

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 5: C & PULSIERENDER SPEICHER

Eric Kunze

`eric.kunze@mailbox.tu-dresden.de`

TU Dresden, 27.11.2020

DIE ACKERMANN-FUNKTION

"Die Ackermannfunktion ist eine 1926 von Wilhelm Ackermann gefundene, extrem schnell wachsende mathematische Funktion, mit deren Hilfe in der theoretischen Informatik Grenzen von Computer- und Berechnungsmodellen aufgezeigt werden können."

Quelle: <https://de.wikipedia.org/wiki/Ackermannfunktion>

Definition von $\text{ack} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

$$\text{ack}(0, y) = y + 1 \quad (y \geq 0)$$

$$\text{ack}(x, 0) = \text{ack}(x - 1, 1) \quad (x > 0)$$

$$\text{ack}(x, y) = \text{ack}(x - 1, \text{ack}(x, y - 1)) \quad (x, y > 0)$$

DIE ACKERMANN-FUNKTION

Definition von $\text{ack} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

$$\text{ack}(0, y) = y + 1 \quad (y \geq 0)$$

$$\text{ack}(x, 0) = \text{ack}(x - 1, 1) \quad (x > 0)$$

$$\text{ack}(x, y) = \text{ack}(x - 1, \text{ack}(x, y - 1)) \quad (x, y > 0)$$

einige Werte

$x \setminus y$	0	1	2	3	4	...	m
0	1	2	3	4	5	...	$m + 1$
1	2	3	4	5	6	...	$m + 2$
2	3	5	7	9	11	...	$2m + 3$
3	5	13	29	61	125	...	$8 * 2^m - 3$
4	13	65533	$2^{65536} - 3$	$\underbrace{2^{2^{\dots^2}}}_{m+3} - 3$

AUFGABE 1

```
1 #include <stdio.h>
2 int ack(int x, int y){
3     int a;
4     if ((x == 0) && (y >= 0))        return y + 1;
5     else if ((x > 0) && (y == 0))    return ack(x-1, 1);
6     else if ((x > 0) && (y > 0)){
7         a = ack(x, y-1);
8         return ack(x-1, a);    }
9 }
10 int main() {
11     int x = 0, y = 0, a;
12     printf("\nAckermannfunktion\n");
13     printf("x = ");    scanf("%d", &x);
14     printf("y = ");    scanf("%d", &y);
15     a = ack(x,y);
16     printf("ack(%i,%i)=%i.\n", x, y, a);
17     return 0;
18 }
```

AUFGABE 2

```
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 }
```

AUFGABE 2 — TEIL (A)

Label	RM	1	2	3	4
label3	—	x 3	y 6		
label1	1			a 3	b 6
label2	1			a 6	b 6
label4	—	x 3	y 6		

AUFGABE 2 — TEIL (B)

```
1 #include <stdio.h>
2 void swap(int *x, int *y){
3     int tmp;
4     tmp = *x;
5     *x = *y;
6     *y = tmp;
7 }
8 int main() {
9     int x = 4, y = 6;
10    printf("x = %d, y = %d \n", x, y);
11    swap(&x, &y);
12    printf("x = %d, y = %d \n", x, y);
13    return 0;
14 }
```

AUFGABE 3

```
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 }
```


AUFGABE 3 — TEIL (A)

Gültigkeitsbereiche

Objektname	Gültigkeitsbereich
a	3 – 39
g	4 – 39
i, j, x in g	19 – 29
u in g	20 – 29
f	6 – 39
i, z in f	6 – 17
u in f	7 – 17
main	31 – 39
b in main	32 – 39

AUFGABE 3 — TEIL (B)

Label	RM	1	2	3	4	5	6	7	8	9	10	11	12
label1	4	a 2		i 2	z 2	u ?							
label1	1:4	a 2					i 1	z 5	u ?				
label1	1:1:4	a 2								i 0	z 8	u ?	
label2	1:4	a 2					i 1	z 5	u 1				
label4	2:1:4	a 2								i 0	j 1	x 5	u ?
label3	1:4	a 2				1	i 1	z 5	u 1				

Label	RM	1	2	3	4	5	6	7	8	9	10	11	12
label2	4	a 2		i 2	z 2	u 1							
label4	2:4	a 2					i 1	j 1	x 2	u ?			
label1	3:2:4	a 2									i 0	z 9	u ?
label5	2:4	a 2					i 1	j 1	x 2	u 1			
label3	4	a 2	2	i 2	z 2	u 1							
label7	ϵ	a 2	b 2										