

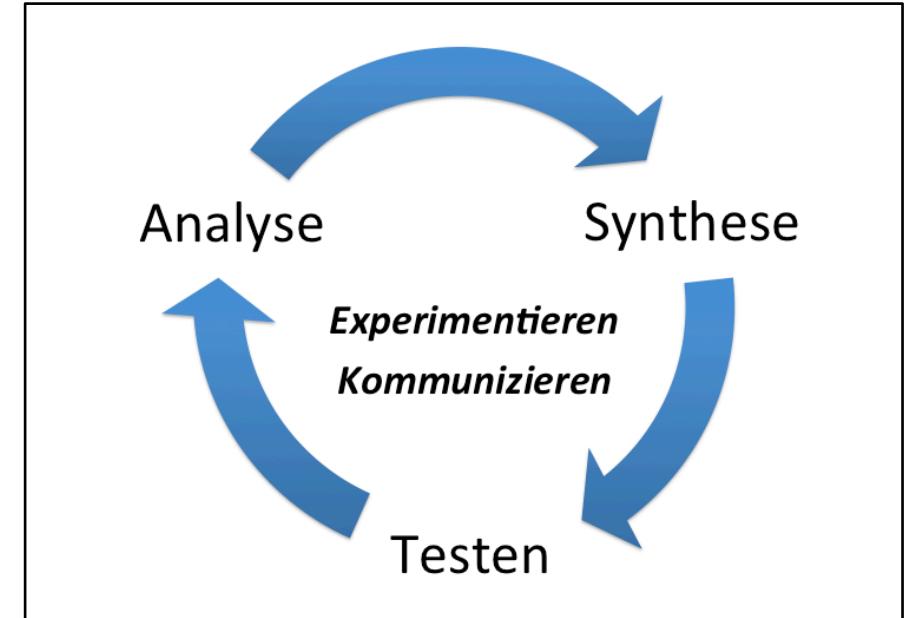
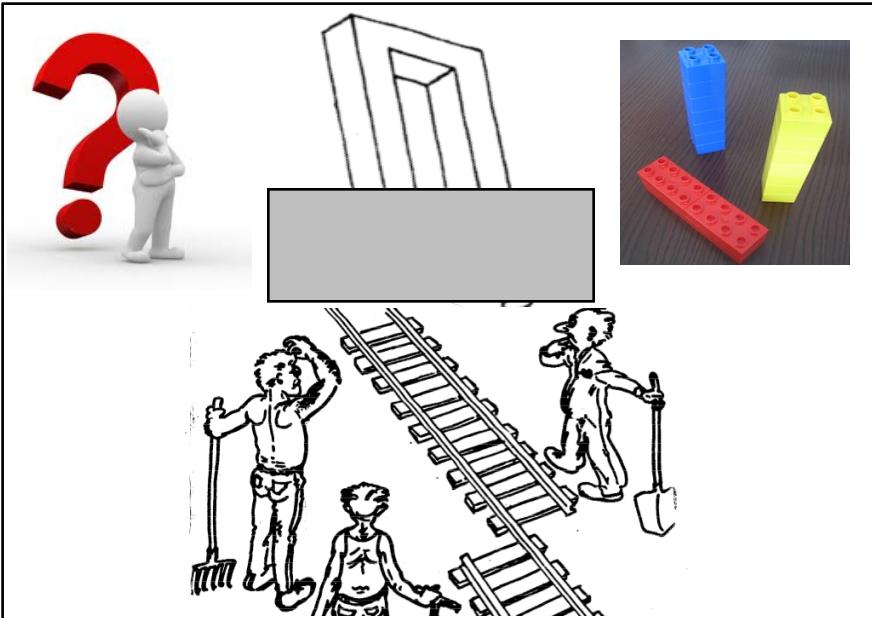
Software-Entwicklung 1

V02: Algorithmisches Denken



Prof. Maalej & Team @maalejw

Was haben wir schon gemacht?



Software-Entwicklung: Einführung

Algorithmus: Lösungsbeschreibung in Einzelschritten



3

Karel the Robot

```

/*
1 F1 = moveForward();
2 F2 = turnLeft();
3 F3 = turnAround();
4 F4 = turnRight();
5 F5 = pickBeeper();
6 F6 = dropBeeper();
*/
7
8
9 void karelsFirstProgram()
10 {
11     // your code here
12 }
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

```

Da war noch was??

Übungsbetrieb Don'ts

Übungsbetrieb Do's

Status der 1. Übungswoche



Zeit	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
Vor mittag	Gruppe 1 Erfüllt: 68%	Gruppe 3 Erfüllt: 73%	Gruppe 5 Erfüllt: 82%	Gruppe 6 Erfüllt: 70%	Gruppe 8 Erfüllt: 84%
Nach mittag	Gruppe 2 Erfüllt: 73%	Gruppe 4 Erfüllt: 73%	Vorlesung	Gruppe 7 Erfüllt: 70%	

Wichtige Hinweise

- Vorher mindestens **Kernbegriffe** lesen
- **Scheinkriterien** lesen
- **TutorInnen** bei Verständnisfragen einbinden
- Nachabnahmen
 - In der nächsten Übungswoche vorbereitet **am Anfang**
 - In der **selben Übungswoche** an einem anderen Termin



Fragen: organisatorisch oder inhaltlich



- An mich vor oder nach der Vorlesung
- An: se-orga@informatik.uni-hamburg.de
- Marlo Häring (D-212) und Mathias Ellmann (D-209)

Überblick

1

Kontrollstrukturen

2

Bedingungen und bedingte Schleifen

3

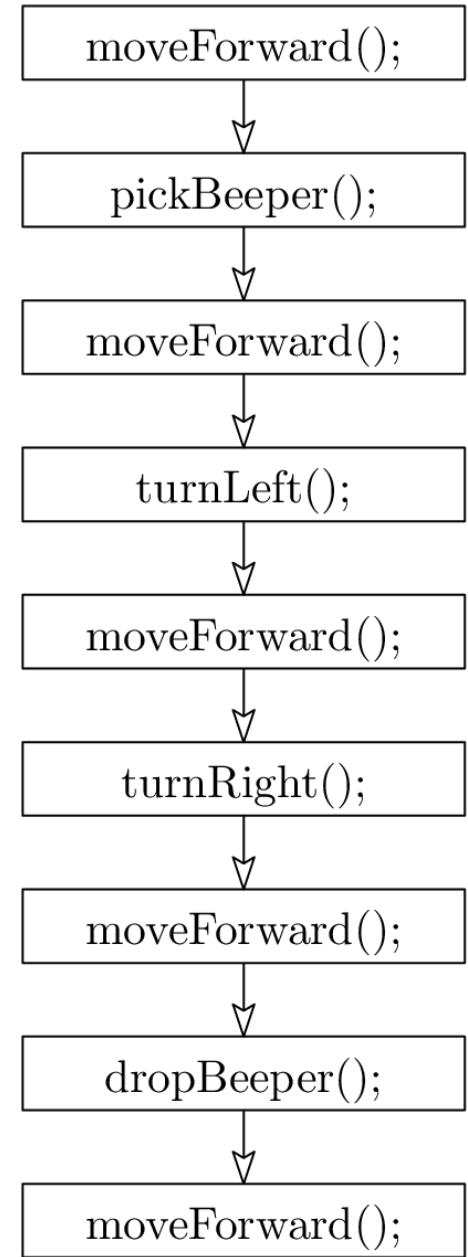
Binärsystem

Kontrollstrukturen

- In welcher Reihenfolge werden Karel's elementare Befehle abgearbeitet?
- Die Abarbeitungsreihenfolge wird durch 5 **Kontrollstrukturen** beeinflusst:
 1. Sequenz
 2. Aufruf zusammengesetzter Befehle
 3. Zählschleife
 4. Fallunterscheidung
 5. Bedingte Schleife (Neu)

Sequenz

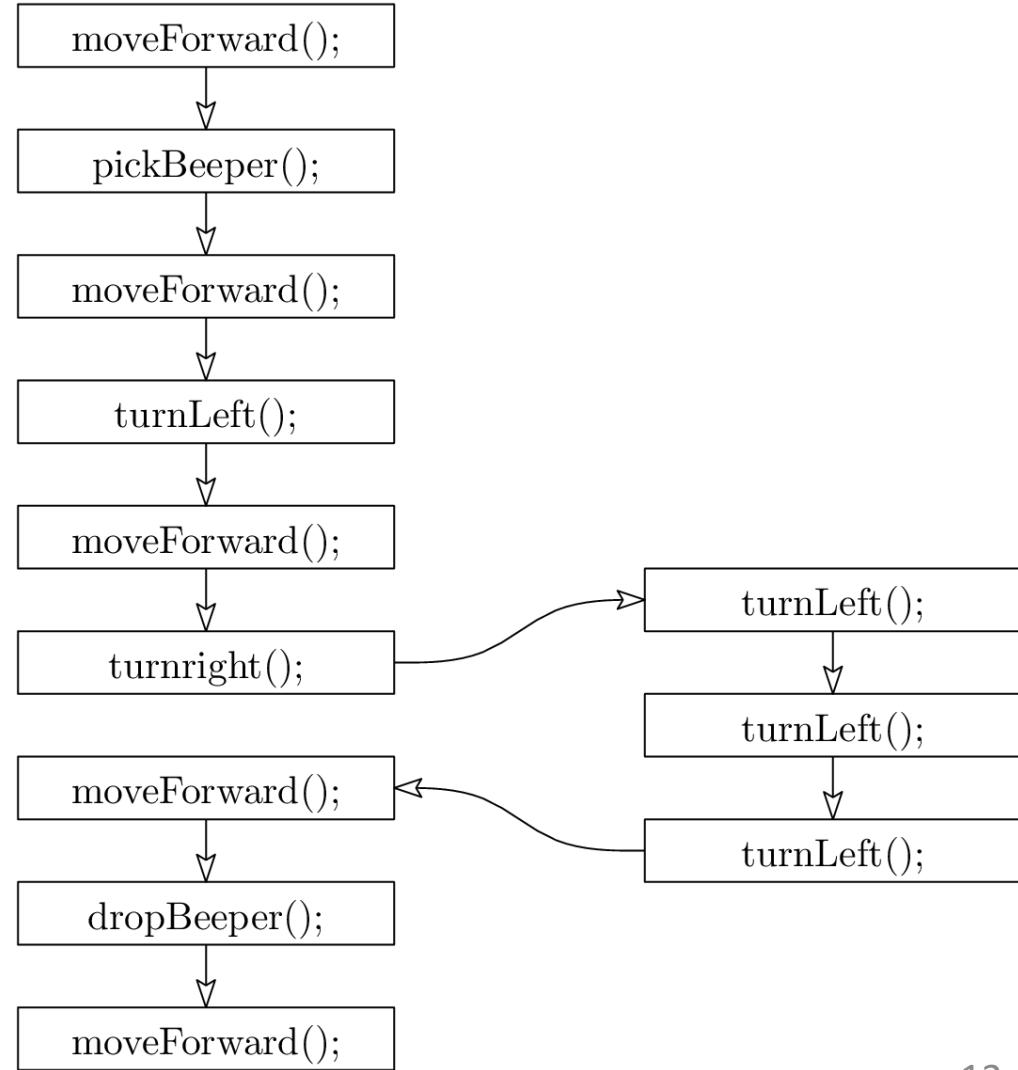
```
void karelsFirstProgram()
{
    moveForward();
    pickBeeper();
    moveForward();
    turnLeft();
    moveForward();
    turnRight();
    moveForward();
    dropBeeper();
    moveForward();
}
```



Aufruf zusammengesetzter Befehle

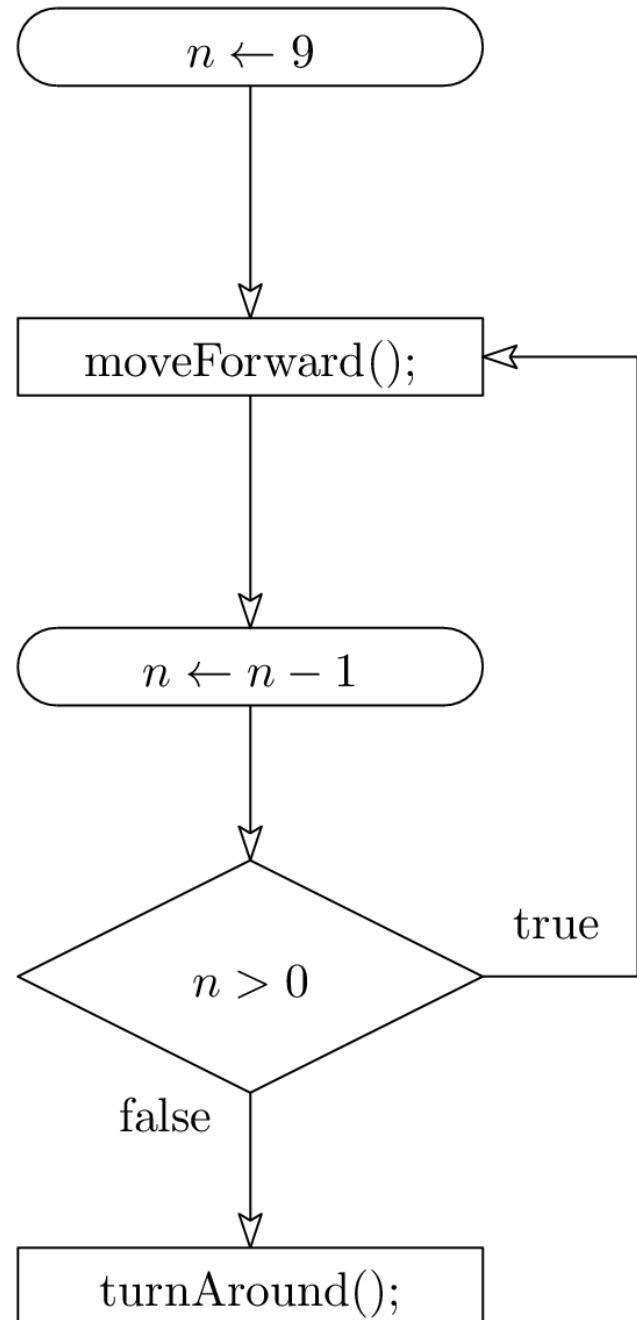
```
void karelsFirstProgram()
{
    moveForward();
    pickBeeper();
    moveForward();
    turnLeft();
    moveForward();
    turnright();
    moveForward();
    dropBeeper();
    moveForward();
}

void turnright()
{
    turnLeft();
    turnLeft();
    turnLeft();
}
```



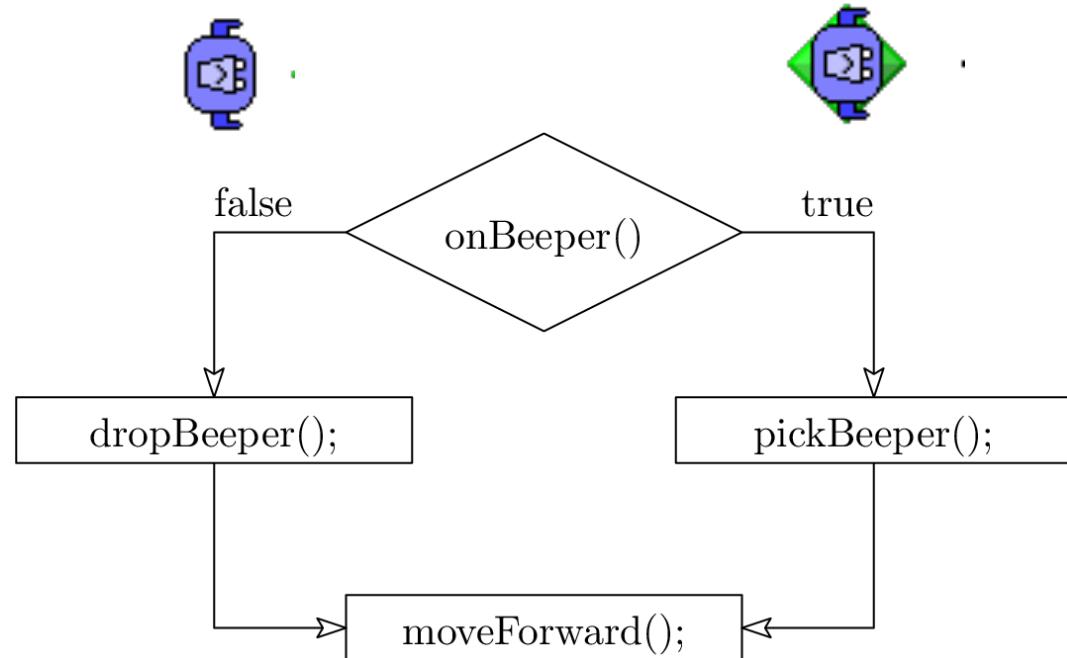
Zählschleife

```
repeat (9)
{
    moveForward();
}
turnAround();
```



Fallunterscheidung mit Alternative

```
if (onBeeper())
{
    pickBeep();
}
else
{
    dropBeep();
}
moveForward();
```



Negation !

- **if/else** mit leerem then-Block kann durch ein **if** ohne **else** ersetzt werden, indem man die Bedingung negiert

```
if (frontIsClear())
{
}
else
{
    turnLeft();
}
```

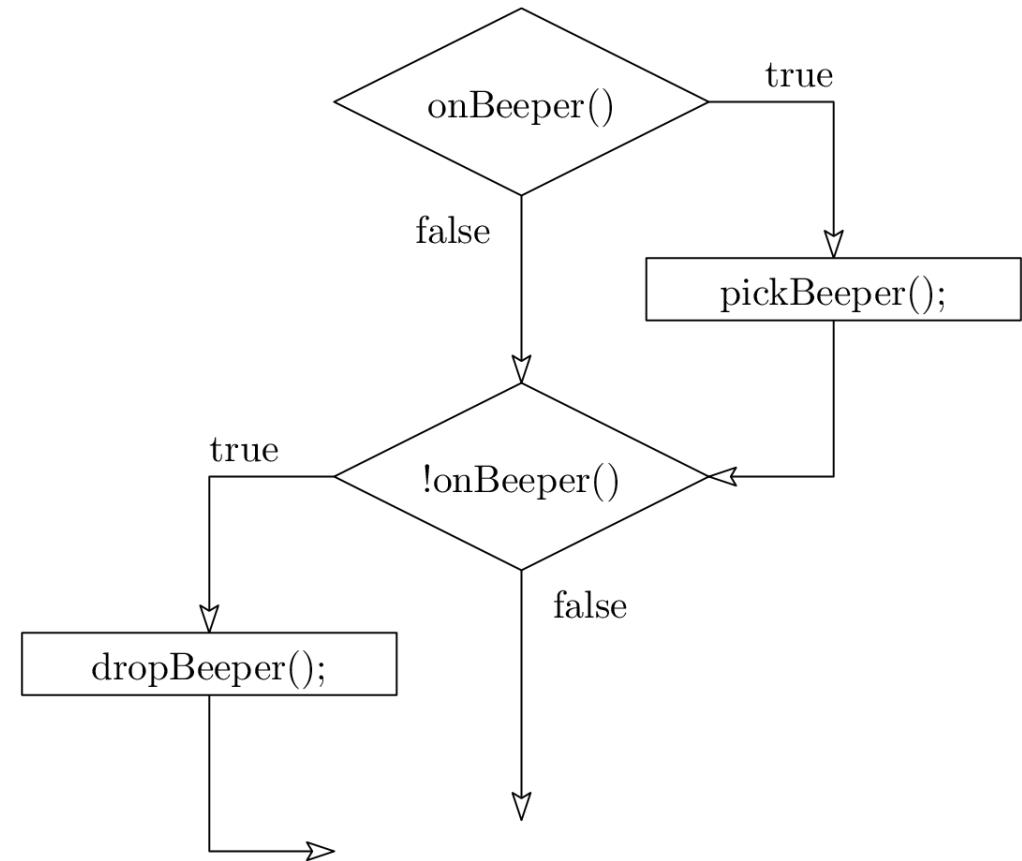


```
if (!frontIsClear())
{
    turnLeft();
}
```

Negierte Fallunterscheidung statt “else”

- Else ist nicht immer durch ein negiertes if ersetzbar

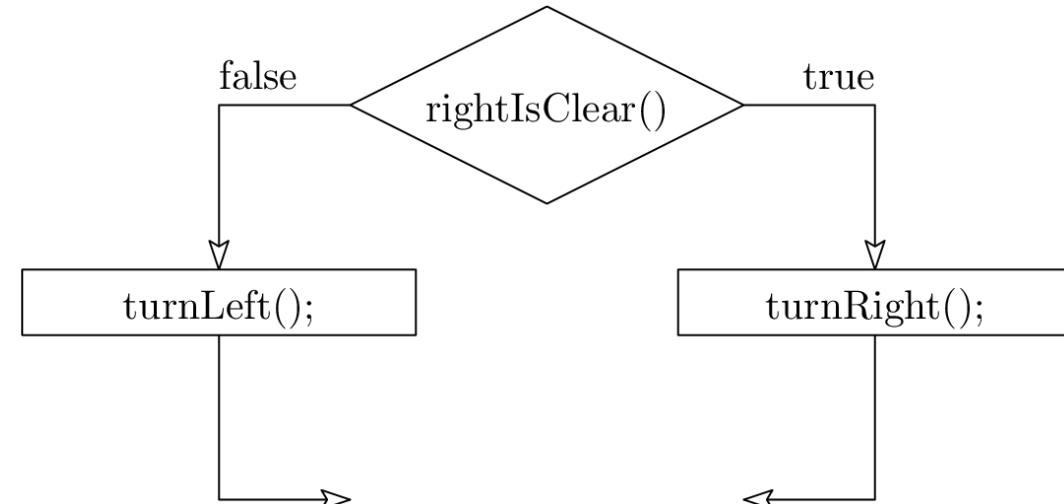
```
void pickOrDrop()
{
    if (onBeep())
    {
        pickBeep();
    }
    if (!onBeep())
    {
        dropBeep();
    }
}
```



- Warum trifft die zweite Bedingung hier immer zu?

I: Wie viele verschiedene Pfade gibt es durch den Code?

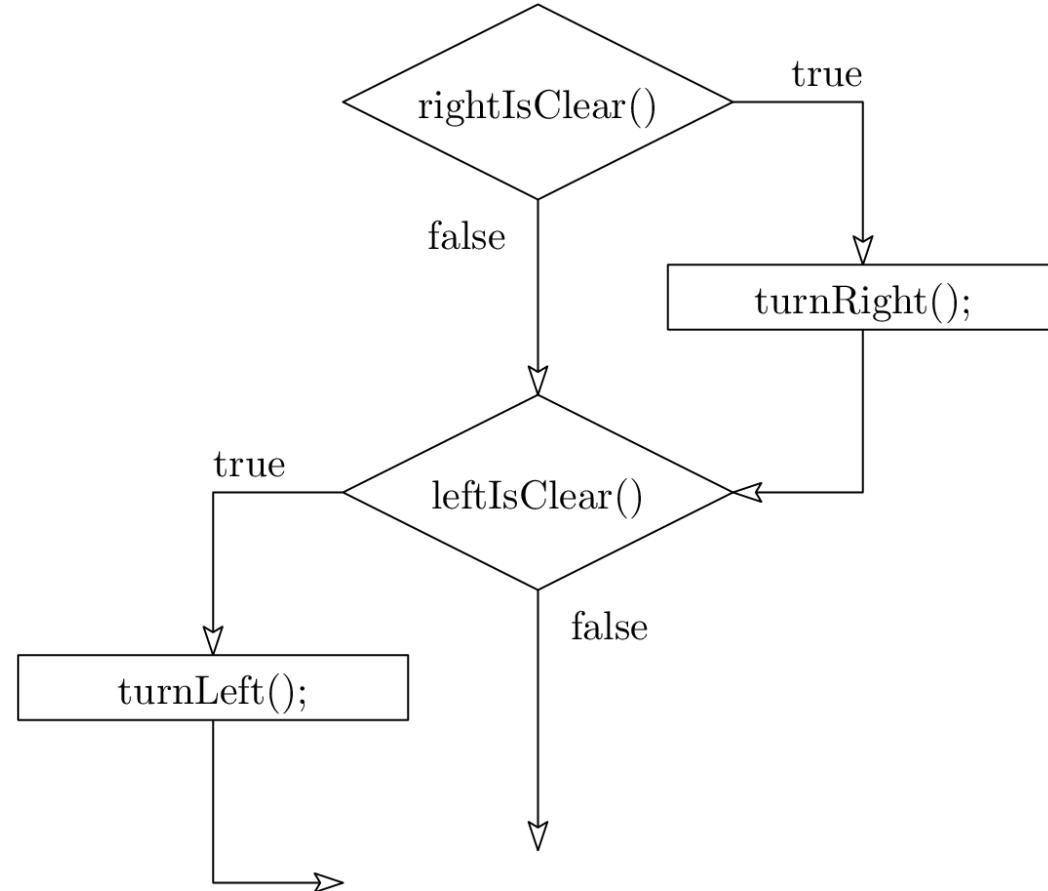
```
void ifelse()
{
    if(rightIsClear())
    {
        turnRight();
    }
    else
    {
        turnLeft();
    }
}
```



2 Pfade: entweder **turnLeft()** oder **turnRight()**

II: Wie viele verschiedene Pfade gibt es durch den Code?

```
void ifif()
{
    if (rightIsClear())
    {
        turnRight();
    }
    if (leftIsClear())
    {
        turnLeft();
    }
}
```



4 Pfade: entweder **keine Ausführung** oder **turnRight** oder **turnLeft** oder **beides**

Demo:

1.3.4 tileTheFloor

Überblick

1

Kontrollstrukturen

2

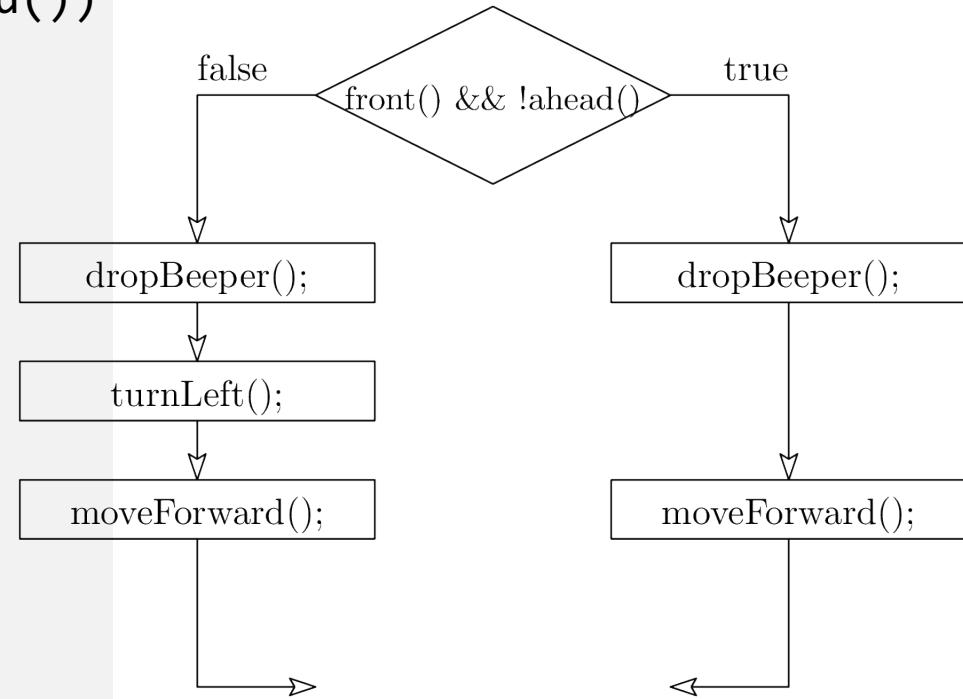
Bedingungen und bedingte Schleifen

3

Binärsystem

tileTheFloor: Fundstücke aus dem Übungsbetrieb

```
if (frontIsClear() && !beeperAhead())
{
    dropBeeper();
    moveForward();
}
else
{
    dropBeeper();
    turnLeft();
    moveForward();
```

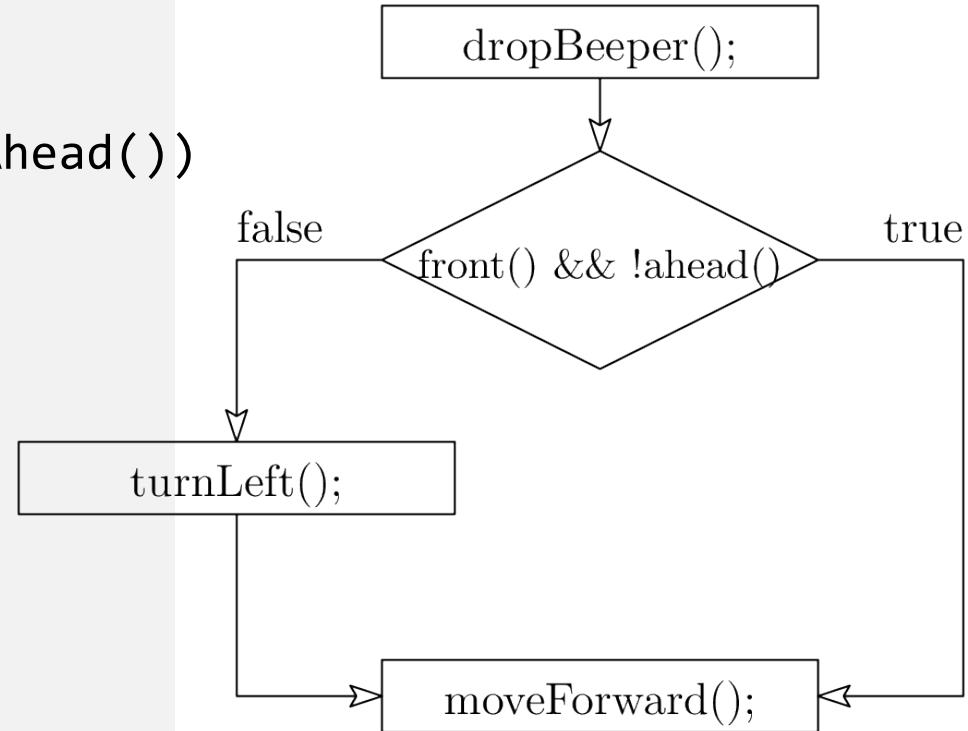


Beobachtung: Die beiden Blöcke sind sich recht ähnlich.

Frage: Kann man die Fallunterscheidung vereinfachen?

tileTheFloor: Fundstücke aus dem Übungsbetrieb

```
dropBeeper();
if (frontIsClear() && !beeperAhead())
{
}
else
{
    turnLeft();
}
moveForward();
```



Beobachtung: Eine Fallunterscheidung mit einem leeren Block ist ungewöhnlich.

Frage: Kann man das **else** loswerden, d.h. das **if/else** durch ein einfaches **if** ersetzen?

tileTheFloor: Fundstücke aus dem Übungsbetrieb

```
// vorher:  
if (frontIsClear() && !beeperAhead())  
{  
}  
}  
else  
{  
    turnLeft();  
}  
  
// nachher:  
if (!frontIsClear() && !beeperAhead())  
{  
    turnLeft();  
}
```

Jetzt haben wir zwei Negationen; geht das nicht einfacher?

De Morgan Beispiel

- Alice schmeckt Kaffee nur
 - **mit Milch und mit Zucker**
 - **Milch && Zucker**
- Wenn Alice den Kaffee **nicht** trinkt, dann ist der Kaffee:
 - **nicht mit Milch und Zucker** ! (Milch **&&** Zucker)
 - **ohne Milch oder ohne Zucker** ! Milch **||** ! Zucker



Gesetze von Augustus De Morgan

- $!(a \&\& b) = !a \mid\mid !b$

Wenn beide Bedingungen nicht zutreffen,
dann trifft mindestens eine Bedingung nicht zu!

- $!(a \mid\mid b) = !a \&\& !b$

Trifft mindestens eine Bedingung nicht zu,
dann treffen beide Bedingungen nicht zu!

- Beispiel in Karel:

```
if (!frontIsClear() && !beeperAhead())
```

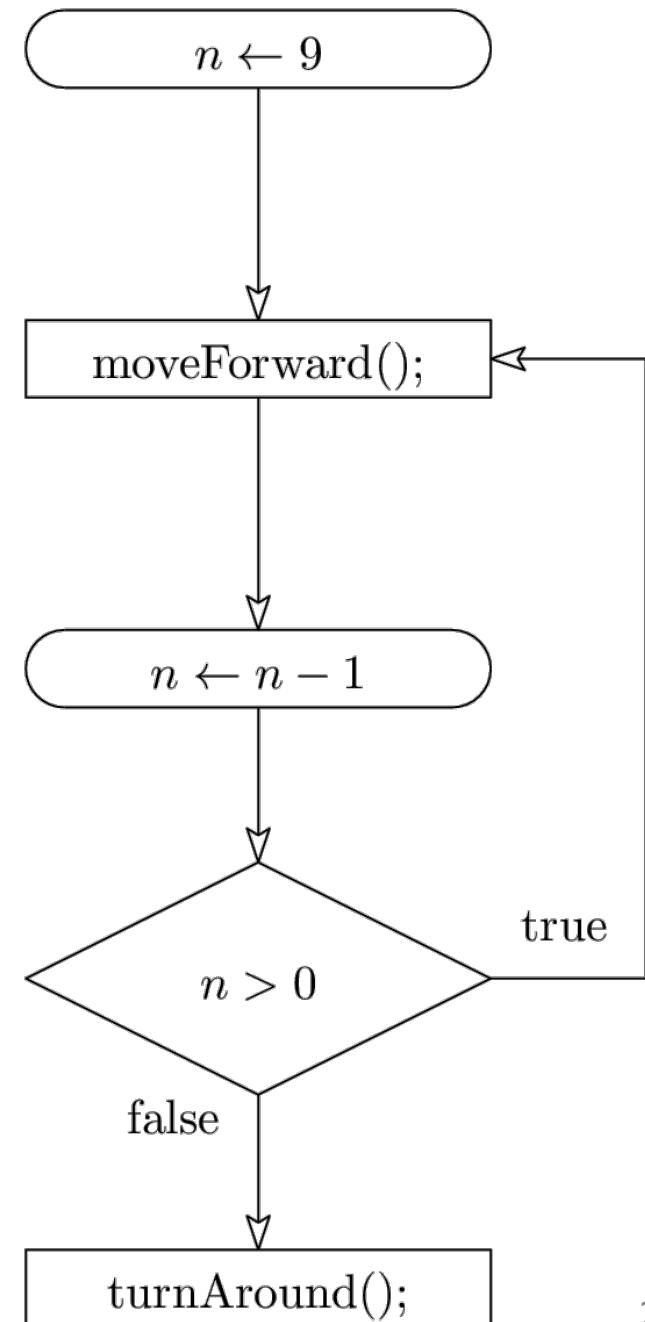
```
if ( !frontIsClear() || beeperAhead())
```



Zählschleife

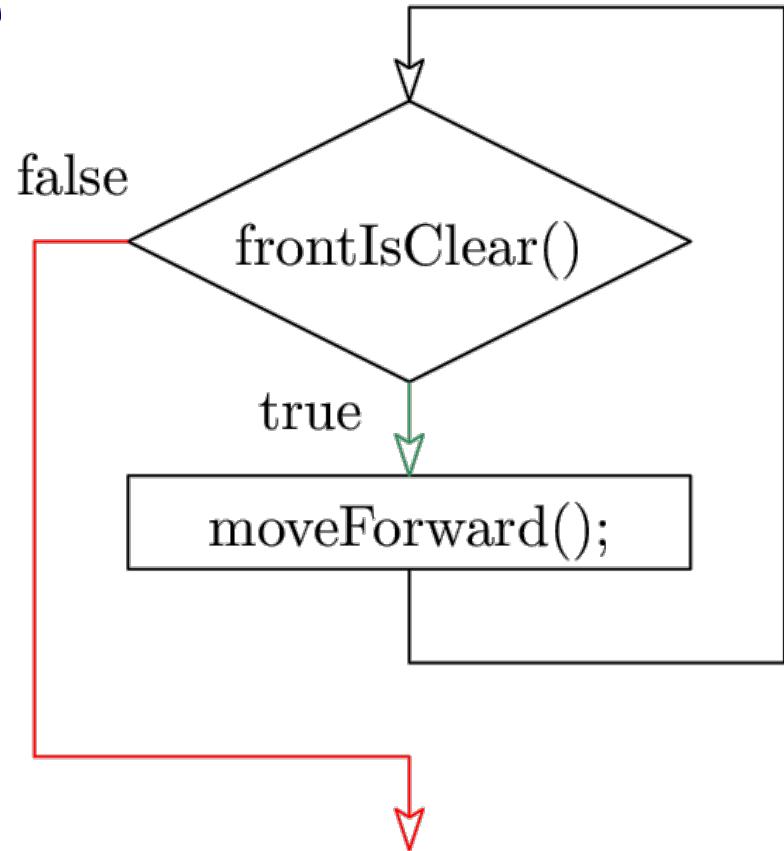
```
repeat (9)
{
    moveForward();
}
turnAround();
```

- Problem:
 - 10 Felder Welt
 - Von einer Seite zur Anderen



Bedingte Schleife

```
void moveToWall()
{
    while (frontIsClear())
    {
        moveForward();
    }
}
```



- Solange die Bedingung wahr ist, wird der Block immer wieder ausgeführt
- Falls die Bedingung bereits am Anfang falsch ist, wird der Block nicht ausgeführt
- Schleifenbedingung wird vor dem nächsten Durchlauf geprüft
- Unbestimmte Anzahl an Ausführungen

Demo while-Schleife: 2.1.1 hangTheLampions

Überblick

1

Kontrollstrukturen

2

Bedingungen und bedingte Schleifen

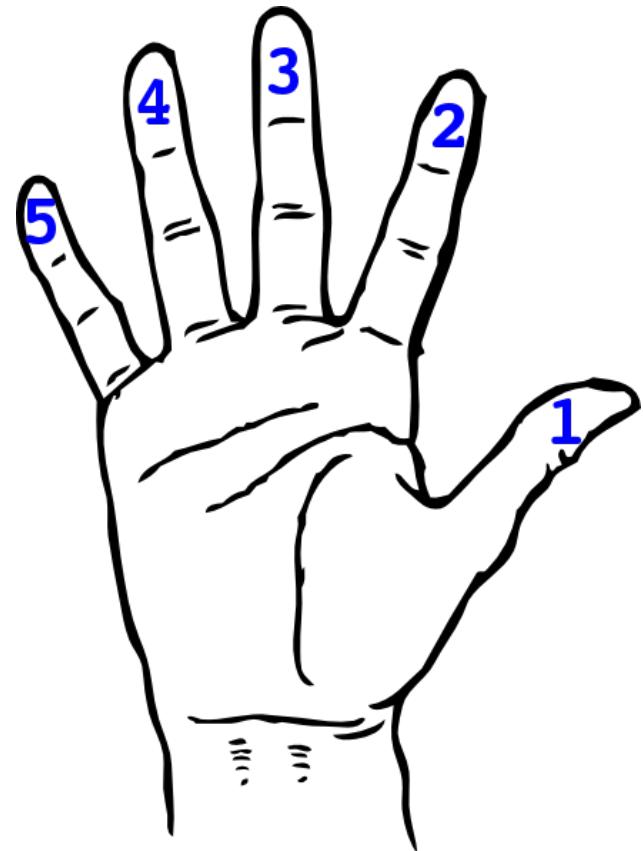
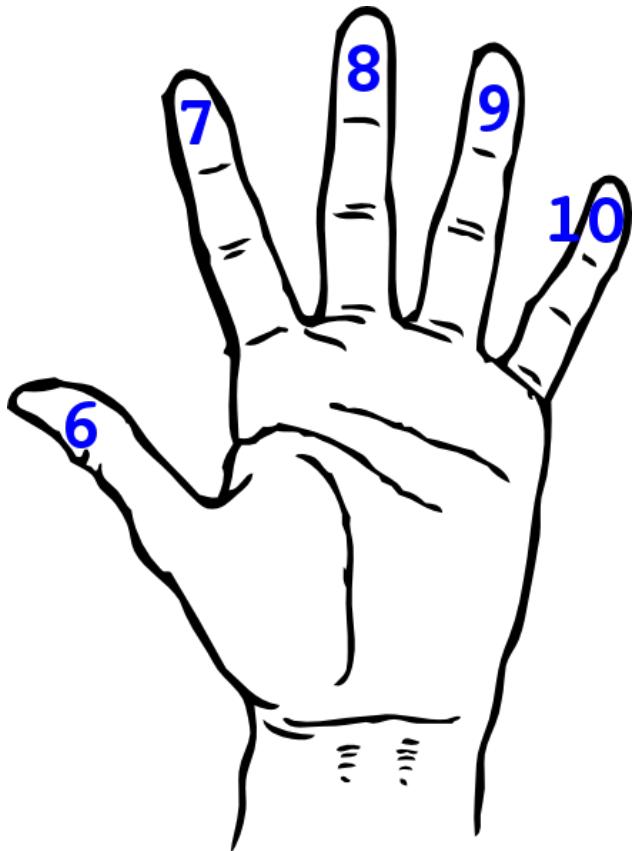
3

Binärsystem

Dezimalsystem

$$\begin{array}{r} \text{Zehntausender} \\ | \\ \text{Tausender} \\ | \\ \text{Hunderter} \\ | \\ \text{Zehner} \\ | \\ \text{Einer} \\ | \\ 2 \quad 0 \quad 5 \quad 9 \quad 7 \\ + \quad 2 \quad 0 \quad 5 \quad 9 \quad 8 \quad 3 \quad 1 \quad 4 \quad 1 \quad 5 \\ \quad 2 \quad 0 \quad 5 \quad 9 \quad 9 \quad + \quad 7 \quad 1 \quad 8 \quad 2 \quad 8 \\ \quad 2 \quad 0 \quad 6 \quad 0 \quad 0 \quad \quad \quad 1 \quad \quad \quad 1 \quad \quad \quad 1 \\ \hline \quad 2 \quad 0 \quad 6 \quad 0 \quad 1 \quad \quad 1 \quad 0 \quad 3 \quad 2 \quad 4 \quad 3 \end{array}$$

Warum gibt es eigentlich zehn verschiedene Ziffern?



Binärsystem

$$\begin{array}{cccccc} & \text{Sechzehner} & & & & \\ & | & | & | & | & | \\ & \text{Achter} & \text{Vierer} & \text{Zweier} & \text{Einer} & \\ \hline 0 & 0 & 0 & 0 & 0 & \\ \\ 0 & 0 & 0 & 0 & 1 & & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & + & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & & 1 & 1 & & & \\ \hline 0 & 0 & 1 & 0 & 0 & & 1 & 1 & 0 & 1 & 0 \end{array}$$

Binärsystem in Karels Welt

- Jede Zeile repräsentiert eine Binärzahl
 - 1 = Beeper
 - 0 = Leeres Feld

512	256	128	64	32	16	8	4	2	1	Σ
+	+	+	+	+	+	+	+	+	+	1
+	+	+	+	+	+	+	+	+	+	8
+	+	+	+	+	+	+	+	+	+	18
+	+	+	+	+	+	+	+	+	+	77

Schriftliche Addition im Binärsystem

- Addition im Binärsystem wie im Dezimalsystem
- A + B werden von rechts nach links aufsummiert
- Eventuell auftretende Überträge (Ü) berücksichtigen
- Blauer Kasten beinhaltet Summe A+B+Ü einer Spalte

A	0101	+	+	+	+ 1	+ 1	+	+	+	+	+	+
B	+	+	+	+	+	+	+	+	+	+	+	+
Ü	+	+	+	+	+	+	+	+	+	+	+	+
S	+	+	+	+	+	+	+	+	+	+	+	+
	+	+	+	+	+	+	+	+	+	+	+	+

A	0101	+	+	+	+ 1	+ 1	+	+	+	+	+	+
B	+	+	+	+	+	+	+	+	+	+	+	+
Ü	+	+	+	+	+	+	+	+	+	+	+	+
S	+	+	+	+	+	+	+	+	+	+	+	+
	+	+	+	+	+	+	+	+	+	+	+	+

Zusammenfassung

- 1 **Kontrollstrukturen:** Sequenz Aufruf zusammengesetzter, Befehle, Zählschleife, Fallunterscheidung, Bedingte Schleife
- 2 **Bedingte Schleifen** erlauben es, Befehle zu wiederholen, solange eine bestimmte Bedingung erfüllt ist
- 3 DeMorgan'schen Regeln helfen bei der Vereinfachung von **negierten zusammengesetzten Bedingungen**
- 4 Darstellung, Überträge und Addieren von Binärzahlen im **Binärsystem** funktionieren wie im Dezimalsystem