Algorithmen & Datenstrukturen

Blatt 1

Dr. Matthias Thimm

Tina Walber, Leon Kastler, Martin Leinberger und Maximilian Strauch

Fachbereich Informatik, Universität Koblenz-Landau

6. Dezember 2013

1 Graphen zeichnen (5 Punkte)

Zeichnen sie die Graphen der folgenden Funktionen über den natürlichen Zahlen $\mathbb{N} = \{0, 1, 2, \dots\ 15\}$. Berechnen Sie dazu die Funktion für kleine Werte von x. Die Graphen dürfen mit der Hand skizziert werden. Beschreiben Sie den Verlauf der Graphen (z.B. linearer Verlauf, exponentieller Verlauf, Parabel, ...).

- $f_1(x) = 3x + 2, \quad f_2(x) = (3x)^2, \quad f_3(x) = 3^x + 2$
- $g(x) =$ **if** x **even** $3(x + 2)$ **else** $3(x - 2)$
- $h(1) = 0$ and $h(x) = 3h(x - 1) + 2,$ for $x \geq 2$.

2 Semantik von Algorithmen (5 Punkte)

Der Binominalkoeffizient gibt an, auf wieviele verschiedene Arten k Objekte aus einer Menge von n Objekten ausgewählt werden können:

$$\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!}$$

Mit dem Paskalschen Dreieck kann die Berechnung von Binominalkoeffizieten graphisch dargestellt werden:

$n = 0$:				1				
$n = 1$:			1		1			
$n = 2$:		1		2		1		
$n = 3$:	1		3		3		1	
$n = 4$:	1	4		6		4		1

Gegeben sind die untenstehenden Algorithmen für die Berechnung von Binominalkoeffizienten. Um die beiden Algorithmen besser zu verstehen, führen Sie die jeweilige semantische Auswertung der beiden Algorithmen mit den Initialwerten ($n = 3, k = 2$) aus. Verwenden Sie dabei die Techniken, die Sie in der Vorlesung gelernt haben (Foliensätze 2 und 3).

a) **Imperativer Algorithmus (2,5 Punkte)**

BINOM var A, B, C, D: int
input A, B;
C := 1;
D := 1;
while D $\leq$ B do C := C * $\frac{A-(D-1)}{D}$; D:= D+1
output C;

b) **Funktionaler Algorithmus (2,5 Punkte)**

$bin(n,k) :=$ if $term(n,k)$ then 1 else $bin(n-1,k) + bin(n-1,k-1)$
$term(n,k) := k = 0 \vee n = k$

3 Fibonacci-Algorithmus (5 Punkte)

In der Vorlesung (Foliensatz 02, Programmierparadigmen) haben Sie die Fibonacci-Zahlen kennengelernt und wie sie mit einem funktionalen Algorithmus `fib(x)` berechnet werden können. Dieser Algorithmus wird aber bereits für relativ kleine Fibonacci-Zahlen sehr langsam, z. B. für `fib(40)`. Es gibt aber auch einen imperativen Algorithmus zur Berechnung der Fibonacci-Zahlen, der deutlich schneller ist:

```
public static int fibonacci(int n) throws IllegalArgumentException {
    if (n < 0) {
       throw new IllegalArgumentException("n muss >= 0 sein.");
    } else if (n == 0) {
        return 0;
    } else {
        int f = 1;
        int f1 = 1;
        int f2 = 0;
        int k = 2;
        while (k <= n) {
            f = f1 + f2;
            f2 = f1;
            f1 = f;
            k = k + 1;
        }
        return f;
    }
}
```

Bringen Sie diesen imperativen Algorithmus in seine äquivalente funktionale Form.

Tipp: Um diese Aufgabe zu lösen ist es nicht unbedingt notwendig zu verstehen, wie dieser imperative Algorithmus konkret funktioniert. Vielmehr geht es darum zu verstehen, wie jeder beliebige imperative Algorithmus auf einen äquivalenten funktionalen Algorithmus abgebildet werden kann. Das dafür notwendige Wissen finden Sie in Foliensatz 02 und 03. Dort wird die Semantik der imperativen Algorithmen zurückgeführt auf eine mathematische bzw. funktionale Definition.

4 Javaprogrammierung: Aufsummieren von Preisen (5 Punkte)

Ihr Chef hat Ihnen die Aufgabe gegeben, die Software für eine Kasse zu programmieren. Unter anderem wird eine Funktion benötigt, die die Gesamteinnahmen des Tages berechnet. Es gibt eine Liste mit den Preisen für die verkauften Produkte und eine Liste mit der Anzahl der verkauften Produkte. Schreiben Sie die Funktion zur Berechnung der Einnahmen. Gehen Sie dabei wie folgt vor:

a) **(1 Punkte)** Formulieren Sie das Problem, das durch die Funktion gelöst werden soll. Dazu gehört es, dem Problem einen passenden Namen bzw. eine Kurzbeschreibung zu geben, die Eingabe- und Ausgabeparameter der Funktion festzulegen und mögliche Probleme bei den Inputvariablen zu identifizieren.

b) **(2 Punkte)** Schreiben Sie eine Funktion im imperativen Programmierparadigma, die das unter a) definierte Problem löst. Fügen Sie mindestens eine Abfrage ein um bei problematischen Inputvariablen eine Fehlermeldung auszugeben und die Ausführung des Algorithmus abzubrechen.

c) **(2 Punkte)** Schreiben Sie eine zweite Funktion im funktionalen Programmierparadigma, die ebenfalls die Summe der Preise bildet. Dafür kann es eventuell notwendig sein, neue Eingabeparameter bzw. eine Unterfunktion einzuführen. Formulieren Sie in dem Fall, in Analogie zu a), das dazugehörige neue Problem bzw. das durch die Unterfunktion gelöste Unterproblem. Fügen Sie auch hier mindestens eine Abfrage ein um bei problematischen Inputvariablen eine Fehlermeldung auszugeben und die Ausführung des Algorithmus abzubrechen.