
Algorithmen und Datenstrukturen

Aufgabe 1 (AGS 3.1.9 ★)

Gegeben sei die Ackermann-Funktion $\text{ack}: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. Implementieren Sie diese in C.

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

Aufgabe 2 (AGS 3.1.16 ★)

(a) Das folgende C-Programm ist gegeben.

```

1      #include <stdio.h>
2
3      void swoop(int a, int b) {
4          /* label 1 */
5          a = b;
6          b = a;
7          /* label 2 */
8      }
9      int main() {
10         int x = 3, y = 6;
11         /* label 3 */
12         swoop(x, y); /*$1*/
13         /* label 4 */
14         printf("x = %d, y = %d", x, y);
15         return 0;
16     }
```

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 (AGS 4.33 ★)

Gegeben sei das folgende C-Programm:

```

1  #include <stdio.h>
2
3  int a;
4  void g (int i, int j, int *x);
5
6  void f (int i, int *z) {
7      int u;
8      /* label 1 */
9      if (i > 0) {
10         f(i - 1, &u); /* $1 */
11         /* label 2 */
12         g(i - 1, u, z) /* $2 */;
13         /* label 3 */
14     }
15     else
16         *z = 1;
17 }
18
19 void g (int i, int j, int *x) {
20     int u;
21     /* label 4 */
22     if (i > 0) {
23         f(i - 1, &u); /* $3 */
24         /* label 5 */
25         *x = u + j;
26     }
27     else
28         *x = 1;
29 }
30
31 int main () {
32     int b;
33     scanf("%i", &a);
34     /* label 6 */
35     f(a, &b); /* $4 */
36     /* label 7 */
37     printf("%d", b);
38     return 0;
39 }
```

- (a) Tragen Sie den Gültigkeitsbereich jedes Objektes in die Tabelle ein. Nutzen Sie dazu die Zeilennummern.
- (b) Führen Sie jedes der zwei folgenden Speicherbelegungsprotokolle um jeweils fünf Schritte weiter. Dokumentieren Sie die aktuelle Situation beim Passieren der Marken *label 1* bis *label 7*. 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 *\$4*.

HP	RM	Umgebung											
		1	2	3	4	5	6	7	8	9	10	11	12
label1	4	a		i	z	u							
		2		2	2	?							

HP	RM	Umgebung											
		1	2	3	4	5	6	7	8	9	10	11	12
label2	4	a		i	z	u							
		2		2	2	1							

Zusatzaufgabe 1 (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	x					
		7	0					

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

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

Zusatzaufgabe 2 (AGS 3.2.22 a,b, AGS 3.2.42 a *)

(a) Gegeben sei folgende Funktionsdefinition:

$$f(0) = 3$$

$$f(1) = 5$$

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

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 $n \geq 0$ ist.

Beispiel: $qs(162) = 1 + 6 + 2 = 9$

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