

C++ Programming II

Getting Started

C++ Programming II
September 16, 2018

Prof. Dr. P. Arnold
Bern University of Applied Sciences

► Getting Started

- Linux
- Windows
- Mac



Getting Started

Linux
Windows
Mac

First Program

CMake

► Getting Started

- Linux
- Windows
- Mac

► First Program

- CMake



Getting Started

Linux
Windows
Mac

First Program

CMake



Getting Started

Getting Started

Linux

Windows

Mac

First Program

CMake

Platform

Which platform to use?

C++ is platform independent, various IDE exists

- ▶ Windows Microsoft - Visual C/C++ , commercial
- ▶ MacOS X - XCode, free
- ▶ Unix - KDevelop, Eclipse, **QtCreator** etc., Open-Source, i.e. source code available
- ▶ Unix - GCC = Gnu Compiler Collection, free compiler

For newcomers, Linux (e.g Ubuntu) is the recommended development platform due to the free and well-engineered C++ 11 compiler.

Alternatively install Virtual Box, although not really convenient for software development!

In this course

K(Ubuntu) and **QT-Creator** are default.

Debian based Distributions

Prof. Dr. P. Arnold

Bern University
of Applied Sciences

The GCC (Gnu Compiler Collection including the gcc and g++ compilers) is usually already installed with Ubuntu (and Mac). To test, open the unix terminal and type “gcc –version”. On my Kubuntu machine this gives the following output:

gcc-version

```
1 $ gcc --version
2
3 gcc (Ubuntu 5.4.0-6ubuntu1~16.04.5) 5.4.0 20160609 Copyright (C)
   2015 Free Software Foundation, Inc. This is free software;
   see the source for copying conditions. There is NO
   warranty; not even for MERCHANTABILITY or FITNESS FOR A
   PARTICULAR PURPOSE.
```

Make sure your compiler version is at least **gcc 4.8** to enable c++11 features.

Getting Started

Linux

Windows

Mac

First Program

CMake

To install the build tools and the complete Qt-Creator/qt5 toolchain with examples and documentation simply run:

Install Qt Creator IDE and tools

```
1 % Build tools
2 sudo apt-get install build-essential gdb cmake cmake-curses-gui
3
4 % Install Qt with examples and openGL support for widgets
5 sudo apt-get install qtcreator qt5-default qttools5-dev-tools
   qt5-doc qtbase5-examples qtbase5-doc-html libgl1-mesa-dev
```

Getting Started

Linux

Windows

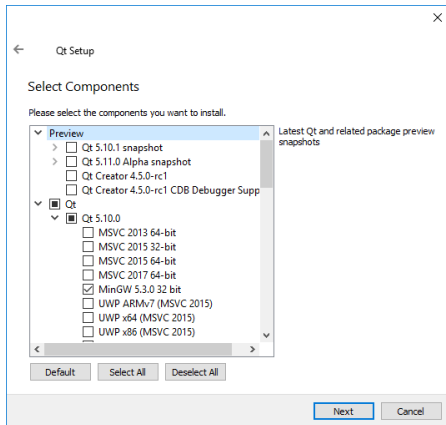
Mac

First Program

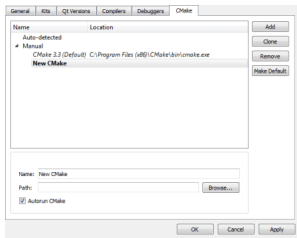
CMake

The installation on Windows with MinGW-Compiler is straight-forward following these instructions:

1. Get the open source version of Qt from:
<https://www.qt.io/download>
2. Follow the instructions of the installer. Skip the account creation
3. Select **5.3.0 32-bit** in the Qt 5.10.0 sub-folder for installation



- ▶ CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice
1. Get CMake from: <https://cmake.org/>
 2. For best experience with Qt-Creator get version 3.7.2:
<https://cmake.org/files/v3.7/cmake-3.7.2-win32-x86.msi>
<https://cmake.org/files/v3.7/cmake-3.7.2-win64-x64.msi>
 3. Start Qt Creator and set up cmake according to the Qt documentation:
<http://doc.qt.io/qtcreator/creator-project-cmake.html>
 4. CMake should get detected automatically by Qt Creator

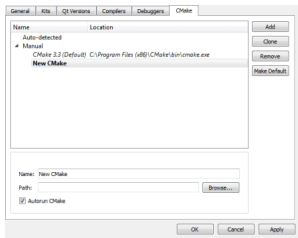


For MacOS X the C++ -Compiler is part of XCode.

1. Install XCode from Apples App Store
2. Get the open source version of Qt from:
`https://www.qt.io/download`
3. Follow the instructions of the installer. Skip the account creation

- ▶ CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice

1. Get CMake from: <https://cmake.org/>
2. For best experience with Qt-Creator get version 3.7.2:
https://cmake.org/files/v3.7/cmake-3.7.2-Darwin-x86_64.dmg
3. Set up cmake according to the official Qt documentation:
<http://doc.qt.io/qtcreator/creator-project-cmake.html>





First Program

Getting Started

Linux

Windows

Mac

First Program

CMake

First Program

Hello World

- ▶ Get QT-Creator ([Homework01.pdf](#))
- ▶ Compile and run the helloworld example in a console
- ▶ Compile with: `g++ helloworld.cpp -o helloworld`
- ▶ In a console run with: `./helloworld`

First Program

Hello World

- ▶ Get QT-Creator ([Homework01.pdf](#))
- ▶ Compile and run the helloworld example in a console
- ▶ Compile with: `g++ helloworld.cpp -o helloworld`
- ▶ In a console run with: `./helloworld`

```
1 #include <iostream>
2 #include <vector>
3 #include <random>
4 #include <algorithm>
5 #include <stack>
6
7
8 static bool odd(int n) { return n % 2; }
9
10
11 int main()
12 {
13     std::priority_queue
14         std::cout << "Hello World" << std::endl;
15     return 0;
16 }
```

```
1 g++ helloworld.cpp -o helloworld
2 ./helloworld
3
4 Hello World!
```

First Program

Hello World - Analysis

```
1 // Pre-processor directive
2 #include <iostream>
3
4 // Start of your program
5 int main()
6 {
7     /* Write to the screen using std::cout */
8     std::cout << "Hello World" << std::endl;
9
10    // Return a value to the OS
11    return 0;
12 }
```



First Program

Hello World - Analysis

```
1 // Pre-processor directive
2 #include <iostream>
3
4 // Start of your program
5 int main()
6 {
7     /* Write to the screen using std::cout */
8     std::cout << "Hello World" << std::endl;
9
10    // Return a value to the OS
11    return 0;
12 }
```

- ▶ The preprocessor directive `#include` command occurs before the actual compilation starts. It tells the preprocessor to include the content of the specified file at the current line. In this example, `iostream` lets us use the `std::cout` and `std::endl` functions to write on the screen.
- ▶ The `int main()` is the body of your Program. The execution of a C++ program always starts here.
- ▶ The `{ }` indicate that everything inside them is part of the function. In this case, they denote that everything inside is a part of the “main” function.

First Program

Hello World - Analysis

```
1 // Pre-processor directive
2 #include <iostream>
3
4 // Start of your program
5 int main()
6 {
7     /* Write to the screen using std::cout */
8     std::cout << "Hello World" << std::endl;
9
10    // Return a value to the OS
11    return 0;
12 }
```

- ▶ The “;” denotes the end of a line. Most lines of C++ code need to end with a semicolon.
- ▶ `cout` (console-out) writes the “Hello World” to the screen. `cout` is a *stream* defined in the standard `std` and therefore `std::cout`. The stream insertion parameter `<<` puts the text in the stream and `std::endl` ends a line.
- ▶ `main()` is a function and always returns an integer: 0 for success and -1 in the event of an error. Other error codes using the available range of integers can be used.

First Program

Hello World - Analysis

```
1 // Pre-processor directive
2 #include <iostream>
3
4 // Start of your program
5 int main()
6 {
7     /* Write to the screen using std::cout */
8     std::cout << "Hello World" << std::endl;
9
10    // Return a value to the OS
11    return 0;
12 }
```

- ▶ C++ supports two styles of comments
 - ▶ `//` indicates the start of a comment until the end of the line
 - ▶ `/* */` indicates that the contained text is a comment
- ▶ Use `using namespace std` in order to use `cout` instead of `std::cout`



```
1 # Name of project and executable
2 project(HelloWorld)
3
4 # set cmake version
5 cmake_minimum_required(VERSION 2.8)
6
7 # activate latest c++ compiler version
8 set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++17")
9
10 # set flags to configure the warning settings
11 # Note: warnings are compiler specific
12 if( CMAKE_COMPILER_IS_GNUCC )
13     set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Wextra")
14 endif()
15
16 # set build type to Debug/Release
17 set(CMAKE_BUILD_TYPE "Debug")
18
19 # Add an executable to the project and sources
20 add_executable(${PROJECT_NAME} "helloworld.cpp")
```

- ▶ Comments are set with #: line 1, 4, 7...
- ▶ Compiler Flags: line 8, 13
- ▶ Check compiler: line 12
- ▶ Set build type: line 17

Thank You

Questions

???



Getting Started

Linux

Windows

Mac

First Program

CMake