TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

WS 11/12

Einführung in die Informatik II **Übungsblatt 1**

Prof. Dr. Andrey Rybalchenko, M.Sc. Ruslán Ledesma Garza 25.10.2011

Dieses Blatt behandelt Kapitel 1.1 - 1.9 aus dem Buch zur Vorlesung!

Aufgabe 1.3 (Signum) Schreiben Sie ein Prozedur `signum : int → int`, die für negative Argumente -1, für positive Argumente 1 und 0 für das Ergebnis 0 liefert.

Lösungsvorschlag 1.3:

```
fun signum(x:int) = if x < 0 then ~1 else if x > 0 then 1 else 0
```

Aufgabe 1.4 Schreiben Sie ein Prozedur `hoch17 : int → int`, die zu einer Zahl x die Potenz x^{17} berechnet. Dabei sollen möglichst wenige Multiplikationen verwendet werden. Schreben Sie die Prozedur auf zwei Arten: Mit einer Hilfsprozedur und mit lokalen Deklarationen.

Lösungsvorschlag 1.4:

```
fun q (y:int) = y*y
fun hoch17 (x:int) = q (q (q (q x))) * x
```

oder

```
fun hoch17 (x:int) =
  let val x2 = x*x
      val x4 = x2*x2
      val x8 = x4*x4
      val x16 = x8*x8
  in
    x16*x
  end
```

Aufgabe 1.5

a) Geben Sie ein Tupel mit 3 Positionen und nur einer Komponente an.

b) Geben Sie einen Tupelausdruck an, der den Typ `int * (bool * (int * unit))` hat.

c) Geben Sie ein Paar an, dessen erste Komponente ein Paar und dessen zweite Komponente ein Tripel ist.

Lösungsvorschlag 1.5:

a) `((), (), ()) : unit * unit * unit`

b) `(1,(true,(0,()))) : int * (bool * (int * unit))`

c) `(((),()),((),(),())) : (unit * unit) * (unit * unit * unit)`

Aufgabe 1.7 Schreiben Sie eine Prozedur `max: int * int * int → int`, die zu drei Zahlen die größte liefert, auf zwei Arten:

a) Benuzen Sie keine Hilfsprozedur und drei Konditionale.

b) Benuzen Sie eine Hilfsprozedur und insgesamt nur ein Konditional.

Lösungsvorschlag 1.7:

a)
```
fun max(x,y,z) =
  if x>y then
    (if x>z then x else z)
  else
    (if y>z then y else z)
```

b)
```
fun max2 (x:int, y:int) = if x<y then y else x
fun max (x:int, y:int, z:int) = max2(max2(x,y),z)
```

Aufgabe 1.10 Schreiben Sie eine Prozedur `teilbar : int * int → bool`, die für (x, y) testet, ob x durch y ohne Rest teilbar ist.

Lösungsvorschlag 1.10:

```
fun teilbar (x:int, y:int) =
  x mod y = 0
```

Aufgabe 1.11 (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 → 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 → 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.

Lösungsvorschlag 1.11:

a)
```
fun sec (h,m,s) = 60*60*h+60*m+s
```

b)
```
fun hms v =
  let
    val h = v div (60*60)
    val m = (v mod (60*60)) div 60
    val s = (v mod (60*60)) mod 60
  in
    (h,m,s)
  end
```

Aufgabe 1.13 Schreiben Sie eine rekursive Prozedur `mul : int * int → int`, die das Produkt einer natürlichen und einer ganzen Zahl durch wiederholte Addition berechnet. Beschreiben Sie den zugrunde liegenden Algorithmus zunächst mit Rekursionsgleichungen.

Lösungsvorschlag 1.13: Rekursionsgleichungen: $(n \in \mathbb{N}, x \in \mathbb{Z})$

$$\begin{aligned} 0 \cdot x &= 0 \\ n \cdot x &= x + (n-1) \cdot x \qquad (n \neq 0) \end{aligned}$$

Prozedur:

```
fun mul (x:int, y:int) =
  if x=0
  then 0
  else y + mul(x-1,y)
```

Aufgabe 1.14 Der ganzzahlige Quotient `x div y` lässt sich aus `x` durch wiederholtes Subtrahieren von `y` bestimmen. Schreiben Sie eine rekursive Prozedur `mydiv : int * int → int`, die für $x \geq 0$ und $y \geq 1$ das Ergebnis `x div y` liefert. Geben Sie zunächst Rekursionsgleichungen für `x div y` an.

Lösungsvorschlag 1.14: Rekursionsgleichungen:

$$\begin{array}{lcll} x \text{ div } y & = & 0 & (0 \leq x < y) \\ x \text{ div } y & = & 1 + (x - y) \text{ div } y & (x \geq y) \end{array}$$

Prozedur:

```
fun mydiv (x:int, y:int) =
  if x>=y then 1+mydiv(x-y,y) else 0
```

Aufgabe 1.15 Auch der ganzzahlige Rest `x mod y` lässt sich aus `x` durch wiederholtes Subtrahieren von `y` bestimmen. Schreiben Sie eine rekursive Prozedur `mymod : int * int → int`, die für $x \geq 0$ und $y \geq 1$ das Ergebnis `x mod y` liefert. Geben Sie dazu zunächst Rekursionsgleichungen für `x mod y` an.

Lösungsvorschlag 1.15: Rekursionsgleichungen:

$$\begin{array}{lcll} x \bmod y & = & x & (0 \leq x < y) \\ x \bmod y & = & (x - y) \bmod y & (x \geq y) \end{array}$$

Prozedur:

```
fun mymod (x:int, y:int) =
  if x<y then x else mymod(x-y,y)
```

Aufgabe 1.16 (Stelligkeit) Schreiben Sie eine rekursive Prozedur `stell : int → int`, die zu einer Zahl die Stelligkeit ihrer Dezimaldarstellung liefert. Beispielsweise soll stell `117 = 3` gelten. Geben Sie zunächst die Rekursionsgleichungen für stell an. Verwenden Sie ganzzahlige Division durch `10`, um die Zahl zu erhalten, die durch Streichen der letzten Ziffer entsteht.

Lösungsvorschlag 1.16: Rekursionsgleichungen:

$$\begin{array}{lcll} \text{stell } x & = & 1 & (0 \leq x \leq 9) \\ \text{stell } x & = & 1 + \text{stell } (x \text{ div } 10) & (x \geq 10) \end{array}$$

Prozedur:

```
fun stell (x:int) =
  if x < 10 then 1 else 1 + stell(x div 10)
```

Aufgabe 1.17 (Quersumme) Schreiben Sie eine rekursive Prozedur `quer : int → 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. Geben Sie zunächst die Rekursionsgleichungen für quer an. Verwenden Sie Restbestimmung modulo 10, um die letzte Ziffer einer Zahl zu bestimmen.

Lösungsvorschlag 1.17:

Bemerkung:

$$x \text{ div } y \triangleq \lfloor x/y \rfloor \Rightarrow -1 \text{ div } 10 = \lfloor -0.1 \rfloor = -1$$

Rekursionsgleichungen:

$$\begin{array}{lcll} \text{quer } x & = & |x| & (0 \leq |x| \leq 9) \\ \text{quer } x & = & |x| \bmod 10 + \text{quer } (|x| \text{ div } 10) & (|x| \geq 10) \end{array}$$

Prozedur:

```
fun quer (x:int) =
  let val abs_x = if x<0 then ~x else x in
    if abs_x=0 then 0 else abs_x mod 10 + quer (abs_x div 10)
end
```