CARL VON OSSIETZKY universität OLDENBURG

Teil

Imperative Programmierung

Unterrichtseinheit 7

Kontrollstrukturen (Hamster-Modell)

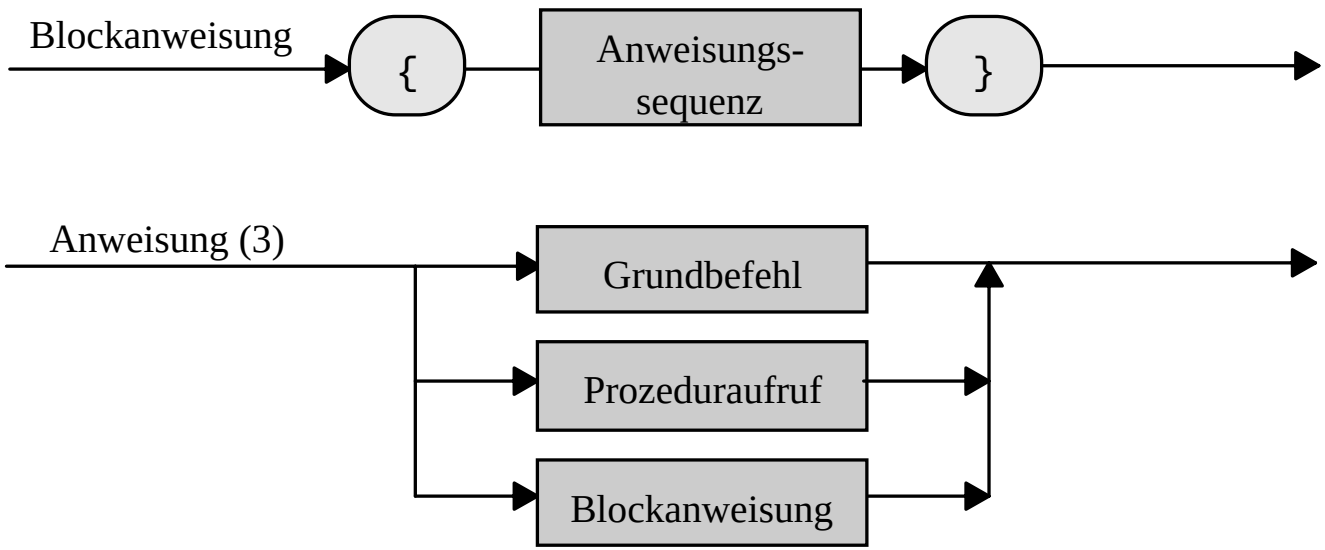
Dr. Dietrich Boles

Gliederung

- Blockanweisung
- Leeranweisung
- Motivation Kontrollstrukturen
- Testbefehle
- Boolesche Ausdrücke
- Bedingte Anweisung
- Alternativanweisung
- Wiederholungsanweisung
- Endlosschleife
- Beispiele
- Codekonventionen
- Zusammenfassung

Blockanweisung

Syntax:

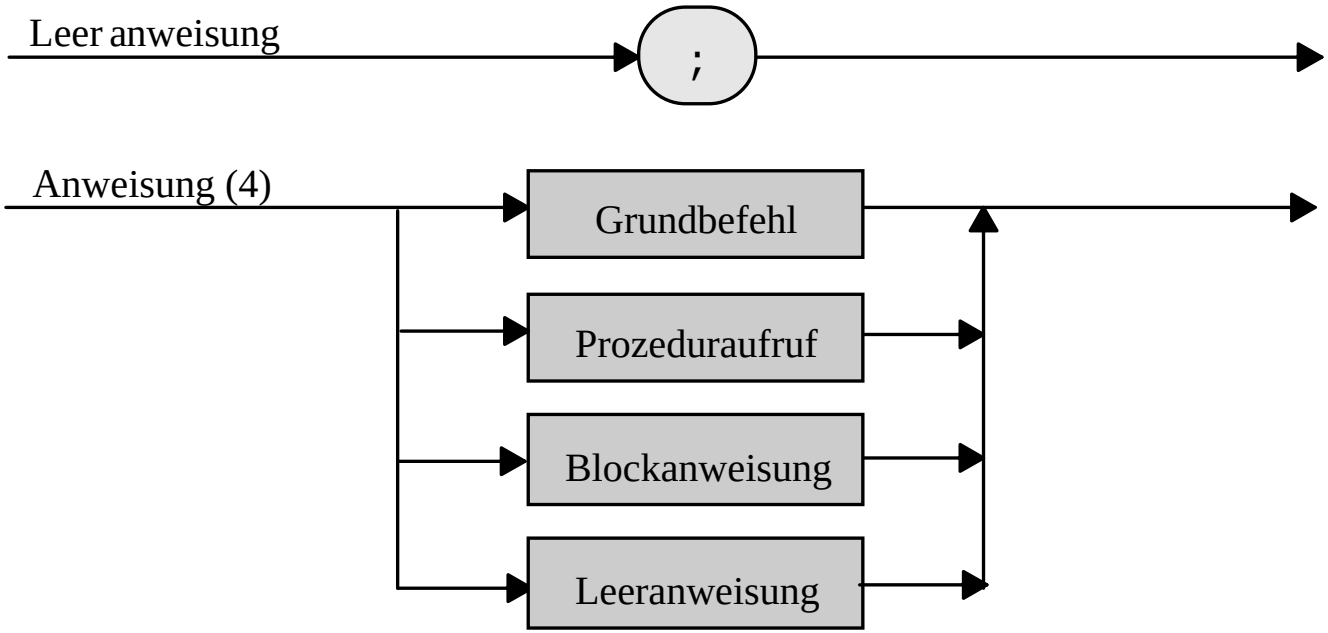


Semantik:

Zusammenfassung mehrerer Anweisungen zu einer (zusammengesetzten) Anweisung

Leeranweisung

Syntax:



Semantik:

keine Auswirkungen auf den Programmablauf

Motivation

- Nur wenn der Hamster Körner im Maul hat, soll er eines davon ablegen
- Nur wenn die Kachel vor dem Hamster frei ist, soll er nach vorne gehen; andernfalls soll er sich linksum drehen.

- Solange der Hamster nicht vor einer Mauer steht, soll er wiederholt eine Kachel nach vorne gehen.
- Solange sich noch Körner auf der Kachel des Hamsters befinden, soll er wiederholt ein Korn aufnehmen.

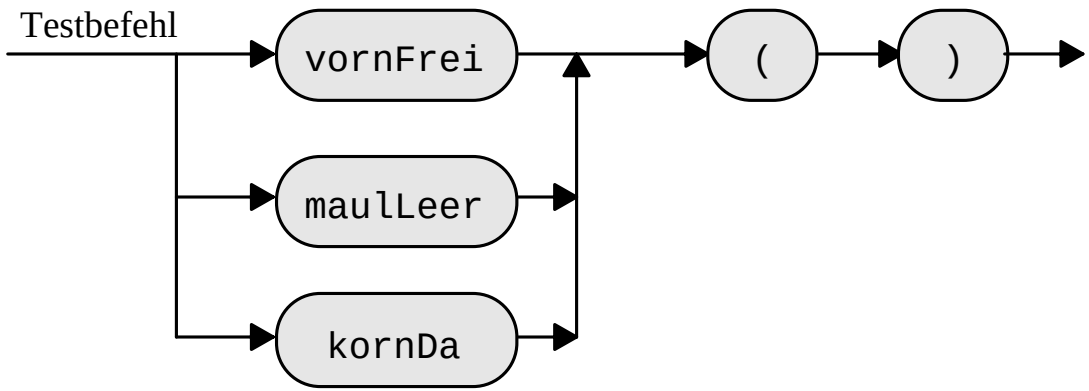
- Solange der Hamster nicht vor einer Mauer steht und er noch Körner im Maul hat, soll er wiederholt eine Kachel nach vorne gehen und jeweils ein Korn ablegen.

- Hamster-Programme für eine bestimmte Klasse von Territorien!

Testbefehle

Drei Testbefehle:

Syntax:



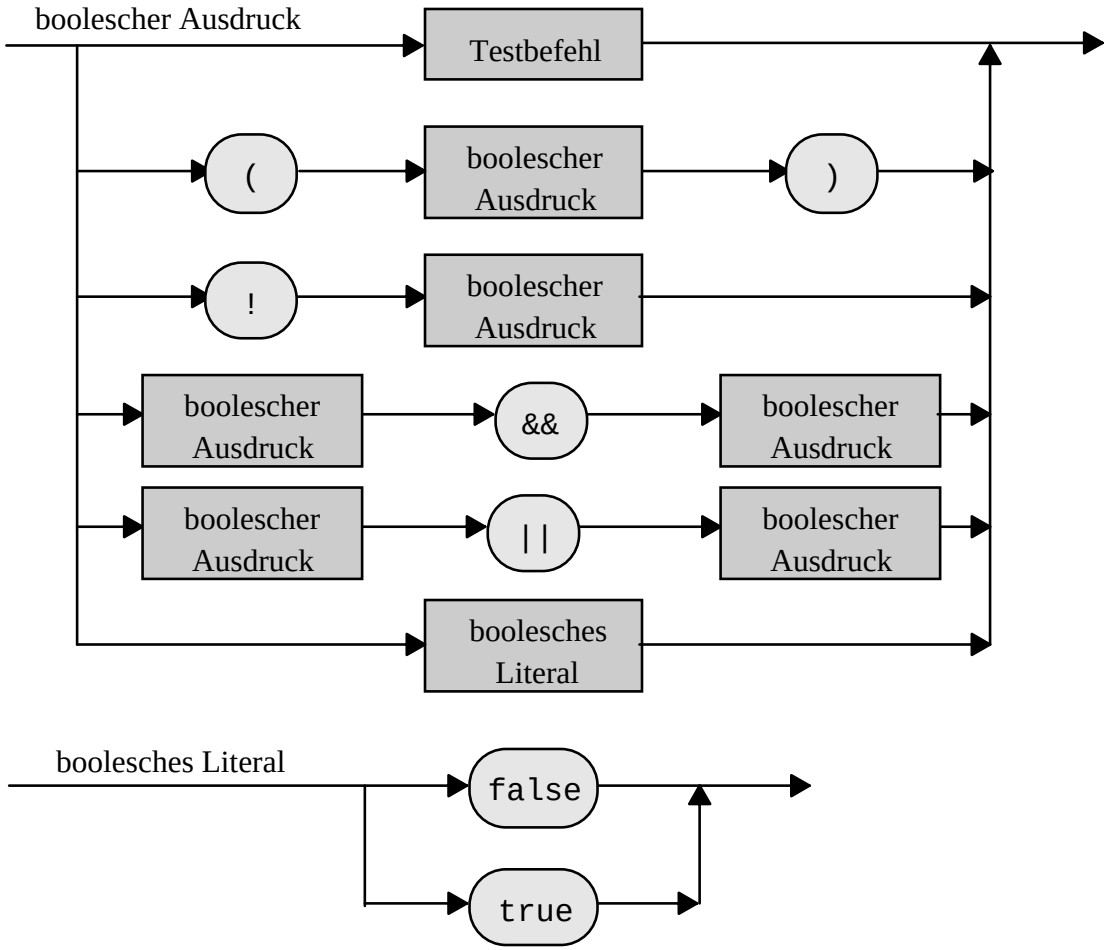
Semantik:

Lieferung von booleschen Werten (**`true`**, **`false`**):

`vornFrei()`	ist das Feld vor dem Hamster blockiert?
`maulLeer()`	ist das Maul des Hamsters leer?
`kornDa()`	liegt ein Korn auf der aktuellen Kachel?

Boolesche Ausdrücke

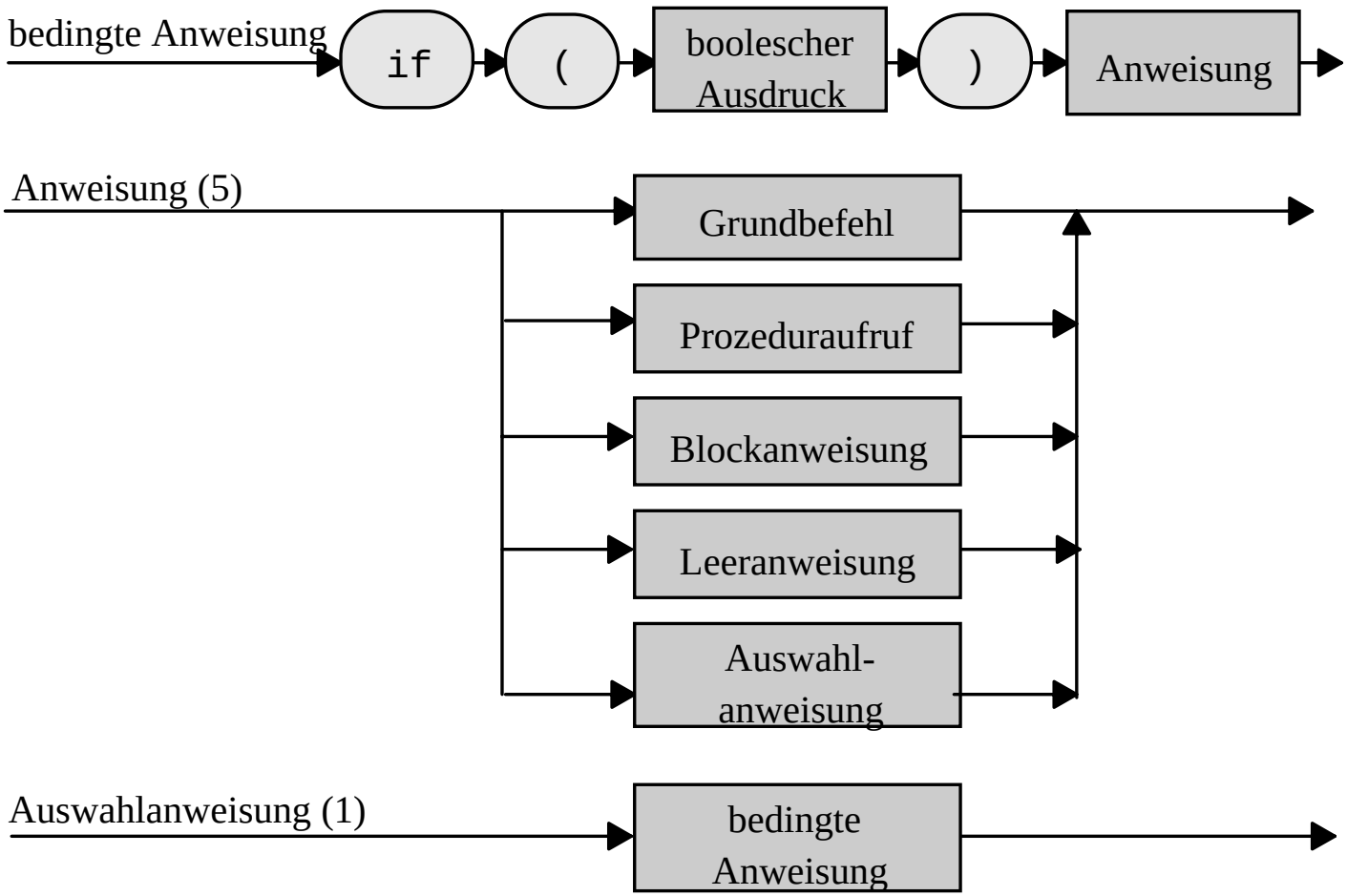
Syntax:



Semantik: siehe Aussagenlogik

Bedingte Anweisung (1)

Syntax:



Bedingte Anweisung (2)

Semantik:

1. Werte den booleschen Ausdruck aus.
2. Nur falls der Ausdruck den Wert **true** liefert, führe die true-Anweisung aus.

Beispiele:

```
if (vornFrei())
    vor();
```

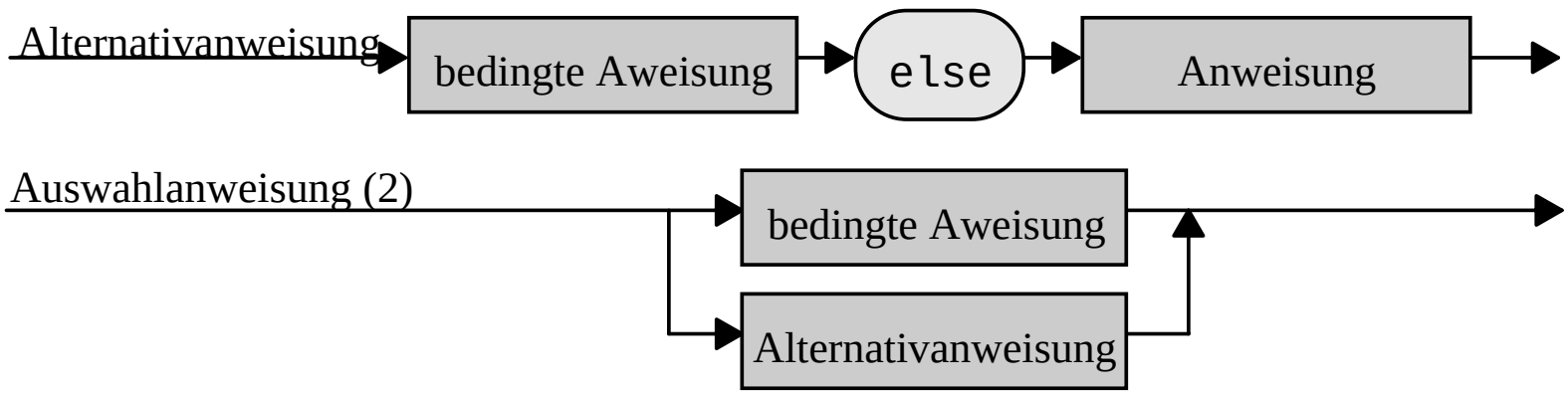
```
if (!maulLeer()) {
    gib();
}
```

```
if (kornDa() && vornFrei()) {
    nimm();
    vor();
}
```

```
if (kornDa())
    if (vornFrei()) {
        nimm();
        vor();
    }
```

Alternativanweisung (1)

Syntax:



Semantik:

1. Werte den booleschen Ausdruck aus.
2. Falls der Ausdruck den Wert **true** liefert, führe die true-Anweisung aus.
3. Andernfalls führe die false-Anweisung aus.

Alternativanweisung (2)

Beispiele:

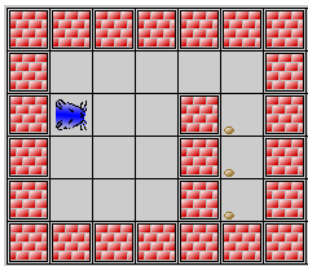
```
if (vornFrei()) {
    vor();
} else {
    linksUm();
}
```

```
if (maulLeer())
    ;
else
    gib();
linksUm();
```

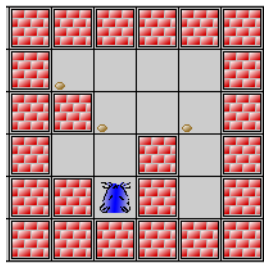
```
if (vornFrei())
    vor();
else if (kornDa())
    nimm();
else if (!maulLeer())
    gib();
else
    linksUm();
```

Wiederholungsanweisung (1)

Motivation: Der Hamster soll bis zur nächsten Wand laufen.



```
void main() {
    vor();
    vor();
}
```

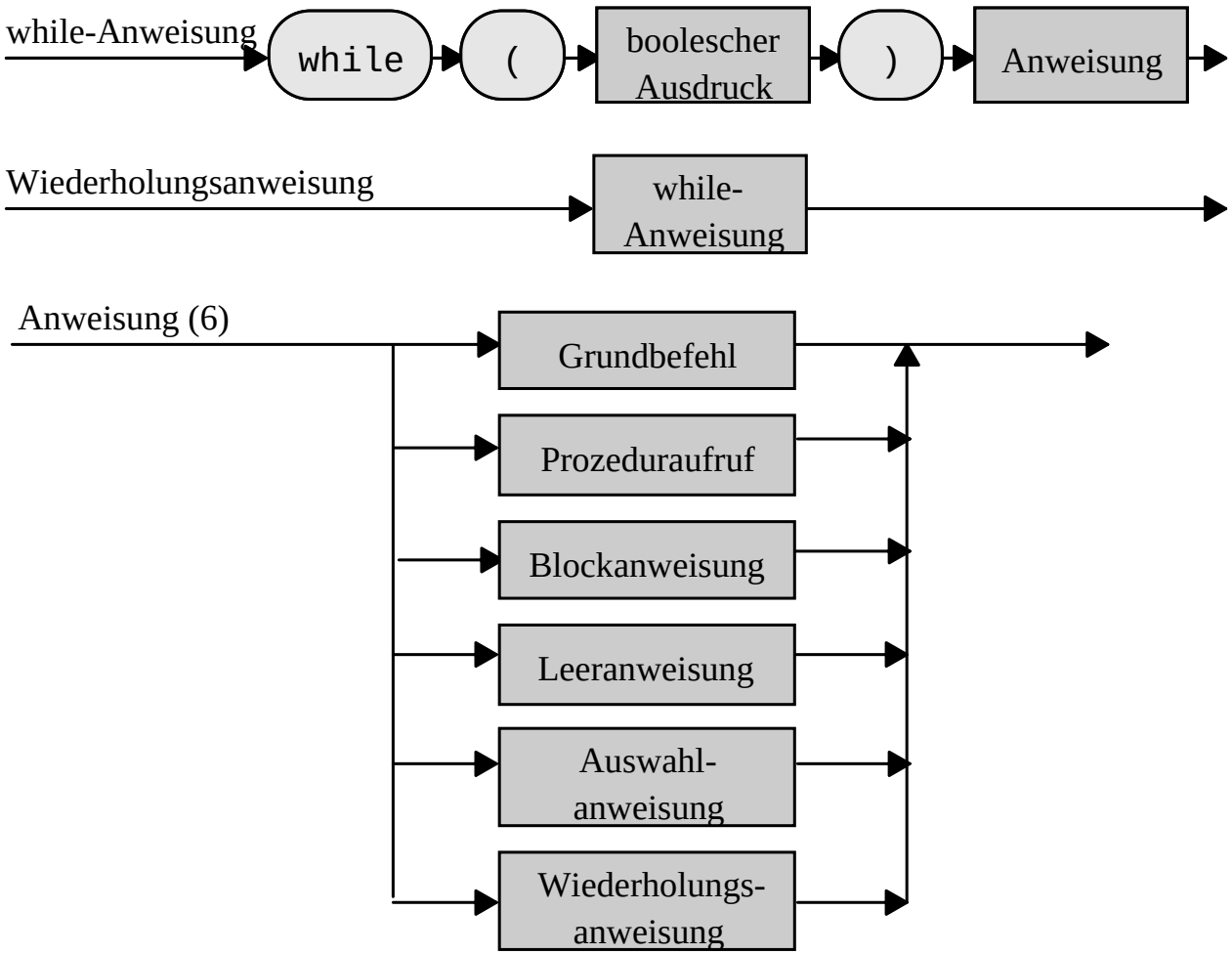


```
void main() {
    vor();
    vor();
    vor();
}
```

Allgemeingültig: `solange vornFrei(): vor();`

Wiederholungsanweisung (2)

Syntax:



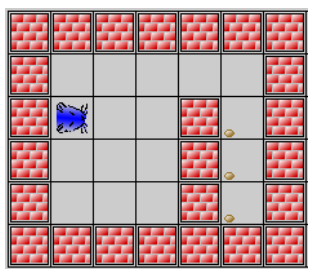
Wiederholungsanweisung (3)

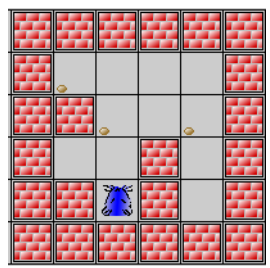
Semantik:

1. Werte den booleschen Ausdruck (*Schleifenbedingung*) aus.
2. Falls die Schleifenbedingung den Wert **false** liefert, beende die Wiederholungsanweisung.
3. Falls die Schleifenbedingung den Wert **true** liefert, führe die Anweisung (Iterationsanweisung) aus und fahre bei (1.) fort.

Beispiel:

```
void main() {
    while (vornFrei()) {
        vor();
    }
}
```

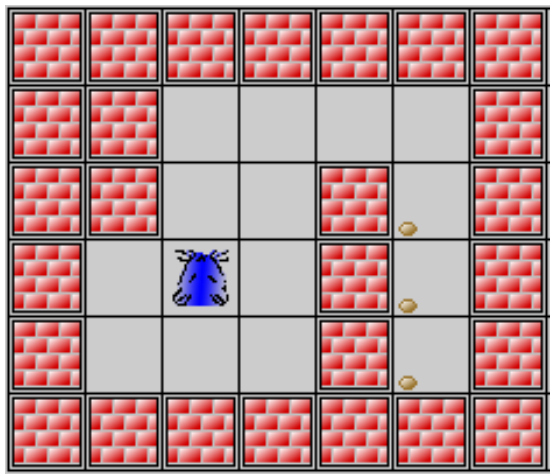




Demo

Endlosschleife

```
void main() {
    while (vornFrei()) {
        linksUm();
    }
}
```



Beispiel 1

Aufgabe: Landschaft beliebig! Der Hamster soll bis zur nächsten Wand laufen und dabei alle Körner fressen.

```
void laufeBisZurNaechstenWandUndSammle() {
    sammle();
    while (vornFrei()) {
        vor();
        sammle();
    }
}

void sammle() {
    while (kornDa()) {
        nimm();
    }
}
```

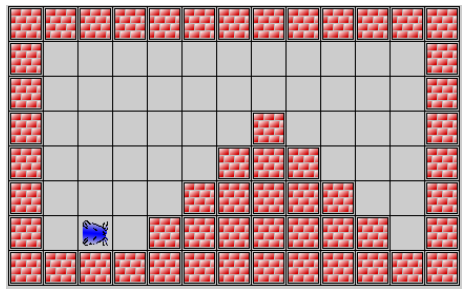
Demo

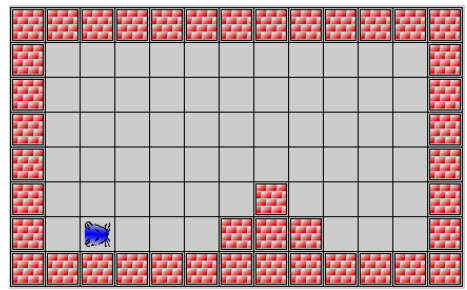
Beispiel 2

Aufgabe:

Der Hamster steht vor einem regelmäßigen Berg unbekannter Höhe. Er soll den Gipfel erklimmen.

Typische Landschaften:





Programm:

```
void main() {
    laufeZumBerg();
    erklimmeGipfel();
}
void laufeZumBerg() {
    while (vornFrei())
        vor();
}
void erklimmeGipfel() {
    while (!vornFrei()) {
        erklimmeEineStufe();
    }
}
```

```
void erklimmeEineStufe() {
    linksUm();
    vor();
    rechtsUm();
    vor();
}

void rechtsUm() {
    linksUm();
    linksUm();
    linksUm();
}
```

Demo

Codekonventionen

- Hinter `if` und `while` ein Leerzeichen
- Vor und hinter duadischen Operatoren ein Leerzeichen
- Für true-, false- und Iterationsanweisung möglichst immer die Blockanweisung verwenden
- falls Blockanweisung:
  - `)` und `{` in dieselbe Zeile, durch Leerzeichen getrennt
  - `}` unter `i` von `if` bzw. `w` von `while`
- innere Anweisungen um 4 Spalten einrücken
- Alternativanweisung: `} else {` (in eine Zeile)
- geschachtelte Schleifen vermeiden (→ Prozeduren)

Zusammenfassung

- Kontrollstruktur: Schema, welches die Reihenfolge der Abarbeitung von Anweisungen festlegt
  - Sequenz: nacheinander
  - if-Anweisung: Ausführung von Anweisungen abhängig von einer Bedingung
  - while-Anweisung: wiederholte Ausführung von Anweisungen, abhängig von einer Bedingung