Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion

Prof. Dr. Michael Hanus, Jan Tikovsky

Institut für Informatik

3. Übung zur Vorlesung „Programmierung“

Serie 3

Abgabe am Freitag, 14. November 2014 - 08:00

Gehen Sie bei allen Programmieraufgaben nach der Konstruktionsanleitung aus der Vorlesung vor:

1. Schreiben Sie einen kurzen Kommentar, der beschreibt was die Prozedur berechnet.
2. Geben Sie die Signaturvereinbarung der Prozedur an.
3. Definieren Sie mindestens drei Testfälle für alle von Ihnen implementierten Prozeduren, wobei insbesondere auch Randfälle getestet werden sollten. Zusätzlich sollten Sie darauf achten, dass alle Programmteile durch ihre Testfälle mindestens einmal ausgeführt werden. Erkennen können Sie das in DrRacket daran, dass nach Drücken des “Start”-Knopfes kein Programmteil mehr dunkel unterlegt ist.

Präsenzaufgabe 1 - Substitutionsmodell

Die Funktionen `factorial`, `number-if` und `new-factorial` sind definiert durch

```
; Berechnet die Fakultät einer Zahl
(: factorial (natural -> natural))
(define factorial
  (lambda (n)
    (if (= n 0)
        1
        (* n (factorial (- n 1))))))

; Führt eine Fallunterscheidung nach einer Bedingung c durch.
; Liefert den zweiten Parameter t, falls c wahr ist und sonst den dritten e.
(: number-if (boolean number number -> number))
(define number-if
  (lambda (c t e)
    (cond (c t)
          (else e))))

; Berechnet die Fakultät einer Zahl
(: new-factorial (natural -> natural))
(define new-factorial
  (lambda (n)
    (number-if (= n 0)
               1
               (* n (new-factorial (- n 1))))))
```

Werten Sie die Terme `(factorial 1)` und `(new-factorial 1)` schrittweise mit Hilfe des Substitutionsmodells aus.

Präsenzaufgabe 2 - Summe und KgV

1. Implementieren Sie eine Funktion `sum-1-to`, die die Summe der ersten `n` natürlichen Zahlen berechnet. Gibt es außer der naiven Lösung auch eine weitere, effizientere?
2. Implementieren Sie eine Funktion `least-common-multiple`, die das kleinste gemeinsame Vielfache zweier natürlicher Zahlen berechnet.

Aufgabe 3 - Pascalsches Dreieck

6 Punkte

Die folgende Anordnung von Zahlen ist als *Pascalsches Dreieck* bekannt:

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
   ...
```

Die Zahlen an den beiden Schenkeln des Dreiecks sind alle 1 und jede Zahl im Innern des Dreiecks ist die Summe der beiden schräglinks und schrägrechts darüberstehenden Zahlen. Schreiben Sie eine Funktion `pascal` mit Parametern `row` und `pos`, die die Zahl, die in der Zeile `row` und der Position `pos` im *Pascalschen Dreieck* steht, berechnet. Dabei wird immer `pos` $\leq$ `row` vorausgesetzt. Die Spitze des Dreiecks hat die Koordinaten `row` $= 0$ und `pos` $= 0$. Der Aufruf `(pascal 4 2)` soll also das Ergebnis 6 liefern. Verwenden Sie für Ihre Lösung nicht die Fakultätsfunktion!

Aufgabe 4 - Substitutionsmodell

6 Punkte

Die Funktionen `double` und `g` sind definiert durch

```
(define double
  (lambda (x) (+ x x)))

(define g
  (lambda (x y) (if (= x y)
                x
                (g x (+ y 2)))))
```

Werten Sie die folgenden Terme schrittweise mit Hilfe des Substitutionsmodells aus. Werten Sie dabei die Teile von Kombinationen im ersten Ausdruck von links nach rechts und die Teile von Kombinationen im zweiten Ausdruck von rechts nach links aus. Achten Sie dabei auf die korrekte Auswertung der Sonderform `if`.

1. `(g (double (* 4 8)) (- (double 35) 8))`
2. `(double (g (- (double 17) 13) (/ (* 7 9) 3)))`

Aufgabe 5 - Schnittpunktberechnung per Newton-Verfahren 8 Punkte

Das in der Vorlesung vorgestellte Newton'sche Approximationsverfahren kann auch zur Näherung des Schnittpunkts zweier Funktionen verwendet werden. Wir betrachten dazu beispielhaft die beiden folgenden Funktionen:

$f(x) = cos(x)$

$g(x) = x^3$

Die Berechnung der Schnittstelle dieser beiden Funktionen lässt sich zurückführen auf die Berechnung der Nullstelle der Funktion $h(x) = f(x) - g(x)$. Mit dem Newton-Verfahren lässt sich eine Folge $(x_i)_{i\geq 0}$ definieren mit

$\lim_{i\to\infty} h(x_i) = 0$

Die Folge sei definiert durch:

$x_0 = 0.5$

$x_i = x_{i-1} - \frac{h(x_{i-1})}{h'(x_{i-1})}$ falls $i > 0$

Definieren Sie eine Prozedur `intersection` mit einem Parameter `eps`, die die Schnittstelle der Funktionen $f(x)$ und $g(x)$ so lange mit Hilfe des Newton-Verfahrens approximiert, bis nach k Iterationen $\mid f(x_k) - g(x_k) \mid$ kleiner als die Schranke *eps* ist, und dann x_k als Näherungswert zurückliefert. Beispielsweise soll der Aufruf `(intersection 0.001)` den Näherungswert für die x-Koordinate des Schnittpunkts von $f(x)$ und $g(x)$ mit einer Genauigkeit von 0.001 liefern.

Orientieren Sie sich bei der Implementierung am in der Vorlesung vorgestellten Verfahren zur Approximation der Quadratwurzel. Das heißt, zerlegen Sie das Problem in Teilprobleme und abstrahieren Sie diese Teilprobleme in eigene Prozeduren.

Zur Berechnung des Sinus bzw. Kosinus dürfen Sie die in Racket vordefinierten Prozeduren `sin` bzw. `cos` verwenden.