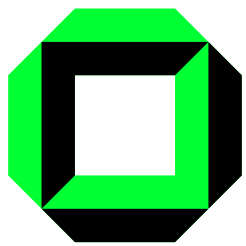
Info I – Übungsblatt 5

Joachim Breitner
mit Aufgaben von Christian Maier

5. Dezember 2005



Unser Programm heute

1. Übungsblatt 4
2. Hornerschema
3. Schleifeninvariante
4. Schleifen-Praxis
5. Fragen

Übungsblatt-Rückblick

. . . kommst nächste Woche. . .

Hornerschema

Das Hornerschema berechnet Polynome auf effizientere Art:
so wird das Polynom

$$2x^5 - x^4 - 4x^3 - 3x^2 + 2$$

nicht etwa so

$$2 \cdot x \cdot x \cdot x \cdot x \cdot x - x \cdot x \cdot x \cdot x - 4 \cdot x \cdot x \cdot x - 3 \cdot x \cdot x + 2$$

sondern so

$$((((2 \cdot x - 1) \cdot x - 4) \cdot x - 3) \cdot x) \cdot x + 2$$

berechnet.

Aufgabe zum Hornerschema

Aufgabe

Berechnet das Polynom $2x^5 - 8x^4 + x^3 - 2x^2 + 2x + 2$ für $x = 4$ einmal normal und einmal mit Hornerschema.
Gebt jeweils auch die Zahl der Additionen und Multiplikationen an.

Aufgabe zum Hornerschema

Aufgabe

Berechnet das Polynom $2x^5 - 8x^4 + x^3 - 2x^2 + 2x + 2$ für $x = 4$ einmal normal und einmal mit Hornerschema.
Gebt jeweils auch die Zahl der Additionen und Multiplikationen an.

Lösung

	Ergebnis	Additionen	Multiplikationen
ohne Hornerschema	42	5	14
mit Hornerschema	42	5	6

Lösung imperativ und ausführlich

Aufgabe

Polynom $2x^5 - 8x^4 + x^3 - 2x^2 + 2x + 2$, $x = 4$

Also: $a_5 = 2$, $a_4 = -8$, $a_3 = 1$, $a_2 = -2$, $a_1 = 2$, $a_0 = 2$

1. $y_6 = 0$
2. $y_5 = x \cdot y_6 + a_5 = 4 \cdot 0 + 2 = 2$
3. $y_4 = x \cdot y_5 + a_4 = 4 \cdot 2 - 8 = 0$
4. $y_3 = x \cdot y_4 + a_3 = 4 \cdot 0 + 1 = 1$
5. $y_2 = x \cdot y_3 + a_2 = 4 \cdot 1 - 2 = 2$
6. $y_1 = x \cdot y_2 + a_1 = 4 \cdot 2 + 2 = 10$
7. $y_0 = x \cdot y_1 + a_0 = 4 \cdot 10 + 2 = 42$

Hornerschema und O-Notation

Aufgabe

Gebt eine geschlossene Formel für die Zahl der Additionen und Multiplikationen bei der Berechnung eines Polynom vom Grad n mit und ohne Hornerschema sowie deren Komplexitätsklassen an.

Hornerschema und O-Notation

Aufgabe

Gebt eine geschlossene Formel für die Zahl der Additionen und Multiplikationen bei der Berechnung eines Polynom vom Grad n mit und ohne Hornerschema sowie deren Komplexitätsklassen an.

Lösung

- ohne Hornerschema:
  - Additionen: $n = O(n)$
  - Multiplikationen: $\sum_{i=1}^{n} i = \frac{n \cdot (n-1)}{2} = O(n^2)$
- mit Hornerschema:
  - Additionen: $n = O(n)$
  - Multiplikationen: $n = O(n)$

Sinn und Zweck

Schleifeninvarianten...

- sind Aussagen, die am Anfang und am Ende eines Schleifendurchlaufs gelten.
- helfen, die Korrektheit eines Programmes zu beweisen.
- beweist man meist durch Induktion. (Wiederholung nötig?)

Aufgabe

```
int a = In.readInt();
int b = In.readInt();
if (a > b) {
  int h = a;
  a = b;
  b = h;
}
int z = b;
int i = 1;
while ((z % a) != 0){ /*1*/
  z += b;             /*2*/
  i++;                /*3*/
}
```

Beweist oder widerlegt die folgendenen Aussagen:

1. $z = b \cdot i$ ist eine Schleifeninvariante für die `while`-Schleife
2. $\mathrm{LCM}(a, b) = \mathrm{LCM}(a, z)$ ist eine Schleifeninvariante für die `while`-Schleife
3. Die `while`-Schleife terminiert

Hinweise: Verwende geeignete Zusicherungen an den Positionen 1,2 und 3.

Lösung Aufgabe 1

```
int a = In.readInt();
int b = In.readInt();
if (a > b) {
  int h = a;
  a = b;
  b = h;
}
int z = b;
int i = 1;
while ((z % a) != 0){ /*1*/
  z += b;             /*2*/
  i++;                /*3*/
}
```

Induktionsanfang: $(n = 1)$
Beim ersten Schleifendurchlauf gilt die Schleifeninvariante, denn $z = b = b \cdot 1 = b \cdot i$.
Induktionsannahme:
Beim n-ten Durchlauf gelte bei Schleifenbegin (/*1*/): $z = b \cdot i$
Inuktionsschluss: $(n \rightarrow n + 1)$
Es gelten folgende Zusicherungen:

/*1*/ $z_{\text{alt}} = b \cdot i_{alt}$.

/*2*/ $z_{\text{neu}} = z_{\text{alt}} + b$.

/*3*/ $i_{\text{neu}} = i_{\text{alt}} + 1$.

Lösung Aufgabe 1 (Fortsetzung)

```
int a = In.readInt();
int b = In.readInt();
if (a > b) {
  int h = a;
  a = b;
  b = h;
}
int z = b;
int i = 1;
while ((z % a) != 0){ /*1*/
  z += b;             /*2*/
  i++;                /*3*/
}
```

...

Es gelten folgende Zusicherungen:

/*1*/ $z_{\text{alt}} = b \cdot i_{alt}$.

/*2*/ $z_{\text{neu}} = z_{\text{alt}} + b$.

/*3*/ $i_{\text{neu}} = i_{\text{alt}} + 1$.

Also gilt:

$$\begin{aligned} z_{\text{neu}} &= z_{\text{alt}} + b = b \cdot i_{\text{alt}} + b \\ &= b \cdot (i_{\text{alt}} + 1) = b \cdot i_{neu} \end{aligned}$$

Lösung Aufgabe 1 (Fortsetzung)

```
int a = In.readInt();
int b = In.readInt();
if (a > b) {
  int h = a;
  a = b;
  b = h;
}
int z = b;
int i = 1;
while ((z % a) != 0){ /*1*/
  z += b;             /*2*/
  i++;                /*3*/
}
```

...

Es gilt also die Schleifeninvariante am Ende des n-ten Durchlauf, damit am Anfang des $n+1$-ten Durchlauf. Duch die Induktion ist nun gezeigt, dass $z = b \cdot i$ am Anfang und am Ende jeders Schleifendurchlaufes gilt. Damit ist bewiesen, dass es eine Schleifeninvariante ist.

Lösung Aufgabe 2

```
int a = In.readInt();
int b = In.readInt();
if (a > b) {
  int h = a;
  a = b;
  b = h;
}
int z = b;
int i = 1;
while ((z % a) != 0){ /*1*/
  z += b;             /*2*/
  i++;                /*3*/
}
```

Beweis duch Gegenbeispiel:
Sei $a = 3$ und $b = 5$.
Bei /*1*/ gilt: $z = b = 5$, also
$\mathrm{LCM}(a, b) = \mathrm{LCM}(a, z)$
Bei /*3*/ gilt:
$z_{\text{neu}} = b \cdot i_{\text{neu}} = 5 \cdot (1 + 1) = 10$
Aber:
$\mathrm{LCM}(a, b) = 15$
$\neq 30 = \mathrm{LCM}(a, z_{\text{neu}})$

Lösung Aufgabe 3

```
int a = In.readInt();
int b = In.readInt();
if (a > b) {
  int h = a;
  a = b;
  b = h;
}
int z = b;
int i = 1;
while ((z % a) != 0){ /*1*/
  z += b;             /*2*/
  i++;                /*3*/
}
```

1. Zu Beginn gilt $z = b$
2. z wächst mit jedem Durchlauf monoton um b
3. $z \mod a = 0$ genau dann, wenn z Vielfaches von a ist
4. Nach a Schritten ist $z = b \cdot a$ Vielfaches von a
5. Die Schleife terminiert nach (spätestens) a Schritten

for-Anweisung

Syntax

"for" "(" Initialisierungsteil ";" Abbruchbedingung ";" Inkrementierungsteil ")" Schleifenrumpf ";"

Beispiel

```
for (int i = 0; i<100; i++) Out.print("*");
```

In welcher Reihenfolge werden die Teile ausgeführt?

2× `while`-Schleife

Die `while`-Schleife realisiert eine Abweisschleife.

```
int n = In.readInt();
// no negative values
if (n < 0) n = -n;
while (n > 0) {
  // rest from n / 10
  Out.print(n % 10);
  n = n / 10;
}
```

Die `do-while`-Schleife realisiert eine Durchlaufschleife.

```
int n = In.readInt();
// no negative values
if (n < 0) n = -n;
do{
  // rest from n / 10
  Out.print(n % 10);
  n = n / 10;
} while (n>0)
```

Worin unterscheiden sich die Programme?

Schleifentransformation

Wandelt folgende `for`-Schleife in eine `while`-Schleife und in eine `do-while`-Schleife um. Verwendet dabei keine `break`-Anweisung:

```
int s = 0;
for (;;) {
  int x = In.readInt();
  if (x<0) break;
  s = s + x;
}
```

break-Anweisung

```
int x = In.readInt();
int y = In.readInt();
while (y > 0) {
  while (x > 0) {
    if (x > 5) {
      if (x == 7) break;
    }
    Out.print("1");
    x--;
  }
  Out.print("2");
  y--;
}
Out.print("3");
```

Der Befehl `break` realsiert den Abbruch eines Rumpfes

- einer `switch`-Anweisung.
- einer `while`-Schleife.
- einer `for`-Schleife.
- eines `do`-Blockes.

Warum sollte man `break` sparsam verwenden?
Was gibt das Program aus bei den Eingaben 2 und 2, 7 und 3, 9 und 1?

Welche Schleife? (1)

Welche Schleife ist für die folgende Aufgabenstellung am geeignetsten:

Aufgabe

Zuerst wird ein positiver ganzzahliger Wert eingelesen. Es werden anschließend genau so viele Zahlen eingelesen, wie der anfangs eingelesene Wert angibt und nach Eingabe jeder Zahl wird die Summe der bis dahin gelesenen Zahlen ausgegeben.

Welche Schleife? (1)

Welche Schleife ist für die folgende Aufgabenstellung am geeignetsten:

Aufgabe

Zuerst wird ein positiver ganzzahliger Wert eingelesen. Es werden anschließend genau so viele Zahlen eingelesen, wie der anfangs eingelesene Wert angibt und nach Eingabe jeder Zahl wird die Summe der bis dahin gelesenen Zahlen ausgegeben.

Lösung

`for`-Schleife, da die Anzahl der Iterationen bereits vorher bekannt ist.

Welche Schleife? (2)

Welche Schleife ist für die folgende Aufgabenstellung am geeignetsten:

Aufgabe

Bis zur Eingabe einer 0 sind ganze Zahlen einzulesen. Sofern es sich bei der Eingabe um keine 0 handelt, wird nach Eingabe die Summe der bisher gelesenen Zahlen ausgegeben.

Welche Schleife? (2)

Welche Schleife ist für die folgende Aufgabenstellung am geeignetsten:

Aufgabe

Bis zur Eingabe einer 0 sind ganze Zahlen einzulesen. Sofern es sich bei der Eingabe um keine 0 handelt, wird nach Eingabe die Summe der bisher gelesenen Zahlen ausgegeben.

Lösung

`while`-Schleife, da die Abbruchbedingung bereits vor der ersten Summation und Ausgabe überprüft werden muss.

Welche Schleife? (3)

Welche Schleife ist für die folgende Aufgabenstellung am geeignetsten:

Aufgabe

Zunächst ist ein positiver ganzzahliger Grenzwert einzulesen. Danach werden ganze Zahlen eingelesen und nach Eingabe jeder Zahl wird die Summe der bisher gelesenen Zahlen ausgegeben, bis die Summe (inklusive der zuletzt eingegebenen Zahl) den Grenzwert überschritten hat.

Welche Schleife? (3)

Welche Schleife ist für die folgende Aufgabenstellung am geeignetsten:

Aufgabe

Zunächst ist ein positiver ganzzahliger Grenzwert einzulesen. Danach werden ganze Zahlen eingelesen und nach Eingabe jeder Zahl wird die Summe der bisher gelesenen Zahlen ausgegeben, bis die Summe (inklusive der zuletzt eingegebenen Zahl) den Grenzwert überschritten hat.

Lösung

`do-while`-Schleife, da die Abbruchbedingung erst nach der Summation und Ausgabe überprüft werden muss.

Fragen

Fragen?