Ludwig-Maximilians-Universität München
Department "Institut für Informatik"
Prof. Dr. Christian Böhm
Elke Achtert, Arthur Zimek

München, 24.11.2006

Informatik 1

WS 2006/07

Übungsblatt 6: Geltungsbereiche von Namen, Umgebung, Endrekursion

Besprechung: 04.12.–08.12.2006

Abgabe aller mit **Hausaufgabe** markierten Aufgaben bis Freitag, 01.12.2006, 18:00 Uhr

Aufgabe 6-1 *Geltungsbereiche von Namen* **(Hausaufgabe)**

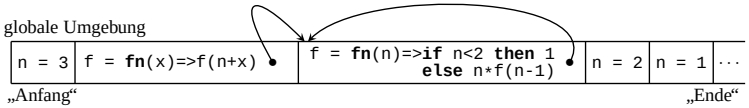
```
val n       =  3;
val k       =  5;

fun f(n)    =  let
                   local
                       val n   =  2 * k * n
                       val k   =  2 * k * n
                   in
                       fun g(n)  =  n + k
                   end
               in
                   g(n + k)
               end;

local
    val n   =  2 * k * n
    and k   =  2 * k * n
in
    fun g(n)  =  n + k

    fun h(n)  =  g(n + k)
end;
```

Die obigen Deklarationen sind in der Datei `6-1.sml` enthalten, die Sie in ein passendes Unterverzeichnis in Ihrem Home-Verzeichnis kopieren können. Geben Sie eine Kopie der Datei ab, in der die Namen so umbenannt sind, dass gilt:

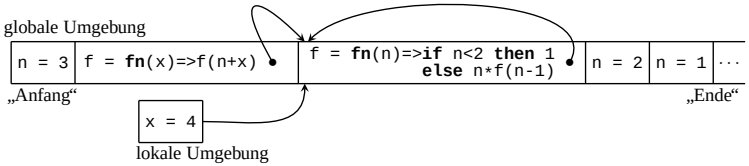
- In der äußeren Umgebung sind die gleichen Namen deklariert wie bisher.
- Diese Namen haben die gleichen Werte wie bisher. Falls die Werte Funktionen sind, liefern diese Funktionen für jedes Argument vom Typ `int` das gleiche Ergebnis wie bisher.
- Für keinen Namen existiert mehr als eine Deklaration in der Datei. Kein formaler Parameter einer Funktion heißt so wie ein formaler Parameter einer anderen Funktion oder wie ein Name, der irgendwo in der Datei deklariert ist.
- Insgesamt müssen 9 Namen umbenannt werden. Benennen Sie sie in der Reihenfolge, in der sie in der Datei vorkommen, mit den neuen Namen `A`, `B`, `C`, `D`, `E`, `F`, `G`, `H`, `I`.

Aufgabe 6-2 *Umgebung und Geltungsbereiche von Namen* **(Hausaufgabe)**

Gesucht ist ein SML-Programm mit folgenden Eigenschaften. Es besteht aus mehreren Deklarationen, nach denen ein Ausdruck kommt, der keine Deklaration ist. Durch die Deklarationen entsteht folgende Umgebung, wobei . . . die Standard-Umgebung beim Start von SML sei:



In dieser globalen Umgebung wird nun der letzte Ausdruck des Programms ausgewertet. Dabei entsteht folgende lokale Umgebung, in der der Ausdruck `f(n+x)` ausgewertet werden muss, um den Wert des ursprünglichen Ausdrucks zu erhalten. Dieser Wert ist übrigens 6!, also `720`.



Durch diese Eigenschaften ist das Programm noch nicht eindeutig bestimmt, weil zum Beispiel der Wert `1` durch beliebige arithmetische Ausdrücke entstanden sein kann. In solchen Fällen soll das Programm jeweils den einfachsten Ausdruck verwenden, also die Zahl selbst. Außerdem sollen Funktionsdeklarationen nicht mit **`fun`** geschrieben werden, sondern mit **`val`** bzw. **`val rec`**. Damit ist das Programm eindeutig bestimmt.

Schreiben Sie das Programm in eine Datei `6-2.sml` und geben Sie diese ab.

Aufgabe 6-3 *Endrekursion* **(Hausaufgabe)**

```
fun summe(n)  =  if n = 0
                 then 0
                 else n + summe(n-1);
```

Definieren Sie in einer Datei `6-3.sml` diese Funktion neu, so dass sie selbst nicht mehr rekursiv ist, aber eine lokale Funktion benutzt, die endrekursiv ist (also einen iterativen Berechnungsprozess auslöst, aber ohne Referenzen).

Hinweis: Die endrekursive Definition der Fakultätsfunktion in der Vorlesung kann Ihnen als Modell dienen.

Aufgabe 6-4 *Endrekursion* **(Hausaufgabe)**

Die Fibonacci-Funktion ist für natürliche Zahlen wie folgt definiert:

$$fib(n) = \begin{cases} 0 & \text{falls } n = 0 \text{ ist} \\ 1 & \text{falls } n = 1 \text{ ist} \\ fib(n-1) + fib(n-2) & \text{falls } n \geq 2 \text{ ist} \end{cases}$$

Geben Sie in einer Datei `6-4.sml` mit rein funktionalen Mitteln eine SML-Definition `fib : int -> int` an, die für $n \in \mathbb{N}$ durch einen iterativen Berechnungsprozess die n-te Fibonacci-Zahl liefert.