1. Übungsblatt zu Programmierung, WS 04/05

Prof. Dr. Gert Smolka, Dipl.-Inform. Guido Tack
http://www.ps.uni-sb.de/courses/prog-ws04/

Lesen Sie im Skript: Schnellkurs

Aufgabe 1.1 (Lexikalische Syntax) Betrachten Sie das folgende Programm:

```
val x = 7+4
val y = x*(x-1)
val z = ~x*(y-2)
```

Welche Bezeichner, Konstanten, Operatoren und Schlüsselwörter kommen in dem Programm vor? An welche Werte bindet das Programm die vorkommenden Bezeichner?

Aufgabe 1.2 (Signum) Schreiben Sie eine Prozedur *signum: int→ int*, die für negative Argumente -1, für positive Argumente 1, und für 0 das Ergebnis 0 liefert.

Aufgabe 1.3 (17. Potenz) Schreiben Sie eine Prozedur *hoch17* : *int* → *int*, die zu einer Zahl x die Potenz x^{17} berechnet. Dabei sollen möglichst wenig Multiplikationen verwendet werden. Schreiben Sie die Prozedur auf zwei Arten: Mit einer Hilfsprozedur und mit lokalen Deklarationen.

Aufgabe 1.4 (min) Schreiben Sie eine Prozedur *min* : *int* ∗ *int* → *int*, die zu zwei Zahlen die kleinere liefert. Deklarieren Sie *min* analog zu *swap* auf 3 verschiedene Arten: mit einem kartesischen Argumentmuster, mit einer lokalen Deklaration, und mit Projektionen.

Aufgabe 1.5 (mid) Schreiben Sie eine Prozedur *mid* des Typs *int* ∗ *int* ∗ *int* → *int*, die zu drei Zahlen die mittlere liefert. Beispielsweise soll *mid*(*3*, *56*, *13*) die Zahl 13 liefern.

Aufgabe 1.6 (Rekursionsfolgen) Gegeben sei die rekursive Prozedurdeklaration

```
fun f(n:int, a:int) : int = if n=0 then a else f(n-1, a*n)
```

a) Geben Sie die Rekursionsfolge für den Aufruf $f(3,1)$ an.
b) Geben Sie ein detailliertes Ausführungsprotokoll für den Ausdruck $f(3,1)$ an. Wenn es mehrere direkt ausführbare Teilausdrücke gibt, soll immer der am weitesten links stehende zuerst ausgeführt werden. Sie sollten insgesamt 18 Ausführungsschritte bekommen.

Aufgabe 1.7 (Größter gemeinsamer Teiler) Schreiben Sie eine rekursive Prozedur $ggt : int * int \to int$, die zu zwei positiven Zahlen den größten gemeinsamen Teiler berechnet. Beispielsweise soll $ggt(24, 40)$ das Ergebnis 8 liefern. Verwenden Sie den durch die folgenden Gleichungen gegebenen Algorithmus:

$$\begin{array}{ll} ggt(x,x) = x & \text{falls } x > 0 \\ ggt(x,y) = ggt(x-y,\ y) & \text{falls } x > y > 0 \\ ggt(x,y) = ggt(x,\ y-x) & \text{falls } y > x > 0 \end{array}$$

Aufgabe 1.8 (Natürliche Quadratwurzel) Deklarieren Sie eine Prozedur $wurzel : int \to int$, die zu einer Zahl $x \geq 0$ die größte Zahl $n \geq 0$ mit $n^2 \leq x$ berechnet. Schreiben Sie dazu eine Hilfsprozedur $wurzel' : int * int \to int$, die zu zwei Zahlen $m, x \geq 0$ die kleinste Zahl $n \geq m$ mit $n^2 > x$ liefert. Geben Sie zuerst die Gleichungen für $wurzel'$ an.

Aufgabe 1.9 (Quersumme) Schreiben Sie eine rekursive Prozedur $quer : int \to int$, die die Quersumme einer ganzen Zahl berechnet. Die Quersumme einer Zahl ist die Summe ihrer Dezimalziffern. Beispielsweise hat die Zahl -3754 die Quersumme 19. Verwenden Sie Restbestimmung modulo 10, um die letzte Ziffer einer positiven Zahl zu bestimmen, und ganzzahlige Division durch 10, um die Zahl zu erhalten, die durch Streichen der letzten Ziffer entsteht.

Aufgabe 1.10 (Zeitangaben) Oft gibt man eine Zeitdauer im *HMS-Format* mit Stunden, Minuten und Sekunden an. Beispielsweise ist 2h5m26s eine hervorragende Zeit für einen Marathonlauf.

a) Schreiben Sie eine Prozedur $sec : int * int * int \to int$, die vom HMS-Format in Sekunden umrechnet. Beispielsweise soll $sec(1,1,6)$ die Zahl 3666 liefern.
b) Schreiben Sie eine Prozedur $hms : int \to int * int * int$, die eine in Sekunden angegebene Zeit in das HMS-Format umrechnet. Beispielsweise soll *hms* 3666 das Tupel $(1,1,6)$ liefern. Berechnen Sie die Komponenten des Tupels mithilfe lokaler Deklarationen.

Aufgabe 1.11 (Fakultäten) Für $n \geq 0$ können wir die so genannte n-te Fakultät $n!$ wie folgt definieren:

$$\begin{array}{ll} 0! = 1 & \\ n! = 1 \cdot \ldots \cdot n & \text{falls } n \geq 1 \end{array}$$

Beispielsweise gilt $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$.

a) Geben Sie zwei Gleichungen an, mit denen $n!$ berechnet werden kann.

b) Realisieren Sie die Gleichungen mit einer Prozedur $fak : int \to int$. Schreiben Sie die Prozedur so, dass ihre Ausführung für negative Argumente wegen Speicherüberschreitung abgebrochen wird.
c) Die Fakultäten werden schnell groß. Beispielsweise gilt $10! = 3628800$. Ermitteln Sie mit einem Interpreter das erste n, für das die Ausführung Ihrer Prozedur zu einem Überlauf führt.

Aufgabe 1.12 (Gleitkomma-Potenzierung) Schreiben Sie eine rekursive Prozedur $power : real * int \to real$, die zu einer reellen Zahl x und einer natürlichen Zahl n die Potenz x^n mittels Gleitkommaoperationen berechnet. Welche Zahl liefert $power(3.0, 100)$? Handelt es sich dabei wirklich um die Zahl 3^{100}?

Aufgabe 1.13 (Newtonsches Verfahren für Quadratwurzeln) Schreiben Sie eine Prozedur $sqrt : real \to real$, die zu x die erste Approximation a_n liefert, die $\sqrt{x}$ mit der Genauigkeit $|x - a_n^2| < 10^{-4}$ approximiert. Schreiben Sie zusätzlich eine Prozedur $sqrt' : real \to real * int$, die zu x das Paar (a_n, n) liefert, damit Sie sehen können, wieviele Approximationsschritte jeweils erforderlich sind.