Karlsruher Institut für Technologie

lukasbach

# Grundbegriffe der Informatik
# Tutorium 33

Lukas Bach, lukas.bach@student.kit.edu | 22.12.2016

Lukas Bach, lukas.bach@student.kit.edu

# Algorithmen

Lukas Bach, lukas.bach@student.kit.edu



# Algorithmen

- Es existiert eine **endliche** Beschreibung

Lukas Bach, lukas.bach@student.kit.edu



# Algorithmen

- Es existiert eine **endliche** Beschreibung
- Es wird zu einer beliebig großen, aber **endlichen** Eingabe eine **endliche** Ausgabe berechnet

Lukas Bach, lukas.bach@student.kit.edu



# Algorithmen

- Es existiert eine **endliche** Beschreibung
- Es wird zu einer beliebig großen, aber **endlichen** Eingabe eine **endliche** Ausgabe berechnet
- Es finden **endlich** viele Schritte statt (der Algorithmus terminiert)

Lukas Bach, lukas.bach@student.kit.edu



# Algorithmen

- Es existiert eine **endliche** Beschreibung
- Es wird zu einer beliebig großen, aber **endlichen** Eingabe eine **endliche** Ausgabe berechnet
- Es finden **endlich** viele Schritte statt (der Algorithmus terminiert)
- Deterministisch (bei mehrmaliger Ausführung kommt immer das selbe raus)

Lukas Bach, lukas.bach@student.kit.edu



# Hier verwendeter Pseudocode

Lukas Bach, lukas.bach@student.kit.edu



# Hier verwendeter Pseudocode

- Zuweisungssymbol $\leftarrow$

Lukas Bach, lukas.bach@student.kit.edu



# Hier verwendeter Pseudocode

- Zuweisungssymbol $\leftarrow$
- Schlüsselwörter für Verzweigungen **if, then, else, fi**

Lukas Bach, lukas.bach@student.kit.edu



# Hier verwendeter Pseudocode

- Zuweisungssymbol $\leftarrow$
- Schlüsselwörter für Verzweigungen **if, then, else, fi**
- Schlüsselwörter für Schleifen **while, do, od, for, to**

Lukas Bach, lukas.bach@student.kit.edu



# Hier verwendeter Pseudocode

- Zuweisungssymbol $\leftarrow$
- Schlüsselwörter für Verzweigungen **if, then, else, fi**
- Schlüsselwörter für Schleifen **while, do, od, for, to**
- Symbole für Konstanten, Funktionen und Relationen

Lukas Bach, lukas.bach@student.kit.edu



## Eine **if**-Verzweigung

```
1 if x < y then
2     s ← x
3 else
4     s ← y
5 fi
```

Lukas Bach, lukas.bach@student.kit.edu



Eine **if**-Verzweigung

```
1 if x < y then
2     s ← x
3 else
4     s ← y
5 fi
```

Eine **while**-Schleife

```
1 while x > 0 do
2     x ← x div 2
3     s ← s + x
4 od
```

Lukas Bach, lukas.bach@student.kit.edu



Eine **if**-Verzweigung

```
1 if x < y then
2     s ← x
3 else
4     s ← y
5 fi
```

Eine **while**-Schleife

```
1 while x > 0 do
2     x ← x div 2
3     s ← s + x
4 od
```

Eine **for**-Schleife

```
1 for i ← 1 to n do
2     s ← s + i
3 od
```

Lukas Bach, lukas.bach@student.kit.edu



# Was kann man mit Algorithmen machen?

Lukas Bach, lukas.bach@student.kit.edu



# Was kann man mit Algorithmen machen?

- Komplexe Algorithmen mit Pseudocode definieren zu Sortierung, Graphen, Datenstrukturen

Lukas Bach, lukas.bach@student.kit.edu



# Was kann man mit Algorithmen machen?

- Komplexe Algorithmen mit Pseudocode definieren zu Sortierung, Graphen, Datenstrukturen, im Modul Algorithmen I

Lukas Bach, lukas.bach@student.kit.edu



# Was kann man mit Algorithmen machen?

- Komplexe Algorithmen mit Pseudocode definieren zu Sortierung, Graphen, Datenstrukturen, im Modul Algorithmen I
- Laufzeitanalyse von Algorithmen

Lukas Bach, lukas.bach@student.kit.edu



# Was kann man mit Algorithmen machen?

- Komplexe Algorithmen mit Pseudocode definieren zu Sortierung, Graphen, Datenstrukturen, im Modul Algorithmen I
- Laufzeitanalyse von Algorithmen, später.

Lukas Bach, lukas.bach@student.kit.edu



# Was kann man mit Algorithmen machen?

- Komplexe Algorithmen mit Pseudocode definieren zu Sortierung, Graphen, Datenstrukturen, im Modul Algorithmen I
- Laufzeitanalyse von Algorithmen, später.
- Korrektheitsbeweise

Lukas Bach, lukas.bach@student.kit.edu



# Was kann man mit Algorithmen machen?

- Komplexe Algorithmen mit Pseudocode definieren zu Sortierung, Graphen, Datenstrukturen, im Modul Algorithmen I
- Laufzeitanalyse von Algorithmen, später.
- Korrektheitsbeweise, jetzt.

Lukas Bach, lukas.bach@student.kit.edu

# Korrektheitsbeweise

Lukas Bach, lukas.bach@student.kit.edu



# Korrektheitsbeweise

Wie findet man heraus, ob ein Algorithmus korrekt funktioniert?

Lukas Bach, lukas.bach@student.kit.edu



# Korrektheitsbeweise

Wie findet man heraus, ob ein Algorithmus korrekt funktioniert?

- Durch den Beweis von Zusicherungen, die an bestimmten Stellen des Algorithmus gelten.

Lukas Bach, lukas.bach@student.kit.edu



# Korrektheitsbeweise

Wie findet man heraus, ob ein Algorithmus korrekt funktioniert?

- Durch den Beweis von Zusicherungen, die an bestimmten Stellen des Algorithmus gelten.

Was sind Zusicherungen?

Lukas Bach, lukas.bach@student.kit.edu



# Korrektheitsbeweise

Wie findet man heraus, ob ein Algorithmus korrekt funktioniert?

- Durch den Beweis von Zusicherungen, die an bestimmten Stellen des Algorithmus gelten.

Was sind Zusicherungen?

- prädikatenlogische Formeln, die Aussagen über (Zusammenhänge zwischen) Variablen machen

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Definition

$\{P\}S\{Q\}$ heißt Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Definition

$\{P\}S\{Q\}$ heißt Hoare-Tripel. Dabei gilt:

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Definition

$\{P\}S\{Q\}$ heißt Hoare-Tripel. Dabei gilt:

- S ist ein Programmstück im Pseudocode

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Definition

$\{P\}S\{Q\}$ heißt Hoare-Tripel. Dabei gilt:

- S ist ein Programmstück im Pseudocode
- P und Q sind Zusicherungen

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Definition

$\{P\}S\{Q\}$ heißt Hoare-Tripel. Dabei gilt:

- S ist ein Programmstück im Pseudocode
- P und Q sind Zusicherungen

- P nennt man Vorbedingung, Q Nachbedingung

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Definition

$\{P\}S\{Q\}$ heißt Hoare-Tripel. Dabei gilt:

- S ist ein Programmstück im Pseudocode
- P und Q sind Zusicherungen

- P nennt man Vorbedingung, Q Nachbedingung
- Prädikatenlogische Formeln

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Definition

$\{P\}S\{Q\}$ heißt Hoare-Tripel. Dabei gilt:

- S ist ein Programmstück im Pseudocode
- P und Q sind Zusicherungen

- P nennt man Vorbedingung, Q Nachbedingung
- Prädikatenlogische Formeln
- Beispiel (Vorausblick): $\{x \doteq 1\}x \leftarrow x + 1\{x \doteq 2\}$

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Definition

$\{P\}S\{Q\}$ heißt Hoare-Tripel. Dabei gilt:

- S ist ein Programmstück im Pseudocode
- P und Q sind Zusicherungen

- P nennt man Vorbedingung, Q Nachbedingung
- Prädikatenlogische Formeln
- Beispiel (Vorausblick): $\{x \doteq 1\}x \leftarrow x + 1\{x \doteq 2\}$
- Meistens in jeder Zeile nur eine Zeile Code oder ein Zusicherungsblock

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Gültigkeit von Hoare-Tripeln

$\{P\}S\{Q\}$ ist gültig, wenn für jede gültige Interpretation $(D, I)$ und Variablenbelegung $\beta$ gilt:

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Gültigkeit von Hoare-Tripeln

$\{P\}S\{Q\}$ ist gültig, wenn für jede gültige Interpretation $(D, I)$ und Variablenbelegung $\beta$ gilt:
Aus

Lukas Bach, lukas.bach@student.kit.edu



# Das Hoare-Tripel

## Gültigkeit von Hoare-Tripeln

$\{P\}S\{Q\}$ ist gültig, wenn für jede gültige Interpretation $(D, I)$ und Variablenbelegung $\beta$ gilt:
Aus

- $val_{D,I,\beta}(P) = w$

# Das Hoare-Tripel

## Gültigkeit von Hoare-Tripeln

$\{P\}S\{Q\}$ ist gültig, wenn für jede gültige Interpretation $(D, I)$ und Variablenbelegung $\beta$ gilt:
Aus

- $val_{D,I,\beta}(P) = w$
- $\beta'$ ist Variablenbelegung nach Ausführung von $S$

# Das Hoare-Tripel

## Gültigkeit von Hoare-Tripeln

$\{P\}S\{Q\}$ ist gültig, wenn für jede gültige Interpretation $(D, I)$ und Variablenbelegung $\beta$ gilt:
Aus

- $val_{D,I,\beta}(P) = w$
- $\beta'$ ist Variablenbelegung nach Ausführung von $S$

folgt $val_{D,I,\beta'}(Q) = w$

Lukas Bach, lukas.bach@student.kit.edu



# Zuweisung

## Axiom HT-A

Lukas Bach, lukas.bach@student.kit.edu



# Zuweisung

## Axiom HT-A

- Sei $x \leftarrow E$ eine Zuweisung

Lukas Bach, lukas.bach@student.kit.edu



# Zuweisung

## Axiom HT-A

- Sei $x \leftarrow E$ eine Zuweisung
- $Q$ eine Nachbedingung von $x \leftarrow E$ und

Lukas Bach, lukas.bach@student.kit.edu



# Zuweisung

## Axiom HT-A

- Sei $x \leftarrow E$ eine Zuweisung
- $Q$ eine Nachbedingung von $x \leftarrow E$ und
- $\sigma_{\{x/E\}}$ kollisionsfrei für Q

Lukas Bach, lukas.bach@student.kit.edu



# Zuweisung

## Axiom HT-A

- Sei $x \leftarrow E$ eine Zuweisung
- $Q$ eine Nachbedingung von $x \leftarrow E$ und
- $\sigma_{\{x/E\}}$ kollisionsfrei für Q

Dann ist $\sigma_{\{x/E\}}(Q)x \leftarrow E\{Q\}$ ein gültiges Hoare-Tripel

Lukas Bach, lukas.bach@student.kit.edu



# Zuweisung

## Axiom HT-A

- Sei $x \leftarrow E$ eine Zuweisung
- $Q$ eine Nachbedingung von $x \leftarrow E$ und
- $\sigma_{\{x/E\}}$ kollisionsfrei für Q

Dann ist $\sigma_{\{x/E\}}(Q)x \leftarrow E\{Q\}$ ein gültiges Hoare-Tripel

**Bemerkung**

Lukas Bach, lukas.bach@student.kit.edu



# Zuweisung

## Axiom HT-A

- Sei $x \leftarrow E$ eine Zuweisung
- $Q$ eine Nachbedingung von $x \leftarrow E$ und
- $\sigma_{\{x/E\}}$ kollisionsfrei für Q

Dann ist $\sigma_{\{x/E\}}(Q)x \leftarrow E\{Q\}$ ein gültiges Hoare-Tripel

**Bemerkung**

- $\sigma_{\{x/E\}}$ ist die Substitution von $x$ mit $E$

Lukas Bach, lukas.bach@student.kit.edu



# Zuweisung

## Axiom HT-A

- Sei $x \leftarrow E$ eine Zuweisung
- $Q$ eine Nachbedingung von $x \leftarrow E$ und
- $\sigma_{\{x/E\}}$ kollisionsfrei für Q

Dann ist $\sigma_{\{x/E\}}(Q)x \leftarrow E\{Q\}$ ein gültiges Hoare-Tripel

**Bemerkung**

- $\sigma_{\{x/E\}}$ ist die Substitution von $x$ mit $E$
- Bei Anwendung der Regel rückwärts vorgehen

Lukas Bach, lukas.bach@student.kit.edu



**Beispiel**

Betrachte die Zuweisung

$x \leftarrow x + 1$

und die Nachbedingung

$\{x \doteq 1\}$

Lukas Bach, lukas.bach@student.kit.edu



**Beispiel**

Betrachte die Zuweisung

$x \leftarrow x + 1$

und die Nachbedingung

$\{x \doteq 1\}$

Nach HT-A gilt

Lukas Bach, lukas.bach@student.kit.edu



**Beispiel**

Betrachte die Zuweisung

$x \leftarrow x + 1$

und die Nachbedingung

$\{x \doteq 1\}$

Nach HT-A gilt

$\{x + 1 \doteq 1\}\ x \leftarrow x + 1\ \{x \doteq 1\}$ ist ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu



# Ableitungsregeln: HT-E

- Verstärkung der Vorbedingung
- Abschwächung der Nachbedingung

Lukas Bach, lukas.bach@student.kit.edu



# Ableitungsregeln: HT-E

- Verstärkung der Vorbedingung
- Abschwächung der Nachbedingung

**HT-E**

Wenn $\{P\}S\{Q\}$ ein gültiges Hoare-Tripel ist und $P' \vdash P$ und $Q \vdash Q'$ gelten, dann folgt:

Lukas Bach, lukas.bach@student.kit.edu



# Ableitungsregeln: HT-E

- Verstärkung der Vorbedingung
- Abschwächung der Nachbedingung

**HT-E**

Wenn $\{P\}S\{Q\}$ ein gültiges Hoare-Tripel ist und $P' \vdash P$ und $Q \vdash Q'$ gelten, dann folgt:
$\{P'\}S\{Q'\}$ ist ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu



# Ableitungsregeln: HT-E

- Verstärkung der Vorbedingung
- Abschwächung der Nachbedingung

**HT-E**

Wenn $\{P\}S\{Q\}$ ein gültiges Hoare-Tripel ist und $P' \vdash P$ und $Q \vdash Q'$ gelten, dann folgt:
$\{P'\}S\{Q'\}$ ist ein gültiges Hoare-Tripel.

**Bemerkung**

$B \vdash A :\Leftrightarrow$ Aussage $A$ ist syntaktisch aus Aussage $B$ ableitbar

**Beispiel**

Angenommen es sei $\{y > 3\}\ x \leftarrow y - 1\ \{x > 1\}$ ein gültiges Hoare-Tripel.
Es gilt $\{(y > 4)\} \vdash \{(y > 3)\}$ und $\{(x > 1)\} \vdash \{(x > 0)\}$.
Also folgt nach HT-E:

**Beispiel**

Angenommen es sei $\{y > 3\}\ x \leftarrow y - 1\ \{x > 1\}$ ein gültiges Hoare-Tripel.
Es gilt $\{(y > 4)\} \vdash \{(y > 3)\}$ und $\{(x > 1)\} \vdash \{(x > 0)\}$.
Also folgt nach HT-E:
$\{y > 4\}\ x \leftarrow y - 1\ \{x > 0\}$ ist ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu



**Beispiel**

Angenommen es sei $\{y > 3\}\ x \leftarrow y - 1\ \{x > 1\}$ ein gültiges Hoare-Tripel.
Es gilt $\{(y > 4)\} \vdash \{(y > 3)\}$ und $\{(x > 1)\} \vdash \{(x > 0)\}$.
Also folgt nach HT-E:
$\{y > 4\}\ x \leftarrow y - 1\ \{x > 0\}$ ist ein gültiges Hoare-Tripel.

**Bemerkung**

Es müssen sich nicht unbedingt beide Bedingungen ändern!
Aus $\{(y > 3)\} \vdash \{(y > 3)\}$ und $\{(x > 1)\} \vdash \{(x > 0)\}$
folgt nach HT-E auch
$\{y > 3\}\ x \leftarrow y - 1\ \{x > 0\}$ ist ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu



# Ableitungsregeln: HT-S

Lukas Bach, lukas.bach@student.kit.edu



# Ableitungsregeln: HT-S

Hintereinanderausführung von durch Hoare-Triple bewiesene Code Segmente sind selbst gültig.

# Ableitungsregeln: HT-S

Hintereinanderausführung von durch Hoare-Triple bewiesene Code Segmente sind selbst gültig.

**HT-S**

Wenn $\{P\}S_1\{Q\}$ und $\{Q\}S_2\{R\}$ gültige Hoare-Tripel sind

# Ableitungsregeln: HT-S

Hintereinanderausführung von durch Hoare-Triple bewiesene Code Segmente sind selbst gültig.

**HT-S**

Wenn $\{P\}S_1\{Q\}$ und $\{Q\}S_2\{R\}$ gültige Hoare-Tripel sind, dann folgt:

# Ableitungsregeln: HT-S

Hintereinanderausführung von durch Hoare-Triple bewiesene Code Segmente sind selbst gültig.

**HT-S**

Wenn $\{P\}S_1\{Q\}$ und $\{Q\}S_2\{R\}$ gültige Hoare-Tripel sind, dann folgt:$\{P\}S_1; S_2\{R\}$ ist ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu



# Ableitungsregeln: HT-S

Hintereinanderausführung von durch Hoare-Triple bewiesene Code Segmente sind selbst gültig.

**HT-S**

Wenn $\{P\}S_1\{Q\}$ und $\{Q\}S_2\{R\}$ gültige Hoare-Tripel sind, dann folgt:$\{P\}S_1; S_2\{R\}$ ist ein gültiges Hoare-Tripel.

**Bemerkung**

";" trennt hier zwei Programmstücke

Lukas Bach, lukas.bach@student.kit.edu



**Beispiel**

Angenommen es seien $\{y > 3\}\ x \leftarrow y - 1\ \{x > 1\}$ und $\{x > 1\}\ z \leftarrow x - 1\ \{z > -1\}$ gültige Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu



**Beispiel**

Angenommen es seien $\{y > 3\}\ x \leftarrow y - 1\ \{x > 1\}$ und $\{x > 1\}\ z \leftarrow x - 1\ \{z > -1\}$ gültige Hoare-Tripel.
Dann folgt nach HT-S:

Lukas Bach, lukas.bach@student.kit.edu



**Beispiel**

Angenommen es seien $\{y > 3\}\ x \leftarrow y - 1\ \{x > 1\}$ und
$\{x > 1\}\ z \leftarrow x - 1\ \{z > -1\}$ gültige Hoare-Tripel.
Dann folgt nach HT-S:
$\{y > 3\}\ x \leftarrow y - 1; z \leftarrow x - 1\ \{z > -1\}$ ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu

# Bedingte Anweisungen

Lukas Bach, lukas.bach@student.kit.edu

# Bedingte Anweisungen



## HT-I

Wenn $\{P \wedge B\} S_1 \{Q\}$ und $\{P \wedge \neg B\} S_2 \{Q\}$ gültige Hoare-Tripel sind

Lukas Bach, lukas.bach@student.kit.edu



# Bedingte Anweisungen

## HT-I

Wenn $\{P \wedge B\} S_1 \{Q\}$ und $\{P \wedge \neg B\} S_2 \{Q\}$ gültige Hoare-Tripel sind, dann folgt:

$\{P\}$
**if** $B$ **then** $S_1$
**else** $S_2$
**fi**
$\{Q\}$

ist ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu



# Beispiel

$\{ x = a \wedge y = b \}$

**if** $x > y$

**then**

$\{ \dots \}$

$z \leftarrow x$

$\{ \dots \}$

**else**

$\{ \dots \}$

$z \leftarrow y$

$\{ \dots \}$

**fi**

$\{ z = \min(a, b) \}$

Lukas Bach, lukas.bach@student.kit.edu



# Beispiel

$\{ x = a \wedge y = b \}$

**if** $x > y$

**then**

$\{ x = a \wedge y = b \wedge \neg(x > y) \}$

$\{ x = \min(a, b) \}$

$z \leftarrow x$

$\{ z = \min(a, b) \}$

**else**

$\{ x = a \wedge y = b \wedge x > y \}$

$\{ y = \min(a, b) \}$

$z \leftarrow y$

$\{ z = \min(a, b) \}$

**fi**

$\{ z = \min(a, b) \}$

Lukas Bach, lukas.bach@student.kit.edu



# Schleifen

## HT-W

Wenn $\{I \wedge B\} S \{I\}$ ein gültiges Hoare-Tripel ist

Lukas Bach, lukas.bach@student.kit.edu



# Schleifen

## HT-W

Wenn $\{I \wedge B\} S \{I\}$ ein gültiges Hoare-Tripel ist, dann folgt:

# Schleifen

## HT-W

Wenn $\{I \wedge B\} S \{I\}$ ein gültiges Hoare-Tripel ist, dann folgt:

$\{I\}$
**while** $B$ **do** $S$
**od**
$\{I \wedge \neg B\}$

ist ein gültiges Hoare-Tripel.

Lukas Bach, lukas.bach@student.kit.edu

# Schleifeninvariante

Lukas Bach, lukas.bach@student.kit.edu



# Schleifeninvariante

- Eine spezielle Zusicherung

Lukas Bach, lukas.bach@student.kit.edu



# Schleifeninvariante

- Eine spezielle Zusicherung
- Schleifeninvarianten müssen **vor**, **während** und **nach** jedem Schleifendurchlauf gelten

Lukas Bach, lukas.bach@student.kit.edu



# Schleifeninvariante

- Eine spezielle Zusicherung
- Schleifeninvarianten müssen **vor**, **während** und **nach** jedem Schleifendurchlauf gelten
- Garantiert, dass die Schleife nicht während einem beliebigen Durchlauf “kaputt” geht.

Lukas Bach, lukas.bach@student.kit.edu



## Beispiel

$\{ x = a \wedge y = b \}$

$\{ \dots \}$

**while** $y \neq 0$

**do**

$\{ \dots \}$

$y \leftarrow y - 1$

$\{ \dots \}$

$x \leftarrow x + 1$

$\{ \dots \}$

**od**

$\{ \dots \}$

$\{ x = a + b \}$

Lukas Bach, lukas.bach@student.kit.edu



## Beispiel

$\{ x = a \wedge y = b \}$

$\{ x + y = a + b \}$

**while** $y \neq 0$

**do**

$\{ x + y = a + b \wedge y \neq 0 \}$

$\{ x + 1 + y - 1 = a + b \}$

$y \leftarrow y - 1$

$\{ x + 1 + y = a + b \}$

$x \leftarrow x + 1$

$\{ x + y = a + b \}$

**od**

$\{ x + y = a + b \wedge \neg(y \neq 0) \}$

$\{ x = a + b \}$


That's all Folks!