beim Programmieren oft dazu, um etwas zu suchen oder zu berechnen. Dabei kannst du nicht immer genau wissen, wie oft sich die Schleife wiederholen muss, bis der Computer das Ergebnis gefunden hat. Deshalb ist es manchmal sinnvoll, eine Schleife mit `break` abzubrechen.

Nehmen wir als Beispiel an, der Computer soll überprüfen, ob 91 eine Primzahl ist. Dazu muss er grundsätzlich alle Zahlen von 2 bis 90 durchgehen und ausrechnen, ob sich 91 ohne Rest durch eine kleinere Zahl teilen lässt. Nachdem der Computer aber festgestellt hat, dass 91 durch 7 teilbar ist, muss er die anderen Zahlen nicht mehr überprüfen. Er hat die Antwort auf unsere Frage und kann daher mit der Berechnung aufhören.

**Das Programm** Das Programm fordert den Anwender in der ersten Zeile dazu auf, eine ganze Zahl einzugeben (erinnerst du dich daran, dass `int` für eine ganze Zahl steht?).

In den Zeilen 2 bis 8 prüft das Programm der Reihe nach alle möglichen Teiler durch. Wenn die eingegebene Zahl durch einen Teiler wirklich teilbar ist, dann wird die Schleife in Zeile 7 abgebrochen. Die Variable `teiler` enthält jetzt den kleinsten Teiler der eingegebenen Zahl (ausser 1 natürlich).

Am Schluss prüfen wir, ob der gefundene Teiler kleiner ist als die Zahl und geben entsprechend aus, dass es eine Primzahl ist oder nicht.

```
1  zahl = inputInt("Bitte gib eine ganze Zahl ein:")
2  teiler = 2
3  repeat zahl-1:
4      rest = zahl % teiler
5      if rest == 0:
6          break
7      teiler += 1
8
9  if teiler < zahl:
10     msgDlg(zahl, "ist durch", teiler, "teilbar!")
11 if teiler == zahl:
12     msgDlg(zahl, "ist eine Primzahl!")
```

Probiere das Programm mit verschiedenen Zahlen aus und beobachte mit dem Debugger, wann was passiert. Achte dabei vor allem darauf, was `break` bewirkt und wo das Programm nach `break` weiterfährt.

**Die wichtigsten Punkte** Schleifen wiederholen einen Programmteil für eine feste Anzahl von Durchläufen. Mit `break` kannst du die Schleife aber auch vorzeitig abbrechen. Bei `break` fährt der Computer also nach dem Schleifencode weiter: Er überspringt sowohl den Rest des Schleifencodes als auch alle Wiederholungen, die noch ausstehen.

Die `break`-Anweisung hat nur innerhalb einer `if`-Bedingung Sinn, weil die Schleife sonst sofort abgebrochen und gar nicht wiederholt würde.

```
repeat n:
    Schleifencode
    if Abbruch-Bedingung:
        break
    Schleifencode
```

## AUFGABEN

**44.** Wie oft musst du $1.5$ mit sich selbst multiplizieren, bis das Ergebnis grösser ist als 100? Wie sieht es auf mit $1.05$ ...?

Schreibe ein Programm, das die Antwort mit einer Schleife sucht. Sobald $1.5^n$ grösser ist als $100$, bricht die Schleife ab und das Programm gibt das Ergebnis aus. Mit `anzahl += 1` kannst du z. B. zählen, wie oft die Schleife tatsächlich wiederholt wird.

**45.** Zerlege eine Zahl in ihre Primfaktoren. Das Grundprinzip funktioniert ähnlich wie beim Primzahltest oben. Wenn allerdings `rest == 0` ist, dann teilst du die Zahl `zahl` mit `/=` durch den Teiler. Und nur wenn der Rest nicht Null ist, erhöhst du den Teiler um Eins (eine Zahl kann ja auch mehrfach durch $3$ teilbar sein).

Woran erkennst du, dass die Zahl fertig zerlegt ist und du die Schleife abbrechen kannst?

**46.** Schreibe eine «Passwort-Schleife», die sich so lange wiederholt, bis jemand das richtige Passwort (oder die richtige Antwort auf eine Quizfrage oder Rechnung) eingegeben hat.