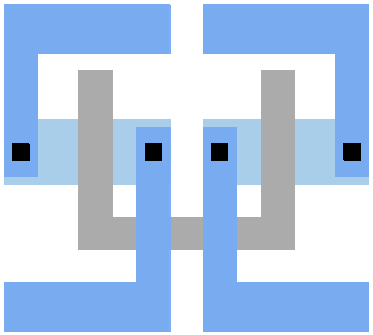
UNIVERSITÄT HEIDELBERG

C++ Kurs
Erläuterungen



Christian Kreidl

christian.kreidl@ziti.uni-heidelberg.de

01.02.2024

Bitmodifikation

```
char bitvector
bitvector = 0b1101;
```

Zahl	7	6	5	4	3	2	1	0
13	0	0	0	0	1	1	0	1

```
char test
test = 1;
test = test << 1;
```

Zahl	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0

```
bitvector & test;
Ergebnis: true
```

bitv.	0	0	0	0	1	1	0	1
test	0	0	0	0	0	0	0	1
&	0	0	0	0	0	0	0	1

```
test = 1<<1;
bitvector & test;
Ergebnis: false
```

bitv.	0	0	0	0	1	1	0	1
test	0	0	0	0	0	0	1	0
&	0	0	0	0	0	0	0	0

Array

Array zugreifen

Beispiel

```
zahlen[0]=100;
zahlen[4]=zahlen[0]+6;
```

int zahlen[5]

[0]	[1]	[2]	[3]	[4]
3	5	4	2	1
100	5	4	2	1
100	5	4	2	106

lese [0], addiere 6, speichere in [4]

```
zahlen[1]=zahlen[1]*2;
```

100	10	4	2	106

lese [1], mult. 2, speichere in [1]

Array zugreifen – Grenzen überschreiten

```
int werte[4];
```

Welcher Inhalt hat ein Array eigentlich am Anfang?

→ unbekannte Werte!!!
Immer zuerst initialisieren!

```
werte[0]=10;
...
werte[4]=14;
```

Es gibt kein Eintrag [4] !!!

Array zugreifen – Grenzen überschreiten

```
int werte[4];
```

?	?	?	?

1245	7642532665235	1	3456327635523

Welcher Inhalt hat ein Array eigentlich am Anfang?

→ unbekannte Werte!!!
Immer zuerst initialisieren!

```
werte[0]=10;
...
werte[4]=14;
```

10	11	12	13	?

Es gibt kein Eintrag [4] !!!

Aber es erfolgt ein Zugriff auf eine Speicherzelle!

```
int test[4];
int x=20;
test[4]=x*2;
cout << x;
```

`int test[4];` `int x;`

1245	-2532665235	1	3456327635523	
1245	-2532665235	1	3456327635523	20
1245	-2532665235	1	3456327635523	40

40 sollte 20 sein!!!

Lese x und mult. 2 .
Gehe vom Array-Anfang 4 Schritte und speichere Ergebnis

g++ Beispiel

Fehlerhafter Zugriff wird durch stack-protector erschwert, daher funktioniert das Beispiel nicht mehr direkt. Dies hängt vom Compiler ab!

```
int main() {
 int x=20;
 int test[1];
 test[1] =30;

 std::cout << "x=" << x;
}
```

g++ -fno-stack-protector -o test test.cpp

Array zugreifen – for Schleife

```
int test[4];
int x=20;
for (int i=0; i<=4; i++ ){
       test[i]=0;
};
cout << x;
```

Schleifen-durchläufe:

i=0 : test[0]=0;

i=1 : test[1]=0;

i=2 : test[2]=0;

i=3 : test[3]=0;

i=4 : test[4]=0;

int test[4];				int x;
1245	-2532665235	1	3456327635523	
1245	-2532665235	1	3456327635523	20
0	-2532665235	1	3456327635523	20
0	0	1	3456327635523	20
0	0	0	3456327635523	20
0	0	0	0	20
0	0	0	0	0

x wurde unerwartet geändert!

Array zugreifen – for Schleife

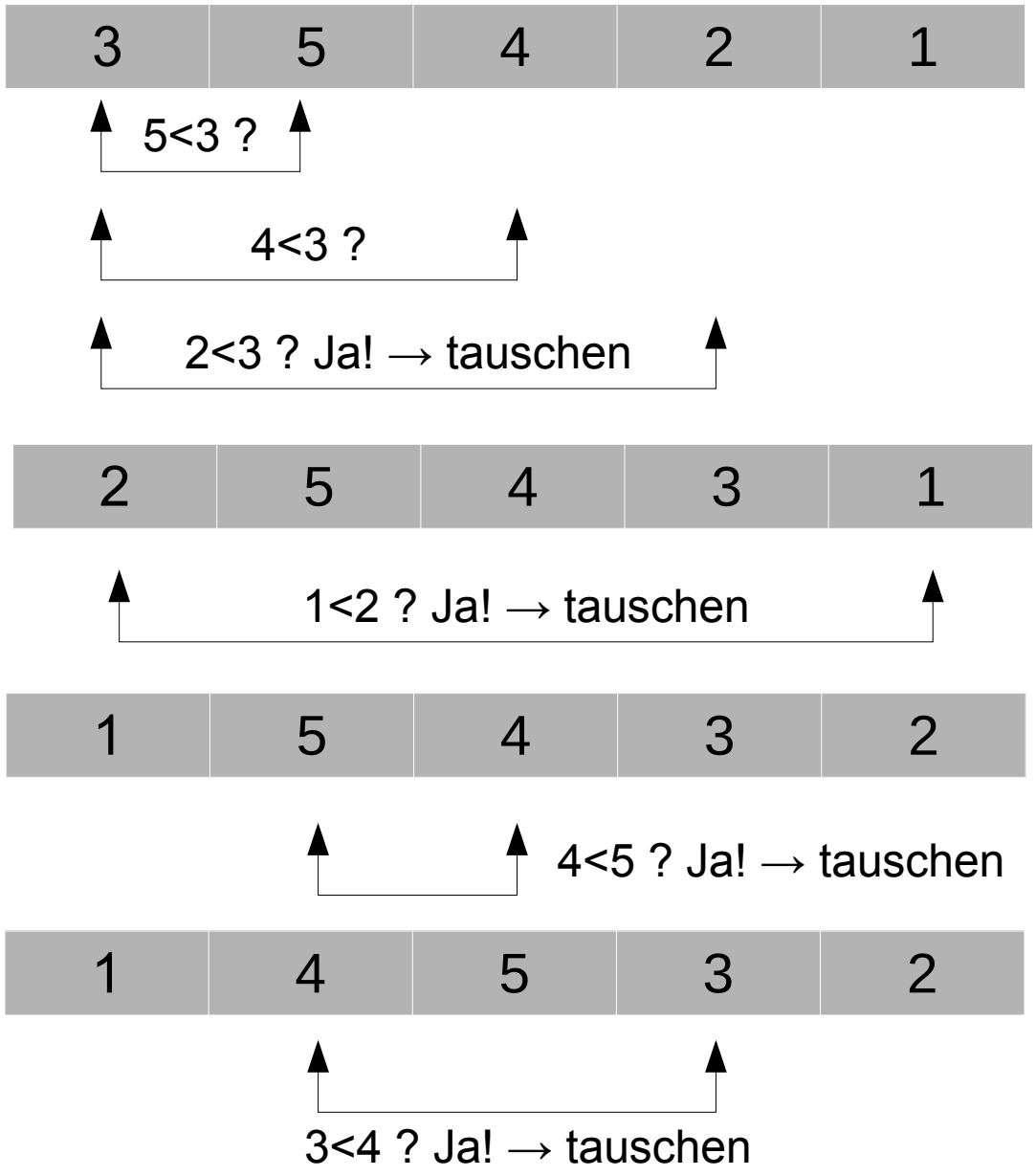
```
int test[4];
for (int i=0; i<=4; i++ ){
       test[i]=0;
};
cout << x;
```

Schleifen-durchläufe:	int test[4];				int i;
	1245	-2532665235	1	3456327635523	
i=0 : test[0]=0;	0	-2532665235	1	3456327635523	0
i=1 : test[1]=0;	0	0	1	3456327635523	1
i=2 : test[2]=0;	0	0	0	3456327635523	2
i=3 : test[3]=0;	0	0	0	0	3
i=4 : test[4]=0;	0	0	0	0	0
i=1 : test[1]=0;	0	0	0	0	1

i wird durch test[4] geändert.
Schleife wird nie enden!

Array: Sortieren

```
int zahlen[5]
```



Pointer

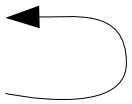
Pointer

```
int i;
int* ptr = &i;

i=10;
*ptr = 20;
```

- erzeuge Speicher für int mit Namen i
- erzeuge Speicher für pointer auf int mit Namen ptr, lese Adresse von i (0) und merke sie in ptr
- speichere 10 in Stelle mit Namen i
- gehe zu ptr und lese Inhalt, gehe an diese Adresse (0) und speichere 20

Name	Addr	Data
`int i`	0	20
`int* ptr`	1	0
	2	
	3	
	4	
	5	



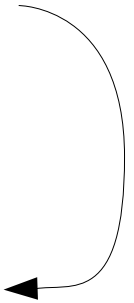
Pointer

```
int i;
int* ptr = &i;

i=10;
*ptr = 20;
ptr = 5;
cout << *ptr;
```

- speichere 5 in Stelle mit Namen ptr
- gehe zu ptr und lese Inhalt, gehe an diese Adresse (5) und lese Inhalt → undefinierter Inhalt!

Name	Addr	Data
`int i`	0	20
`int* ptr`	1	5
	2	
	3	
	4	
	5	?

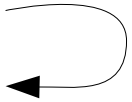


Pointer

```
int i;
int* ptr = &i;
i=10;
*ptr = 20;
ptr = new int;
*ptr = 5;
```

- erzeuge Speicher für int ohne Namen und merke Adresse in ptr
- gehe zu ptr und lese Inhalt, gehe an diese Adresse (2) und speichere 5

Name	Addr	Data
`int i`	0	20
`int* ptr`	1	2
`int`	2	5
	3	
	4	
	5	



Pointer

```
int i;
int* ptr = &i;
i=10;
*ptr = 20;
ptr = new int;
*ptr = 5;
```

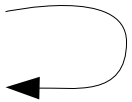
```
char* c = new char;
```

- erzeuge Speicher für Pointer auf char, dann erzeuge Speicher für char ohne Namen und merke Adresse in c

```
*c = 'a';
```

- gehe zu c und lese Inhalt, gehe an diese Adresse (4) und speichere ‘a’

Name	Addr	Data
`int i`	0	20
`int* ptr`	1	2
`int`	2	5
`char* c`	3	4
`char`	4	a
	5	

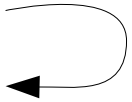


Pointer

```
int i;
int* ptr = &i;
i=10;
*ptr = 20;
ptr = new int;
*ptr = 5;
char* c = new char;
*c = 'a';
delete ptr;
```

•gebe durch ptr verwendeten Speicher frei

Name	Addr	Data
int i	0	20
int* ptr	1	2
	2	5
char* c	3	4
char	4	a
	5	



Referenz

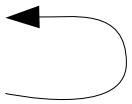
```
int i;
int &ptr = i;

i=10;
ptr = 20;
```

- erzeuge Speicher für int mit Namen i
- erzeuge Speicher für pointer auf int mit Namen ptr, lese Adresse von i (0) und merke sie in ptr
- speichere 10 in Stelle mit Namen i
- gehe zu ptr und lese Inhalt, gehe an diese Adresse (0) und speichere 20

Referenz ist wie ein Pointer, dem automatisch gefolgt (dereferenziert) wird.

Name	Addr	Data
`int i`	0	20
`int &ptr`	1	0
	2	
	3	
	4	
	5	



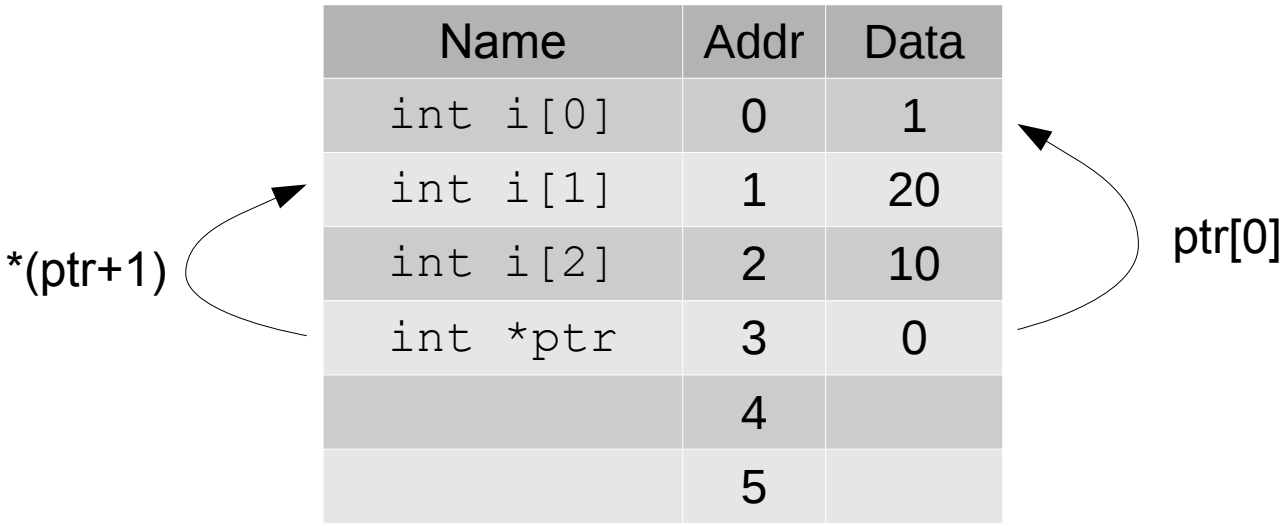
Array und Pointer

```
int i[3];
int *ptr = i;

i[2]=10;
*(ptr+1) = 20;

ptr[0]=1;
```

- erzeuge Speicher für Array aus int mit Namen i und 3 Einträge
- erzeuge Speicher für pointer auf int mit Namen ptr, lese Adresse von i (0) und merke sie in ptr
- speichere 10 in Stelle mit Namen i
- gehe zu ptr und lese Inhalt, gehe an diese Adresse (0) und speichere 20



Name	Addr	Data
int i[0]	0	1
int i[1]	1	20
int i[2]	2	10
int *ptr	3	0
	4	
	5	

Danke!