Grundlagen der Programmierung (Vorlesung 13)

Ralf Möller, FH-Wedel

- Vorige Vorlesung
  - Verifikation von Anweisungen und Anweisungsfolgen
  - Fallunterscheidung
- Inhalt dieser Vorlesung
  - Fortsetzung: Schleifen
  - Entwicklung von Programmen aus der Spezifikation
- Lernziele
  - Grundlagen der systematischen Programmentwicklung

4.2.4 Schleifenanweisungen

Wiederholung

while **–Schleife**	aus einer Bedingung und einer Anweisung kann eine while –Schleife aufgebaut werden

Syntax	while *Bedingung* do
	    Anweisung
	end while

Bedeutung	informell
(1)	die *Bedingung* wird ausgewertet
(2a)	ist das Resultat false, so wird die Ausführung des Programms hinter der Schleife fortgesetzt
(2b)	ist das Resultat true, so wird die Anweisung (der Schleifenrumpf) ausgeführt und anschließend die Schleife erneut ausgeführt.
$\hookrightarrow$	der Schleifenrumpf wird $0, 1, 2, \ldots$ mal ausgeführt.
$\hookrightarrow$	in Programmiersprachen gibt es häufig noch weitere Schleifenarten (repeat –, for –Schleifen). Diese können alle auf die while –Schleife zurückgeführt werden.

Konstruktion	von `while` –Schleifen
Initialisierungs-anweisungen	für Variable, die in der Schleife verwendet werden, häufig $\hookrightarrow$ eine Laufvariable $\hookrightarrow$ eine Variable für das Resultat
Abbruchkriterium	eine Bedingung, die bestimmt, wie lange der Schleifenrumpf wiederholt ausgeführt wird. Diese Bedingung muß mindestens eine Variable enthalten, die im Schleifenrumpf verändert wird, häufig $\hookrightarrow$ Test der Laufvariablen gegen einen Endewert
Rumpf	enthält Anweisungen zur Veränderung von Variablen, häufig $\hookrightarrow$ Inkrementieren der Laufvariable $\hookrightarrow$ Akkumulieren des Resultats in der zugehörigen Variablen

Endlosschleife	Wird in einem Schleifenrumpf keine Variable verändert, die im Abbruchkriterium vorkommt, so erhält man eine Endlosschleife
Konstruktion	aus Spezifikationen mit Quantoren
$\hookrightarrow$	an Quantoren gebundene Variablen werden zu Laufvariablen
$\hookrightarrow$	aus Bereichsgrenzen werden Initialisierung und Abbruchkriterium abgeleitet

Beweisregel für while –Schleifen

falls

(1) $\{I \wedge B\}\ \ S\ \ \{I\}$

dann gilt $\{I\}\ \ \mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{end\ while}\ \ \ \{I \wedge \neg B\}$

Invariante I ist die Schleifeninvariante, d.h. eine Eigenschaft, die

- $\hookrightarrow$ vor der Ausführung der Schleife
- $\hookrightarrow$ vor jeder Ausführung des Rumpfes
- $\hookrightarrow$ nach jeder Ausführung des Rumpfes
- $\hookrightarrow$ nach der Ausführung der Schleife

gilt

Konstruieren der Invarianten aus der Spezifikation

Faustregel Invariante ist eine Verallgemeinerung der Anfangs- und Endesituation

hier wird noch nichts über die Terminierung ausgesagt

$\hookrightarrow$ Regel noch unvollständig

While-Schleife (1)

- Initialisierung
- while Bedingung do

  Rumpf

  end while

Abbruchkriterium

Zuweisung an Schleifenvariable

While-Schleife (2) Summe berechnen

- var f : array [0..n-1] of R;
  i : N_0;
  s : R
- i,s := 0,0 ;
- while i < n do
  i,s := i+1, s + f[i]
  end
- Wo können Fehler gemacht werden?

Zählschleife (Muster)

- var   i : N0;

- { i □ n }
- i   := 0   ;
- while   i < n do
      i   := i+1
  end while
- { i = n }

Spezifikation: Summe berechnen

- var
  - f : array [0..n-1] of Z;
  - s : Z
- { }
-
-
- { s = $\sum_{j=0}^{n-1} f[j]$ }

Realisierung

- var i : N0;
  f : array [0..n-1] of Z;
  s : Z
- { i □ n }
- i , s := 0, 0 ;
- while i < n do
  i , s := i+1, s + f[i]
  end while
- { i = n ∧ s = $\sum_{j=0}^{n-1} f[j]$ }

Beweisregel: Beispiel

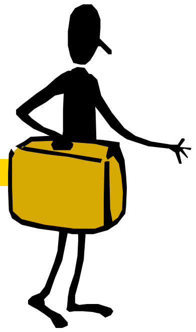
- var $i : N_0$;
- $\{ i \leq n \}$
- $i := 0$; $\{ i \leq n \}$
- while $i < n$ do
  $\{ i < n \wedge i \leq n \}$ $i := i+1$ $\{ i \leq n \}$
  end while
- $\{ i \leq n \wedge i \geq n \}$
- Invariante: $I = i \leq n$
- Bedingung: $B = i < n$, $\neg B = i \geq n$ $(I \wedge \neg B \equiv i = n)$

Terminierung	zusätzliche Bedingungen
falls	
Invariante	$\{I \wedge B\}\ \ S\ \ \{I\}$
Fortschritt	$\{I \wedge B \wedge t > T\}\ \ S\ \ \{t = T\}$
Beschränkung	$I \wedge t \leq 0 \Rightarrow \neg B$
dann	gilt die Nachbedingung $I \wedge \neg B$ und die Schleife terminiert: $\{I\}$ while B do S end while $\{I \wedge \neg B\}$
Variante	t ist eine ganzzahlige Funktion T ist eine Konstante
$\hookrightarrow$	wenn I erfüllt ist vor der Ausführung von S, dann auch nach der Ausführung von S
$\hookrightarrow$	$\{t > T\}\ \ S\ \ \{t = T\}$ beschreibt den Fortschritt der Schleife, da t mindestens um 1 verkleinert wird
$\hookrightarrow$	$I \wedge t \leq 0 \Rightarrow \neg B$: wird $t \leq 0$ so terminiert die Schleife
$\hookrightarrow$	Anfangswert von t liefert obere Grenze für die Anzahl der Schleifendurchläufe
Beschränkung	ist gleichwertig mit der Bedingung $I \wedge B \Rightarrow t > 0$

Wie wird die Terminierung gezeigt?

- Beispiel:
  - $i := 0;\ \text{while } i < n \text{ do } i := i + 1 \text{ end while}$
- Angabe einer Variante: Wir nehmen $(n - i)$
- Fortschritt: $\{ n - i - 1 \}\ i := i + 1\ \{ n - i = T \}$
- Einsetzen der rechten Seite der Zuweisung in Nachbedingung ergibt:
  - $n - (i+1) = T \equiv n - i = T+1$ also $n - i > T$
- Beschränkung: Zu Zeigen: $(I \wedge (t \Box 0)) \rightarrow \neg B$ gültig
  - $\underbrace{(i \Box n)}_{I} \wedge \underbrace{(n - i \Box 0)}_{(t \Box 0)} \equiv i = n \models n \Box i \equiv \neg(i < n) = \neg B$
    und damit $(I \wedge (t \Box 0)) \rightarrow \neg B$ gültig

Zusammenfassung, Kernpunkte



- Logik und die systematische Entwicklung von Programmen
- Zuweisung
- Kontrollstrukturen,
  - Fallunterscheidung
  - Schleifen

Was kommt beim nächsten Mal?

- Terminierung von Schleifen
- Definition von Funktionen
- Rekursion