# C++

Daniel Zint, Rafael Ravedutti, Harald Köstler

23.09.2019

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

## Organization

- 23.09 – 27.09 (*Daniel Zint*)
  Tutorial covering basics in C++
- 01.10 – 02.10 (*Daniel Zint and Rafael Ravedutti*)
  Project: Tetris
- 03.10 – 05.10 (*Harald Köstler*)
  Lectures covering advanced topics in C++

# Lecture 1

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

## Schedule 23.09.2019

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

- Introduction:
  C++: Complicated Programming since 1979
- Setting up a coding environment
  - Hello World
  - Compilation and Execution
  - Setting up an IDE
- Practical session
- Variables, basic types, and structs
- if, while, for
- Practical session
- Debugging tools (gdb)
- Practical session

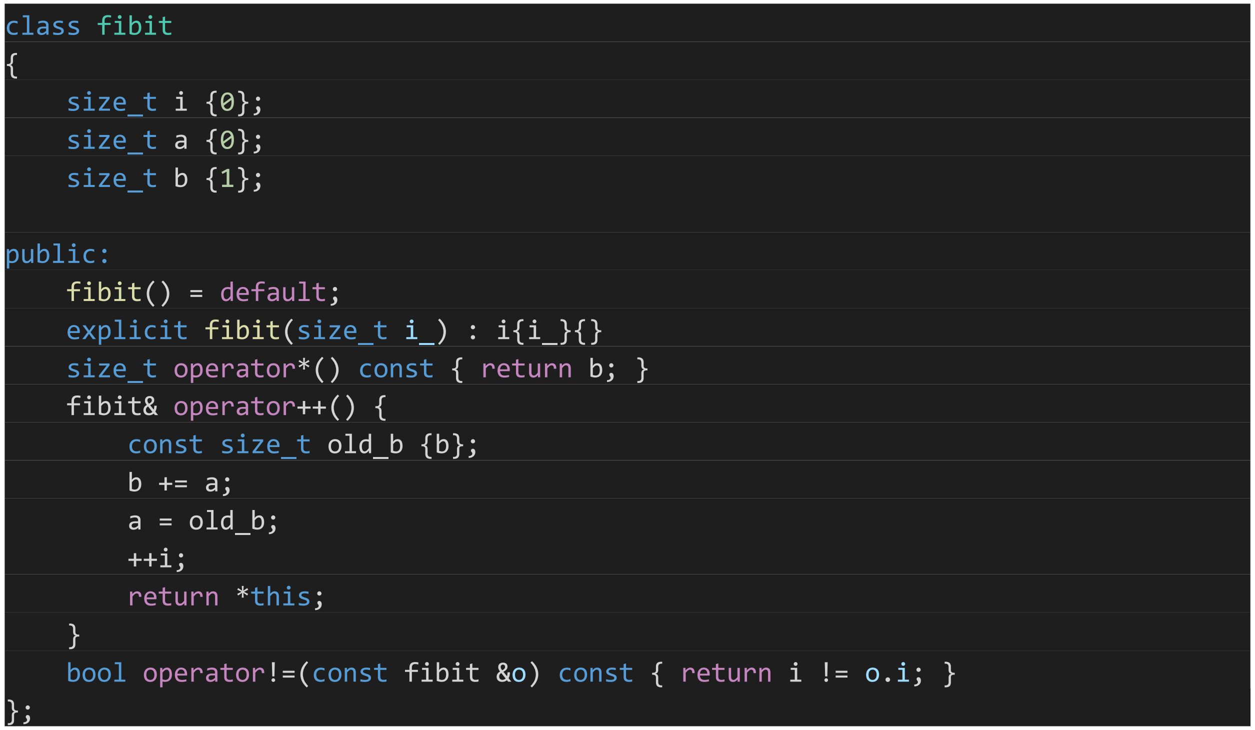
## Literature

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

- S. Lippman et al, *C++ Primer*, 5th edition. Addison Wesley, 2012 (www.awprofessional.com/cpp_primer) http://www.informit.com/store/c-plus-plus-primer-9780321714114

# Why C++?

- A typical C++ Code in C++17:



```cpp
class fibit
{
    size_t i {0};
    size_t a {0};
    size_t b {1};

public:
    fibit() = default;
    explicit fibit(size_t i_) : i{i_}{}
    size_t operator*() const { return b; }
    fibit& operator++() {
        const size_t old_b {b};
        b += a;
        a = old_b;
        ++i;
        return *this;
    }
    bool operator!=(const fibit &o) const { return i != o.i; }
};
```

- Looks way more complicated than it actually is.
- It won't get better after time. It will always look complicated.

## So, why C++??

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

- It’s easy to write efficient code.
- There are plenty different ways for doing the same thing (you can write in your own coding style, e.g. imperative, object-oriented, functional).
- Backward compatibility, even to C code.
- Hard to get rid of: C++ wide spread.
- So far no real alternative that delivers comparable performance.

## What is so special about C++?

- Static language → everything is compiled before execution (not like Java, Python, C#, JavaScript, etc.)
  → fast!
- Object Orientation: the most important advantage over classic C
- Extremely flexible, C++ is used for
  - Operating systems
  - Embedded systems
  - Drivers
  - Signal processors
  - Applications
  - Games
  - Compilers
  - …
- Compatible to C

## C++ Toolchain

- Write some C++ Code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!" << std::endl;
}
```

- Compile
  - Compile all external files
  - Compile main file and link all external files
- Run executable
  - Linux: .out
  - Windows: .exe

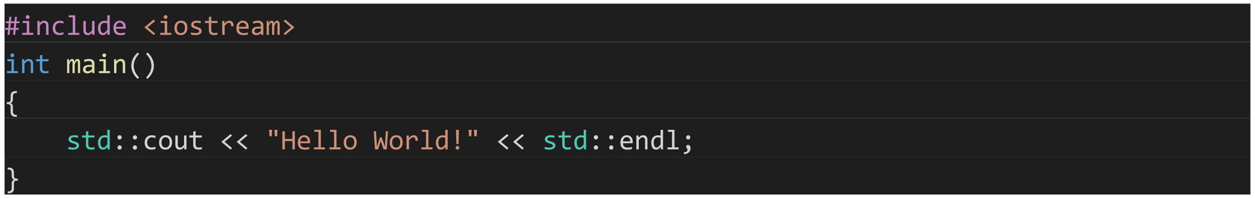
## Practical Session (1)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

## Practical Session (1) – Notes

- How does the code work?



```
#include <iostream>
int main()
{
    std::cout << "Hello World!" << std::endl;
}
```

- iostream: [http://www.cplusplus.com/reference/iostream/](http://www.cplusplus.com/reference/iostream/)
- main-function
  - The only function in C++ that does not require a return.
  - Can be defined with or without arguments.
  - A function body is always marked by curly brackets { }.
- std::cout
  - Object representing the *standard output stream* (in C: stdout).
- <<
  - Insertion operator. “Sends” right-hand side to the output stream on the left.
- std::endl
  - Manipulator adding a newline character to the stream and flushing it.

# Practical Session (1) – Notes

- Compilation with g++
  - g++
    Run the gnu C++ compiler.
  - -std=c++17
    Use the C++17 standard.
  - helloWorld.cpp
    Name of the cpp-file that should be compiled.
  - -o helloworld.out
    Set the output name to helloworld.out
- Other flags
  - -Wall
    Print all warnings
  - -O3
    Enable compiler optimizations
  - …

## Types

| Type | Meaning | Minimum Size |
|---|---|---|
| bool | boolean | |
| char | character | 8 bits |
| wchar_t | wide character | 16 bits |
| char16_t | Unicode character | 16 bits |
| char32_t | Unicode character | 32 bits |
| short | short integer | 16 bits |
| int | integer | 16 bits |
| long | long integer | 32 bits |
| long long | long integer | 64 bits |
| float double | singe-precision | 6 significant digits |
| double | double-precision | 10 significant digits |
| long double | extended-precision | 10 significant digits |

# Variables

- Signed and Unsigned Types

```
int a;
unsigned int b;
```

- Type Conversions

```
bool b = 42;            // b is true
int j = b;              // j has value 1
double pi = 3.14;       // pi has value 3.14
j = pi;                 // j has value 3
unsigned char c = -1;   // assuming 8-bit chars, c has value 255
i = c;  // the character with value 255 is an unprintable character
        // assigns value of c (i.e., 255) to an int
```

- Integer Literals

```
20      // decimal
024     // octal
0x14    // hexadecimal
```

- Floating-Point Literals

```
3.14159     0.313159e1   0.     0e0    .001
3.13159f                 0.f
```

## Variables

- List Initialization

```
int u = 0;
int u = {0};
int u{0};
int u(0);

double d = 3.14159
int a{d}, b = {d};  // error: narrowing conversion required
int c(d), e = (d);  // ok: but value will be truncated
```

- Conventions for Variable Names
  - Indication of meaning
  - Lowercase
  - Classes are uppercase
  - Use underscore or camelCase for variables with multiple words: `student_loan` or `studentLoan`
- Scoping
  - global, block

# Structs

- A container for multiple variables (and also functions)

```
struct myStruct {
    int a;
    int b;
    double c;
    void divide(){ c = (double)a / (double)b; }
};
```

- A struct is a class with default public members.
  (More about classes later)

# Flow of Control

- if

```
bool t = true;
if(t) {
    std::cout << "t is true" << std::endl;
}
else {
    std::cout << "t is false" << std::endl;
}
```

- while

```
int i = 0;
while (i < 10){
    std::cout << i << std::endl;
    i++;
}
```

- for

```
for(int i = 0; i < 10; ++i){
    std::cout << i << std::endl;
}
```

## Practical Session (2)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

## IDEs

- IDE = Integrated Development Environment
- Especially for C++ there are plenty
  - VIM
  - Emacs
  - Notepad++
  - Visual Studio (Windows only)
  - Visual Studio Code
  - Eclipse
  - Code::Blocks
  - Qt Creator
  - …
- Try different some but in the end choose one!
- If you work on Windows, I suggest Visual Studio.

## IDEs

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

- Know your shortcuts
- Know what your IDE can do (auto completion, split-screen, debugging, performance analysis, …)

## Debugging Tools

- Please, use debugging!
- In very short programs like the ones of today it might not be necessary but in the future it will be!
- It is easy to get a C++ program to compile. It is hard to get a C++ program working correctly!

## Practical Session (3)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

# Thank you for your Attention!