



Outline

- 1 Einleitung
- 2 Einführung in C
- 3 C - Fehler finden und vermeiden
- 4 Fortgeschrittenes in C
- 5 C - Binärer Suchbaum
- 6 Einführung in Common Lisp
- 7 Lisp - Beispiele
- 8 Einführung in Prolog
- 9 Formale Semantik**



- Mathematischer Nachweise über Korrektheit von Programmen
- Formale Beschreibung der Bedeutung von Programmen
 - Übersetzersemantik
 - Operationale Semantik
 - Denotationelle Semantik
 - Axiomatische Semantik
- Literatur:
Appelrath, Ludewig, *Skriptum Informatik*, B. G. Teubner, 2000.



- Die Operationale Semantik definiert für eine Programmiersprache eine mathematische Maschine, welche als Programminterpretier dient. Eine solche *mathematische Maschine* M ist ein Tupel $M = (I, O, K, \alpha, \omega, \tau, \pi)$ mit folgenden Elementen:

I ... Eingabedaten und Programmanweisungen

O ... Menge von Ausgabedaten

K ... Menge von Konfigurationen

$\alpha : I \rightarrow K$... Eingabefunktion

$\omega : K \rightarrow O$... Ausgabefunktion

$\tau : K \rightarrow K$... Übergangsfunktion

$\pi : K \rightarrow \{\text{false}, \text{true}\}$... Halteprädikat



- $K = A \times Z$, wobei A die Menge aller Anweisungsfolgen und Z die Menge aller möglichen Programmezustände ist.
- Ein MINI-Programm terminiert genau dann, wenn die aktuelle Anweisungsfolge leer ist. Also: $\pi(a, z) = true$ gdw. $a = \varepsilon$
- *Beispiel:* Das folgende MINI-Programm liefert die Summe zweier Zahlen x und y , falls $y \geq 0$ gilt. Andernfalls terminiert es nicht.

```
read x, y;  
while y != 0 do x := x + 1; y := y - 1 end;  
write x.
```

Im Folgenden berechnen wir schrittweise die Konfigurationsübergänge dieses Programms.



Übergangsfunktion τ für Mini

alter Zustand	alte Anweisungsfolge	neuer Zustand	neue Anweisungsfolge
z	$\varepsilon; \langle A \rangle$	z	$\langle A \rangle$
z	$b := 0; \langle A \rangle$	$z \langle b \leftarrow 0 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 + 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) + 1 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 - 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) - 1 \rangle$	$\langle A \rangle$
z	if $b = 0$ then $\langle A_1 \rangle$ else $\langle A_2 \rangle$ end; $\langle A \rangle$	z	$\langle A_1 \rangle; \langle A \rangle$ falls $z(b) = 0$ $\langle A_2 \rangle; \langle A \rangle$ sonst
z	while $b \neq 0$ do $\langle A_1 \rangle$ end; $\langle A \rangle$	z	$\langle A \rangle$ falls $z(b) = 0$ $\langle A_1 \rangle; \text{while } b \neq 0$ sonst do $\langle A_1 \rangle$ end; $\langle A \rangle$

Hierbei z ist die Abbildung $z : B \rightarrow \mathbb{N}$, die jedem Bezeichner einen Wert zuweist. $z \langle b \leftarrow w \rangle$ bezeichnet die Variablenblegung die mit z übereinstimmt, wobei nur die Variable b den Wert w hat.



Übergangsfunktion τ für MINI

alter Zustand	alte Anweisungsfolge	neuer Zustand	neue Anweisungsfolge
z	$\varepsilon; \langle A \rangle$	z	$\langle A \rangle$
z	$b := 0; \langle A \rangle$	$z \langle b \leftarrow 0 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 + 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) + 1 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 - 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) - 1 \rangle$	$\langle A \rangle$
z	if $b = 0$ then $\langle A_1 \rangle$ else $\langle A_2 \rangle$ end; $\langle A \rangle$	z	$\langle A_1 \rangle; \langle A \rangle$ falls $z(b) = 0$ $\langle A_2 \rangle; \langle A \rangle$ sonst
z	while $b \neq 0$ do $\langle A_1 \rangle$ end; $\langle A \rangle$	z	$\langle A \rangle$ falls $z(b) = 0$ $\langle A_1 \rangle; \text{while } b \neq 0$ sonst do $\langle A_1 \rangle$ end; $\langle A \rangle$

(while $y \neq 0$ do $x := x + 1; y := y - 1$ end; ε, z_0)

$z_0(x) = 3, z_0(y) = 2$

Hinweis: Beachten Sie, dass die Anweisungsfolge durch die leere Anweisung ε ergänzt wird.



Übergangsfunktion τ für MINI

alter Zustand	alte Anweisungsfolge	neuer Zustand	neue Anweisungsfolge
z	$\varepsilon; \langle A \rangle$	z	$\langle A \rangle$
z	$b := 0; \langle A \rangle$	$z \langle b \leftarrow 0 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 + 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) + 1 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 - 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) - 1 \rangle$	$\langle A \rangle$
z	if $b = 0$ then $\langle A_1 \rangle$ else $\langle A_2 \rangle$ end; $\langle A \rangle$	z	$\langle A_1 \rangle; \langle A \rangle$ falls $z(b) = 0$ $\langle A_2 \rangle; \langle A \rangle$ sonst
z	while $b \neq 0$ do $\langle A_1 \rangle$ end; $\langle A \rangle$	z	$\langle A \rangle$ falls $z(b) = 0$ $\langle A_1 \rangle; \text{while } b \neq 0$ sonst do $\langle A_1 \rangle$ end; $\langle A \rangle$

$(x := x + 1; y := y - 1; \text{while } y \neq 0 \text{ do } x := x + 1; y := y - 1 \text{ end}; \varepsilon, z_0)$
 $z_0(x) = 3, z_0(y) = 2$



Übergangsfunktion τ für MINI

alter Zustand	alte Anweisungsfolge	neuer Zustand	neue Anweisungsfolge
z	$\varepsilon; \langle A \rangle$	z	$\langle A \rangle$
z	$b := 0; \langle A \rangle$	$z \langle b \leftarrow 0 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 + 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) + 1 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 - 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) - 1 \rangle$	$\langle A \rangle$
z	if $b = 0$ then $\langle A_1 \rangle$ else $\langle A_2 \rangle$ end; $\langle A \rangle$	z	$\langle A_1 \rangle; \langle A \rangle$ falls $z(b) = 0$ $\langle A_2 \rangle; \langle A \rangle$ sonst
z	while $b \neq 0$ do $\langle A_1 \rangle$ end; $\langle A \rangle$	z	$\langle A \rangle$ falls $z(b) = 0$ $\langle A_1 \rangle; \text{while } b \neq 0$ sonst do $\langle A_1 \rangle$ end; $\langle A \rangle$

$(y := y - 1; \text{while } y \neq 0 \text{ do } x := x + 1; y := y - 1 \text{ end}; \varepsilon, z_1)$

$z_1(x) = 4, z_1(y) = 2$



Übergangsfunktion τ für MINI

alter Zustand	alte Anweisungsfolge	neuer Zustand	neue Anweisungsfolge
z	$\varepsilon; \langle A \rangle$	z	$\langle A \rangle$
z	$b := 0; \langle A \rangle$	$z \langle b \leftarrow 0 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 + 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) + 1 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 - 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) - 1 \rangle$	$\langle A \rangle$
z	if $b = 0$ then $\langle A_1 \rangle$ else $\langle A_2 \rangle$ end; $\langle A \rangle$	z	$\langle A_1 \rangle; \langle A \rangle$ falls $z(b) = 0$ $\langle A_2 \rangle; \langle A \rangle$ sonst
z	while $b \neq 0$ do $\langle A_1 \rangle$ end; $\langle A \rangle$	z	$\langle A \rangle$ falls $z(b) = 0$ $\langle A_1 \rangle; \text{while } b \neq 0$ sonst do $\langle A_1 \rangle$ end; $\langle A \rangle$

(while $y \neq 0$ do $x := x + 1; y := y - 1$ end; ε, Z_2)

$z_2(x) = 4, z_2(y) = 1$



Übergangsfunktion τ für MINI

alter Zustand	alte Anweisungsfolge	neuer Zustand	neue Anweisungsfolge
z	$\varepsilon; \langle A \rangle$	z	$\langle A \rangle$
z	$b := 0; \langle A \rangle$	$z \langle b \leftarrow 0 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 + 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) + 1 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 - 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) - 1 \rangle$	$\langle A \rangle$
z	if $b = 0$ then $\langle A_1 \rangle$ else $\langle A_2 \rangle$ end; $\langle A \rangle$	z	$\langle A_1 \rangle; \langle A \rangle$ falls $z(b) = 0$ $\langle A_2 \rangle; \langle A \rangle$ sonst
z	while $b \neq 0$ do $\langle A_1 \rangle$ end; $\langle A \rangle$	z	$\langle A \rangle$ falls $z(b) = 0$ $\langle A_1 \rangle; \text{while } b \neq 0$ sonst do $\langle A_1 \rangle$ end; $\langle A \rangle$

$(x := x + 1; y := y - 1; \text{while } y \neq 0 \text{ do } x := x + 1; y := y - 1 \text{ end}; \varepsilon, z_2)$
 $z_2(x) = 4, z_2(y) = 1$



Übergangsfunktion τ für MINI

alter Zustand	alte Anweisungsfolge	neuer Zustand	neue Anweisungsfolge
z	$\varepsilon; \langle A \rangle$	z	$\langle A \rangle$
z	$b := 0; \langle A \rangle$	$z \langle b \leftarrow 0 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 + 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) + 1 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 - 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) - 1 \rangle$	$\langle A \rangle$
z	if $b = 0$ then $\langle A_1 \rangle$ else $\langle A_2 \rangle$ end; $\langle A \rangle$	z	$\langle A_1 \rangle; \langle A \rangle$ falls $z(b) = 0$ $\langle A_2 \rangle; \langle A \rangle$ sonst
z	while $b \neq 0$ do $\langle A_1 \rangle$ end; $\langle A \rangle$	z	$\langle A \rangle$ falls $z(b) = 0$ $\langle A_1 \rangle; \text{while } b \neq 0$ sonst do $\langle A_1 \rangle$ end; $\langle A \rangle$

$(y := y - 1; \text{while } y \neq 0 \text{ do } x := x + 1; y := y - 1 \text{ end}; \varepsilon, z_3)$

$z_3(x) = 5, z_3(y) = 1$



Übergangsfunktion τ für MINI

alter Zustand	alte Anweisungsfolge	neuer Zustand	neue Anweisungsfolge
z	$\varepsilon; \langle A \rangle$	z	$\langle A \rangle$
z	$b := 0; \langle A \rangle$	$z \langle b \leftarrow 0 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 + 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) + 1 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 - 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) - 1 \rangle$	$\langle A \rangle$
z	if $b = 0$ then $\langle A_1 \rangle$ else $\langle A_2 \rangle$ end; $\langle A \rangle$	z	$\langle A_1 \rangle; \langle A \rangle$ falls $z(b) = 0$ $\langle A_2 \rangle; \langle A \rangle$ sonst
z	while $b \neq 0$ do $\langle A_1 \rangle$ end; $\langle A \rangle$	z	$\langle A \rangle$ falls $z(b) = 0$ $\langle A_1 \rangle; \text{while } b \neq 0$ sonst do $\langle A_1 \rangle$ end; $\langle A \rangle$

(while $y \neq 0$ do $x := x + 1; y := y - 1$ end; ε, z_4)

$z_4(x) = 5, z_4(y) = 0$



Übergangsfunktion τ für MINI

alter Zustand	alte Anweisungsfolge	neuer Zustand	neue Anweisungsfolge
z	$\varepsilon; \langle A \rangle$	z	$\langle A \rangle$
z	$b := 0; \langle A \rangle$	$z \langle b \leftarrow 0 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 + 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) + 1 \rangle$	$\langle A \rangle$
z	$b_1 := b_2 - 1; \langle A \rangle$	$z \langle b_1 \leftarrow z(b_2) - 1 \rangle$	$\langle A \rangle$
z	if $b = 0$ then $\langle A_1 \rangle$ else $\langle A_2 \rangle$ end; $\langle A \rangle$	z	$\langle A_1 \rangle; \langle A \rangle$ falls $z(b) = 0$ $\langle A_2 \rangle; \langle A \rangle$ sonst
z	while $b \neq 0$ do $\langle A_1 \rangle$ end; $\langle A \rangle$	z	$\langle A \rangle$ falls $z(b) = 0$ $\langle A_1 \rangle; \text{while } b \neq 0$ sonst do $\langle A_1 \rangle$ end; $\langle A \rangle$

(ε, Z_4)

$$z_4(x) = 5, z_4(y) = 0$$



- ... Maschinenmodell zur Beschreibung der Zustandsänderungen aller Variablen
- Sei Z_p die Menge aller möglichen Zustände
 $Z_p \subseteq W(v_1) \times W(v_2) \times \dots \times W(v_n)$, wobei v_1, \dots, v_n die im Programm auftretenden Variablen und $W(v_i)$ der jeweilige Wertebereich der Variable v_i ist.

Jede Anweisung kann dann als Funktion gesehen werden, die von einem Zustand in einem anderen abbildet:

$$F[\text{Anweisung}] : Z_p \rightarrow Z_p$$

- ⇒ F ist die *semantische Funktion*. Sie entspricht der Abstraktion von der Wirkung von Programmen auf Variablenbelegungen, hin zu der Wirkung von Einzelanweisungen und Anweisungsfolgen auf die Variablenbelegungen.



- Zur Vorbereitung auf das Übungsblatt ein kleines Beispiel zu Anweisungssequenzen:

$x := x + 1; y := y - 1$

Hieraus ergibt sich die folgende semantische Funktion F :

$$F[x := x + 1; y := y - 1](z) = F[y := y - 1](F[x := x + 1](z))$$

- ... an dieser Stelle endet das Beispiel, denn die semantischen Funktion für eine Zuweisung soll im Selbststudium, das heißt im Rahmen des aktuellen Übungsblatts, definiert werden.



- Zur Vorbereitung auf das Übungsblatt noch ein kleines Beispiel zu Schleifen:

WHILE $y \neq 0$ DO $\langle A \rangle$ END

Hieraus ergibt sich die folgende semantische Funktion F:

$$F[\text{WHILE } y \neq 0 \text{ DO } \langle A \rangle \text{ END}](z) = \begin{cases} z & \forall z \in Z_p : z(y) = 0 \\ F[\text{WHILE } y \neq 0 \text{ DO } \langle A \rangle \text{ END}](F[\langle A \rangle]z) & \text{sonst} \end{cases}$$



- Meist für imperative Programmiersprachen
- Bekannteste Vertreter: wp-Kalkül, Hoare-Kalkül
- Statt konkreter Zustände: Eigenschaften von Zuständen oder Aussagen über Zustände.



Schwächste Vorbedingung

- Wenn vor der Anweisung S , die Bedingung P gilt, so gilt nach ihr die Bedingung Q .

$\{P\}$

S

$\{Q\}$

- Beispiel:

$\{x = N\}$

$x := x + 1$

$\{x = N + 1\}$



Schwächste Vorbedingung

- Problem: Zu einer Nachbedingung kann es viele verschiedene Vorbedingungen geben:

$\{?\}$

$x := x + y;$

$\{x > 0\}$

- Gültige Vorbedingungen:

$\{x \geq -3 \wedge y > 3\}$

$\{x + y > 0\}$

$\{x > 0 \wedge y > 0\}$



Schwächste Vorbedingung

- Gesucht ist diejenige Vorbedingung P , für die gilt: Jede Vorbedingung P' für S , unter der nach Ausführung von S die Nachbedingung Q gilt, impliziert P
- Wir schreiben auch: $P' \rightarrow P$
- P ist schwächer als P' (da spezieller) und P' ist stärker als P (da allgemeiner)
- Solch eine schwächste Vorbedingung heisst: $wp(S, Q)$



Anwendung Schwächste Vorbedingung

- Gibt es zu einem gegebenen Programm S , spezifizierte Vor- und Nachbedingungen P_{SPEC} und Q_{SPEC} , sowie die Bedingungen P und Q , so dass $P_{SPEC} \rightarrow P$, $P\{S\}Q$, und $Q \rightarrow Q_{SPEC}$ gilt, so ist das Programm korrekt.
- Finde zu einem Programm S und einer Nachbedingung Q_{SPEC} die schwächste Vorbedingung $wp(S, Q_{SPEC})$. Das Programm ist dann korrekt, wenn $P_{SPEC} \rightarrow wp(S, Q_{SPEC})$ gilt. Statt Q_{SPEC} darf auch ein stärkeres Q gewählt werden.



Schwächste Vorbedingung

- Leere Anweisung:
 $wp(\varepsilon, Q) \equiv Q$



Schwächste Vorbedingung

- Leere Anweisung:

$$wp(\varepsilon, Q) \equiv Q$$

- Sequenz:

$$wp(A_1; A_2, Q) \equiv wp(A_1, wp(A_2, Q))$$



Schwächste Vorbedingung

- Leere Anweisung:

$$wp(\varepsilon, Q) \equiv Q$$

- Sequenz:

$$wp(A_1; A_2, Q) \equiv wp(A_1, wp(A_2, Q))$$

- Bedingte Verzweigung:

$$wp(\text{IF } B \text{ THEN } \langle A_1 \rangle \text{ ELSE } \langle A_2 \rangle \text{ END, } Q) \equiv$$

$$(B \wedge wp(\langle A_1 \rangle, Q)) \vee (\neg B \wedge wp(\langle A_2 \rangle, Q))$$



Schwächste Vorbedingung

- Leere Anweisung:

$$wp(\varepsilon, Q) \equiv Q$$

- Sequenz:

$$wp(A_1; A_2, Q) \equiv wp(A_1, wp(A_2, Q))$$

- Bedingte Verzweigung:

$$\begin{aligned} wp(\text{IF } B \text{ THEN } \langle A_1 \rangle \text{ ELSE } \langle A_2 \rangle \text{ END, } Q) &\equiv \\ (B \wedge wp(\langle A_1 \rangle, Q)) \vee (\neg B \wedge wp(\langle A_2 \rangle, Q)) \end{aligned}$$

- Zuweisung:

$$wp(x := V, Q(x)) \equiv Q(x \rightarrow V)$$



Schwächste Vorbedingung

- Leere Anweisung:

$$wp(\varepsilon, Q) \equiv Q$$

- Sequenz:

$$wp(A_1; A_2, Q) \equiv wp(A_1, wp(A_2, Q))$$

- Bedingte Verzweigung:

$$\begin{aligned} wp(\text{IF } B \text{ THEN } \langle A_1 \rangle \text{ ELSE } \langle A_2 \rangle \text{ END, } Q) &\equiv \\ (B \wedge wp(\langle A_1 \rangle, Q)) \vee (\neg B \wedge wp(\langle A_2 \rangle, Q)) \end{aligned}$$

- Zuweisung:

$$wp(x := V, Q(x)) \equiv Q(x \rightarrow V)$$

- Schleife:

$$\begin{aligned} wp(\text{WHILE } B \text{ DO } \langle A \rangle \text{ END, } Q) &\equiv \\ (\neg B \wedge Q) \vee ((B \wedge I) \wedge (\neg B \wedge I \rightarrow Q) \wedge (B \wedge I \rightarrow wp(\langle A \rangle, I))) \end{aligned}$$

I nennt man die Schleifeninvariante



Schwächste Vorbedingung - Beispiele

- Zuweisung: $y = 2$ muss vor $x := y + 1$ gelten, damit anschließend $x = 3$ gilt:

$$wp(x := y + 1, x = 3) \equiv (y + 1 = 3) \equiv (y = 2)$$

- Bedingte Verzweigung: Das folgende Programm liefert für *jede* Variablenbelegung die Nachbedingung $x = |y|$.

$$\begin{aligned} wp(\text{IF } y < 0 \text{ THEN } x := -y \text{ ELSE } x := y \text{ END}, x = |y|) &\equiv \\ (y < 0 \wedge wp(x := -y, x = |y|)) \vee (y \geq 0 \wedge wp(x := y, x = |y|)) &\equiv \\ (y < 0 \wedge -y = |y|) \vee (y \geq 0 \wedge y = |y|) &\equiv \\ y < 0 \vee y \geq 0 &\equiv \\ \text{true} \end{aligned}$$



Schwächste Vorbedingung - Beispiele

- Sei $Q \equiv x \leq n \wedge t = x^i$ die Nachbedingung für das Programm
 $t := t * x; i := i + 1$:

$$wp(t := t * x; i := i + 1, i \leq n \wedge t = x^i) \equiv$$

$$wp(t := t * x, wp(i := i + 1, i \leq n \wedge t = x^i)) \equiv$$

$$wp(t := t * x, i + 1 \leq n \wedge t = x^{i+1}) \equiv$$

$$i + 1 \leq n \wedge t * x = x^{i+1} \equiv$$

$$i < n \wedge t = x^i$$

- Die Vorbedingung im zweiten Teil stimmt mit der Nachbedingung Q trotz der Zuweisung überein und wird als Invariante bezeichnet.



Schwächste Vorbedingung - Beispiele

- Auf den folgenden beiden Folien suchen wir die schwächste Vorbedingung für eine Schleife:

```
WHILE( i < n ) DO
  t := t * x
  i := i + 1
END
```

- Dazu betrachten wir zuerst nur die einzelnen Terme, ohne den gesamten logischen Ausdruck für die wp der Schleife am Ende zu notieren.



Schwächste Vorbedingung - Beispiele

- Die Nachbedingung soll, analog zur vorletzten Folie, wie folgt lauten:

$$\begin{aligned}Q &\equiv i = n \wedge t = x^i \\ &\equiv t = x^n.\end{aligned}$$

- kein Schleifendurchlauf

$$\begin{aligned}(\neg B \wedge Q) &\equiv i \geq n \wedge t = x^n \\ &\equiv t = x^n \\ &\equiv Q.\end{aligned}$$

- vor Schleifeneintritt:

$$\begin{aligned}(B \wedge I) &\equiv (i < n) \wedge (i \leq n \wedge t = x^i) \\ &\equiv i < n \wedge t = x^i\end{aligned}$$



Schwächste Vorbedingung - Beispiele

- Schleifenende:

$$\begin{aligned}(\neg B \wedge I) &\equiv (i \geq n) \wedge (i \leq n \wedge t = x^i) \\ &\equiv i = n \wedge t = x^i \\ &\equiv Q\end{aligned}$$

$(\neg B \wedge I)$ impliziert also die Nachbedingung Q .

- Schleifendurchlauf (siehe vorher):

$$wp(t := t * x; i := i + 1, i \leq n \wedge t = x^i) \equiv i < n \wedge t = x^i$$

$(B \wedge I)$ impliziert also wieder die Vorbedingung für einen weiteren Schleifendurchlauf.



Schwächste Vorbedingung - Programmkonstruktion

- Wie aus der Vorlesung bekannt, kann man mittels axiomatischer Semantik auch Programme konstruieren. Dabei geht man von einer Nachbedingung aus und fügt sukzessive Anweisungen zum Programm hinzu, so dass die schwächste Vorbedingung allgemeiner ist als die aktuelle Zwischenbedingung oder spezialisiert die Zwischenbedingungen Q zu Q' , so dass $Q' \rightarrow Q$ gilt.
- Im Folgenden finden Sie ein Beispiel zur Programmkonstruktion, welches Ihnen als Beispiel beim weiteren Selbststudium helfen soll.



Ziel: Konstruiere ein Programm, das den Binomialkoeffizienten b aus n und k berechnet: $\binom{n}{k} = \prod_{i=1}^k \frac{n-(k-i)}{i}$.

- $Q_{SPEC} \equiv b = \binom{n}{k}$



Ziel: Konstruiere ein Programm, das den Binomialkoeffizienten b aus n und k berechnet: $\binom{n}{k} = \prod_{i=1}^k \frac{n-(k-i)}{i}$.

- $Q_{SPEC} \equiv b = \binom{n}{k}$
- wähle spezielleres $Q \equiv b = \binom{n}{k} \wedge 0 \leq k \leq n$



Ziel: Konstruiere ein Programm, das den Binomialkoeffizienten b aus n und k berechnet: $\binom{n}{k} = \prod_{i=1}^k \frac{n-(k-i)}{i}$.

- $Q_{SPEC} \equiv b = \binom{n}{k}$
- wähle spezielleres $Q \equiv b = \binom{n}{k} \wedge 0 \leq k \leq n$
- *Idee:* löse iterativ mittels Schleife und suche nach Schleifeninvariante I mit $(\neg B \wedge I \rightarrow Q) \wedge (B \wedge I \rightarrow wp(A, I))$



Ziel: Konstruiere ein Programm, das den Binomialkoeffizienten b aus n und k berechnet: $\binom{n}{k} = \prod_{i=1}^k \frac{n-(k-i)}{i}$.

- $Q_{SPEC} \equiv b = \binom{n}{k}$
- wähle spezielleres $Q \equiv b = \binom{n}{k} \wedge 0 \leq k \leq n$
- *Idee:* löse iterativ mittels Schleife und suche nach Schleifeninvariante I mit $(\neg B \wedge I \rightarrow Q) \wedge (B \wedge I \rightarrow wp(A, I))$
- wähle als Schleifeninvariante $I \equiv b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n$ und als Abbruchbedingung der while-Schleife $B \equiv i < k$



Ziel: Konstruiere ein Programm, das den Binomialkoeffizienten b aus n und k berechnet: $\binom{n}{k} = \prod_{i=1}^k \frac{n-(k-i)}{i}$.

- $Q_{SPEC} \equiv b = \binom{n}{k}$
 - wähle spezielleres $Q \equiv b = \binom{n}{k} \wedge 0 \leq k \leq n$
 - *Idee:* löse iterativ mittels Schleife und suche nach Schleifeninvariante I mit $(\neg B \wedge I \rightarrow Q) \wedge (B \wedge I \rightarrow wp(A, I))$
 - wähle als Schleifeninvariante $I \equiv b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n$ und als Abbruchbedingung der while-Schleife $B \equiv i < k$
- $\Rightarrow \neg B \wedge I \equiv (i \geq k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$



Ziel: Konstruiere ein Programm, das den Binomialkoeffizienten b aus n und k berechnet: $\binom{n}{k} = \prod_{i=1}^k \frac{n-(k-i)}{i}$.

- $Q_{SPEC} \equiv b = \binom{n}{k}$
 - wähle spezielleres $Q \equiv b = \binom{n}{k} \wedge 0 \leq k \leq n$
 - *Idee:* löse iterativ mittels Schleife und suche nach Schleifeninvariante I mit $(\neg B \wedge I \rightarrow Q) \wedge (B \wedge I \rightarrow wp(A, I))$
 - wähle als Schleifeninvariante $I \equiv b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n$ und als Abbruchbedingung der while-Schleife $B \equiv i < k$
- $\Rightarrow \neg B \wedge I \equiv (i \geq k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n) \equiv (b = \binom{n}{i} \wedge 0 \leq i = k \leq n)$



Ziel: Konstruiere ein Programm, das den Binomialkoeffizienten b aus n und k berechnet: $\binom{n}{k} = \prod_{i=1}^k \frac{n-(k-i)}{i}$.

- $Q_{SPEC} \equiv b = \binom{n}{k}$
 - wähle spezielleres $Q \equiv b = \binom{n}{k} \wedge 0 \leq k \leq n$
 - *Idee:* löse iterativ mittels Schleife und suche nach Schleifeninvariante I mit $(\neg B \wedge I \rightarrow Q) \wedge (B \wedge I \rightarrow wp(A, I))$
 - wähle als Schleifeninvariante $I \equiv b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n$ und als Abbruchbedingung der while-Schleife $B \equiv i < k$
- $\Rightarrow \neg B \wedge I \equiv (i \geq k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n) \equiv (b = \binom{n}{i} \wedge 0 \leq i = k \leq n)$
 $\rightarrow b = \binom{n}{k} \wedge 0 \leq k \leq n \equiv Q$



Ziel: Konstruiere ein Programm, das den Binomialkoeffizienten b aus n und k berechnet: $\binom{n}{k} = \prod_{i=1}^k \frac{n-(k-i)}{i}$.

- $Q_{SPEC} \equiv b = \binom{n}{k}$
 - wähle spezielleres $Q \equiv b = \binom{n}{k} \wedge 0 \leq k \leq n$
 - *Idee:* löse iterativ mittels Schleife und suche nach Schleifeninvariante I mit $(\neg B \wedge I \rightarrow Q) \wedge (B \wedge I \rightarrow wp(A, I))$
 - wähle als Schleifeninvariante $I \equiv b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n$ und als Abbruchbedingung der while-Schleife $B \equiv i < k$
- $\Rightarrow \neg B \wedge I \equiv (i \geq k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n) \equiv (b = \binom{n}{i} \wedge 0 \leq i = k \leq n)$
 $\rightarrow b = \binom{n}{k} \wedge 0 \leq k \leq n \equiv Q$
- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i) / (i + 1); i := i + 1$



Anwendung axiomatischer Semantik - Binomialkoeffizient

- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i) / (i + 1); i := i + 1$
- $wp(b := b * (n - i) / (i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i) / (i + 1); i := i + 1$
- $wp(b := b * (n - i) / (i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv wp(b := b * (n - i) / (i + 1), wp(i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i)/(i + 1); i := i + 1$
- $wp(b := b * (n - i)/(i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv wp(b := b * (n - i)/(i + 1), wp(i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := b * (n - i)/(i + 1), b = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n)$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i)/(i + 1); i := i + 1$
- $wp(b := b * (n - i)/(i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv wp(b := b * (n - i)/(i + 1), wp(i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := b * (n - i)/(i + 1), b = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n)$
 $\equiv b \frac{n-i}{i+1} = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i)/(i + 1); i := i + 1$
- $wp(b := b * (n - i)/(i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv wp(b := b * (n - i)/(i + 1), wp(i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := b * (n - i)/(i + 1), b = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n)$
 $\equiv b \frac{n-i}{i+1} = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge 0 \leq i + 1 \leq k \leq n$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i)/(i + 1); i := i + 1$
- $wp(b := b * (n - i)/(i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv wp(b := b * (n - i)/(i + 1), wp(i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := b * (n - i)/(i + 1), b = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n)$
 $\equiv b \frac{n-i}{i+1} = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge -1 \leq i < k \leq n$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i)/(i + 1); i := i + 1$
- $wp(b := b * (n - i)/(i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv wp(b := b * (n - i)/(i + 1), wp(i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := b * (n - i)/(i + 1), b = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n)$
 $\equiv b \frac{n-i}{i+1} = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge -1 \leq i < k \leq n$
- $B \wedge I$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i)/(i + 1); i := i + 1$
- $wp(b := b * (n - i)/(i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv wp(b := b * (n - i)/(i + 1), wp(i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := b * (n - i)/(i + 1), b = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n)$
 $\equiv b \frac{n-i}{i+1} = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge -1 \leq i < k \leq n$
- $B \wedge I$
 $\equiv i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i)/(i + 1); i := i + 1$
- $wp(b := b * (n - i)/(i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv wp(b := b * (n - i)/(i + 1), wp(i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := b * (n - i)/(i + 1), b = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n)$
 $\equiv b \frac{n-i}{i+1} = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge -1 \leq i < k \leq n$
- $B \wedge I$
 $\equiv i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge 0 \leq i < k \leq n$



- Suche nach Anweisungen A , so dass $B \wedge I \rightarrow wp(A, I)$ gilt.
- wähle für A : $b := b * (n - i)/(i + 1); i := i + 1$
- $wp(b := b * (n - i)/(i + 1); i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv wp(b := b * (n - i)/(i + 1), wp(i := i + 1, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := b * (n - i)/(i + 1), b = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n)$
 $\equiv b \frac{n-i}{i+1} = \binom{n}{i+1} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge 0 \leq i + 1 \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge -1 \leq i < k \leq n$
- $B \wedge I$
 $\equiv i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n$
 $\equiv b = \binom{n}{i} \wedge 0 \leq i < k \leq n \rightarrow b = \binom{n}{i} \wedge -1 \leq i < k \leq n$



- Unser Programm P' lautet bis hierher:

```
WHILE  $i < k$  DO
```

```
 $b := b * (n - i) / (i + 1);$ 
```

```
 $i := i + 1$ 
```

```
END
```

... mit $B \equiv (i < k, l) \equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$

und $Q \equiv (b = \binom{n}{k} \wedge 0 \leq k \leq n)$



- Unser Programm P' lautet bis hierher:

```
WHILE  $i < k$  DO
```

```
 $b := b * (n - i) / (i + 1);$ 
```

```
 $i := i + 1$ 
```

```
END
```

... mit $B \equiv (i < k)$, $I \equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$

und $Q \equiv (b = \binom{n}{k} \wedge 0 \leq k \leq n)$

- Suche schwächste Vorbedingung von P' :

$wp(P', Q) \equiv (\neg B \wedge Q) \vee ((B \wedge I) \wedge (\neg B \wedge I \rightarrow Q) \wedge (B \wedge I \rightarrow wp(A, I))).$



- Unser Programm P' lautet bis hierher:

```
WHILE  $i < k$  DO
```

```
   $b := b * (n - i) / (i + 1);$ 
```

```
   $i := i + 1$ 
```

```
END
```

... mit $B \equiv (i < k, I) \equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$

und $Q \equiv (b = \binom{n}{k} \wedge 0 \leq k \leq n)$

- Suche schwächste Vorbedingung von P' :

$wp(P', Q) \equiv (\neg B \wedge Q) \vee ((B \wedge I) \wedge (\neg B \wedge I \rightarrow Q) \wedge (B \wedge I \rightarrow wp(A, I)))$.

- Wir haben bereits gezeigt:

$\neg B \wedge I \rightarrow Q$ und $(B \wedge I) \rightarrow wp(A, I)$.



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (b = \binom{n}{i} \wedge 0 \leq i < k \leq n)$



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (b = \binom{n}{i} \wedge 0 \leq i < k \leq n)$
 $\equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
- erweitere Programm mit $b := 1; i := 0$



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (b = \binom{n}{i} \wedge 0 \leq i < k \leq n)$
 $\equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
- erweitere Programm mit $b := 1; i := 0$
- $wp(b := 1; i := 0; wp(P', Q))$



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (b = \binom{n}{i} \wedge 0 \leq i < k \leq n)$
 $\equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
- erweitere Programm mit $b := 1; i := 0$
- $wp(b := 1; i := 0; wp(P', Q))$
 $\equiv wp(b := 1, wp(i := 0, wp(P', Q)))$



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (b = \binom{n}{i} \wedge 0 \leq i < k \leq n)$
 $\equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
- erweitere Programm mit $b := 1; i := 0$
- $wp(b := 1; i := 0; wp(P', Q))$
 $\equiv wp(b := 1, wp(i := 0, wp(P', Q)))$
 $\equiv wp(b := 1, wp(i := 0, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (b = \binom{n}{i} \wedge 0 \leq i < k \leq n)$
 $\equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
- erweitere Programm mit $b := 1; i := 0$
- $wp(b := 1; i := 0; wp(P', Q))$
 $\equiv wp(b := 1, wp(i := 0, wp(P', Q)))$
 $\equiv wp(b := 1, wp(i := 0, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := 1, b = \binom{n}{0} \wedge 0 \leq 0 \leq k \leq n)$



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (b = \binom{n}{i} \wedge 0 \leq i < k \leq n)$
 $\equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
- erweitere Programm mit $b := 1; i := 0$
- $wp(b := 1; i := 0; wp(P', Q))$
 $\equiv wp(b := 1, wp(i := 0, wp(P', Q)))$
 $\equiv wp(b := 1, wp(i := 0, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := 1, b = \binom{n}{0} \wedge 0 \leq 0 \leq k \leq n)$
 $\equiv 1 = \binom{n}{0} \wedge 0 \leq 0 \leq k \leq n$



- $wp(P', Q) \equiv (\neg B \wedge Q) \vee (B \wedge I)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (i < k \wedge b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
 $\equiv (i \geq k \wedge b = \binom{n}{k} \wedge 0 \leq k \leq n) \vee (b = \binom{n}{i} \wedge 0 \leq i < k \leq n)$
 $\equiv (b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n)$
- erweitere Programm mit $b := 1; i := 0$
- $wp(b := 1; i := 0; wp(P', Q))$
 $\equiv wp(b := 1, wp(i := 0, wp(P', Q)))$
 $\equiv wp(b := 1, wp(i := 0, b = \binom{n}{i} \wedge 0 \leq i \leq k \leq n))$
 $\equiv wp(b := 1, b = \binom{n}{0} \wedge 0 \leq 0 \leq k \leq n)$
 $\equiv 1 = \binom{n}{0} \wedge 0 \leq 0 \leq k \leq n$
 $\equiv 0 \leq k \leq n$



- Das finale Programm:

```
// Vorbedingung:  $0 \leq k \leq n$   
b := 1;  
i := 0;  
while i < k do  
    b := b * (n-i) / (i+1);  
    i := i + 1  
end  
// Nachbedingung: b = n ueber k
```