TECHNISCHE UNIVERSITÄT DRESDEN

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 ack : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

$$\mathbf{ack}(0, y) = y + 1 \qquad (y \geq 0)$$

$$\mathbf{ack}(x, 0) = \mathbf{ack}(x - 1, 1) \qquad (x > 0)$$

$$\mathbf{ack}(x, y) = \mathbf{ack}(x - 1, \mathbf{ack}(x, y - 1)) \qquad (x, y > 0)$$

DIE ACKERMANN-FUNKTION

Definition von ack : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

$$\mathbf{ack}(0, y) = y + 1 \qquad (y \geq 0)$$

$$\mathbf{ack}(x, 0) = \mathbf{ack}(x - 1, 1) \qquad (x > 0)$$

$$\mathbf{ack}(x, y) = \mathbf{ack}(x - 1, \mathbf{ack}(x, y - 1)) \qquad (x, y > 0)$$

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	$2^{65536} - 3$	...	...	...	$\underbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}_{m+3} - 3$

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

```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;
}
```

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)

```c
#include <stdio.h>
void swap(int *x, int *y){
  int tmp;
  tmp = *x;
  *x = *y;
  *y = tmp;
}
int main() {
  int x = 4, y = 6;
  printf("x = %d, y = %d \n", x, y);
  swap(&x, &y);
  printf("x = %d, y = %d \n", x, y);
  return 0;
}
```

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										