Aufgaben Programmiervorkurs
Übungsblatt 4

TECHNISCHE UNIVERSITÄT DARMSTADT

von Uli Fahrer & Dennis Albrecht

Wintersemester 2014/2015

Aufgabe 1 Rekursion

Aufgabe 1.1 Fakultät rekursiv (Stufe 1)

Einnert euch an die Fakultät, diese ist wie folgt definiert: $n! = \begin{cases} 1, & \text{falls } n = 0 \\ n \cdot (n-1)!, & \text{sonst} \end{cases}$

a) Berechne diese nun mit Hilfe einer rekusiven Funktion.

b) Vergleiche das Ergebnis **15!** mit den gestern berechneten Wert.

Antwort

```java
public class Fakultaet {
    public static void main(String[] args) {
        System.out.println(fac(15));
    }
    public static long fac(int n) {
        return (n<=0)?1:fac(n-1)*n;
    }
}
```

Aufgabe 1.2 Pascalsches Dreieck (Stufe 2)

Schreibe eine Funktion, die eine Zahl im Pascalschen Dreieck berechnet. Deine Funktion bekommt zwei Parameter, wobei der erste die Zeile und der zweite die Position in der Zeile repräsentiert. Zur Berechnung soll die Summe zweier Zellen in der darüberliegenden Zeile gebildet werden (Animation auf der Wikipedia-Seite). Zeile 0 soll dabei die oberste Zeile sein. Sei n die Zeile, dann soll der zweite Parameter die Zahlen 0 bis n annehmen können.

Antwort

```
1  public class Pascal {
2      public static void main(String[] args) {
3          System.out.println("Zeile: ");
4          int zeile = Input.readInt();
5          System.out.println("Spalte: ");
6          int spalte = Input.readInt();
7          System.out.println(pascal(zeile, spalte));
8      }
9      public static int pascal(int row, int column) {
10         if (row < 0 || column > row)
11             return -1;
12         if (column == 0 || column == row)
13             return 1;
14         return pascal(row - 1, column - 1) + pascal(row - 1, column);
15     }
16 }
```

Aufgabe 1.3 Fibonacci-Folge (Stufe 2)

Die Fibonacci-Folge ist eine Folge von Zahlen die nach einer festen Regel berechnet werden. Die ersten beiden Fibonacci-Zahlen sind definiert mit 0 und 1. Jede weitere Fibonacci-Zahl ist die Summe der beiden „Vorgänger“-Fibonacci-Zahlen. Der Anfang der Folge ist also 0,1,1,2,3,5,8,13,...
Schreibe eine Funktion die jeweils die n-te Fibonacci-Zahl mit Hilfe von Rekursion berechnet. Gib anschließend die ersten 20 Zahlen aus.

Antwort

```
1  public class Fibonacci {
2      public static void main(String[] args) {
3          for (int i = 1; i <= 20; i ++)
4              System.out.println(fib(i));
5      }
6      public static int fib(int n) {
7          return (n<=0)?0:
8                  (n==1)?1:fib(n-1) + fib(n-2);
9      }
10 }
```

Aufgabe 1.4 Ultimate Taschenrechner (Stufe 3)

Wir wollen den Taschenrechner nun auf Rekursion umstellen. Du kannst zur Vereinfachung davon ausgehen, dass beide Operanden positiv sind (für Testeingaben musst du dich selbst dann aber auch daran halten).
Im Folgenden werden an die verschiedenen Operatoren Bedingungen gestellt (die Funktionsnamen sind so gewählt wie in der Musterlösung von Tag 3, wenn deine Addierfunktion beispielsweise „add“ heißt, musst du das plus in der Aufgabenstellung duch add ersetzen):

- **plus:** verwende „+1“, „-1“ und plus
- **minus:** verwende „+1“, „-1“ und minus
- **mal:** verwende plus, minus und mal
- **teilen:** verwende plus, minus und teilen (schreibe nur die Ganzzahldivision)
- **potenz:** verwende plus, minus, mal, teilen und potenz

Du darfst nur die genannten Funktionen (wenn plus erlaubt ist, darfst du „plus(x,y)“ aufrufen), konstante Zahlen, die Variablen zahl1 und zahl2 (also die Parameter) und das Prinzip der Rekursion verwenden. Du darfst keine Schleifen und keine arithmetischen Operatoren (+, -, *, /, ... ; nur +1 und -1) verwenden. Du darf die if-Anweisung verwenden. Wenn du einen Operator nicht umsetzten kannst, dann versuch dich an einem anderen. Du kannst beispielsweise die Multiplikation rekursiv umschreiben ohne eine rekursive Addition zu schreiben; nutze dann einfach die bestehende Additionsfunktion.

Antwort

siehe nächste Teilaufgabe

Aufgabe 1.5 Ultimate Taschenrechner II (Stufe 3)

Erweitere den Taschenrechner im selben Stil noch um weitere Funktionen (in Klammern ist angegeben, wie die Funktion aufgerufen werden sollte):

- **modulo (%):** verwende plus, minus und modulo
- **ggT (T):** verwende plus, minus, modulo und ggT
- **min (_):** verwende plus, minus und min
- **max (|):** verwende plus, minus und max

Für den ggT (größten gemeinsamen Teiler) gilt:

$$ggT(a,b) = \begin{cases} \text{b} & \text{wenn a=0,} \\ \text{a} & \text{wenn b=0,} \\ \text{ggt(b, a \% b)} & \text{sonst} \end{cases}$$

Antwort

```
1  public class Taschenrechner {
2      public static void main(String[] args) {
3          System.out.print("Wie lautet der erste Operand: ");
4          int op1 = Input.readInt();
5          System.out.print("Wie lautet der zweite Operand: ");
6          int op2 = Input.readInt();
7          System.out.print("Wie lautet die Operator: ");
8          char op = Input.readChar();
9          switch(op){
10         case '+':
11             System.out.println("Das Ergebnis lautet: " + plus(op1, op2));
12             break;
13         case '-':
14             System.out.println("Das Ergebnis lautet: " + minus(op1, op2));
15             break;
16         case '*':
17             System.out.println("Das Ergebnis lautet: " + mal(op1, op2));
18             break;
19         case '/':
20             System.out.println("Das Ergebnis lautet: " + teilen(op1, op2));
21             break;
22         case '%':
23             System.out.println("Das Ergebnis lautet: " + modulo(op1, op2));
24             break;
25         case '^':
26             System.out.println("Das Ergebnis lautet: " + potenz(op1, op2));
27             break;
28         case 'T':
29             System.out.println("Das Ergebnis lautet: " + ggT(op1, op2));
30             break;
31         case '_':
32             System.out.println("Das Ergebnis lautet: " + min(op1, op2));
33             break;
34         case '|':
35             System.out.println("Das Ergebnis lautet: " + max(op1, op2));
36             break;
37         default:
38             System.out.println("Keine gueltige Eingabe!");
39         }
40     }
41     public static int plus(int op1, int op2) {
42         return (op1 == 0)?op2:
43                 (op2 == 0)?op1:plus(op1 + 1, op2 - 1);
44     }
45     public static int minus(int op1, int op2) {
46         return (op2 == 0)?op1:minus(op1 - 1, op2 - 1);
47     }
48     public static int mal(int op1, int op2) {
```

```
49          return (op1 == 0 || op2 == 0)?0:
50                      plus(mal(op1, minus(op2, 1)), op1);
51      }
52      public static int teilen(int op1, int op2) {
53          return (op2 == 0)?-1:
54                  (op1 < op2)?0:
55                          plus(teilen(minus(op1, op2), op2), 1);
56      }
57      public static int modulo(int op1, int op2) {
58          return (op2 == 0)?-1:
59                  (op1 < op2)?op1:
60                          modulo(minus(op1, op2), op2);
61      }
62      public static int potenz(int op1, int op2) {
63          return (op2 == 0)?1:
64                  mal(potenz(op1, minus(op2, 1)), op1);
65      }
66      public static int ggT(int op1, int op2) {
67          return (op1 == 0)?op2:
68                  (op2 == 0)?op1:
69                          ggT(op2, modulo(op1, op2));
70      }
71      public static int min(int op1, int op2) {
72          return (op1 == 0 || op2 == 0)?0:
73                      plus(min(minus(op1, 1), minus(op2, 1)), 1);
74      }
75      public static int max(int op1, int op2) {
76          return (op1 == 0)?op2:
77                  (op2 == 0)?op1:
78                          plus(max(minus(op1, 1), minus(op2, 1)), 1);
79      }
80  }
```

Aufgabe 1.6 Gerade oder Ungerade (Stufe 3)

Schreibe zwei Funktionen even und odd, welche jeweils eine Zahl als Parameter entgegennehmen und ein boolean zurückgeben wenn bei even die Zahl gerade und bei odd die Zahl ungerade ist. Für diese Funktionen dürft ihr weder den %-Operator noch den /-Operator nutzen.

Antwort

```
1  public class EvenOdd {
2      public static void main(String[] args) {
3          System.out.println("even(42)=" + even(42));
4          System.out.println("odd(42)=" + odd(42));
5          System.out.println("even(23)=" + even(23));
6          System.out.println("odd(23)=" + odd(23));
7      }
8      public static boolean even(int n) {
9          return (n==0)?true:odd(n-1);
10     }
11     public static boolean odd(int n) {
12         return (n==0)?false:even(n-1);
13     }
14 }
```

Aufgabe 2 Wiederholung: Schleifen

Aufgabe 2.1 Quadrat (Stufe 1)

a) Schreibe ein Programm, das die Seitenlänge eines Quadrats erfragt. Und dieses anschließend auf die Konsole ausgibt. Gibt der Benutzer beispielsweise 5 ein soll folgende Ausgabe erstellt werden:

```
*****
*   *
*   *
*   *
*****
```

b) Gib auch den Flächeninhalt aus

Antwort

```java
public class Quadrat {
    public static void main(String[] args) {
        System.out.print("Gib eine Seitenlaenge ein: ");
        int a = Input.readInt();
        if (a < 1)
            return;
        paint(a);
        System.out.println("\nFlaeche: " + (a * a));
    }
    public static void paint(int length) {
        fullLine(length);
        for (int i = 2; i < length; i ++)
            halfLine(length);
        if (length > 1)
            fullLine(length);
    }
    public static void fullLine(int length) {
        for (int i = 0; i < length; i ++)
            System.out.print("*");
        System.out.println();
    }
    public static void halfLine(int length) {
        System.out.print("*");
        for (int i = 2; i < length; i ++)
            System.out.print(" ");
        if (length > 1)
            System.out.println("*");
    }
}
```

Aufgabe 2.2 Harshad-Zahlen (Stufe 3)

Eine Harshad-Zahl oder Niven-Zahl ist eine natürliche Zahl, die durch ihre Quersumme teilbar ist.
Schreibe ein Programm, welches zu einer beliebigen Eingabe erkennt, ob es sich dabei um eine solche Harshad-Zahl handelt.

Antwort

```
1  public class Harshad {
2      public static void main(String[] args) {
3          System.out.println("Gib␣eine␣Zahl␣ein:");
4          int zahl = Input.readInt();
5          int copy = zahl;
6          int quersumme = 0;
7          while(copy > 0) {
8              quersumme += copy % 10;
9              copy /= 10;
10         }
11         if (zahl % quersumme == 0)
12             System.out.println(zahl + "␣ist␣eine␣Harshad-Zahl!");
13         else
14             System.out.println(zahl + "␣ist␣keine␣Harshad-Zahl!");
15     }
16 }
```

Aufgabe 3 Zusatz

Wenn du die Aufgaben erledigt hast, kannst du dich gerne noch an den Schleifen-Varianten der gegebenen Algorithmen versuchen oder für Aufgaben der vergangenen Tage eine rekursive Lösung suchen. Die Tutoren stehen dir dabei gerne bei, allerdings werden wir dafür keine Lösungen bereit stellen. Du kannst die jeweiligen Versionen der Algorithmen in den Lösungen als Kontrolle für deine Versuche nutzen.