

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

# Programming in C++ - Primer

## Lesson 3 - Beyond the Pointers

Jakub 'Eremiell' Marek  
<marekj14@fel.cvut.cz>

Silicon Hill C++ Academy

2013/11/11

## C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

**1** Revision

**2** Strings

**3** Containers

**4** Structs

**5** Tricks

# Welcome!

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- `type name[size]`

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- `type name[size]`
- fixed size

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- `type name[size]`
- fixed size
- 0 to size - 1



# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- `type name[size]`
- fixed size
- 0 to size - 1

## Variable visibility

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- `type name[size]`
- fixed size
- 0 to size - 1

## Variable visibility

- global/local

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- type name[size]
- fixed size
- 0 to size - 1

## Variable visibility

- global/local
- block visibility

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- `type name[size]`
- fixed size
- 0 to size - 1

## Variable visibility

- global/local
- block visibility
- overshadowing

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- type name[size]
- fixed size
- 0 to size - 1

## Variable visibility

- global/local
- block visibility
- overshadowing

## Overloading

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- `type name[size]`
- fixed size
- 0 to size - 1

## Variable visibility

- global/local
- block visibility
- overshadowing

## Overloading

- same name, different function

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- `type name[size]`
- fixed size
- 0 to size - 1

## Variable visibility

- global/local
- block visibility
- overshadowing

## Overloading

- same name, different function
- must have different input types

# Diving deeper

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

## Arrays

- `type name[size]`
- fixed size
- 0 to size - 1

## Variable visibility

- global/local
- block visibility
- overshadowing

## Overloading

- same name, different function
- must have different input types
- doesn't care about names



C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Passing value by:

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Passing value by:

- Value

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Passing value by:

- Value
- Reference

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Passing value by:

- Value
- Reference

Pointers are:

Passing value by:

- Value
- Reference

Pointers are:

- Addresses of memory

Passing value by:

- Value
- Reference

Pointers are:

- Addresses of memory
- Variables like any other

Passing value by:

- Value
- Reference

Pointers are:

- Addresses of memory
- Variables like any other

Two kinds of pointers:



Passing value by:

- Value
- Reference

Pointers are:

- Addresses of memory
- Variables like any other

Two kinds of pointers:

- Pointers \*

Passing value by:

- Value
- Reference

Pointers are:

- Addresses of memory
- Variables like any other

Two kinds of pointers:

- Pointers \*
- References &

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Two pointer related operators:

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Two pointer related operators:

- Reference &

Two pointer related operators:

- Reference &
- Defererence \*

Two pointer related operators:

- Reference &
- Defererence \*

Segfaults

Two pointer related operators:

- Reference &
- Defererence \*

Segfaults

Command line arguments



Two pointer related operators:

- Reference &
- Defererence \*

Segfaults

Command line arguments

Two kinds of memory allocation:

Two pointer related operators:

- Reference &
- Defererence \*

Segfaults

Command line arguments

Two kinds of memory allocation:

- Static

Two pointer related operators:

- Reference &
- Defererence \*

Segfaults

Command line arguments

Two kinds of memory allocation:

- Static
- Dynamic

Two pointer related operators:

- Reference &
- Defererence \*

Segfaults

Command line arguments

Two kinds of memory allocation:

- Static
- Dynamic

Dynamic allocation operators:

Two pointer related operators:

- Reference &
- Defererence \*

Segfaults

Command line arguments

Two kinds of memory allocation:

- Static
- Dynamic

Dynamic allocation operators:

- new

Two pointer related operators:

- Reference &
- Deference \*

Segfaults

Command line arguments

Two kinds of memory allocation:

- Static
- Dynamic

Dynamic allocation operators:

- new
- delete, delete[]

# C Strings

C++ Primer

Jakub Marek

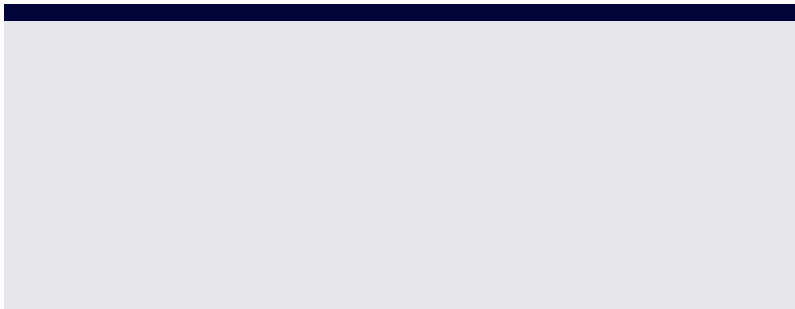
Revision

**Strings**

Containers

Structs

Tricks



# C Strings

C++ Primer

Jakub Marek

Revision

**Strings**

Containers

Structs

Tricks

arrays of chars



# C Strings

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

```
arrays of chars  
terminated by null byte \0
```

# C Strings

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

arrays of chars  
terminated by null byte `\0`  
if not terminated properly, will brake

# C Strings

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

arrays of chars  
terminated by null byte `\0`  
if not terminated properly, will brake

```
#include <iostream>
```

```
int main() {  
    char cs[6] = {'a', 'b', 'c', 'd', 'e', '\0'};  
    char s[] = "deHlorW !";  
    std::cout << s[2] << s[1] << s[3] << s[3] << s[4] << s[7];  
    std::cout << s[6] << s[4] << s[5] << s[3] << s[0] << s[8] << std::endl;  
    std::cout << sizeof(s) << std::endl;  
    return 0;  
}
```

# Why C++ Strings?

C++ Primer

Jakub Marek

Revision

**Strings**

Containers

Structs

Tricks



# Why C++ Strings?

C++ Primer

Jakub Marek

Revision

**Strings**

Containers

Structs

Tricks

C strings are inflexible

# Why C++ Strings?

C++ Primer

Jakub Marek

Revision

**Strings**

Containers

Structs

Tricks

C strings are inflexible  
they're arrays = they're fixed size

# Why C++ Strings?

C++ Primer

Jakub Marek

Revision

**Strings**

Containers

Structs

Tricks

C strings are inflexible  
they're arrays = they're fixed size  
easy to miss the end `\0`

# C++ Strings

C++ Primer

Jakub Marek

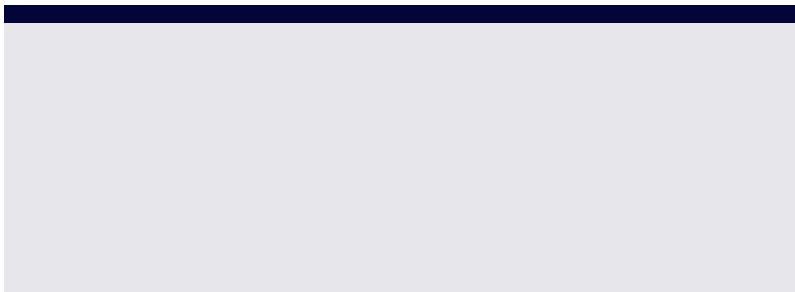
Revision

**Strings**

Containers

Structs

Tricks





# C++ Strings

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

```
class string in library <string>
```

# C++ Strings

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

```
class string in library <string>  
fully mutable
```

# C++ Strings

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

```
class string in library <string>  
fully mutable  
encapsulated object
```

# C++ Strings

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

class string in library `<string>`  
fully mutable  
encapsulated object  
easily convertible

# C++ Strings

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

```
class string in library <string>
fully mutable
encapsulated object
easily convertible

std::string s(<cstring>)
s.c_str()
```

# Questions?

C++ Primer

Jakub Marek

Revision

**Strings**

Containers

Structs

Tricks

C++ Primer

Jakub Marek

Revision

Strings

**Containers**

Structs

Tricks

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

just as strings are encapsulated char arrays, vectors are encapsulated arrays in general



# Vectors

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

just as strings are encapsulated char arrays, vectors are encapsulated arrays in general  
class vector in library `<vector>`

# Vectors

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

just as strings are encapsulated char arrays, vectors are  
encapsulated arrays in general  
class vector in library `<vector>`  
fully mutable

just as strings are encapsulated char arrays, vectors are encapsulated arrays in general  
class vector in library `<vector>`  
fully mutable

```
#include <vector>
```

```
#include <iostream>
```

```
int main() {  
    std::vector<int> v();  
    v.push_back(5);  
    std::cout << v[0] << std::endl;  
    return 0;  
}
```

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

**Containers**

Structs

Tricks

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

**Containers**

Structs

Tricks

A lot of useful stuff

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues
- stacks



# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues
- stacks
- lists

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues
- stacks
- lists
- C libraries

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues
- stacks
- lists
- C libraries
- file streams

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues
- stacks
- lists
- C libraries
- file streams
- iterators

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues
- stacks
- lists
- C libraries
- file streams
- iterators
- algorithms

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues
- stacks
- lists
- C libraries
- file streams
- iterators
- algorithms
- constants

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues
- stacks
- lists
- C libraries
- file streams
- iterators
- algorithms
- constants
- regexes

# Standard Template Library (STL)

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

A lot of useful stuff

- maps
- queues
- stacks
- lists
- C libraries
- file streams
- iterators
- algorithms
- constants
- regexes
- multithreading



# Questions?

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

# Structs

C++ Primer

Jakub Marek

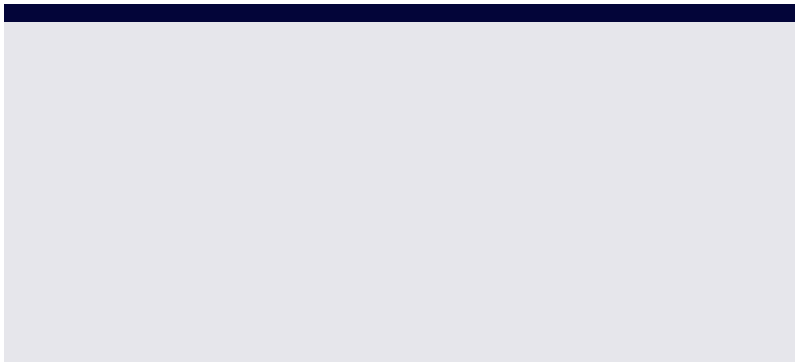
Revision

Strings

Containers

**Structs**

Tricks



C++ Primer

Jakub Marek

Revision

Strings

Containers

**Structs**

Tricks

simple objects

# Structs

C++ Primer

Jakub Marek

Revision

Strings

Containers

**Structs**

Tricks

simple objects  
can only have members, no methods

C++ Primer

Jakub Marek

Revision

Strings

Containers

**Structs**

Tricks

simple objects  
can only have members, no methods  
from C

C++ Primer

Jakub Marek

Revision

Strings

Containers

**Structs**

Tricks

simple objects  
can only have members, no methods  
from C  
form new data types

simple objects  
can only have members, no methods  
from C  
form new data types

```
struct s {  
    int i;  
    double d;  
};
```

C++ Primer

Jakub Marek

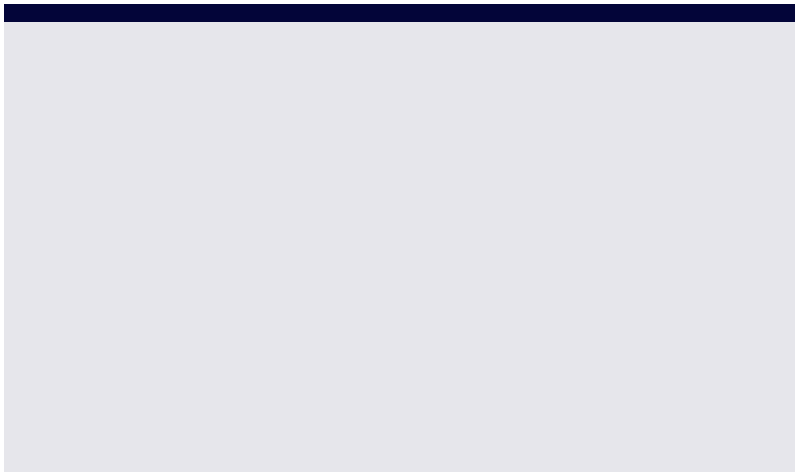
Revision

Strings

Containers

**Structs**

Tricks





C++ Primer

Jakub Marek

Revision

Strings

Containers

**Structs**

Tricks

structs with overlaying memory

C++ Primer

Jakub Marek

Revision

Strings

Containers

**Structs**

Tricks

structs with overlaying memory  
make different data types save and load into same space

C++ Primer

Jakub Marek

Revision

Strings

Containers

**Structs**

Tricks

structs with overlaying memory  
make different data types save and load into same space  
good for coding/decoding various HW flags etc.

structs with overlaying memory  
make different data types save and load into same space  
good for coding/decoding various HW flags etc.

```
union u {  
    int i;  
    char a, b, c, d;  
    unsigned b0: 1;  
    unsigned b1: 1;  
    unsigned b2: 1;  
    unsigned b3: 1;  
};
```

# Questions?

C++ Primer

Jakub Marek

Revision

Strings

Containers

**Structs**

Tricks

# Partial using

C++ Primer

Jakub Marek

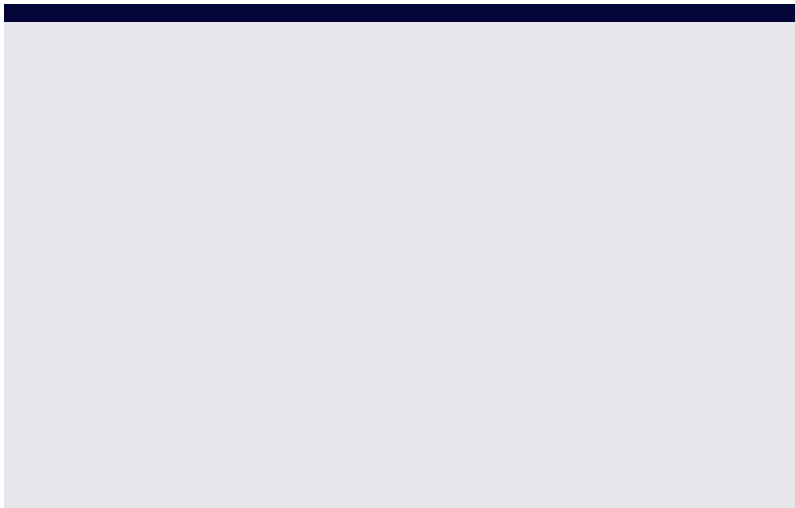
Revision

Strings

Containers

Structs

Tricks



# Partial using

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

using namespace is bad

# Partial using

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

using namespace is bad  
writing std:: is a bit tedious



# Partial using

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

using namespace is bad  
writing std:: is a bit tedious  
you can use just some elements

# Partial using

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

using namespace is bad  
writing std:: is a bit tedious  
you can use just some elements

```
#include <iostream>
```

```
using cout;
```

```
using endl;
```

```
int main() {  
    cout << "abc" << endl;  
    return 0;  
}
```

# Whole Number Logic

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

# Whole Number Logic

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Boolean is a number

# Whole Number Logic

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Boolean is a number  
Can only be 0 or 1

# Whole Number Logic

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Boolean is a number

Can only be 0 or 1

Any other type gets explicit conversion

# Whole Number Logic

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Boolean is a number

Can only be 0 or 1

Any other type gets explicit conversion

- $0 \rightarrow 0$  (False)

# Whole Number Logic

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Boolean is a number

Can only be 0 or 1

Any other type gets explicit conversion

- 0 -> 0 (False)
- anything else -> 1 (True)



# Whole Number Logic

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Boolean is a number

Can only be 0 or 1

Any other type gets explicit conversion

- 0 -> 0 (False)
- anything else -> 1 (True)

NULL, EOF, \0 etc. classifies as 0

# Whole Number Logic

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

Boolean is a number

Can only be 0 or 1

Any other type gets explicit conversion

- 0 -> 0 (False)
- anything else -> 1 (True)

NULL, EOF, \0 etc. classifies as 0  
so you can:

```
if (i) {...}  
if (!i) {...}  
if (getState()) {...}  
while (line = instr.getline()) {...}  
if (ptr) {...}  
if (!ptr) {...}
```

# Questions?

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks

# Break!

C++ Primer

Jakub Marek

Revision

Strings

Containers

Structs

Tricks