5. Übung (08.–12. November 2021)

Algorithmen und Datenstrukturen

Aufgabe 1 (AGS 3.1.9, AGS 3.1.10 b)

(a) Gegeben sei die Ackermann-Funktion ack: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Implementieren Sie diese in C.

$$\begin{aligned} \text{ack}(0, y) &= y + 1 && (y \geq 0) \\ \text{ack}(x, 0) &= \text{ack}(x - 1, 1) && (x > 0) \\ \text{ack}(x, y) &= \text{ack}(x - 1, \text{ack}(x, y - 1)) && (x, y > 0) \end{aligned}$$

(b) Schreiben Sie eine Funktion `palindrom` mit zwei Argumenten: ein Array über **`char`**-Symbolen `str` und eine natürliche Zahl `n`, wobei `n` die Länge von `str` ist. Die Funktion soll prüfen ob das Array `str` ein Palindrom ist und das Ergebnis zurückgeben.

Geben Sie einen Aufruf der Funktion für eine eingegebene Zeichenkette an.

Aufgabe 2

(a) Das folgende C-Programm ist gegeben.

```c
#include <stdio.h>

void swoop(int a, int b) {
  /* label 1 */
  a = b;
  b = a;
  /* label 2 */
}

int main() {
  int x = 3, y = 6;
  /* label 3 */
  swoop(x, y); /*$1*/
  /* label 4 */
  printf("x = %d, y = %d", x, y);
  return 0;
}
```

Geben Sie ein Speicherbelgungsprotokoll für dieses Programm an.

(b) Schreiben Sie eine C-Funktion `swap` mit zwei Parametern, welche die Werte der aktuellen Parameter vertauscht.

Aufgabe 3

Gegeben sei das folgende C-Programm:

```
1  #include <stdio.h>
2  int a;
3
4  void g(int a, int *b);
5
6  void f(int *i, int j) {
7    /*label1*/
8    if (*i + j < a) {
9      *i = *i + 1;
10     f(i, j);            /*$1*/
11   }
12   /*label2*/
13 }
14
15 void g(int a, int *b) {
16   int i = 2;
17   /*label3*/
18   while (a != 1) {
19     f(&i, a);           /*$2*/
20     a = a / 2;
21     *b = *b + 1;
22     /*label4*/
23   }
24 }
25
26 int main() {
27   int x = 0;
28   scanf("%i", &a);
29   /*label5*/
30   g(a, &x);             /*$3*/
31   /*label6*/
32   return 0;
33 }
```

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

(b) Führen Sie jedes das folgende Speicherbelegungsprotokoll weiter bis das Programm endet. Dokumentieren Sie die aktuelle Situation beim Passieren der Marken `/*label1*/` bis `/*label6*/`. Geben Sie jeweils den Rücksprungmarkenkeller und die **sichtbaren** Variablen mit ihrer Wertebelegung an. Die Inhalte von Speicherzellen nicht sichtbarer Variablen müssen Sie nur bei Änderungen eintragen. Die bereits festgelegten Rücksprungmarken sind `/*$1*/` bis `/*$3*/`.

HP	RM	Umgebung										
		1	2	3	4	5	6	7	8	9	10	11
`label5`	—	`a` 7	`x` 0									

Zusatzaufgabe 1 (AGS 3.2.22 a,b ⋆, AGS 3.2.42 a ⋆, AGS 3.2.49 a ⋆)

(a) Gegeben sei folgende Funktionsdefinition:

$$f(0) = 3 \qquad f(1) = 5 \qquad f(n) = f(n-1) + 2 \cdot f(n-2) \quad \text{für } n \geq 2$$

Geben Sie in C eine rekursive und eine iterative Funktion zur Berechnung von $f(n)$ für $n \geq 0$ an.

(b) Schreiben Sie eine Funktion **`int qs(int n)`**, welche die Quersumme der Zahl `n` berechnet und zurückgibt. Sie dürfen davon ausgehen, dass `n` ≥ 0 ist.

Tipp: Nutzen Sie die ganzzahlige Division `/` mit Rest `%`.

(c) Vervollständigen Sie die C-Funktion `fill_occurrences`, so dass sie jeder Position `j` des Arrays `b` die Anzahl der Vorkommen von `a[j]` im Array `a` zuweist. Beide Arrays, `a` und `b`, haben die Länge `bound`.

```
void fill_occurrences(int a[], int b[], int bound) {
```

Zusatzaufgabe 2 (AGS 4.32 ⋆)

Gegeben sei das folgende C-Programm:

```
1  #include <stdio.h>
2  int a;
3
4  void g(int x, int *y);
5
6  void f(int *i, int j) {
7    /*label1*/
8    if (*i + j < a) {
9      *i = *i + 1;
10     f(i, j);            /* $1 */
11   }
12   /*label2*/
13 }
14
15 void g(int x, int *y) {
16   int i = 2;
17   /*label3*/
18   while (x != 1) {
19     f(&i, x);           /* $2 */
20     x = x / 2;
21     *y = *y + 1;
22     /*label4*/
23   }
24 }
25
26 int main() {
27   int x = 0;
28   scanf("%i", &a);
29   /*label5*/
30   g(a, &x);             /* $3 */
31   /*label6*/
32   return 0;
33 }
```

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

(b) Führen Sie jedes der drei folgenden Speicherbelegungsprotokolle um jeweils vier Schritte weiter. Dokumentieren Sie die aktuelle Situation beim Passieren der Marken *label1* bis *label6*. Geben Sie jeweils den Rücksprungmarkenkeller und die **sichtbaren** Variablen mit ihrer Wertebelegung an. Die Inhalte von Speicherzellen nicht sichtbarer Variablen müssen Sie nur bei Änderungen eintragen. Die bereits festgelegten Rücksprungmarken sind *$1* bis *$3*.

HP	RM	Umgebung						
		1	2	3	4	5	6	7
label5	—	a 7	x 0					

HP	RM	Umgebung										
		1	2	3	4	5	6	7	8	9	10	11
label1	2:3	a 8	 2	 2	 2	 4	i 5	j 2				

HP	RM	Umgebung										
		1	2	3	4	5	6	7	8	9	10	11
label2	1:1:2:3	a 9	 2	 2	 2	 7	 5	 2	 5	 2	i 5	j 2