5. Übung AuD

Dominic Deckert

6. Januar 2017

C

imperative Sprache
eingeführte Methoden:

- `scanf("%i", &n);`
- `prinf("%i", n);`
- `if(n == 1) { ... } else{ ... }`
- `for(n=1; n<j; n++){ ... }`

Rekursion

Funktion ruft sich selbst mit veränderten Parametern erneut auf Struktur:

```
int rec(int i){
    if(i == c){ ... } //Abbruchfälle
    else{
        rec(i--);  //neuer Aufruf
    }
}
```

Alternative: `while(i != c){ ...}`

a)

- Eingabe: $n \in \mathbb{N}$
- Ausgabe: $fib(n)$

mit $fib(0) = 0$, $fib(1) = 1$,
$fib(n) = fib(n-1) + fib(n-2)$
Ansätze: Kann mit `for`, `while` oder rekursiv gelöst werden

b)

- Eingabe: $n \in \mathbb{N}$
- Ausgabe: $G(n)$

mit $G(0) = 0, G(n) = n - G(G(n-1))$

c)

Rekursion über 2 Funktionen

Sichtbarkeit

Deklaration $\neq$ Initialisierung
Sichtbarkeit:

- Funktion: Ab Deklaration bis Programmende
- globale Variablen: wie Funktion (kann lokal überdeckt werden)
- lokale Variablen: ab Deklaration bis Funktionsende (kann überdeckt werden)

Pointer

Gekennzeichnet als `int*` o.Ä.
Adresse einer anderen Speicherzelle (evtl. außerhalb des privaten Speicherbereiches)

- Adresse (bzw. Pointer) einer Variable `v`: `&v`
- Wert in einer Adresse `a`: `*a`

a)

Objekt	Bereich

a)

Objekt	Bereich
g	3 - 34
f	5 - 34
main	25 - 34
x	5 - 14
y	5 - 14
x	16 - 23
y	16 - 23
a	26 - 34
b	26 - 34

Speicherbelegungsprotokoll

protokolliere Speicher bei Berechnungsvorgang:

- Stoppe bei *label*
  - schreibe alle Rücksprungmarken auf
  - kennzeichne alle sichtbaren Variablen (mit ihrem Namen)
  - gib für alle Variablen Werte an

c)

label	Rücksprungmarke	1	2	3	4	5	6	7	8
label 6	-	a 3	b 6						

c)

label	Rücksprungmarke	1	2	3	4	5	6	7	8	
label 6	-	a 3	b 6							
label 1	3			x #1	y 6					
label 2	3	 9		x #1	y 6					
label 3	1 : 3					x 9	y #4			

c) - Fortsetzung

label	Rücksprungmarke	1	2	3	4	5	6	7	8
label 4	1 : 3				12	x 9	y #4		
label 5	1 : 3					x 9	y #4		
label 2	3	27		x #1	y 12				
label 3	1 : 3					x 27	y #4		

b) - Fortsetzung

label	Rücksprungmarke	1	2	3	4	5	6	7	8
label 4	1 : 3				24	x 27	y #4		
label 1	2 : 1 : 3							x #5	y 24
label 5	1 : 3					x 27	y #4		
label 7	-	a 27	b 6						

a)

Funktion `void swap( ... ){ ... }`:

- vertauscht `x, y` Variablen (außerhalb des eigenen Speichers)
- falls `x` ungerade: erhöhe `y` um eins