Große Übung
Praktische Informatik 1

2006-02-02

Holger Füßler
fuessler@informatik.uni-mannheim.de
http://www.informatik.uni-mannheim.de/pi4/people/fuessler

1: Announcements / Orga

Termine etc.

- Scheinklausur: 9. Februar S108 15:30 pünktlich 90Minuten
- Modulklausur: 9. März 100Minuten
- 2-3 Tutoren werden eine Q&A Sitzung vor der Modulklausur machen. Genaue Termine werden noch bekannt gegeben

Klausuren

- Stoff ist die gesamte Vorlesung!
- Schwerpunkt liegt auf Fragestellungen, die im Rahmen des Übungsbetriebes angesprochen wurden.
- Modul und Scheinklausur haben eine gewisse Themen-Überdeckung, jedoch werden auch unterschiedliche Themen zu finden sein.

Klausurzulassung

- heute morgen waren es so grob Hundert zugelassene
- Herzlichen Glückwunsch!
- die „nicht zugelassenen“ sollten darauf achten, die Übung nachzuholen, wenn sie das Modul schreiben wollen

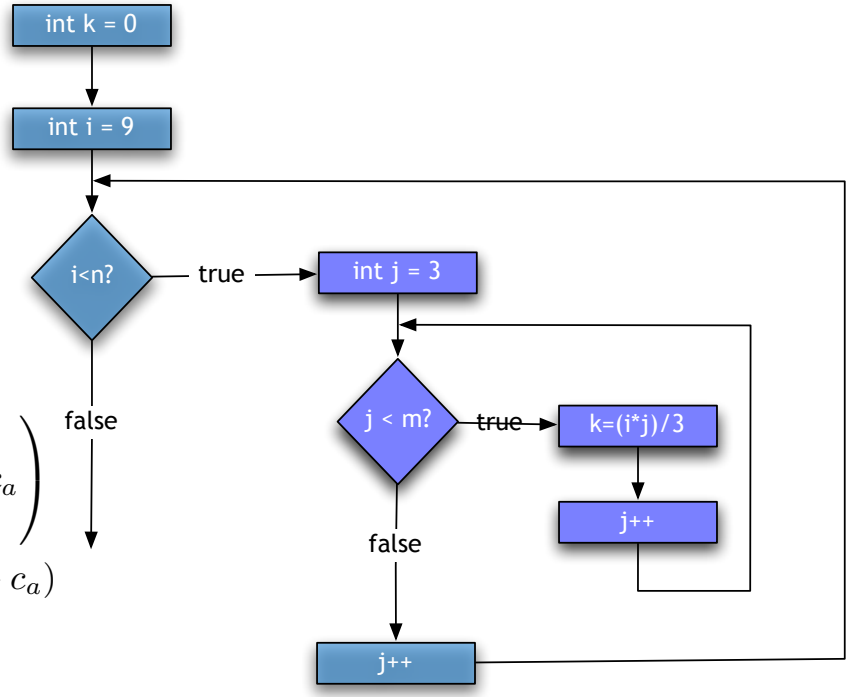
Evaluation

- noch nicht statistisch ausgewertet
- auf den ersten Blick:
  - Robert Schiele: sehr gut
  - Holger Füßler: mäßig

2: Übungsblatt 11

Blatt 11 / Aufgabe 1

```
public static void methA(int n, int m) {
   int k = 0;
   for (int i = 9; i < n; i++)
      for (int j = 3; j < m; j++)
        k = (i*j)/3;
}
```



$$
\begin{aligned}
C &= 2c_z + c_c + \sum_{i=9}^{n-1} \left(c_z + c_c + \sum_{j=3}^{m-1} (3c_a + c_z + c_c) + c_a \right) \\
&= 2c_z + c_c + (n-9)\left(c_z + c_c + (m-3)\left(2c_a + c_z\right) + c_a\right) \\
&= c_1 + (n-9)\left(c_2 + (m-3)\left(c_3\right)\right) \\
&= c_1 + (n-9)\left(c_4 + c_3 m 3\right) \\
&= c_1 + c_4 n - 9c_4 + c_3 nm - 9c_3 m \\
&= c_1 + c_4 n + c_5 m + c_3 nm \\
O(C) &= O(nm)
\end{aligned}
$$

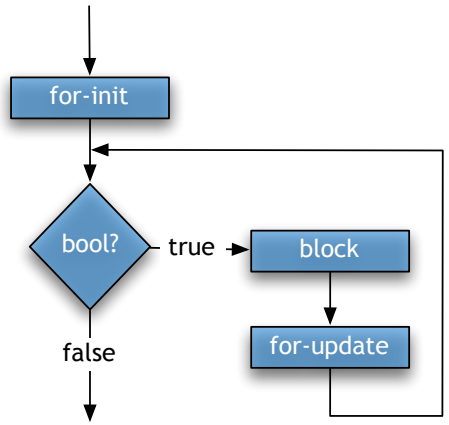
+ einmal Methodenaufruf

methA Komplexität

- Fallunterscheidungen sind für Komplexität irrelevant, da O()-Notation eine Grenzwertbetrachtung ist.
- Anwendung der O-Kalkül Rechenregeln

methB

```
public static void methB(int n, int m) {
   int k = 0;
   m = 150;
   for (int i = 2; i < n; i++)
      for (int j = 2; j < m; j++)
         k = (i* j )/2 + 9;
   }
```



O(n)

methC

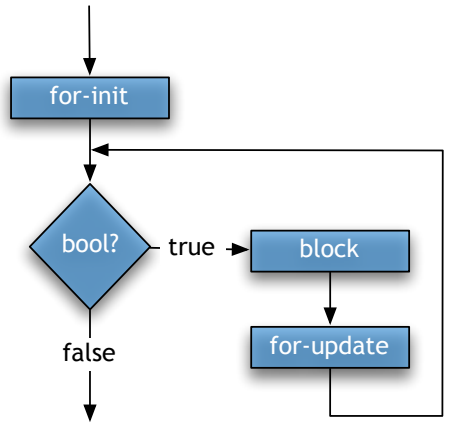
```
public static void methC(int n, int m)
{
   int k = 0;
   for (int i = 1; i < n; i++)
      for (int j = 1; j < m; j <<= 1)
         k = (i* j )/3 – 4;
}
```

m=2,3,4	1
m=5,6,7,8	2
m=9,10,11,12,13,14,15,16	3

O(n log m)

methD

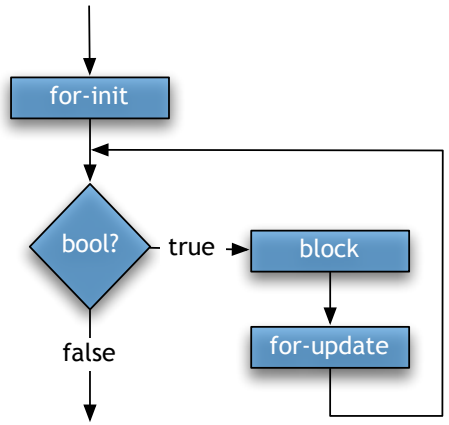
```
public static void methD(int n, int m)
{
   int k = 0;
   n = 150;
   for (int i = 1; i < m; i++)
      for (int j = i % n; j > 0; j--)
         k = (i*j)/3;
}
```



O(m)

methE

```
public static void methE(int n, int m)
{
   int k = 0;
   n = 150;
   for (int i = 1; i < m; i++)
      for (int j = i ; j > 0; j --)
         k = (i * ( j+2 ) ) / 3;
}
```



O(m^2)

Blatt11 / Aufgabe 4a

Aufgabe 4 a) 4 Punkte

Eine Methode mit der Signatur `boolean isNMMatrix(int[][] matrix, int n, int m)` soll prüfen, ob eine übergebene `int[][]`-Variable eine gültige $n \times m$ Matrix ist. Analysieren Sie, was alles schief gehen kann, und spezifizieren Sie einen sinnvollen `black-box-test` für diese Methode in Form eines Java-Programms, das eine Fehlermeldung ausgibt, wenn die Methode nicht wie erwartet arbeitet.

- Black-Box Test: Ich kenne den Inhalt eines Moduls nicht, sondern weiß nur, wie ich es benutze und was es machen soll.
- Also: Ich schreibe meine Annahmen über das Programmverhalten auf, benutze das Modul entsprechend, und überprüfe ob der Effekt wie gewünscht ist.
- Beispiel: Die Methode sollte false zurück geben, wenn man sie mit **`isNMMatrix(null, n, m)`** aufruft.

Blatt11 / Aufgabe 4a

- vollständig: geht nicht
- sinnvoll: Ich überprüfe Sonderfälle
  - matrix = null -> false
  - anzahl zeilen ungleich n -> false
  - matrix mit unterschiedlich langen zeilen -> false
  - anzahl spalten ungleich m -> false
  - korrekte Matrix -> true
- für n,m sollte bei allen tests größer als eins sein

Nachteil

- ja nach Implementierung könnten noch kompliziertere Bugs drin sein. (siehe Microsoft Windows)
- Deshalb testet man in der Regel unter Kenntnis des Source-Codes. (siehe Linux)

Blatt 1 / Aufgabe 4b

Aufgabe 4 b) 6 Punkte

Gegeben sei folgende Methode zur Umrechnung von Oktalziffern in eine Dezimalzahl. Testen Sie diese Methode als *white-box*, einmal als Anweisungs- und einmal als Wege-Test, d.h. definieren Sie eine Menge von Eingabedaten / Funktionsaufrufen, die die für diese Tests erforderlichen Kriterien erfüllen. Erstellen Sie hierfür zuerst ein Flussdiagramm. Wie hängt die Anzahl der Wege durch den Programmfluss von `octals.length` ab?

```
public static int getDecFromOct(int[] octals) {
    int retval = 0;
    if (octals == null)
        retval = -1;
    else {
        for (int i = 0; i < octals.length; i++) {
            if (octals[i] < 0)
                retval = -1;
            if (octals[i] > 7)
                retval = -1;
        }
        if (retval == 0) {
            for (int expo = 0; expo < octals.length; expo++)
                retval += (octals[i] << (3 * expo));
        }
    }
    return retval;
}
```

3: Requested Topics

Prolog / Scheme

- Klausurrelevant

Skript rumblättern

Vielen Dank!