technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2013/14

Übungsblatt 4

Bearbeitungszeit:
25.–29.11.2013

Aufgabe 4.1 – If-Anweisung

Auf dem letzten Übungszettel haben Sie in Aufgabe 3.6 ein Programm geschrieben, das prüft, ob eine Zahl durch drei teilbar ist. Verändern Sie das Programm so, dass es dem Benutzer je nach eingegebener Zahl den Text „`Die Zahl ist durch drei teilbar.`“ oder „`Die Zahl ist nicht durch drei teilbar.`“ ausgibt. Verwenden Sie dazu folgende *Kontrollstruktur*:

```
if (bedingung) {
  ...
} else {
  ...
}
```

Dabei ist `bedingung` ein Ausdruck vom Typ `boolean`.

Aufgabe 4.2 – Logische Operatoren

In der Vorlesung haben Sie die logischen Operatoren `&&`, `||` und `!` kennengelernt. Tragen Sie in die untenstehenden Tabellen die Ergebnisse der angegebenen Ausdrücke ein:

Ausdruck	Ergebnis
`true && true`	
`false && true`	
`true && false`	
`false && false`	
`!false`	
`!true`	

Ausdruck	Ergebnis
`true || true`	
`false || true`	
`true || false`	
`false || false`	
`!(false || true) && true`	
`true || (!false && false)`	

Aufgabe 4.3 – If-Anweisung mit logischen Operatoren

Erweitern Sie nun das Programm aus Aufgabe 4.1 so, dass es zwei (statt nur einer) ganze Zahlen vom Benutzer einliest und – je nach Eingabe – einen der folgenden Texte ausgibt:

- „`Keine der Zahlen ist durch drei teilbar.`“
- „`Genau eine Zahl ist durch drei teilbar.`“
- „`Beide Zahlen sind durch drei teilbar.`“

Verwenden Sie hierzu logische Operatoren.

Aufgabe 4.4 – `while`-Schleifen

Betrachten Sie folgendes Programmfragment:

```
int i = 0;
while (i <= 11) {
  System.out.print(i + ", ");
  i++;
}
```

Die Ausgabe ist die folgende:

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
```

Programmieren Sie analog zu diesem Beispiel Schleifen, die die folgenden Zahlenreihen ausgeben.

a) $8, 9, 10, 11, 12, \ldots, 20$

b) $1, 4, 7, 10, 13, 16, \ldots, 31$

c) $1, 2, 4, 8, 16, 32, \ldots, 128$

d) $50, 45, 40, 35, \ldots, -10$

Aufgabe 4.5 – `do-while`-Schleifen

Suchen Sie sich eine der Schleifen aus, die Sie in den Aufgaben 4.4 a) bis d) programmiert haben. Ändern Sie sie so, dass eine `do-while`-Schleife verwendet wird.

Aufgabe 4.6 – Noch mal Schleifen

a) Schreiben Sie ein Programm, welches die Summe der Zahlen von 1 bis 10 berechnet. Verwenden Sie für die Implementierung eine `while`-Schleife. Benutzen Sie eine Variable, die in der Schleife schrittweise von 1 bis 10 hochgezählt wird, und eine andere Variable, in der die Summe schrittweise berechnet wird. Geben Sie das Ergebnis am Ende des Programms aus.

b) Ändern Sie das Programm so ab, dass die Summe der natürlichen Zahlen von `m` bis `n` berechnet wird. Beachten Sie mögliche Fehlerfälle!

c) Testen Sie Ihr Programm mit folgenden Eingaben:
`m` = 1, `n` = 10
`m` = 1, `n` = 100
`m` = -4, `n` = 45
`m` = 20, `n` = 15

Aufgabe 4.7 – Prä- und Postinkrement/dekrement

Betrachten Sie folgendes Programmfragment:

```
1 int i = 23;
2 int j = 42;
3 int k = ++i;
4 int l = i++;
5 k = --i + j++;  // Zusatzaufgabe
6 j = k++ - i--;  // Zusatzaufgabe
```

Tragen Sie in die unten stehende Tabelle ein, welchen Wert `i`, `j` und `k` jeweils nach der Ausführung der einzelnen Zeilen haben.

	Zeile					
Variable	1	2	3	4	5	6
`i`	23					
`j`	—					
`k`	—	—				
`l`	—	—	—			

Aufgabe 4.8 – Plusgleich-Operator und andere

Wenn Sie in einem Java-Programm auf diese Weise den Wert einer Variablen verändern:

```
x = x + 15;
```

Dann können Sie auch folgende abkürzende Schreibweise benutzen:

```
x += 15;
```

Dasselbe ist auch für andere Operatoren möglich, z. B. `-`, `*`, und `/`, indem Sie `-=`, `*=` und `/=` benutzen. Auch die Benutzung des Modulo-Operators `%` können Sie so abkürzen.

Verändern Sie das Programm von Aufgabe 4.6 so, dass an geeigneter Stelle der „+="-Operator verwendet wird.

Aufgabe 4.9 – Mittelwert

Schreiben Sie ein Programm, das das arithmetische Mittel aus mehreren Gleitkommazahlen berechnet. Die Zahlen sollen über die Tastatur eingegeben werden (mit `scanner.nextDouble()`). Die Anzahl der Zahlen ist vorher nicht bekannt, daher soll nach jeder eingegebenen Zahl gefragt werden, ob noch eine Zahl eingegeben werden soll.

Tipp 1: Ihr Programm muss sich die einzelnen eingegebenen Zahlen nicht merken! Es reicht, wenn die Summe und die Anzahl der eingegebenen Zahlen gespeichert werden.

Tipp 2: Die Abfrage, ob noch eine weitere Zahl eingegeben werden soll, können Sie z. B. folgendermaßen programmieren: Bitten Sie den Benutzer, entweder `0` oder `1` einzugeben (0 um aufzuhören und 1 um weiterzumachen). Lesen Sie anschließend die Zahl mit `scanner.nextInt()` ein.

Ergänzende Aufgaben

Aufgabe 4.10 – Exklusiv-Oder

In der Aufagebe 4.2 wurden logische Operatoren behandelt. In den neueren Versionen von Java hat ein zusätzlicher logischer Operator Einzug gefunden: ^ (Exklusiv-Oder). Im Sprachgebrauch entspricht es einer „entweder ... oder“ Beziehung. In der folgenden Tabelle sehen Sie die entsprechende Wertebelegung:

Ausdruck	**Ergebnis**
`true ^ true`	false
`false ^ true`	true
`true ^ false`	true
`false ^ false`	false

Verwendung mit booleschen Variablen
`a ^ b`

Da es in den früheren Versionen von Java diesen Operator noch nicht gab, musste man auf die Ausdrücke mit üblichen logischen Operatoren (`!`, `&&`, `||`) zurückgreifen, die das Exklusiv-Oder ersetzen. Überlegen Sie sich welche logische Ausdrücke das gleiche Ergebnis liefern wie das Exklusiv-Oder und tragen Sie diese in die obige Tabelle ein.

Aufgabe 4.11 – Fakultät

Die *Fakultät* $n!$ einer Zahl n ist folgendermaßen rekursiv definiert:

$$n! := \begin{cases} 1 & \text{für } n = 0 \\ n \cdot (n-1)! & \text{für } n \geq 1 \end{cases}$$

Alternativ kann die Fakultät auch wie folgt definiert werden:

$$n! := n \cdot (n-1) \cdots 1$$

So ist zum Beispiel

$$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120.$$

Schreiben Sie ein Programm, das die Fakultät einer Zahl berechnet. Verwenden Sie eine Schleife. Die Zahl soll vorher von der Tastatur eingelesen werden.

Aufgabe 4.12 – Fibonacci-Zahlen

Die *Fibonacci-Folge* $(f_1, f_2, \ldots)$ ist folgendermaßen definiert:

$$f_n := \begin{cases} 1 & \text{für } n = 1 \\ 1 & \text{für } n = 2 \\ f_{n-1} + f_{n-2} & \text{für } n > 2 \end{cases}$$

Zum Beispiel ist das vierte Glied der Fibonacci-Folge

$$f_4 = f_3 + f_2 = (f_2 + f_1) + f_2 = (1 + 1) + 1 = 3.$$

Schreiben Sie ein Programm, was zunächst n von der Tastatur einliest und schließlich (mit einer Schleife), den Wert von f_n berechnet und ausgibt.

Noch mehr ergänzende Aufgaben

Auf der Webseite `http://projecteuler.net/` finden Sie sehr viele Programmieraufgaben, deren Lösung Sie versuchen können.