

3 Wiederholung

3.1 Wiederholung mit fester Anzahl

Übung 3.1:

Karol soll vier Ziegel in einer Reihe legen. Erstelle ein Programm hierfür.

Wahrscheinlich wird Karol zuerst einen Stein hinlegen und dann einen Schritt gehen. Nun legt Karol einen weiteren Stein und geht wiederum einen Schritt. Diese Prozedur durchführt Karol insgesamt vier Mal. Den entsprechenden Algorithmus zu schreiben wird dir nicht schwer gefallen sein.

Hinlegen	1. Mal ausgeführt
Schritt	
Hinlegen	2. Mal ausgeführt
Schritt	
Hinlegen	3. Mal ausgeführt
Schritt	
Hinlegen	4. Mal ausgeführt
Schritt	

Nun soll Karol jedoch beispielsweise 80 Ziegelsteine in Reihe legen. Es wird recht mühselig, die Sequenz *Hinlegen - Schritt* 80-mal aufzuschreiben. Solche Wiederholungen kommen in Programmen sehr häufig vor. Deshalb gibt es hierfür eine spezielle Anweisung, die *Wiederholung mit fester Anzahl*.

```
wiederhole 4 mal
Hinlegen
Schritt
```

Dieser Vorschlag ist jedoch unklar formuliert. Was soll Karol nun wiederholen:

- a) Nur die Anweisung *Hinlegen* oder
- b) alle beiden Anweisungen *Hinlegen* und *Schritt*

Übung 3.2:

Überlege dir, was Karol wohl macht, wenn er

- a) nur die Anweisung *Hinlegen* viermal ausführt,
- b) alle beide Anweisungen *Hinlegen* und *Schritt* viermal ausführt.

Teste deine Überlegungen.

Du siehst nun ein, dass Wiederholungen eindeutig formuliert werden müssen. In Übung 3.1 erwartet Karol folgende Anweisungen:

```
wiederhole 4 mal
  Hinlegen
  Schritt
*wiederhole
```

Übung 3.3:

```
wiederhole 4 mal
  Hinlegen
*Wiederhole
Schritt
LinksDrehen
```

```
wiederhole 4 mal
  Hinlegen
  Schritt
*Wiederhole
LinksDrehen
```

```
wiederhole 4 mal
  Hinlegen
  Schritt
LinksDrehen
*Wiederhole
```

Oben stehen drei verschiedene Wiederholungen. Überlege zuerst, was Karol wohl jedes Mal machen wird. Anschließend solltest du deine Überlegungen überprüfen.

Die Einrückungen sind zwar nicht notwendig, aber sie verbessern die Lesbarkeit. Die Programmierumgebung hilft dir bei der Formatierung von Anweisungen unter dem Menüpunkt *Bearbeiten* > *Formatieren* bzw. *Str-F*.

Die Programme können auch in einer grafischen Notation, dem sogenannten Struktogramm dargestellt werden. Die Programmierumgebung zeigt dir unter dem Menüpunkt *Struktogramm* > *Anzeigen* die grafische Darstellung deines Programms:



Abbildung 3.1: Struktogramme der Programme in Übung 3.3

In einem Struktogramm kannst du sehr gut erkennen, welche Sequenz wiederholt werden soll

Übung 3.4

Erzeuge die in Abbildung 3.2 dargestellte Welt (im Direktmodus). Karol hat nun die Aufgabe, um diese Säule einen Weg aus Steinen zu legen. Die Endposition ist in Abbildung 3.3 dargestellt.

- Versuche diese Aufgabe zuerst im Direktmodus zu erfüllen und notiere die Anweisungen.
- Erstelle nun ein Programm ohne Wiederholung, so dass Karol den Weg um die Säule legt.

- c) Überlege, welche Sequenz 4-mal wiederholt wird. Erstelle nun das Programm unter Verwendung der Kontrollstruktur *Wiederhole ... mal*.

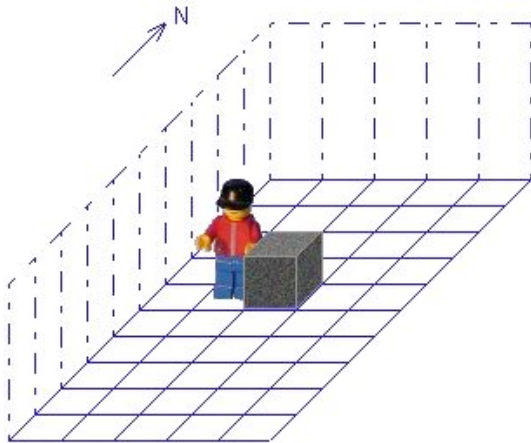


Abbildung 3.2 : Eine Säule
(Anfangszustand)

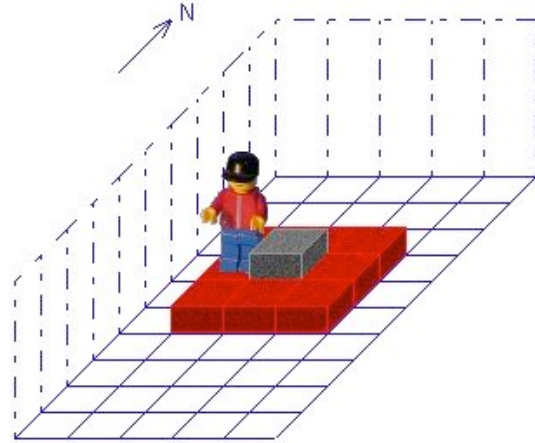


Abbildung 3.3: Weg um die Säule
(Endzustand)

Wahrscheinlich wird Karol zuerst einen Stein legen, dann einen Schritt gehen und sich links herum drehen. Nun legt Karol einen weiteren Stein hin und geht wieder einen Schritt nach vorne. Karol hat jetzt also die in Abbildung 3.4 dargestellte Position erreicht und somit den ersten Bauabschnitt fertig gestellt.

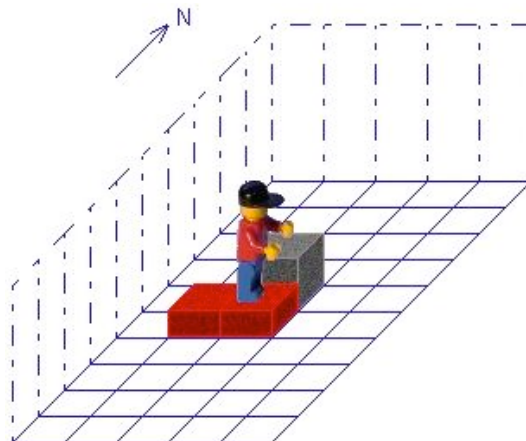


Abbildung 3.4: Erster Bauabschnitt

Diese Sequenz der Anweisungen muss insgesamt 4-mal abgearbeitet werden. Karol muss also immer wieder denselben Bewegungsablauf ausführen. Mit *kopieren und einfügen* könnte man also schnell das Programm erstellen. Sinnvoll ist es jedoch, die Kontrollstruktur *Wiederhole ... mal* zu verwenden. Hierdurch wird der Quelltext kürzer und übersichtlich. Diese

Wiederholungsanweisung kannst du wieder über das Kontextmenü (rechte Maustaste) aufrufen oder direkt eingeben.

Die Abbildung 3.5 zeigt die Lösung des Programms. Ich habe hier gleichzeitig Kommentare eingefügt, die das Lesen des Programms vereinfachen.

```

Programm
  wiederhole 4 mal
    //Einen Bauabschnitt erstellen
    Hinlegen
    Schritt
    LinksDrehen
    Hinlegen
    Schritt
  *wiederhole
*Programm

```

Abbildung 3.5: Quelltext des Programms *Saeule01*

Übung 3.5:

Rufe den Menüpunkt *Struktogramm* > *Anzeigen* auf. Erkläre die Darstellung des Struktogramms.

Übung 3.6:

Nun speichere den Quelltext als *Saeule01* in einen geeigneten Ordner. Verwende dazu den Menüpunkt *Menü* > *Speichern* oder die Schaltfläche *Programm speichern*.

Erzeuge wieder die Ausgangsposition wie in Abbildung 1.5. Speichere diese Welt als *Saeule01*. Verwende dazu den Menüpunkt *Welt* > *Welt speichern* oder die Schaltfläche *Welt speichern*.

Abbildung 3.6 zeigt dir noch einmal die Kontrollstruktur *wiederhole ... mal* in der allgemeinen Form.



Abbildung 3.6: Die Kontrollstruktur *wiederhole ... mal*

Die Anweisungen im Wiederholungsteil werden nacheinander mehrfach ausgeführt. (entsprechend der angegebenen Anzahl)

3.2 Wiederholen von Wiederholungen

Nun wird Karol dasselbe Problem lösen. Um eine Säule soll er einen Weg aus Ziegelsteinen legen. Allerdings zeigt Abbildung 3.7, dass Karol nun an einer anderen Position beginnt. Diese neue Ausgangsposition hat weit reichende Folgen in der Programmstruktur.

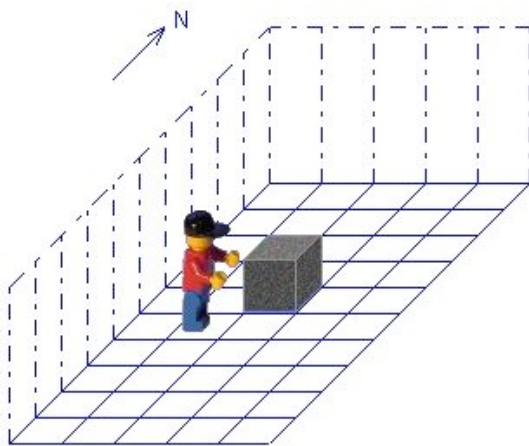


Abbildung 3.7: Eine Säule
(Anfangszustand)

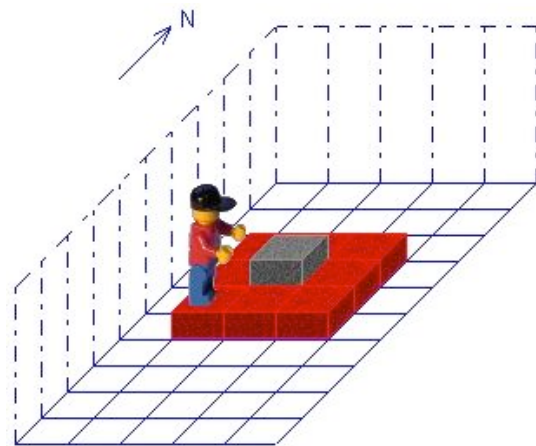


Abbildung 3.8: Weg um die Säule
(Endzustand)

Übung 3.7

Erzeuge die in Abbildung 3.7 dargestellte Welt (im Direktmodus). Karol hat nun die Aufgabe, um diese Säule einen Weg aus Steinen zu legen. Die Endposition ist in Abbildung 3.8 dargestellt.

- Überlege zuerst, was Karol gemacht hat und wo er nun steht, wenn er die rechts abgebildete Sequenz abgearbeitet hat.
 - Hinlegen
 - Schritt
 - Hinlegen
 - Schritt
 - LinksDrehen
- Vervollständige dein Programm, so dass Karol den Weg fertig stellen kann, wie in Abbildung 3.8 dargestellt.
- Finde innerhalb der Wiederholung mehrmals vorkommende Teile dieser Sequenz. Ersetze diese durch eine Wiederholung innerhalb der Wiederholung (geschachtelte Wiederholung)
- Speichere nun dein Programm unter dem Namen *Saeule02*.

Übung 3.8:

Karol soll einen doppelt so hohen Ring um die Säule bauen (vgl. Abbildung 3.9). Verwende einen der beiden Anfangszustände und erstelle das Programm *Saeule03*.

Übung 3.9:

Karol soll einen doppelt so breiten Ring um die Säule bauen (vgl. Abbildung 3.10). Verwende einen der beiden Anfangszustände und erstelle das Programm *Saeule04*.

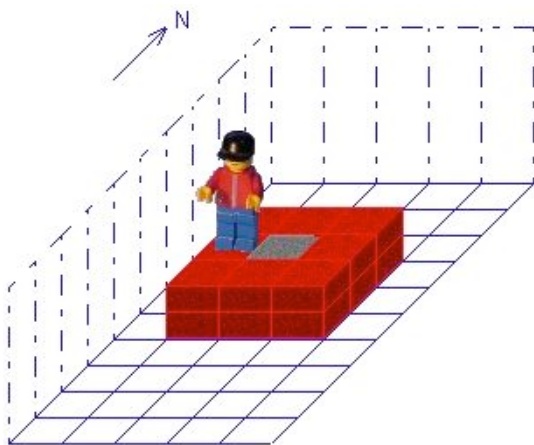


Abbildung 3.9: Programm *Saeule03*

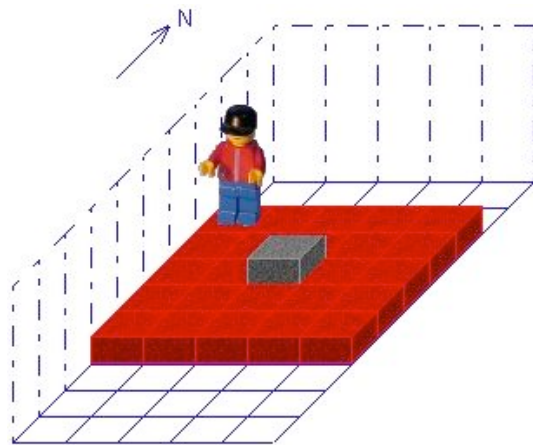


Abbildung 3.10: Programm *Saeule04*

Übung 3.10:

- a) Karol soll eine Ziegelreihe der Länge 5 links neben sich hinlegen. (*ReiheLinks*)
- b) Karol soll ein Quadrat legen. (*Quadrat01*)
- c) Karol soll eine 3 Ziegel hohe quadratische Mauer bauen. (*Quadrat02*)
- d) Karol soll eine 6 Ziegel lange, 4 Ziegel hohe Mauer bauen. (*Mauer01*)
- e) Karol soll zwei solche Mauern parallel nebeneinander bauen mit einem Feld Abstand. (*Mauer02*)

3.3 Wiederholung mit Anfangsbedingung

Karol ist nun schon recht eigenständig geworden. Mit den bisher erlernten Methoden kann er bereits bestimmte Vorgänge ausführen. Nun soll er zwischen WAHR und FALSCH unterscheiden lernen.

Übung 3.11:

Stelle Karol auf eine beliebige Startposition und gebe ihm den Auftrag: *Gehe bis zur Wand!*

Warum kann Karol diesen Auftrag nicht mit der Kontrollstruktur *Wiederhole ... mal* durchführen? Was muss Karol in Erfahrung bringen können?

Da du Karol auf eine beliebige Startposition setzen darfst, ändert sich jedes Mal auch die Entfernung zur Wand. Natürlich kannst du Karols Entfernung zur Wand an Hand der Quadrate abzählen und entsprechend oft Schritte mit Hilfe der Kontrollstruktur *Wiederhole ... mal* gehen. Jedoch versagt die Möglichkeit bei einer geänderten Zimmergröße oder Ausgangsstellung. Um unabhängig von der Ausgangsposition zu sein, muss Karol eine Methode kennen, die eine Anfrage, ob er noch weiter gehen kann, mit WAHR oder FALSCH beantwortet. Einen Aufruf solcher Methoden bezeichnet man als *Bedingung*. Im Kontextmenü (rechte Maustaste) wirst du einige solcher Bedingungen finden, beispielsweise *IstWand*, *NichtIstWand*, *IstZiegel*, *NchtIstZiegel*, usw.

Methode	Bedeutung
IstWand	Liefert wahr, wenn vor Karol eine Wand ist, und falsch, wenn vor ihm keine Wand ist
IstZiegel	Liefert wahr, wenn auf dem Feld vor Karol ein Ziegel liegt, und falsch, wenn auf dem Feld vor ihm kein Ziegel liegt.

Übung 3.12:

Formuliere die Bedeutung der Bedingungen *NichtIstWand* und *NchtIstZiegel*.

Karol wiederholt eine Sequenz, solange die angegebene Bedingung wahr ist. Damit Karol bis zur nächsten Wand gehen kann, musst du also die Bedingung *NichtIstWand* verwenden. Solange diese Bedingung WAHR liefert, darf Karol einen Schritt weitergehen.

```

Programm
  wiederhole solange NichtIstWand
    Schritt
  *wiederhole
*Programm

```



Abbildung 3.11: Quelltext und Struktogramm

Abbildung 3.12 zeigt dir noch einmal die Kontrollstruktur *wiederhole solange* in der allgemeinen Form.

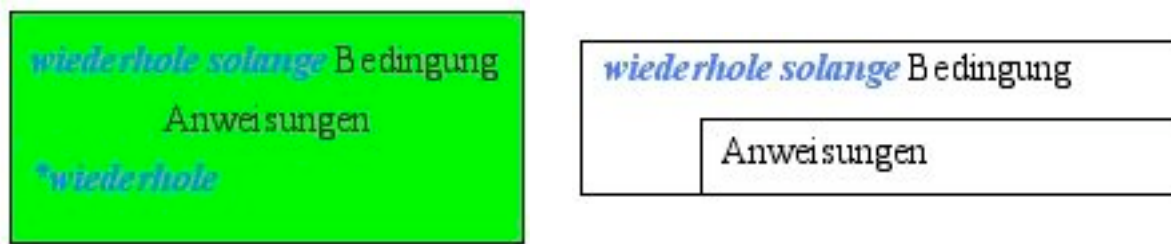


Abbildung 3.12: Die Kontrollstruktur *wiederhole solange*

Die Anweisungen im Wiederholungsteil werden so oft wiederholt, solange die Bedingung WAHR ergibt. Die Überprüfung der Bedingung erfolgt am Anfang jeder Wiederholung.

Bemerkung:

Dabei kann es vorkommen, dass die Bedingung schon bei der ersten Prüfung nicht erfüllt ist und somit die zu wiederholende Sequenz überhaupt nicht abgearbeitet wird.

Es ist auch möglich, dass die Bedingung immer erfüllt ist. Dann endet die Wiederholung zumindest theoretisch nie.

Übung 3.12:

Karol hat die Aufgabe, entlang seiner Westwand eine Leiste aus Ziegelsteinen zu legen. (*Leiste01*)

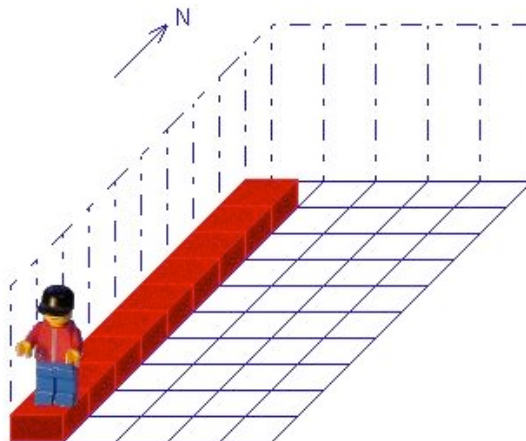


Abbildung 3.13: Programm *Leiste01*

Übung 3.13:

Karol hat nun die Aufgabe, entlang der vier Wände seines Zimmers eine Fußbodenleiste zu legen. Du sollst diese Aufgabe wieder in Bauabschnitte zerlegen. Abbildung 3.13 zeigt, dass zuerst nur entlang einer Wand die Fußbodenleiste gelegt wird. (*Leiste02*)

Übung 3.14:

- Karol soll parallel zu einer Wand durch sein Zimmer laufen. Verbaue dabei seinen Weg mit einem Quader. (*Weg01*)
- Karol soll dabei nur auf jedes zweite Feld einen Ziegel legen. (*Weg02*)
- Karol soll seine Welt mit einer Mauer (z.B. 3 Ziegelreihen aufeinander) umgeben. (*Weg03*)
- Karol wurde wie in Abbildung 3.14 eingemauert. Er soll sich befreien, indem er alle Ziegelsteine abbaut. (*Kerker01*)
- Die Höhe der Seitenteile ist nun verschieden (vgl. Abbildung 3.15). Karol soll wiederum alle Ziegelsteine abbauen. (*Kerker02*)

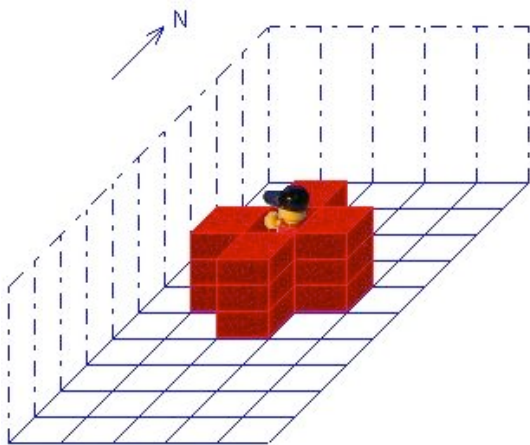


Abbildung 3.14: Programm *Kerker01*

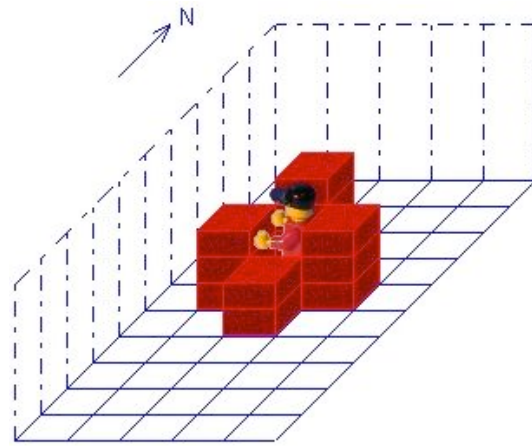


Abbildung 3.15: Programm *Kerker02*

3.4 Wiederholung mit Endbedingung

Bemerkung:

Die Wiederholung mit Endbedingung muss nicht unbedingt im Unterricht behandelt werden. Manchmal lässt sich jedoch hiermit die Lösung einer Aufgabenstellung leichter finden.

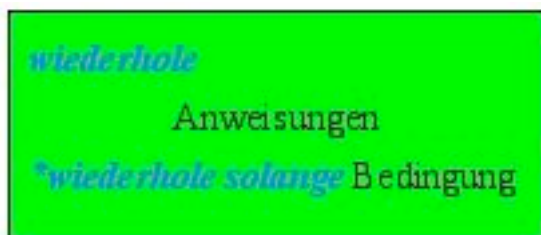


Abbildung 3.16: Die Kontrollstruktur **wiederhole solange*

Die Anweisungen im Wiederholungsteil werden so oft wiederholt, solange die Bedingung WAHR ergibt. Die Überprüfung der Bedingung erfolgt am Ende jeder Wiederholung.

Übung 3.15:

Karol soll bis zur Wand laufen. Formuliere den Unterschied zwischen beiden Varianten und überprüfe deine Überlegungen.

```
wiederhole solange NichtIstWand
schritt
*solange
```

```
wiederhole
schritt
*solange NichtIstWand
```

3.5 Weitere Wiederholungen**Bemerkung:**

Auch diese Wiederholungen müssen nicht unbedingt im Unterricht behandelt werden. Manchmal lässt sich jedoch hiermit die Lösung einer Aufgabenstellung leichter finden.



Abbildung 3.17: Das Kontextmenü *Wiederholungen*

Das Kontextmenü *Wiederholungen* zeigt noch weitere Arten von Wiederholungen, die manchmal recht nützlich sind. Ich überlasse es dir, diese Arten auszuprobieren und eigene Programme zu entwerfen.

Übung 3.16:

- a) Erstelle eine Welt der Breite 6 und der Länge 10. In dieser Welt soll Karol ein Schachbrettmuster legen. (*Schachbrett01*)
- b) Erstelle eine Welt der Breite 6 und der Länge 9. In dieser Welt soll Karol ein Schachbrettmuster legen. (*Schachbrett02*)
- c) ... weitere Aufgaben ...

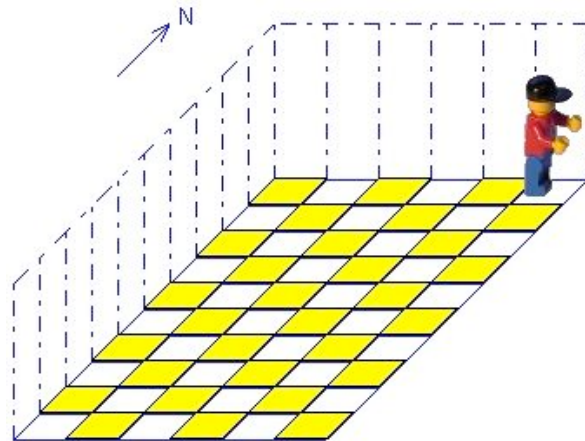


Abbildung 3.18: Programm *Schachbrett*