TECHNISCHE UNIVERSITÄT DRESDEN

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

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

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

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

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

$n \setminus m$	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

```c
#include <stdio.h>

void g(int x, int *y);

void f(int *x, int y){
  /* label1 */
  while (*x < y){
    *x = *x * 3;
    /* label2 */
    g(*x, &y);    /* $1 */
  }
}

void g(int x, int *y){
  /* label3 */
  if (*y < x){
    *y = *y * 2;
    /* label4 */
    if (x > *y)
      f(&x, *y);  /* $2 */
  }
  /* label5 */
}

int main(){
  int a, b;
  a = 3;
  b = 6;
  /* label6 */
  f(&a, b);       /* $3 */
  /* label7 */
  printf("%d", a);
  return 0;
}
```

AUFGABE 3 — TEIL (A)

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						