Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Peer Kröger
Michael Fromm, Florian Richter

Einführung in die Programmierung

WS 2018/19

Übungsblatt 4: Eigenschaften von Algorithmen, Ganze Zahlen, While

Besprechung: 19.11 - 23.11.2018

Aufgabe 4-1 *Eigenschaften von Algorithmen*

Stellen Sie sich einen Karteikasten vor, in dem Sie nach einer Karteikarte mit einem bestimmten Titel suchen. Gehen Sie dabei von folgenden Annahmen aus:

- Der Karteikasten enthält nicht mehrere Karteikarten mit dem selben Titel.
- Initial enthalte der Karteikasten $n \in \mathbb{N}$ Karteikarten. Also mindestens eine Karteikarte.
- Eine gültige Eingabe ist auch ein Titel, der nicht im Karteikasten vorkommt. In diesem Fall soll nichts als Ergebnis zurück gegeben werden.
- Die Eingabe beinhaltet den gesuchten Titel.
- Die Ausgabe sei die Karte, die sich bei Terminierung in Ihrer Hand befindet.

Suchverfahren 1:
(1) Sie nehmen die erste Karte aus dem Karteikasten.

Suchverfahren 2:
(1) Sie nehmen die erste Karte aus dem Karteikasten heraus.
(2) Hat diese den gewünschten Titel, so sind Sie fertig ansonsten legen Sie die Karte auf die Seite
(3) Ist der Kasten leer, so beenden Sie die Suche, sonst wiederholen Sie das Verfahren ab Schritt 1.

Suchverfahren 3:
(1) Sie greifen zufällig eine Karte heraus.
(2) Hat diese den gewünschten Titel, so sind Sie fertig, ansonsten legen Sie sie zurück und wiederholen das Verfahren ab Schritt 1.

Suchverfahren 4:
(1) Sie greifen zufällig eine Karte heraus.
(2) Hat diese den gewünschten Titel, so sind Sie fertig, ansonsten legen Sie sie zurück.
(3) Sind seit Beginn der Suche zehn Minuten vergangen, geben Sie auf und beenden die Suche.
(4) Ansonsten wiederholen Sie das Verfahren ab Schritt 1.

Suchverfahren 5: (Vorbedingung: Karteikasten ist lexikographisch sortiert)
(1) Sie greifen die Karte, die in der Mitte liegt. Bei n Karten ist dies die $DIV(n+1,2)$-te Karte.
(2) Trägt diese den gewünschten Titel, beenden die Suche, ansonsten:
(3) Ist der gesuchte Titel alphabetisch vor dem Titel auf der ausgewählten Karte, so suchen Sie nur noch in der ersten Hälfte nach dem gleichen Verfahren ab Schritt 1 weiter.
(4) Ist der gesuchte Titel alphabetisch hinter dem Titel auf der ausgewählten Karte, so suchen Sie nur noch in der zweiten Hälfte nach dem gleichen Verfahren ab Schritt 1 weiter.
(5) Das Verfahren ist erfolglos beendet, wenn der letzte Kartenabschnitt leer ist.

Zu Aufgabe 4-1:
Welcher Algorithmus erfüllt welche der folgenden Eigenschaften? Begründen Sie Ihre Entscheidungen!

(a) terminierend (b) deterministisch (c) determiniert (d) partiell korrekt (e) total korrekt

Aufgabe 4-2 *Ganze Zahlen in Java*

In der Vorlesung haben Sie die maschinelle Darstellung von Zahlen kennengelernt. Außerdem haben Sie von mehreren Sorten oder Typen gehört, die diese Zahlendarstellungen repräsentieren. Einer der primitiven Typen zur Speicherung von ganzen Zahlen ist Integer oder kurz `int`. Sie haben ihn bereits in den vorherigen Übungen verwendet. Der Variablentyp kann dem Compiler so vermitteln, was für einen Wert er erwarten soll. Der Compiler kann dann einen Bereich im Speicher reservieren, um den Variablenwert dort abzulegen und für spätere Berechnungen zu verwenden. Das Benennen und die Typzuweisung einer Variablen nennt man Deklaration. Es gilt meistens, dass eine Variable erst deklariert werden muss, bevor man sie mit einem Wert initialisiert, d.h. ihr einen Wert zuweist.

```
int x;        /*Deklaration einer Integervariablen x*/
x = 2016;     /*Initialisierung mit 2016*/

int y = 42;   /*Deklaration und Zuweisung in einem Ausdruck*/

int z = x+y; /*Zuweisung durch andere Variablen*/
```

(a) Variablen des Typs `int` reservieren im Speicher 4 Byte. Ein Byte besteht aus 8 Bit, also belegt eine Integervariable 32 Bit. Können damit alle ganzen Zahlen dargestellt werden? Wenn nicht, wieviele Zahlen können dargestellt werden?

(b) Welche Ausgabe liefert der folgende Code? Warum?
Hinweis: `==` ist der Vergleichsoperator in Java, der auf Gleichheit prüft und `True/False` ausgibt. Das einfache `=` weist der linken Seite den Wert der rechten Seite zu.

```
public class GanzeZahlen1 {
        public static void main(String[] args) {
                int i = 0b00000000000000000000000000000001;
                int j = 0b11111111111111111111111111111111;
                int k = 0b00000000000000000000000000000000;
                System.out.println(i+j==k);
        }
}
```

(c) Welche Ausgabe liefert der folgende Code? Warum?

```
public class GanzeZahlen2 {
        public static void main(String[] args) {
                int x = 2147483647;
                System.out.println(x+1);

                int y = -2147483648;
                System.out.println(y-1);
        }
}
```

(d) Geben Sie jede Ausgabe `System.out.println(...)` im folgenden Javacode an. Begründen Sie jeweils in einem kurzen Satz, wie es zu diesem Wert kam. Versuchen Sie, sich als erstes selbst zu überlegen, welche Ausgabe Sie erwarten und überprüfen Sie Ihre Erwartung mit dem Java-Compiler.

```
1                   int a = 0b101;
2                   System.out.println(a);
3                   a = a / 2;
4                   System.out.println(a);
5                   ;;;;;;;;
6                   System.out.println(a);
7                   a += 1;
8                   System.out.println(a);
9                   a = 2;
10                  a *= a;
11                  System.out.println(a == 2);
12                  System.out.println(a = 2);
13                  a++;
14                  System.out.println(a);
15                  System.out.println(5-2+2);
16                  System.out.println(5/2*2);
17                  a = (int) 13.7;
18                  System.out.println(a + 5.3);
19                  System.out.println(a);
```

Aufgabe 4-3 *Primzahlgenerator,* ***`While`***

In dieser Aufgabe werden Sie einen Primzahlgenerator programmieren. Benutzen Sie die Javadatei `FindPrimes.java` als Grundgerüst. Geben Sie Ihre Lösung in `FindPrimes.java` ab.

(a) Sie sollen den euklidischen Algorithmus verwenden. Dazu gibt es im vorliegenden Code bereits ein Basiskonstrukt. Ergänzen Sie den fehlenden Code gemäß:
https://de.wikipedia.org/wiki/Euklidischer_Algorithmus#Der_klassische_Algorithmus
Die Syntax für eine `while`-Schleife können Sie hier bereits sehen. Die `main`-Methode enthält zwei exemplarische Aufrufe von `ggT`, die natürlich nicht die vollständige Korrektheit beweisen, aber Ihnen einen Hinweis darauf geben, ob Ihre Lösung zumindest teilweise richtig arbeitet.

(b) In der `main`-Methode ist bereits ein wenig Code vorgegeben. Ergängen Sie Ihren Code an den mit `TODO` markierten Stellen. An der ersten Stelle sollen Sie eine `while`-Schleife benutzen, um alle Vorgänger des aktuellen Prim-Kandidaten auf gemeinsame Teiler zu prüfen. Dazu können Sie mit `ggT()` die in (a) vervollständigte Methode nutzen. Die Faktoren von `ggTProdukt` sind alle ggT's des Kandidaten n und seiner Vorgänger $2, \ldots, n-1$. Wenn eine Zahl prim ist, so sind alle größten gemeinsamen Teiler 1 und damit ist auch das Produkt gleich 1:

$$ggTProdukt(n) = ggT(2,n) \cdot ggT(3,n) \cdot \ldots \cdot ggT(n-1,n) = 1 \iff n \text{ prim}$$

(c) Falls `ggTProdukt == 1` ist, so ist der Kandidat prim. Erweitern Sie Ihr Programm, sodass alle Primzahlen zwischen 1 und 1000 ausgegeben werden.