Informatik III WS2004/2005

Übungsblatt 4

Institut Intelligente Systeme
Leiter: Prof. Dr. E. Lehmann
Dietmar Lippold

Besprechungstermin: 15.11.04 - 19.11.04

Aufgabe 1

Hinweis: Für diese Aufgabe stellen Sie bitte bei *drscheme* im Menü *Sprache, Sprache auswählen...* den Wert *PLT → Text (MzScheme)* ein, damit die Prozedur `random` zur Verfügung steht.

Die Punkte des Einheitskreises gehorchen der Gleichung $x^2 + y^2 = 1$. Seine Fläche hat den Wert π. Wählt man einen zufälligen Punkt aus dem Einheitsquadrat im ersten Quadranten (also einen Punkt (x, y) mit $0 \leq x \leq 1$ und $0 \leq y \leq 1$), so liegt dieser mit der Wahrscheinlichkeit $\pi/4$ im Einheitskreis (im Teil des Einheitskreises im ersten Quadranten). Durch das wiederholte zufällige Auswählen eines Punktes und Ermitteln der empirischen Wahrscheinlichkeit, mit der der Punkt im Einheitskreis liegt (d.h. der Anzahl der Fälle, in denen der Punkt im Einheitskreis liegt, geteilt duch die Anzahl der Versuche), kann man einen Näherungswert für π ermitteln.

1. Definieren Sie eine Prozedur `(random1)`, die einen zufälligen Wert zwischen 0 und 1 liefert.

   Hinweis: Benutzen Sie die Prozedur `(random n)`, wobei sie für n eine Konstante `max-rand` verwenden, der Sie vorher den für `random` maximal zulässigen Wert von $2^{31} - 1$ zuweisen.

2. Formen Sie die Gleichung des Einheitskreises für den ersten Quadranten in eine Funktion $f : y = f(x)$ um und implementieren Sie $f(x)$ als Prozedur `(ekreis-y x)`.

3. Definieren Sie eine Prozedur `(punkttest)`, die einen zufälligen Punkt erzeugt und für diesen ermittelt, ob er im Einheitskreis liegt und entsprechend *#t* oder *#f* liefert.

4. Definieren Sie eine Prozedur `(enthalten-anzahl versuche)`, die *versuche* mal die Prozedur `punkttest` aufruft und die Anzahl der Versuche zählt, bei denen ein Punkt im Einheitskreis enthalten war.

5. Definieren Sie eine Prozedur `(pi-naeherung versuche)`, die mittels der Prozedur `(enthalten-anzahl versuche)` einen Näherungswert für π ermittelt. Ermitteln Sie konkrete Näherungswerte für einen geeigneten Wert von *versuche*.

Aufgabe 2

In der Vorlesung wurde besprochen, wie man die Multiplikation von natürlichen Zahlen auf die Addition zurückführen kann. Nun sollen die Subtraktion und Division mit den noch grundlegenderen Prozeduren zum Increment (add1) und Decrement (sub1) realisiert werden.

1. Schreiben Sie eine Prozedur `(sub x y)` zur Subtraktion und eine Prozedur `(div x y)` zur Division zweier natürlicher Zahlen x und y mit $x >= y$, wobei `sub` nur `add1` und `sub1` und `div` nur zusätzlich noch `sub` benutzt.

2. Die Prozedur `div` kann einen rekursiven oder einen iterativen Prozeß beschreiben. Schreiben Sie zusätzlich zu der Version aus dem vorangegangenen Aufgabenteil die andere Version.

3. Führen Sie für `(div 17 5)` die Auswertung mit beiden Versionen von Hand durch. Notieren Sie die Prozeduraufrufe, wartende Operationen und Werte von Aufrufen.

Aufgabe 3 (schriftlich)

Die Entwicklung geeigneter Kalender erwies sich in der Kulturgeschichte der Menschheit als äußerst bedeutsam und folgenreich. Der *Gregorianische Kalender*[1] ist heute ein in Europa und vielen anderen Ländern verbreitetes System der Zeitrechnung.

1. Definieren Sie eine Prozedur `(schaltjahr? jahr)`, die zurückgibt, ob das Jahr `jahr` im Gregorianischen Kalender ein Schaltjahr ist.

   Zur Erinnerung: Alle 4 Jahre ist ein Schaltjahr, alle durch 100 teilbaren Jahre fällt es aus, bei den durch 400 teilbaren findet es wiederum statt.

2. Definieren Sie eine Prozedur `(tage monat jahr)`, die Monat und Jahr als Zahl erhält und die Anzahl der Tage in diesem Monat zurückgibt.

3. Definieren Sie eine Prozedur `(tage-bis tag1 monat1 jahr1 tag2 monat2 jahr2)`, die die Anzahl der Tage zwischen dem ersten und dem zweiten Datum berechnet.

4. Definieren Sie eine Prozedur `(wochentag tag monat jahr)`, die den Wochentag des gegebenen Datums bestimmt. (Mit `tage-bis` und der Kenntnis des heutigen Wochentags ist das möglich.)

Aufgabe 4

Gegeben sei folgende Prozedurdefinition:

```
(define (f g)
  (g (g 2)) )
```

1. Wie verhält sich diese Prozedur bei den unten aufgeführten Aufrufen? Wie läßt sich dieses Verhalten erklären?

   ```
   > (f -)
   > (f (lambda (x) (* x x)))
   > (f (lambda (x y) (* x y)))
   > (f f)
   ```

2. Wie kann man die “erlaubten” Parameter für die Prozedur `f` beschreiben?

[1] `http://www.ortelius.de/kalender/greg_de.php`