

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 6: PULSIERENDER SPEICHER

Eric Kunze

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

TU Dresden, 28.11.2019

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  void g(int x, int *y);
4
5  void f(int *x, int y){
6      /* label1 */
7      while (*x < y){
8          *x = *x * 3;
9          /* label2 */
10         g(*x, &y);    /* $1 */
11     }
12 }
13
14 void g(int x, int *y){
15     /* label3 */
16     if (*y < x){
17         *y = *y * 2;
```

```
18         /* label4 */
19         if (x > *y)
20             f(&x, *y);    /* $2 */
21     }
22     /* label5 */
23 }
24
25 int main(){
26     int a, b;
27     a = 3;
28     b = 6;
29     /* label6 */
30     f(&a, b);            /* $3 */
31     /* label7 */
32     printf("%d", a);
33     return 0;
34 }
```


Gültigkeitsbereiche

Objektname	Gültigkeitsbereich
g	3 – 34
f	5 – 34
x,y in f	5 – 12
x,y in g	14 – 23
main	25 – 34
a,b in main	26 – 34

AUFGABE 3 — TEIL (B)

Label	RM	1	2	3	4	5	6	7	8
label6	—	a 3	b 6						
label1	3			x 1	y 6				
label2	3	9		x 1	y 6				
label3	1 : 3					x 9	y 4		
label4	1 : 3				12	x 9	y 4		
label5	1 : 3					x 9	y 4		
label2	1 : 3	27		x 1	y 12				
label3	1 : 3					x 27	y 4		
label4	1 : 3				24	x 27	y 4		
label1	2 : 1 : 3							x 5	y 24
label5	2 : 1 : 3					x 27	y 4		
label7	1 : 3	a 27	b 6						