marc-oliver pahl

Rekursion vs. Iteration

Anhand einer Beispielfunktion werden das Konzept der Rekursion und das der Iteration, sowie Korrektheitsbeweis-verfahren (Induktion, Floyd) erläutert.

28.11.2003

Ablauf der Rekursion

marc-oliver pahl

```
sum_rek(n){
  if (n == 0) then return 0; fi
  return ( n + summe_rek( n-1 ) );
}
```

return: In Folge eines Aufrufs des return-statements wird der Rest des Codes nicht mehr ausgeführt.

Betrachten wir den Ablauf der Rekursion bei Eingabe **n=3**:

sum_rek(3) => return (3 + sum_rek(2))
=> return (3 + (return (2 + sum_rek(1))))
=> return (3 + (return (2 + (return (1 + sum_rek(0))))))
=> return (3 + (return (2 + (return (1 + (return 0))))))
=> return (3 + (return (2 + (return (1 + 0)))))
=> return (3 + (return (2 + 1)))
=> return (3 + 3)
=> 6

Rekursion vs. Iteration

marc-oliver pahl

$$sum(n) = \sum_{i=0}^{n} i$$

Anforderung: $n \in N$

Zusicherung: Das Ergebnis ist $\sum_{i=0}^{n} i$

```
sum_rek(n){
  if (n == 0) then return 0; fi
  return ( n + summe_rek( n-1 ) );
}
```

```
sum_iter(n){
  k := n; erg := 0;
  while ( k >= 0 ) do {
    erg := k + erg;
    k := k - 1;
  } od
  return erg;
}
```

Ablauf der Iteration

marc-oliver pahl

```
sum_iter(n){
  k := n; erg := 0;
  while ( k >= 0 ) do { erg := k + erg; k := k - 1; } od
  return erg;
}
```

Betrachten wir den Ablauf der Iteration bei Eingabe **n=3**:

sum_iter(3) => [erg = 0; k = 3;]
=> < (k >= 0)? > Ja =>=> [erg = 3; k =2;]
=> < (k >= 0)? > Ja =>=> [erg = 5; k =1;]
=> < (k >= 0)? > Ja =>=> [erg = 6; k =0;]
=> < (k >= 0)? > Ja =>=> [erg = 6; k =-1;]
=> < (k >= 0)? > Nein => return erg;
=> 6

Korrektheitsbeweis der Rekursion (vollst. Induktion)

marc-oliver pahl

```
sum_rek(n){
  if (n == 0) then return 0; fi
  return ( n + summe_rek( n-1 ) );
}
```

Vollständige Induktion:

InduktionsAnfang: $sum\_rek(0) = 0 = \sum_{i=0}^{0} i$

InduktionsVoraussetzung: $sum\_rek(n-1) = \sum_{i=0}^{n-1} i$

InduktionsSchritt:

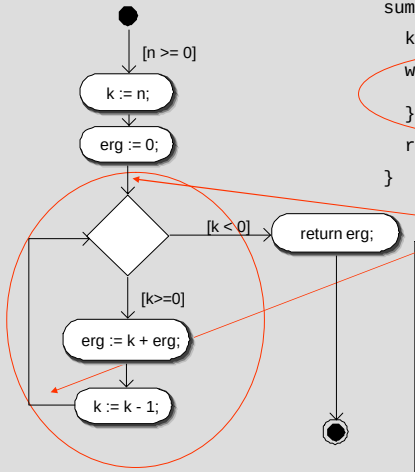
$$sum\_rek(n) \overset{n.V.}{=} n + sum\_rek(n-1) = n + \sum_{i=0}^{n-1} i = \sum_{i=0}^{n} i$$

□

Mit n > (n-1) > ... > 0 folgt die Terminierung des Verfahrens.

Korrektheitsbeweis der Iteration (Floyd)

marc-oliver pahl




```
sum_iter(n){
  k := n; erg := 0;
  while ( k >= 0 ) do {
    erg := k + erg; k := k - 1;
  }od
  return erg;
}
```

Finden der **Invarianten**, die hier gelten muss:

„Was wollen wir?“ = „getan“ + „zu tun“

$$\sum_{i=0}^{n} i = \sum_{i=k+1}^{n} i + \sum_{i=0}^{k} i$$

erg

Was haben wir mit der Induktion gezeigt?

marc-oliver pahl

Es gibt ein **Element am Anfang**, für welches die **Behauptung gilt** (InduktionsAnfang).

Wenn die Behauptung für irgendein Element gilt (**InduktionsVoraussetzung**),

so **folgt daraus, dass sie auch für das Nachfolgeelement gilt** (InduktionsSchritt; zu zeigen!).

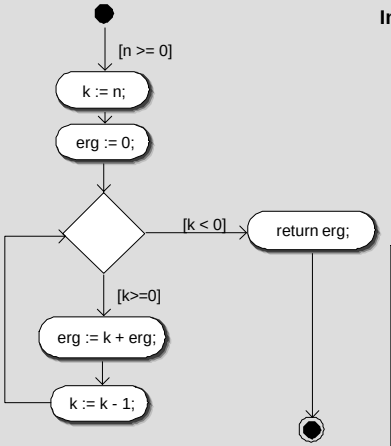
Daraus folgt, das die Behauptung für alle Elemente gilt, denn:

Aus dem ersten Element kann ich mit der Behauptung das zweite bestimmen und daraus das dritte u.s.w.

Dass die Behauptung für das Nachfolgeelement gilt haben wir bewiesen!

Korrektheitsbeweis der Iteration (Floyd)

marc-oliver pahl



Invariante:

$$\sum_{i=0}^{n} i = \text{erg} + \sum_{i=0}^{k} i$$

$$\sum_{i=k+1}^{n} i$$

(n-k gibt an, wie oft die Schleife schon durchlaufen worden ist)

Beispiel:

Wir suchen **sum_iter(3)** und haben die **Schleife genau einmal durchlaufen**:

$$sum\_iter(3) = \sum_{i=3}^{3} i + \sum_{i=0}^{2} i = (3) + (0+1+2)$$

Korrektheitsbeweis der Iteration (Floyd)

marc-oliver pahl

Invariante: $\sum_{i=0}^{n} i$ = erg + $\sum_{i=0}^{k} i$ [=INV((erg, k))]

1. Gilt die Invariante vor dem ersten Schleifendurchlauf?

$$\sum_{i=0}^{n} i = 0 + \sum_{i=0}^{n} i$$

2. Gilt die Invariante auch nach einem beliebigen Schleifendurchlauf noch?

$$INV((erg,k)) \wedge (k \geq 0) => INV((erg',k'))?$$

Korrektheitsbeweis der Iteration (Floyd)

marc-oliver pahl

Invariante: $\sum_{i=0}^{n} i$ = erg + $\sum_{i=0}^{k} i$ [=INV((erg, k))]

1. Gilt die Invariante vor dem ersten Schleifendurchlauf?

$$\sum_{i=0}^{n} i = 0 + \sum_{i=0}^{n} i$$

2. Gilt die Invariante auch nach einem beliebigen Schleifendurchlauf noch?

$$INV((erg,k)) \wedge (k \geq 0) => INV((erg',k'))!$$

3. Stimmt das Ergebnis, wenn die Invariante nach dem letzten Schleifendurchlauf gilt?

$$[\sum_{i=0}^{n} i + \sum_{i=0}^{-1} i = \sum_{i=0}^{n} i]$$

Die Terminierung folgt wieder aus n > n-1 > 0 > -1.

Korrektheitsbeweis der Iteration (Floyd)

marc-oliver pahl

2. Gilt die Invariante auch nach einem beliebigen Schleifendurchlauf noch?

$INV((erg,k)) \wedge (k \geq 0) => INV((erg',k'))?$

Vorher: $[\sum_{i=0}^{n} i = erg + \sum_{i=0}^{k} i]$

$(k \geq 0)$

=> $erg' + \sum_{i=0}^{k'} i$

=> $(k + erg) + \sum_{i=0}^{k'} i$

=> $(k + erg) + \sum_{i=0}^{k-1} i$

Nachher: $\underbrace{\sum_{i=k}^{n} i + \sum_{i=0}^{k-1} i = \sum_{i=0}^{n} i}$

Invariante: $[\sum_{i=0}^{n} i = erg' + \sum_{i=0}^{k'} i]$