

Algorithmen

(Schleifen 2)

Jetzt kennen wir schon Schleifen, die bei bestimmten Bedingungen terminieren. Es gibt noch eine weitere Schleifenart, nämlich die Zählschleife.

Wie der Name schon sagt, zählt diese Schleifenart die Durchläufe und beendet sich dann nach der entsprechenden Anzahl Wiederholungen.

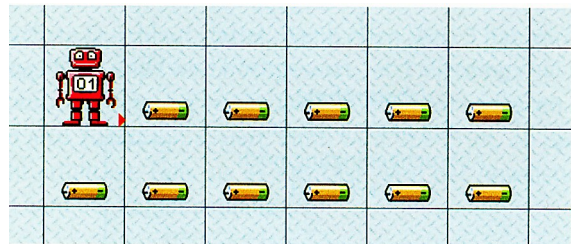
Auch hier gibt es eine Bedingung, die erfüllt werden muss, damit die Schleife läuft, allerdings wird hier ein Zähler überprüft, der einen bestimmten Wert erreichen soll.

Umgangssprachlich könnte man dem Roboter sagen:

„Nimm vier Akkus auf.“

Im Pseudocode würde das lauten:

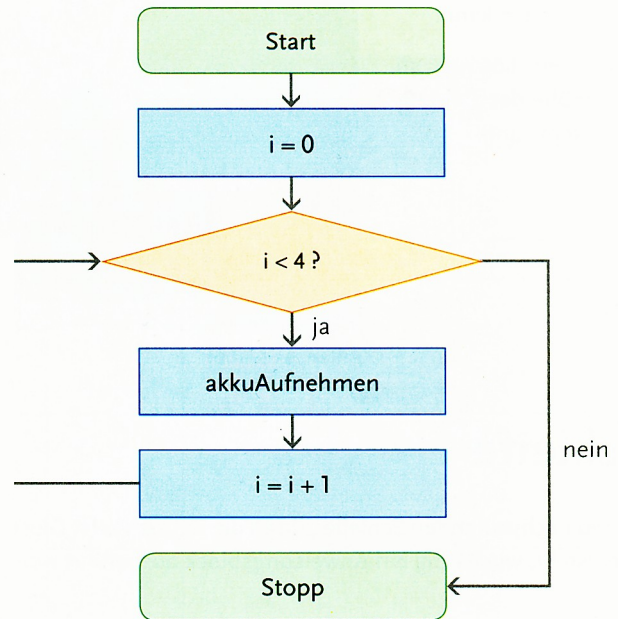
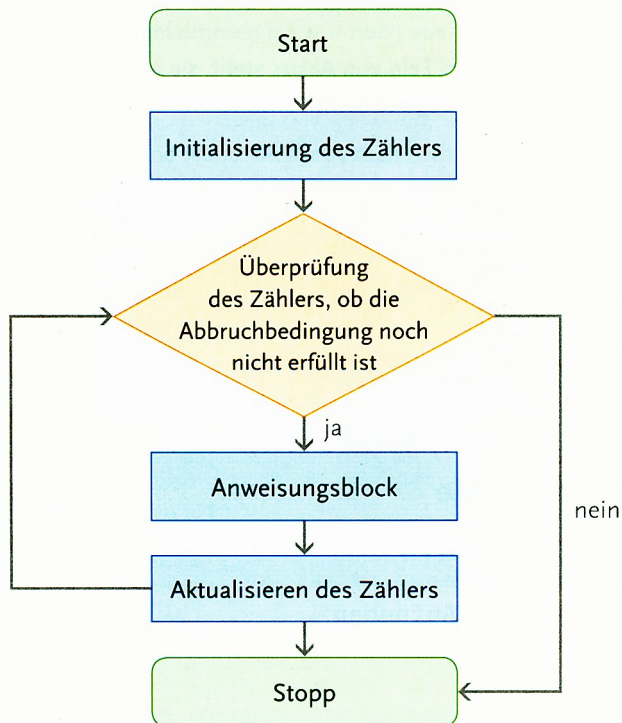
wiederhole 4 mal
 Roboter bewegen
 Roboter Akku aufnehmen



Oder allgemein:

wiederhole n mal
 Anweisungsblock

Oder auch als Programmablaufplan:



Das Ablaufdiagramm einer Zählschleife sieht also in der allgemeinen Form so aus.

In Java wird das folgendermaßen aufgeschrieben:

```

for (int i = 0; i < 4; i++) {
    bewegen();
    akkuAufnehmen();
}
  
```

Allgemein:

```
for (Deklaration & Initialisierung; Abbruchbedingung; Aktualisierung) {  
    Anweisungsblock  
}
```

Wie schon in den anderen Schleifen, kann auch hier ein ganzer Block von Anweisungen in den geschweiften Klammern stehen. Wichtig ist hier der Bedingungsteil. Der Zähler heißt hier i . Eine Zählschleife durchläuft folgende Schritte:

1. Der Zähler i startet mit dem Initialisierungswert.
2. Die Abbruchbedingung wird überprüft.
 - a) Bedingung ist falsch: Der Anweisungsblock wird übersprungen und es geht nach der Schleife weiter im Programm.
 - b) Bedingung ist wahr: Der Anweisungsblock wird ausgeführt und es folgt 3.
3. Der Zähler wird aktualisiert. Es folgt Schritt 2.

Also für unser Beispiel hier konkret:

Der Zähler i wird mit 0 initialisiert. i ist also kleiner als 4 und die Anweisungen `bewegen()` und `akkuAufnehmen()` werden ausgeführt. Dann wird i aktualisiert, also gilt jetzt $i=1$. Der nächste Durchlauf startet. Wenn i den Wert 3 hat, wird die Anweisung noch einmal ausgeführt und i wird hoch gesetzt. Jetzt hat i den Wert 4 und somit ist die Abbruchbedingung erfüllt. Die Schleife wird verlassen.

Also zuerst wird der Zähler initialisiert, dann wird die Abbruchbedingung geprüft. Ist sie noch nicht erfüllt, wird der Anweisungsblock ausgeführt. Erst dann wird der Zähler aktualisiert.

Aufgaben

1. Die erste Zählschleife
 - a) Vervollständigt die Tabelle für das obige Beispiel bis zum Abbruch der Schleife.

Zähler	Abbruchbedingung erfüllt?	Zähler aktualisieren	Ausgeführte Anweisungen

b) Ändert die Zählschleife so ab, dass eine Endlosschleife entsteht.

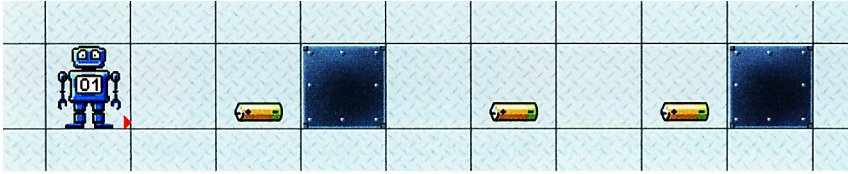
2. Euer Roboter
 - a) Implementiert die Zählschleife in eurer eigenen Klasse in der Methode `act()` oder in einer eigenen Methode.
 - b) Ergänzt euer Programm dahingehend, dass euer Roboter an seinen Ausgangsort zurückgeht, wenn er die Akkus aufgenommen hat. Ergänzt auch den PAP dahingehend.
3. Auch ein anderer Roboter möchte Akkus einsammeln. Allerdings weiß er nicht, wo die Reihe der Akkus anfängt. Aber er weiß, dass in „seiner“ Reihe auf jeden Fall welche liegen. Er weiß außerdem nicht, wie viele es sind. Natürlich ist er bestrebt, möglichst viele, maximal aber fünf, zu finden und mitzunehmen.



Implementiert Algorithmen, die das Problem mit folgenden Aufgaben lösen.

- a) Der Roboter soll fünf Akkus aufnehmen.
- b) Die Zählvariable soll mit 23 initialisiert werden (`int i=23`). Verändert die übrigen Teile der Zählschleife so, dass sie noch das gleiche Ergebnis wie die Lösung aus a) erzielt.
- c) Mit `i--` oder `i=i-1` wird die Zählvariable in der Aktualisierung jeweils um 1 reduziert. Erläutert, welche Auswirkungen das auf die Abbruchbedingung hat.

- d) Erweitert die Lösung aus a) so, dass der Roboter wieder zu seinem Ausgangspunkt zurückkehrt.
- e) Erweitert euren Algorithmus so, dass er auch funktioniert, wenn die Akkus nicht aufeinander folgen, sondern „leere Felder“ dazwischen sind.
- f) Mauern versperren den Weg, aber irgendwo in der Reihe befinden sich neben leeren Feldern auch noch Akkus.



- 4. Euer Roboter hat gute Laune und möchte aus Schrauben Muster legen.
 - a) Programmiert einen Algorithmus, mit dem ein Quadrat gelegt wird.
 - b) Programmiert einen Algorithmus, mit dem ein Rechteck gelegt wird.
 - c) Programmiert einen Algorithmus, mit dem eine eckige Spirale gelegt wird.
 - d) Entwickelt in Partnerarbeit ein Muster und einen Algorithmus in Pseudocode. Dann gebt den Entwurf an eine andere Gruppe. Diese soll dann mit eurem Pseudocode einen PAP erstellen und den Algorithmus programmieren.