

ProgrammierParadigmen

Übung - Gruppe 6 & 7

Tobias Kahlert

C Warmup: Datentypen

a) `sizeof(char)` = ?

b) `sizeof(int)` = ?

c) `sizeof(float)` = ?

C Warmup: const

- a) `const int* x = ...;`
- b) `const int const* x = ...;`
- c) `int const* const x = ...;`

Welches von den Folgenden kompiliert?

- a) `x = NULL;`
- b) `*x = 5;`

C Warmup: const

a) `const int* x = ...;` `x = NULL;`
 `*x = 5;`

b) `const int const* x = ...;`

c) `int const* const x = ...;`

Welches von den Folgenden kompiliert?

a) `x = NULL;`

b) `*x = 5;`

C Warmup: const

a) `const int* x = ...;` `x = NULL;`
 `*x = 5;`

b) `const int const* x = ...;` `x = NULL;`
 `*x = 5;`

c) `int const* const x = ...;`

Welches von den Folgenden kompiliert?

a) `x = NULL;`

b) `*x = 5;`

C Warmup: const

a) `const int* x = ...;` `x = NULL;`
 `*x = 5;`

b) `const int const* x = ...;` `x = NULL;`
 `*x = 5;`

c) `int const* const x = ...;` `x = NULL;`
 `*x = 5;`

Welches von den Folgenden kompiliert?

a) `x = NULL;`

b) `*x = 5;`

C Warmup: const

a) `const int* x = ...;` `x = NULL;`
 `*x = 5;`

b) `const int const* x = ...;` `x = NULL;`
 `*x = 5;`

c) `int const* const x = ...;` `x = NULL;`
 `*x = 5;`

Welches von den Folgenden kompiliert?

a) `x = NULL;`

b) `*x = 5;`

d) `int*** p; // Wie komplett konstant machen?`

C Warmup: const

a) `const int* x = ...;` `x = NULL;`
 `*x = 5;`

b) `const int const* x = ...;` `x = NULL;`
 `*x = 5;`

c) `int const* const x = ...;` `x = NULL;`
 `*x = 5;`

Welches von den Folgenden kompiliert?

a) `x = NULL;`

b) `*x = 5;`

d) `int*** p; // Wie komplett konstant machen?`
 `-> int const *const *const *const p;`

C Warmup: Typedefinitionen

a) `int* a, *b, c;`

Welchen Typ haben a, b und c?

C Warmup: Typedefinitionen

a) `int* a, *b, c;`

Welchen Typ haben a, b und c?

`a = int*`

`b = int*`

`c = int`

C Warmup: Arrays

```
void test1() {  
    char str[] = "hello";  
    printf("%i", sizeof(str));  
}
```

```
void test2() {  
    char* str = "hello";  
    printf("%i", sizeof(str));  
}
```


```
void test3(char str[]) {  
    printf("%i", sizeof(str));  
}
```

```
void test4(char str[9]) {  
    printf("%i", sizeof(str));  
}
```

```
void test5(char* str) {  
    printf("%i", sizeof(str));  
}
```

C Warmup: Arrays

```
void test1() {  
    char str[] = "hello";  
    printf("%i", sizeof(str));  
}
```



```
void test2() {  
    char* str = "hello";  
    printf("%i", sizeof(str));  
}
```


```
void test3(char str[]) {  
    printf("%i", sizeof(str));  
}
```

```
void test4(char str[9]) {  
    printf("%i", sizeof(str));  
}
```


```
void test5(char* str) {  
    printf("%i", sizeof(str));  
}
```

C Warmup: Arrays

```
void test1() {  
    char str[] = "hello";  
    printf("%i", sizeof(str));  
}
```



```
void test2() {  
    char* str = "hello";  
    printf("%i", sizeof(str));  
}
```




```
void test3(char str[]) {  
    printf("%i", sizeof(str));  
}
```

```
void test4(char str[9]) {  
    printf("%i", sizeof(str));  
}
```


```
void test5(char* str) {  
    printf("%i", sizeof(str));  
}
```

C Warmup: Arrays


```
void test1() {  
    char str[] = "hello";  
    printf("%i", sizeof(str));  
}
```



```
void test2() {  
    char* str = "hello";  
    printf("%i", sizeof(str));  
}
```



```
void test3(char str[]) {  
    printf("%i", sizeof(str));  
}
```




```
void test4(char str[9]) {  
    printf("%i", sizeof(str));  
}
```


```
void test5(char* str) {  
    printf("%i", sizeof(str));  
}
```

C Warmup: Arrays


```
void test1() {  
    char str[] = "hello";  
    printf("%i", sizeof(str));  
}
```




```
void test2() {  
    char* str = "hello";  
    printf("%i", sizeof(str));  
}
```



```
void test3(char str[]) {  
    printf("%i", sizeof(str));  
}
```




```
void test4(char str[9]) {  
    printf("%i", sizeof(str));  
}
```




```
void test5(char* str) {  
    printf("%i", sizeof(str));  
}
```

C Warmup: Arrays


```
void test1() {  
    char str[] = "hello";  
    printf("%i", sizeof(str));  
}
```




```
void test2() {  
    char* str = "hello";  
    printf("%i", sizeof(str));  
}
```




```
void test3(char str[]) {  
    printf("%i", sizeof(str));  
}
```



```
void test4(char str[9]) {  
    printf("%i", sizeof(str));  
}
```



```
void test5(char* str) {  
    printf("%i", sizeof(str));  
}
```



C Warmup: Arrays

```
void test4(char str[9]) {  
    printf("%i", sizeof(str));  
}
```

4

Aber mit Referenzen in C++ gehts

C++ 4.9.2 (Gcc-4.9.2)

```
1  #include <iostream>
2  using namespace std;
3
4  void func(const char(&str)[7])
5  {
6      // ...
7  }
8
9  int main()
10 {
11     func("test");
12     return 0;
13 }
```

```
prog.cpp: In function 'int main()':
prog.cpp:11:16: error: invalid initialization of reference of type
               'const char (&)[7]' from expression of type 'const char [5]'
               func("test"); // Works, N is 5
```

C Warmup: Arrays

```
void test() {  
    char* str = "hello world!";  
    str[0] = toupper(str[0]);  
    printf("%s", str);  
}
```

C Warmup: Arrays

Liegt im evtl. im Datensegment
und kann nicht geschrieben
werden!

```
void test() {  
    char* str = "hello world!";  
    str[0] = toupper(str[0]);  
    printf("%s", str);  
}
```

C Warmup: Arrays

Müsste eigentlich einen Fehler
oder eine Warnung geben!

Cast: `const char* -> char*`

Liegt im evtl. im Datensegment
und kann nicht geschrieben
werden!

```
void test() {  
    char* str = "hello world!";  
    str[0] = toupper(str[0]);  
    printf("%s", str);  
}
```

C Warmup: Arrays

Müsste eigentlich einen Fehler
oder eine Warnung geben!

Cast: `const char* -> char*`

Liegt im evtl. im Datensegment
und kann nicht geschrieben
werden!

```
void test() {  
    char* str = "hello world!";  
    str[0] = toupper(str[0]);  
    printf("%s", str);  
}
```

Durch Verwendung von Array
kann der Fehler behoben
werden.

Pointers



<https://xkcd.com/138/>

Blocking/Non-Blocking vs Buffering

- Es gibt 8 MPI Befehle zum senden

Blocking/Non-Blocking vs Buffering

- Es gibt 8 MPI Befehle zum senden
- 4 davon sind blocking
 - MPI_Send - Standard-mode
 - MPI_Bsend - Buffered-mode
 - MPI_Ssend - Synchronous-mode
 - MPI_Rsend - Ready-mode

Blocking/Non-Blocking vs Buffering

- Es gibt 8 MPI Befehle zum senden
- 4 davon sind blocking
 - MPI_Send - Standard-mode
 - MPI_Bsend - Buffered-mode
 - MPI_Ssend - Synchronous-mode
 - MPI_Rsend - Ready-mode
- 4 sind non-blocking
 - MPI_Isend
 - MPI_Ibsend
 - MPI_Issend
 - MPI_Irsend

Blocking/Non-Blocking vs Buffering

- Es gibt 8 MPI Befehle zum senden
- 4 davon sind blocking
 - MPI_Send - Standard-mode
 - MPI_Bsend - Buffered-mode
 - MPI_Ssend - Synchronous-mode
 - MPI_Rsend - Ready-mode
- 4 sind non-blocking
 - MPI_Isend
 - MPI_Ibsend
 - MPI_Issend
 - MPI_Irsend

Mehr Infos:

<https://stackoverflow.com/a/47041382/1393971>

Blocking/Non-Blocking vs Buffering

- Es gibt 8 MPI Befehle zum senden
- 4 davon sind blocking
 - MPI_Send - Standard-mode
 - MPI_Bsend - Buffered-mode
 - MPI_Ssend - Synchronous-mode
 - MPI_Rsend - Ready-mode
- 4 sind non-blocking
 - MPI_Isend
 - MPI_Ibsend
 - MPI_Issend
 - MPI_Irsend

Gute Nachricht:
Es gibt nur zwei zum Empfangen:
MPI_Recv und MPI_Irecv

Blatt 10

- C: Zeiger-Arithmetik, Arrays
- MPI: Punkt-zu-Punkt-Kommunikation
- MPI: Reduce
- MPI: Scatter & Allgather
- (MPI: Matrizenmultiplikation)