Sven Eric Panitz

Übungsblatt 3

(26. Oktober 2009)

Aufgabe 1 In C gibt es einen weiteren in der Vorlesung nicht behandelten arithmetischen Operator. Er wird mit dem Prozentzeichen % bezeichnet. Machen Sie Testaufrufe für Ausdrücke mit diesem binären Operator. Was berechnet der Operator.

Aufgabe 2 Betrachten Sie folgende C-Funktion:

Eins.c

```
1 int f(){
2   int x = 0;
3   x=x+1;
4   return x;
5 }
```

a) Schreiben Sie eine Hauptfunktion, in der Sie die Funktion `f` mehrfach aufrufen und das Ergebnis auf dem Bildschirm ausgeben.

b) Schreiben Sie jetzt vor die Variablendeklaration `int x = 0;` das Wort `static`.

Führen Sie die Hauptmethode erneut aus. Inwiefern hat sich die Ausgabe verändert? Ist die Funktion `f` damit noch eine Funktion im mathematischen Sinne?

Aufgabe 3 Mit Hilfe statischer lokaler Variablen lassen sich in C Funktionen schreiben, die Zahlenfolgen darstellen. Jeder Aufruf der C-Funktion gibt das nächste Element der Folge zurück. Mathematiker schreiben Zahlenfolgen gerne in runden Klammern, z.B.:

Die Folge der natürlichen Zahlen:

$$(1, 2, 3, 4, 5, 6, \ldots)$$

Die Folge der ungeraden Zahlen:

$$(1, 3, 5, 7, 9, 11, \ldots)$$

Die Folge der Quadratzahlen:

$$(1, 4, 9, 16, 25, 36, \ldots)$$

Schreibweise allgemein für Folgen:

$$(x_0, x_1, x_2, x_3, x_4, \ldots)$$

In der vorherigen Aufgabe wurde so eine Funktion realisiert, die bei jedem Aufruf ein nächstes Element der natürlichen Zahlen zurückgibt.

Benutzen Sie jetzt die Erfahrungen über statische lokale Variablen aus Aufgabe 2 um die folgenden C-Funktionen zu realisieren, die Zahlenfolgen berechnen:

a) Eine C-Funktion `odds()`, die bei jedem Aufruf eine neue ungerade Zahl zurückgibt.

b) Eine C-Funktion `squares()`, die bei jedem Aufruf eine neue nächst höhere Quadratzahl zurückgibt.

Sie können, solange noch keine Schleifenkonstrukte in der Vorlesung behandelt wurden, die Funktionen die Folgen darstellen, mit Hilfe der folgenden rekursiven Ausgabefunktion testen,

```
1  int schleife(int i){
2    //is calling the function f()
3    //please adjust to the function you want to test
4    printf("%d ",f());
5    return i>0?schleife(i-1):0;
6  }
7  int main(){
8    schleife(100);
9    return 0;
10 }
```

Aufgabe 4 Für Folgen werden in der Mathematik gerne rekursive Bildungsprinzipien angegeben. Dann wird das $i+1$-Element einer Folge mit Hilfe der bereits vorher in der Folge auftretenden Elemente berechnet.

Beispiele:

Rekusive Definition der Folge der natürlichen Zahlen:

$$x_0 = 1$$

$$x_{i+1} = x_i + 1$$

Rekusive Definition der Folge der ungeraden Zahlen:

$$x_0 = 1$$

$$x_{i+1} = x_i + 2$$

Schreiben Sie jetzt C-Funktionen, die die folgenden rekursiv definierten Folgen realisieren:

a)

$$x_0 = 0$$

$$x_{i+1} = x_i + (i + 1)$$

b)

$$x_0 = 1$$

$$x_{i+1} = x_i * (i + 1)$$

Aufgabe 5 (1 Punkt) Realisieren Sie weitere rekursiv definierte Folgen:

a) Auch die schon im letzten Übungsblatt betrachteten Fibonaccizahlen sind als Zahlenfolge rekursiv definiert:

$$x_0 = 0$$

$$x_1 = 1$$

$$x_{i+2} = x_i + x_{i+1}$$

b) Folgende Folge hat ein relativ gleichverteiltes zufällig erscheinendes Auftreten der Zahlen zwischen 0 und 99[1]:

$$x_0 = 42$$

$$x_{i+1} = (21 * x_i + 17) \bmod 100$$

[1] mit mod ist der der Modulo-Operator aus der ersten Aufgabe gemeint.

Aufgabe 6 Schreiben Sie ein C Programm, das die folgenden Ausgabe auf der Kommandozeile hat:

```
sep@pc305-4:~/fh/c$ ./a.out
.xxxxxxxxx.xxxxxxxxx.
x.xxxxxxxx.xxxxxxxx.x
xx.xxxxxxx.xxxxxxx.xx
xxx.xxxxxx.xxxxxx.xxx
xxxx.xxxxx.xxxxx.xxxx
xxxxx.xxxx.xxxx.xxxxx
xxxxxx.xxx.xxx.xxxxxx
xxxxxxx.xx.xx.xxxxxxx
xxxxxxxx.x.x.xxxxxxxx
xxxxxxxxx...xxxxxxxxx
.....
xxxxxxxxx...xxxxxxxxx
xxxxxxxx.x.x.xxxxxxxx
xxxxxxx.xx.xx.xxxxxxx
xxxxxx.xxx.xxx.xxxxxx
xxxxx.xxxx.xxxx.xxxxx
xxxx.xxxxx.xxxxx.xxxx
xxx.xxxxxx.xxxxxx.xxx
xx.xxxxxxx.xxxxxxx.xx
x.xxxxxxxx.xxxxxxxx.x
.xxxxxxxxx.xxxxxxxxx.
sep@pc305-4:~/fh/c$
```