

Aufgabenblatt zur 5. Übung

Zeitraum: 14.11. bis 18.11.2011

1. Aufgabe: (AGS 3.9)

Es soll geprüft werden, ob der Inhalt des Zeichenfeldes `feld` der Länge `l` ein Palindrom ist, das heißt, ob die Zeichenkette sowohl von vorne als auch von hinten gelesen gleich lautet.

Ist die Zeichenkette ein Palindrom, so soll der Parameter `korrekt` den Wert `true` repräsentieren, sonst `false`.

(a) Von folgender Funktion wird behauptet, dass sie diese Aufgabe löst:

```
Palindrom1(char feld[], int l, int korrekt) {  
    int i;  
    i = 1; l = l - 1;  
    while (i < l && korrekt) {  
        korrekt = feld[i] == feld[l];  
        i = i + 1;  
    }  
    return korrekt;  
}
```

Prüfen Sie diese Funktion. Korrigieren Sie eventuell enthaltene Fehler.

(b) Schreiben Sie eine Funktion `Palindrom2`, die rekursiv arbeitet. Überlegen Sie sich hierbei als erstes einen geeigneten Funktionskopf.

2. Aufgabe: (AGS 3.6*)

Schreiben Sie für die Berechnung von Fibonacci-Zahlen ein C-Programm, welches eine natürliche Zahl als Eingabe fordert und den zugeordneten Funktionswert ausgibt. Die Berechnung der Fibonacci-Zahl selbst soll mit Hilfe einer Funktion `int fib(int z)` erfolgen.

Realisieren Sie zwei Varianten der Funktion `fib`:

- (a) einen iterativ arbeitenden Berechnungsalgorithmus und
- (b) einen rekursiv arbeitenden Berechnungsalgorithmus.

Machen Sie Aussagen zur Effizienz beider Programmvarianten.

3. Aufgabe: (AGS 4.8)

Gegeben sei das folgende C-Programm:

```
(1) #include <stdio.h>  
(2) int e;  
(3)  
(4) void g(int y, int z, int *x);  
(5)  
(6) void f(int y, int *z) {  
(7)     int u; /* label1 */  
(8)     if (y > 0) {
```

```

(9)     f(y - 1, &u); /* $2 */ /* label2 */
(10)    g(y - 1, u, z); /* $3 */ /* label3 */
(11)    }
(12)    else *z = 1;
(13) }
(14)
(15) void g(int y, int z, int *x) {
(16)     int u; /* label4 */
(17)     if (y > 0) {
(18)         f(y - 1, &u); /* $4 */ /* label5 */
(19)         *x = u + z;
(20)     }
(21)     else *x = 1;
(22) }
(23)
(24) int main() {
(25)     int a;
(26)     scanf("%i", &e); /* label6 */
(27)     f(e, &a); /* $1 */ /* label7 */
(28)     printf("a = %d\n", a);
(29)     return 0;
(30) }

```

(a) Geben Sie den Gültigkeitsbereich jedes Objektes des Programmes an. Nutzen Sie dazu die angegebenen Zeilennummern.

(b) Stellen Sie die Rechnung des Programmes für die Eingabe `e = 1` als pulsierenden Speicher dar, wobei die aktuelle Situation bei jedem Passieren der Marken `label1` bis `label7` gezeigt werden soll. Dokumentieren Sie hierzu jeweils alle *sichtbaren* Variablen mit ihren Wertebelegungen. Hat eine Variable noch keinen Wert erhalten, so geben Sie anstelle des Wertes ein `?` an (also z. B. `x = ?`, wenn `x` zum Zeitpunkt der aktuellen Marke noch unbelegt ist). Führen Sie des Weiteren auch ein Rücksprungmarkenprotokoll.

Beachten Sie weiter: `$1` bis `$4` sind die bereits festgelegten Rücksprungmarken.

Zusatzaufgabe: (AGS 4.2*)

Gegeben sei das folgende C-Programm:

```

1  #include <stdio.h>
2  int a, b, c;
3
4  void g(int x, int y, int *z) {
5      /*label1*/
6      if (x > 0) {
7          g(x - 1, y, z); /*$2*/ /*label7*/
8          *z = *z + 1;
9      }
10     else *z = y;
11 }
12
13 void f(int x, int y, int *z) {
14     int u; /*label2*/
15     if (x > 0) {
16         f(x - 1, y, &u); /*$3*/ /*label3*/
17         g(u, y, z); /*$4*/ /*label4*/
18     }

```

```
19  else *z = 0;
20  }
21
22  int main() {
23      printf("\n Zahl a: ");
24      scanf("%d", &a);
25      printf("\n Zahl b: ");
26      scanf("%d", &b); /*label5*/
27      f(a, b, &c); /*$1*/ /*label6*/
28      printf("\nDas Ergebnis lautet: %d\n\n", c);
29      return 0;
30  }
```

(a) Geben Sie den Gültigkeitsbereich jedes Objektes des Programmes an. Nutzen Sie dazu die angegebenen Zeilennummern.

(b) Stellen Sie die Rechnung des Programmes für die Eingabe $a = 2$ und $b = 2$ als pulsierenden Speicher dar, wobei die aktuelle Situation bei jedem Passieren der Marken `label11` bis `label17` gezeigt werden soll. Führen Sie des Weiteren ein Rücksprungmarkenprotokoll. Dokumentieren Sie hierzu jeweils alle *sichtbaren* Variablen mit ihren Wertebelegungen. Hat eine Variable noch keinen Wert erhalten, so geben Sie anstelle des Wertes ein `?` an (also z. B. $x = ?$, wenn x zum Zeitpunkt der aktuellen Marke noch unbelegt ist). Führen Sie des Weiteren auch ein Rücksprungmarkenprotokoll.

Beachten Sie: `$1` bis `$4` sind die bereits festgelegten Rücksprungmarken.