



Einfuehrung strukturierte  
Programmierung

[Tutorium]

vision

& [Gruppe 06,07]



# Fragen

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

Offene Punkte aus der Vorlesung





# Gmias - Gruppenmeldung

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

## Bis 29.10.2009

Assignments				
Typ	Gruppengröße	Anmeldung bis	Gruppe/Anmeldung	Aufgaben/Abgabe
Übungsbeispiele	4	2007-10-29 23:59:59	<a href="#">Als Gruppe anmelden</a>	
Hausübungen	1	2007-10-09 23:59:59	Als Gruppe gemeldet [Gruppenstatus] Gruppe: 548	[Details]



**Benutzer**  
Abmelden  
Passwort ändern  
**Studierende**  
meine LVen  
**Tutoren**

Gruppenanmeldung für  
Einführung in die Strukturierte Programmierung - WS06

**Familienname**

**Matrikelnummer**

Altinger

n.a.

Mr. X

0630000

Mr. right

0705333

Mr. wrong

42

[als Gruppe Anmelden](#)

G'mIAS - mein IICM Abgabe



# Zahlendarstellung

1x1.c

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

- <http://www.arndt-bruenner.de/mathe/scripts/Zahlensysteme.htm>

- zB: 123

- Hex: 0x7B

- Oktal: 0173

- Binär: 0b1111011

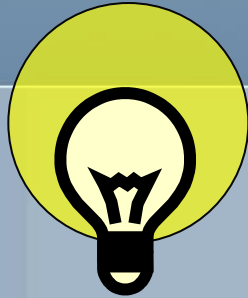
```
altihar@pluto: uname -a
```

```
|  
X --> 32 bit
```

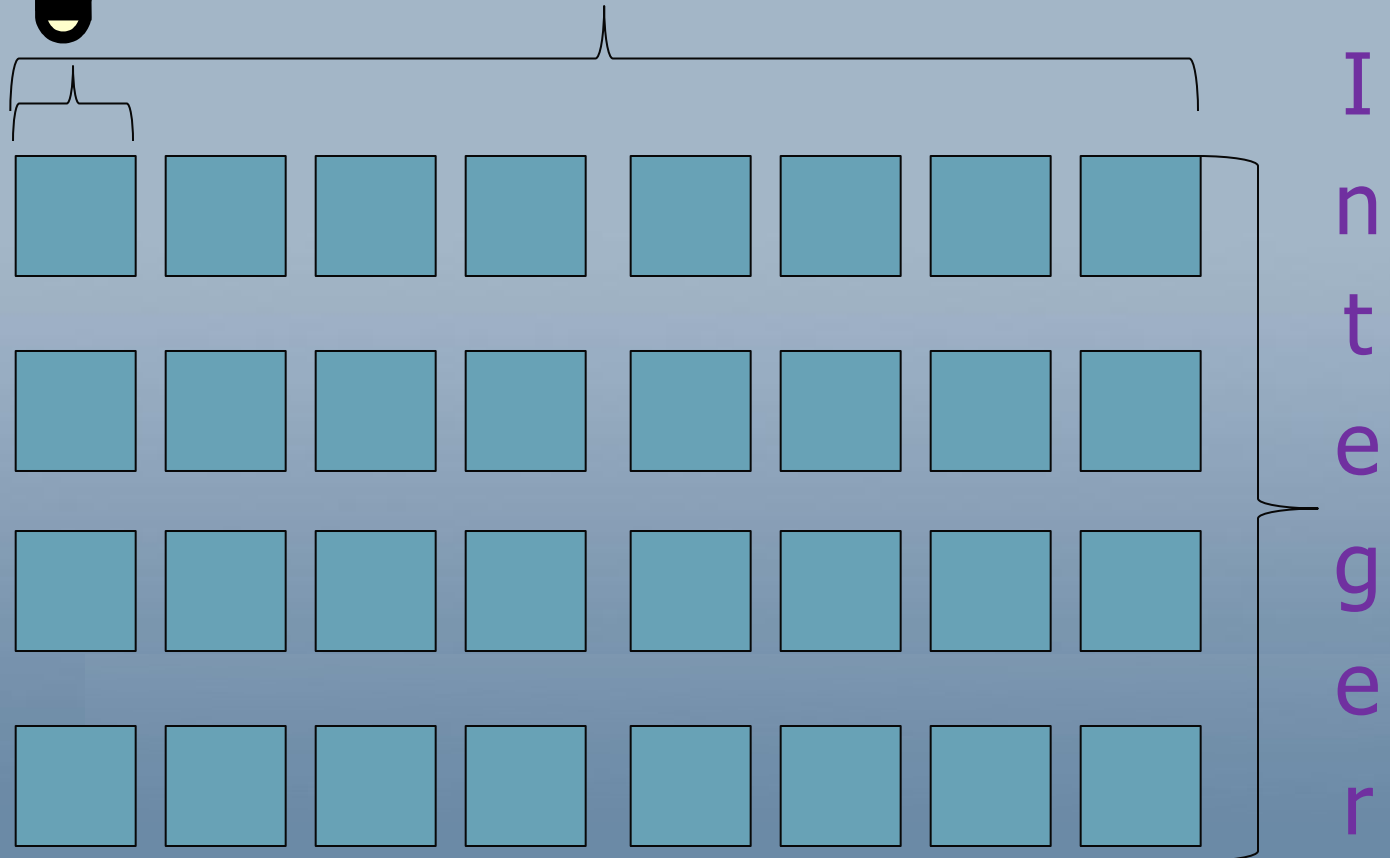


# Zahlendarstellung im RAM

Simplescanf.c



0 ... 255



Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.



# Sinn von Bitoperationen

bitshift\_demo.c, bitshift\_demoII.c

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

- Hardware Anwendungen:
  - setzen eines Bits
  - loeschen eines Bits
  - Auslesen einer Bitfolge
- Software:
  - Multiplizieren
  - Dividieren
- HW3?



# Arten von Bitoperationen

bitshift\_demo.c

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

- $\sim$  bitweises NOT
- $|$  bitweises OR
- $\&$  bitweises AND
- $\wedge$  bitweises XOR
  
- $\ll$  Shift left
- $\gg$  Shift right



# Sinn von Bitoperationen

bitshift\_demo.c

Fragen

Gmias

Zahlen

**Bitop.**

Testen

Funkt.

## Einzelnes Bits pruefen:

```
    01001011 Information
AND 00001000 Bitmaske (1 Bit gesetzt)
```

-----

```
= 00001000 Ergebnis (TRUE,
                        also gesetzt)
```

```
    01000011 Information
AND 00001000 Bitmaske (1 Bit gesetzt)
```

-----

```
= 00000000 Ergebnis (FALSE,
                        also ungesetzt)
```



# Sinn von Bitoperationen

bitshift\_demo.c

Fragen

Gmias

Zahlen

**Bitop.**

Testen

Funkt.

## Einzelnes Bits setzen:

```
      01000011 Information
OR 00001000 Bitmaske
-----
= 01001011 Ergebnis
```



# Sinn von Bitoperationen

bitshift\_demo.c

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

## Einzelnes Bits löschen:

```
NOT  00001000  Bitmaske
-----
= 11110111  invertierte Bitmaske

01001011  Information
AND  11110111  invertierte Bitmaske
-----
= 01000011  Ergebnis
```



# Sinn von Bitoperationen

bitshift\_demo.c

Fragen

## Bits toggeln (umschalten):

Gmias

Zahlen

**Bitop.**

Testen

Funkt.

```
      01000111 Information
XOR  00001100 Bitmaske
-----
      = 01001011 Ergebnis
```

## Bitmasken zusammenfassen

```
      00000100 Bitmaske I
OR   00001000 Bitmaske II
-----
      = 00001100 Ergebnis
```



# Sinn von Bitoperationen

bitshift\_demo.c

Fragen

## Bit-Shifts

Gmias

Zahlen

Bitop.

Testen

Funkt.

<< (left)

00000**1**00 Wert (4)

-----

0000**1**000 Ergebnis (4 \* 2 = 8)

>> (right)

00000**1**00 Wert (4)

-----

000000**1**0 Ergebnis (4 / 2 = 2)



# Logische Operatoren

bitshift\_demo.c

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

■ `!2 == ~2 ?`

■ `(1 & 2) == (1 && 2) ?`

■ Unterschied zwischen Bitweisem und Logischem Operator?



# HW 3

hw3.zip || hw3.tar.gz

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

## ■ Hausuebung 3:

- Scanf

- Bitschift

- Zahlendarstellung

Please enter an integer: 0

00000000|00000000|00000000|00000000

Please enter an integer: 1

00000000|00000000|00000000|000000001

Please enter an integer: 10

00000000|00000000|00000000|00001010

Please enter an integer: -1

11111111|11111111|11111111|11111111



# Wie wir testen

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

compile

- `gcc -o hw3 -Wall hw3.c 2> log_comp`

run

- `./hw3 < testcase_1 > log_tc1`

compare

- `diff log_comp log_ref`
- `diff log_tc1 ref_tc1`

3c3

< you entered: 0

---

> you entered: 10

# Funktionen

asci\_tableII.c asci\_table.c

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

- Conquer and Divide
- Recycling
- Lesbarkeit

```
82 //-----
83 // funktion checks how many lines per funktion are written and
84 // will output an error if they are more than codingstandrad spec.
85 int checkLinesPerFunktion(int count_lines_per_funktion, int line_of_actuall_funktion)
86 {
87     //based on codingstandard 60 lines is maximum
88     if (count_lines_per_funktion > 60)
89         printf("Funktion at line %d has got %d lines\n",line_of_actuall_funktion,
90             count_lines_per_funktion);
91
92     return 0;
93 }
94
```



# Funktionen

functions.c

Fragen

Gmias

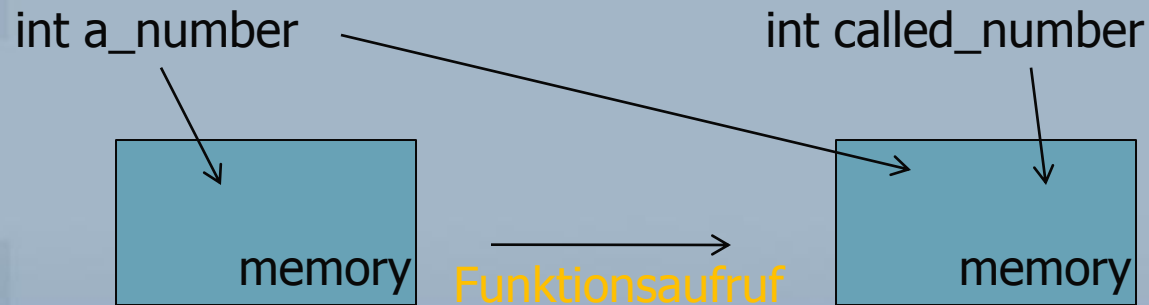
Zahlen

Bitop.

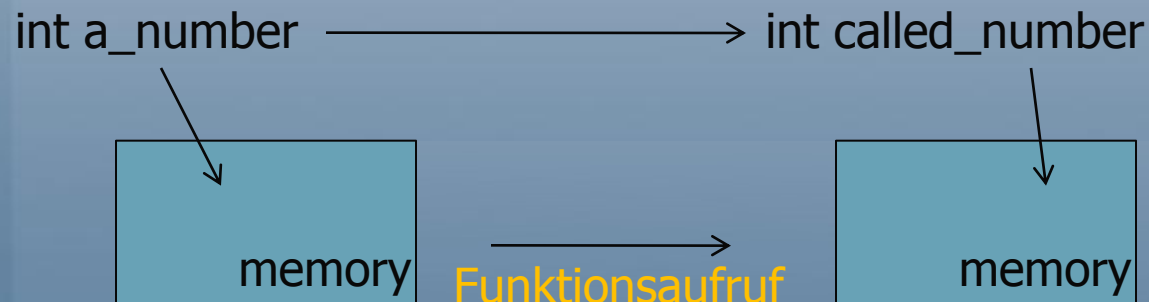
Testen

**Funkt.**

## ■ Call by Value



## ■ Call by Reference





# Arbeit fuer jetzt

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt.

- Demos verstehen
- Den Tutor Fragen wenn was unklar ist!
- HW 3?



**Naechstes Tutorium:  
10.11.2009**



# Fragen

Fragen

Gmias

Zahlen

Bitop.

Testen

Funkt .

? Alle Klarheiten beseitigt ?

