



COMSATS University
Islamabad
(Lahore Campus)

CSC103-Programming Fundamentals

MS. MAHWISH WAQAS

MAWISH.WAQAS@CUILAHORE.EDU.PK

Introduction to C++

Course Objectives

To develop the skills to analyze, design, test and translate **problems** into computer programs;

- Discuss and apply different problem solving techniques such as top-down design, algorithm refinement etc.

To present the fundamental programming concepts such as the 3 types of statements, functions, arrays etc.

To demonstrate basic coding, testing and debugging techniques;

To provide an implementation of the concepts.

Outline

Very brief history of C++

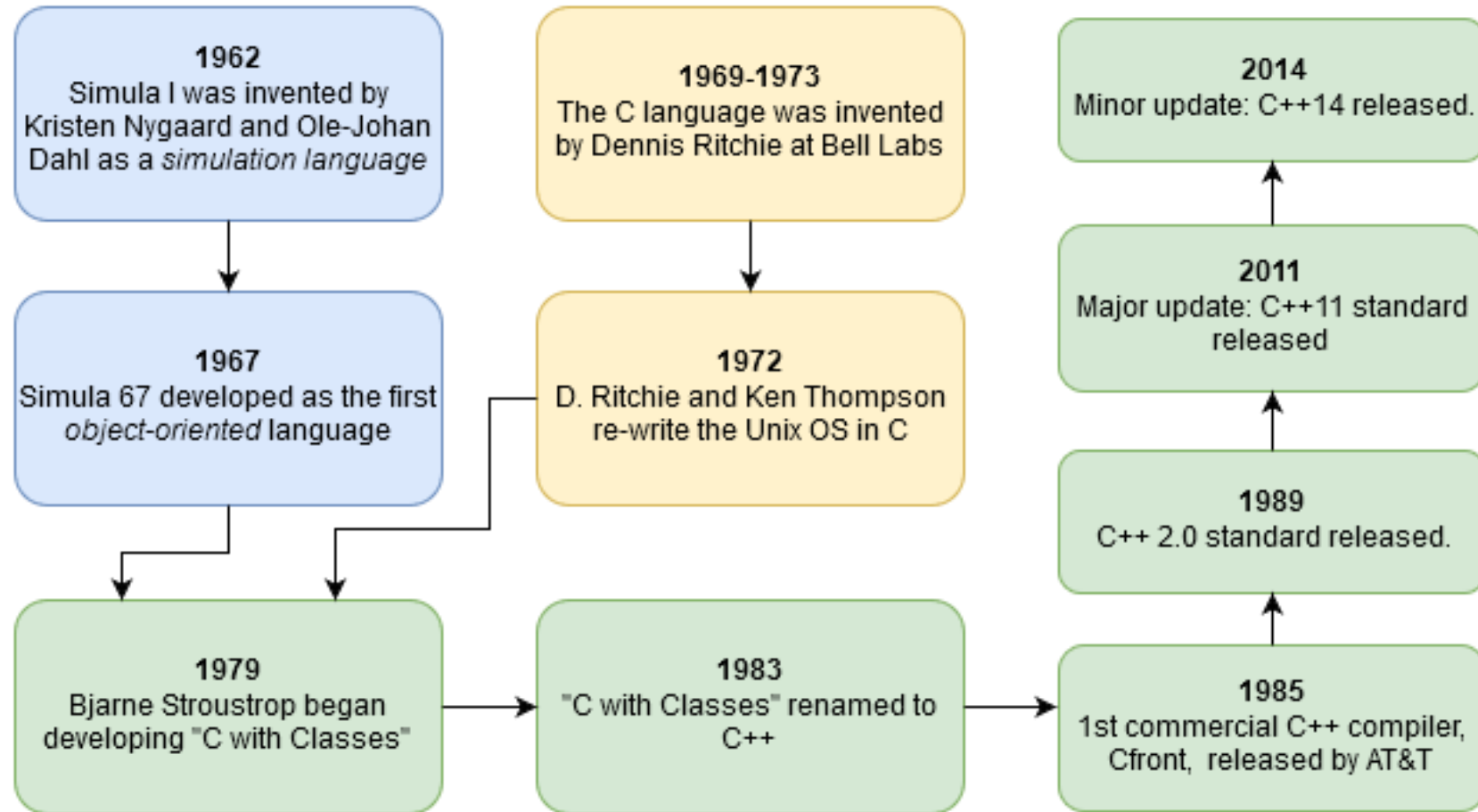
When C++ is a good choice

The Code::Blocks IDE

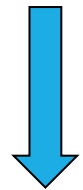
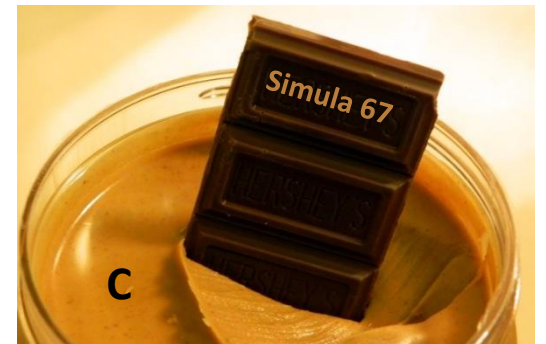
First program!

Some C++ syntax

Very brief history of C++



For details more check out [A History of C++: 1979–1991](#)



C++

Characteristics of C++

C++ is...

- Compiled.
 - A separate program, the compiler, is used to turn C++ source code into a form directly executed by the CPU.
- Strongly typed and unsafe
 - Conversions between variable types must be made by the programmer (strong typing) but can be circumvented when needed (unsafe)
- C compatible
 - call C libraries directly and C code is nearly 100% valid C++ code.
- Capable of very high performance
 - The programmer has a very large amount of control over the program execution
- Object oriented
 - With support for many programming styles (procedural, functional, etc.)
- No automatic memory management
 - The programmer is in control of memory usage

When to choose C++

“If you’re not at all interested in performance, shouldn’t you be in the Python room down the hall?”
— Scott Meyers (author of [Effective Modern C++](#))

Despite its many competitors C++ has remained **popular for ~30 years** and will continue to be so in the foreseeable future.

Why?

- Complex problems and programs can be effectively implemented
- Object Oriented Programming capability
- No other language quite matches C++’s combination of **performance, expressiveness**, and ability to **handle complex programs**.

Choose C++ when:

- **Program performance matters**
 - Dealing with large amounts of data, multiple CPUs, complex algorithms, etc.
- **Programmer productivity is less important**
 - It is faster to produce working code in Python, R, Matlab or other scripting languages!
- **Access to libraries**
 - Ex. Nvidia’s CUDA Thrust library for GPUs
- **Your organization uses it already!**

Code::Blocks

In this course, we will use the Code::Blocks integrated development environment (IDE) for writing and compiling C++

About C::B

- cross-platform: supported on Mac OSX, Linux, and Windows
- Short learning curve compared with other IDEs such as Eclipse or Visual Studio

Supports advanced features such as automated code building system (e.g. cmake).

- You may need this in future.

Project homepage: <http://www.codeblocks.org>

Download page:

<https://sourceforge.net/projects/codeblocks/files/Binaries/20.03/Windows/codeblocks-20.03mingw-setup.exe/download>

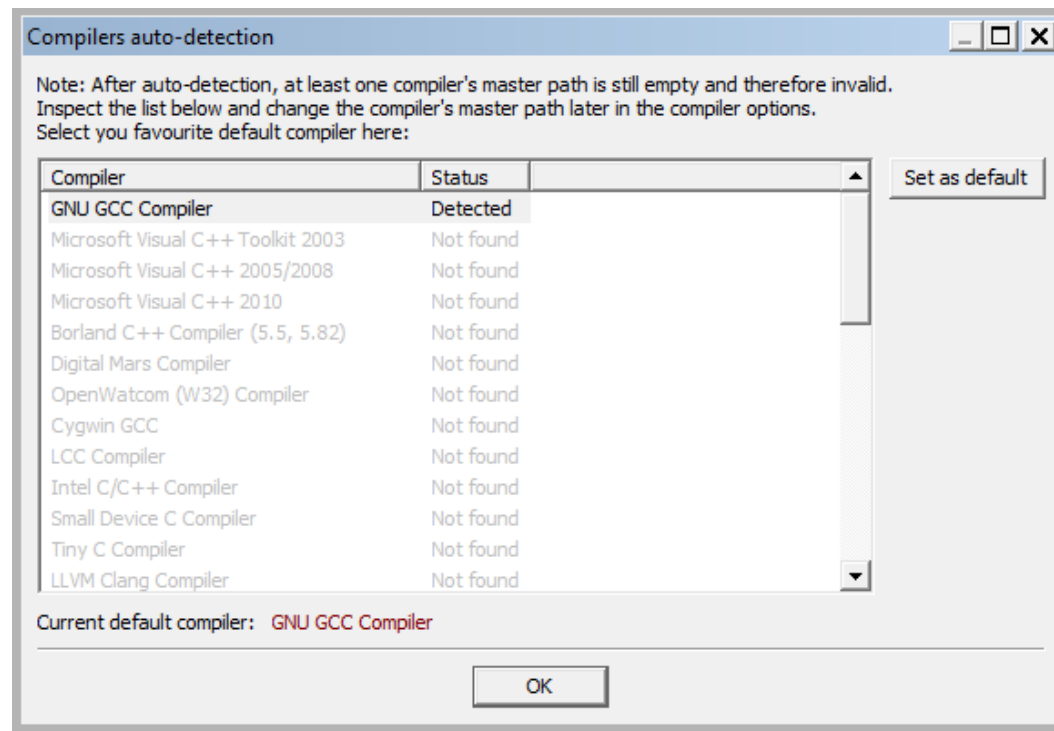
IDE Advantages

- Handles build process for you
- Syntax highlighting and live error detection
- Code completion (fills in as you type)
- Creation of files via templates
- Built-in debugging
- Code refactoring (ex. Change a variable name everywhere in your code)
- Higher productivity

Opening C::B

The 1st time it is opened C::B will search for compilers it can use.

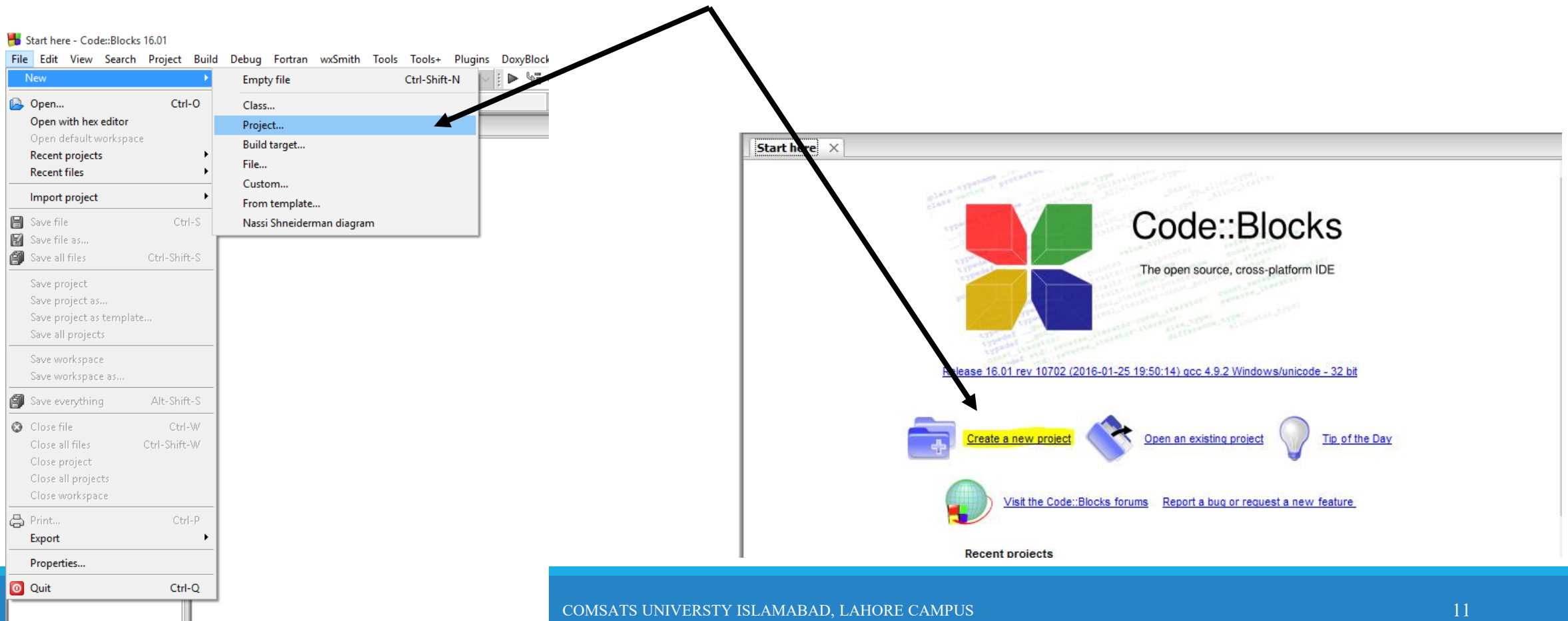
A dialog that looks like this will open. Select GCC if there are multiple options:



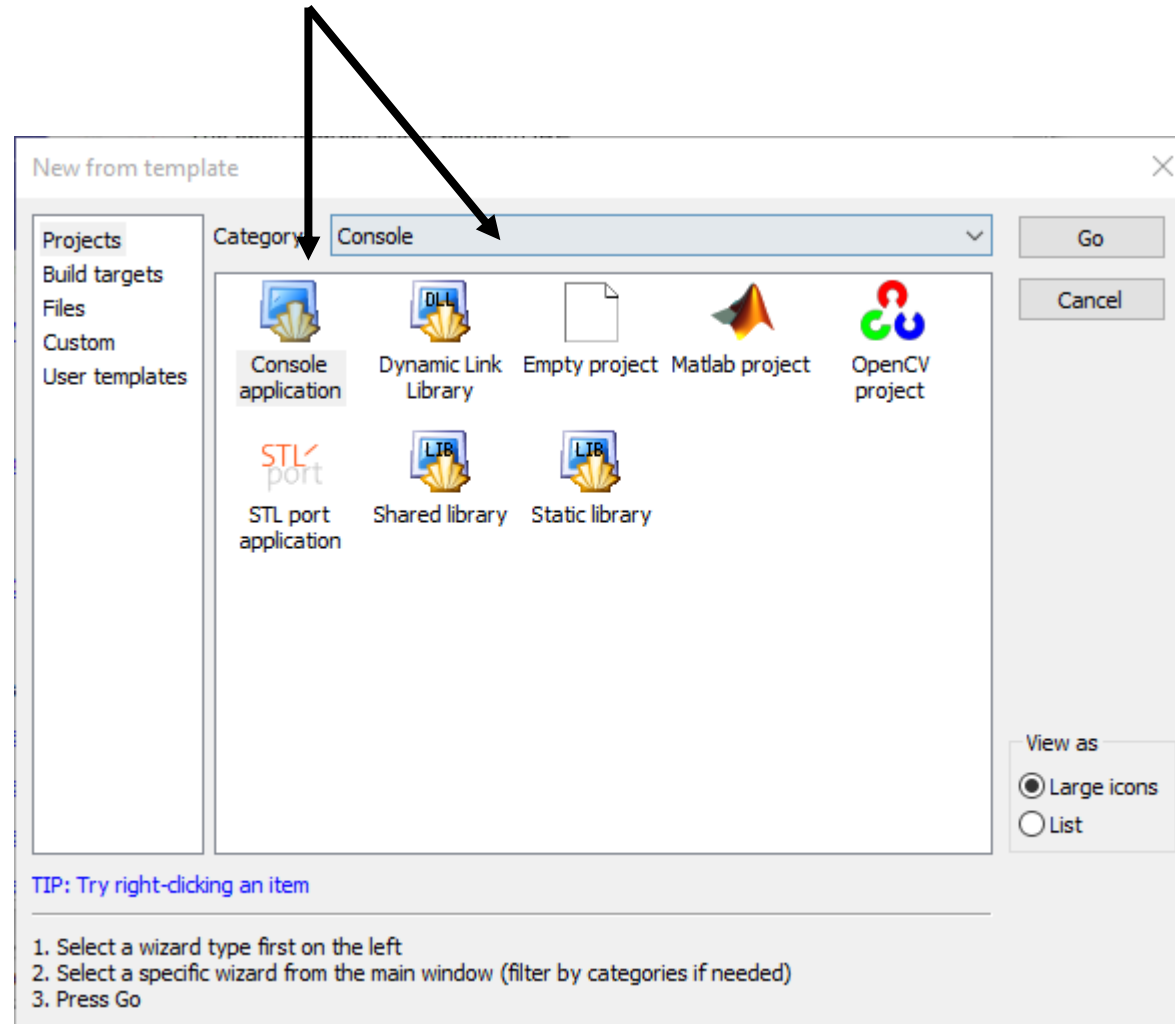
And click OK.

Opening C::B and creating a 1st C++ project...

Step 1. Create a project from the File menu or the Start Here tab:



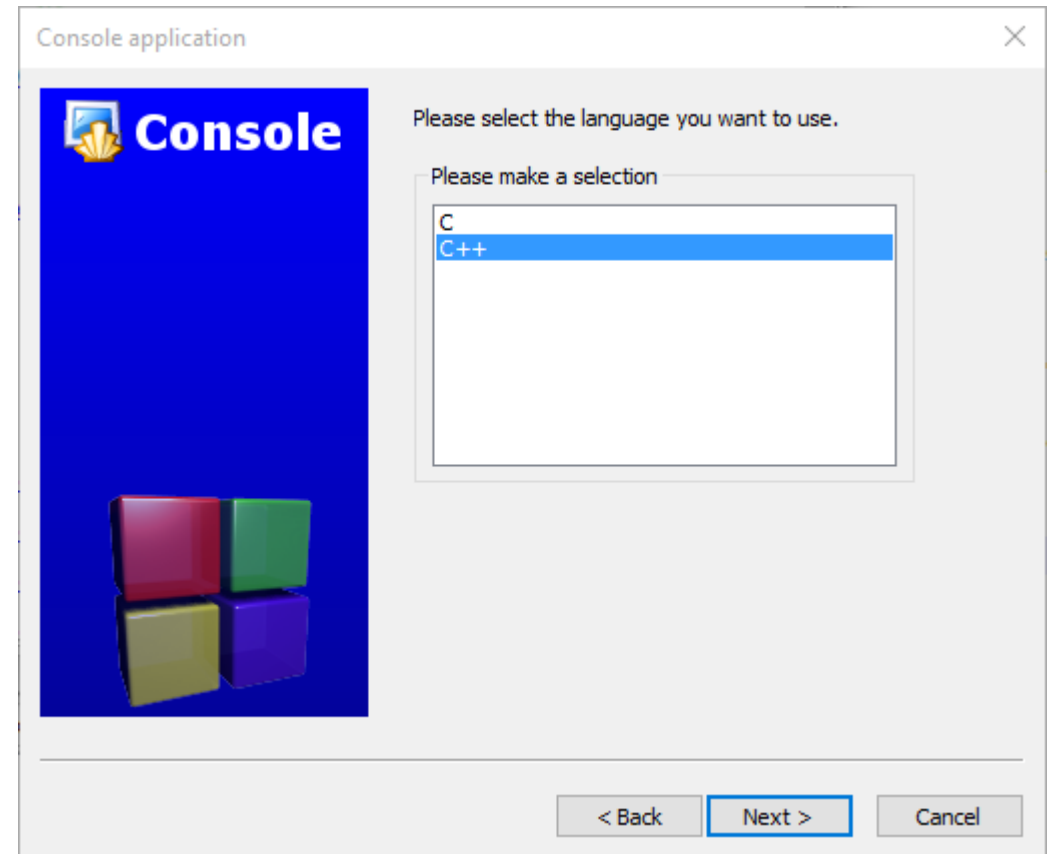
Step 2. Choose the Console category and then the Console application and click Go.



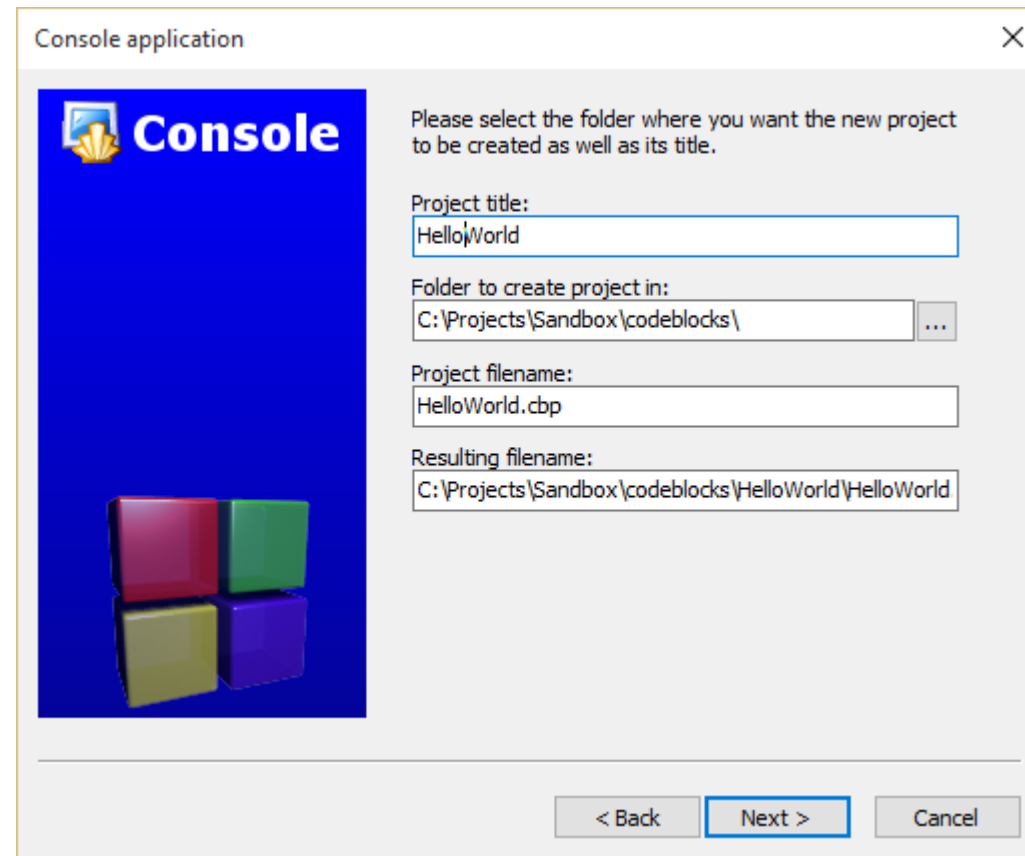
Step 3: Click Next on the “Welcome to the new console application wizard!” screen.

Step 4: Choose C++!

...then click Next.



Step 5. Enter a project title. Let C::B fill in the other fields for you. If you like you can change the default folder to hold the project. Click Next.



Console application

Please select the folder where you want the new project to be created as well as its title.

Project title:
HelloWorld

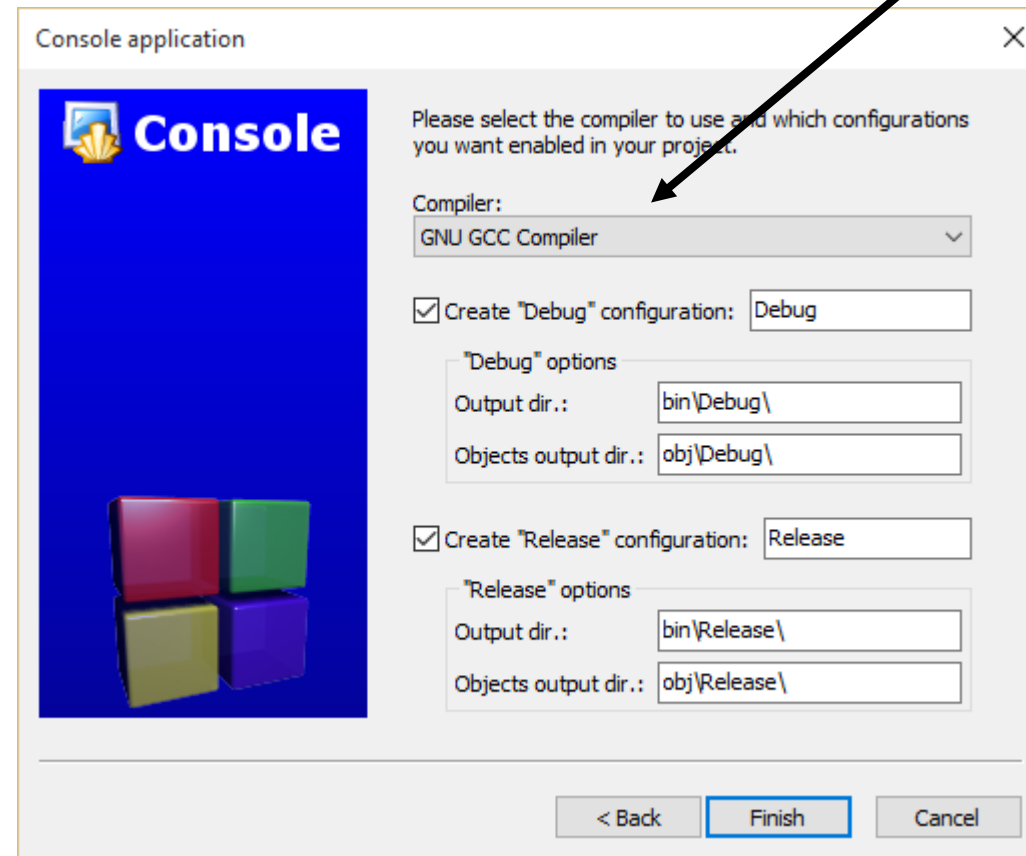
Folder to create project in:
C:\Projects\Sandbox\codeblocks\ ...

Project filename:
HelloWorld.cbp

Resulting filename:
C:\Projects\Sandbox\codeblocks\HelloWorld\HelloWorld

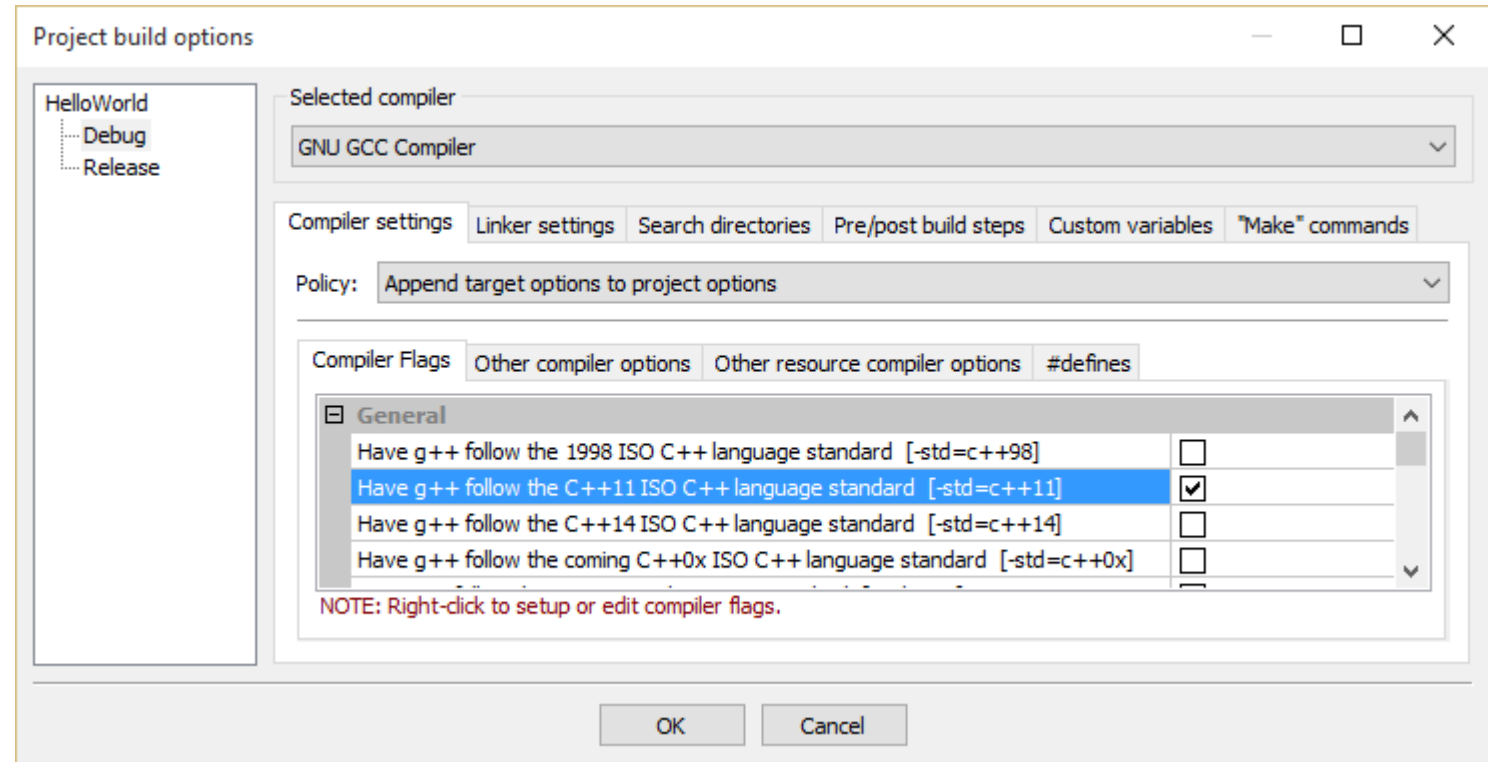
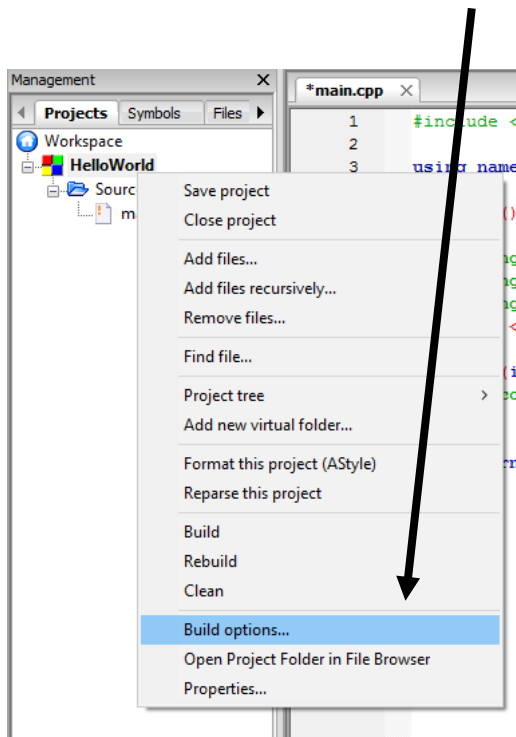
< Back Next > Cancel

Step 6: Choose the compiler. For this tutorial, choose GNU GCC as the compiler. Click Next.



Enable C++11 standard

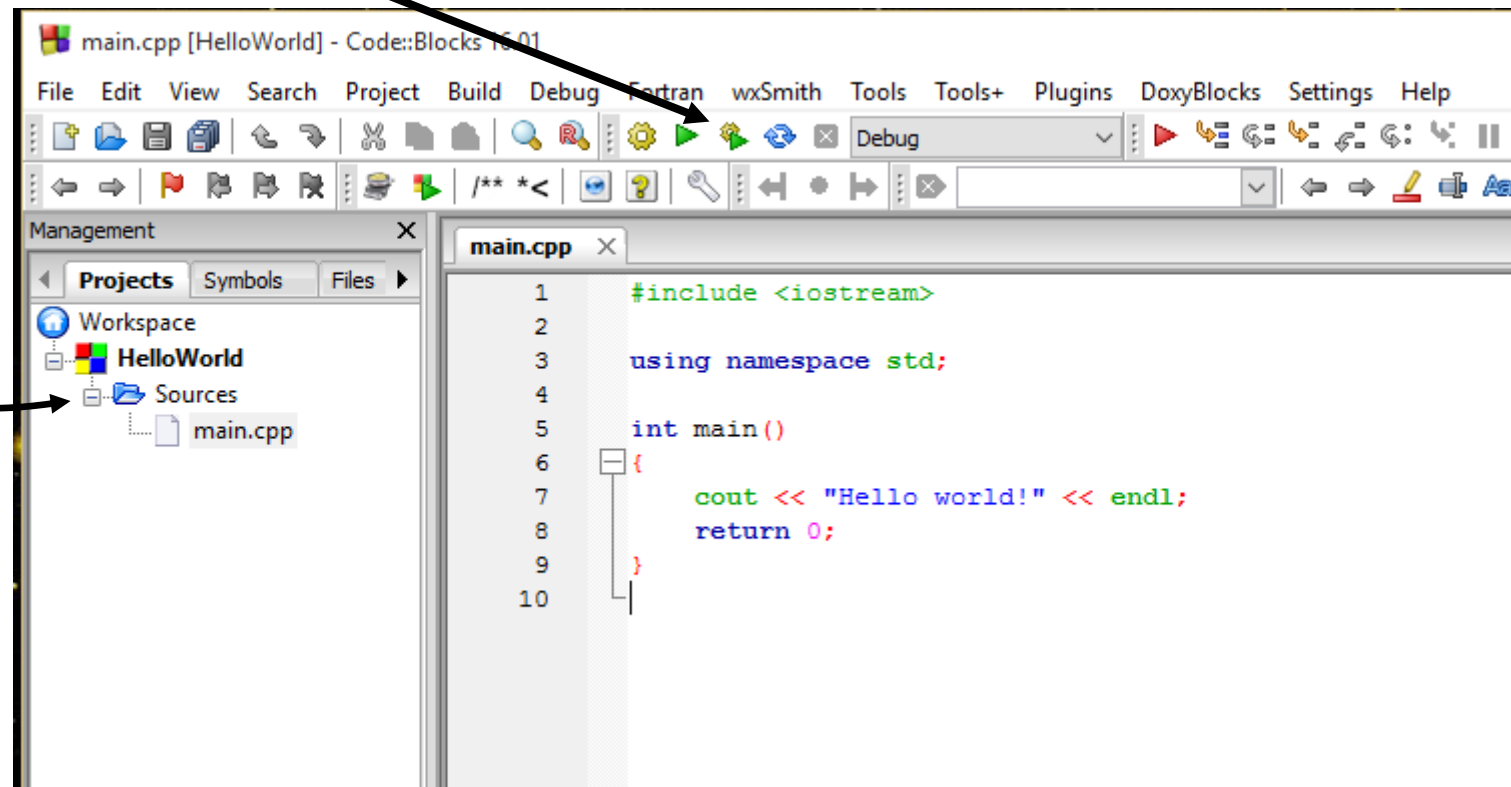
Step 7.1 Right-click on your project name and choose Build options



- Check off the C++11 option. Click *Release* on the left and do the same there as well.
- Do this anytime we create a project in C++

Step 8: Your project is now created! Click on Sources in the left column, then double-click *main.cpp*.

Click the  icon in the toolbar or press F9 to compile and run the program.

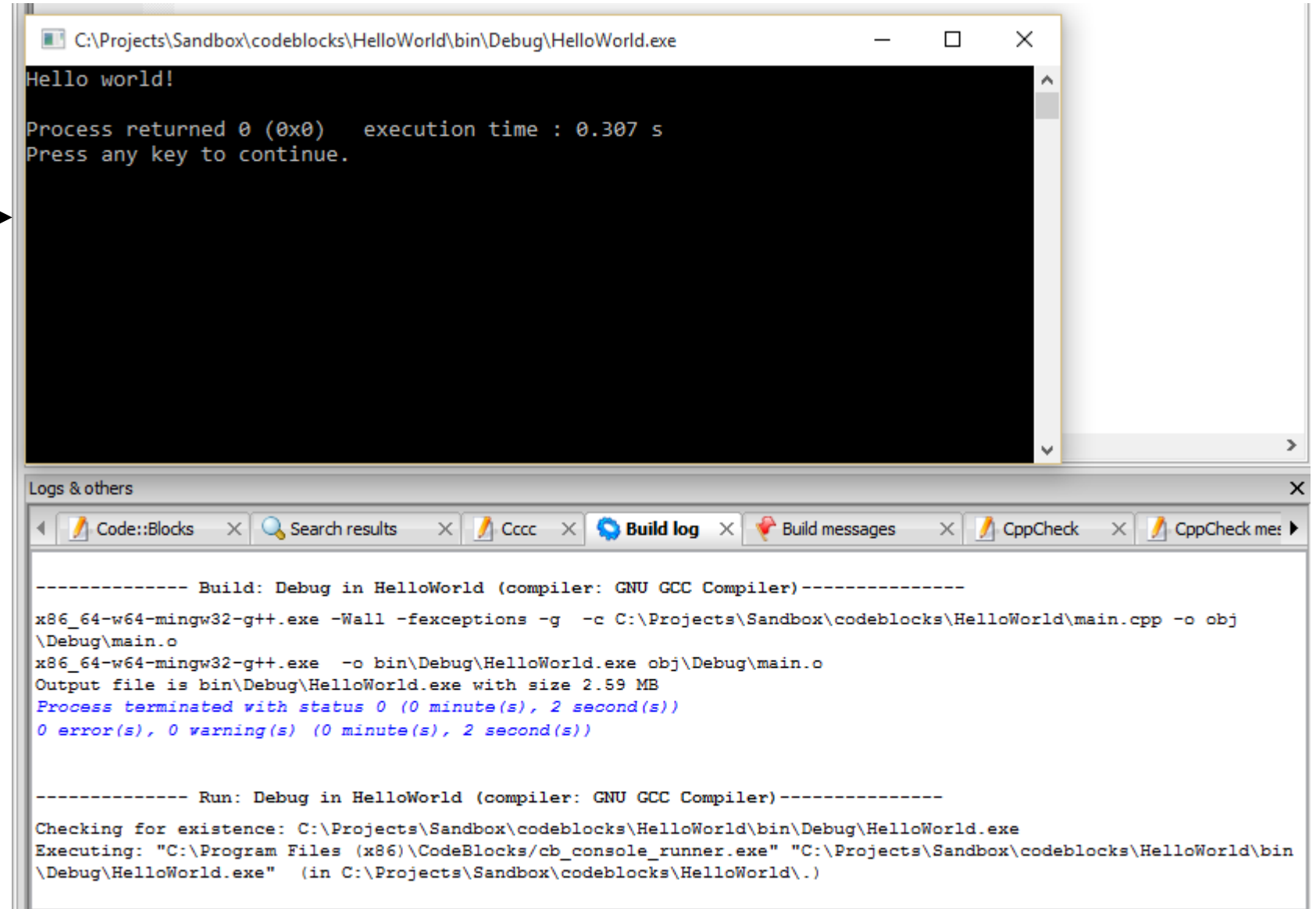


Hello, World!

Console window:



Build and compile messages



The screenshot shows the Code::Blocks IDE interface. The top window is the console, titled "C:\Projects\Sandbox\codeblocks\HelloWorld\bin\Debug\HelloWorld.exe". It displays the output of a C++ program: "Hello world!", followed by "Process returned 0 (0x0) execution time : 0.307 s" and "Press any key to continue.". The bottom window is the "Logs & others" panel, which contains several tabs. The "Build log" tab is active, showing the build process for the "HelloWorld" project using the GNU GCC Compiler. The log includes the compiler command, the output file path, and the execution status, indicating a successful build with no errors or warnings. The "Run" section shows the execution of the program, confirming its existence and successful execution.

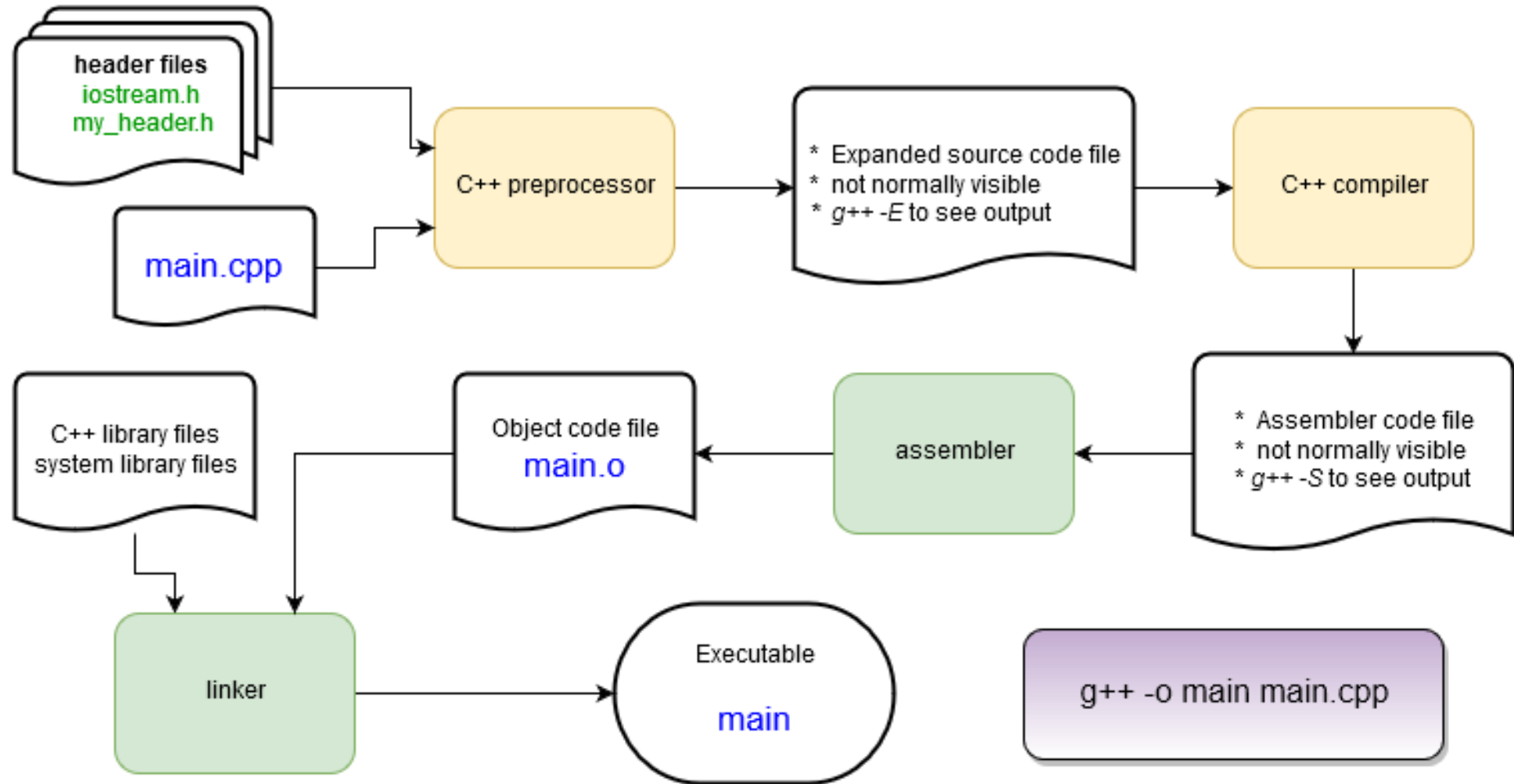
```
C:\Projects\Sandbox\codeblocks\HelloWorld\bin\Debug\HelloWorld.exe
Hello world!
Process returned 0 (0x0) execution time : 0.307 s
Press any key to continue.

Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck CppCheck me

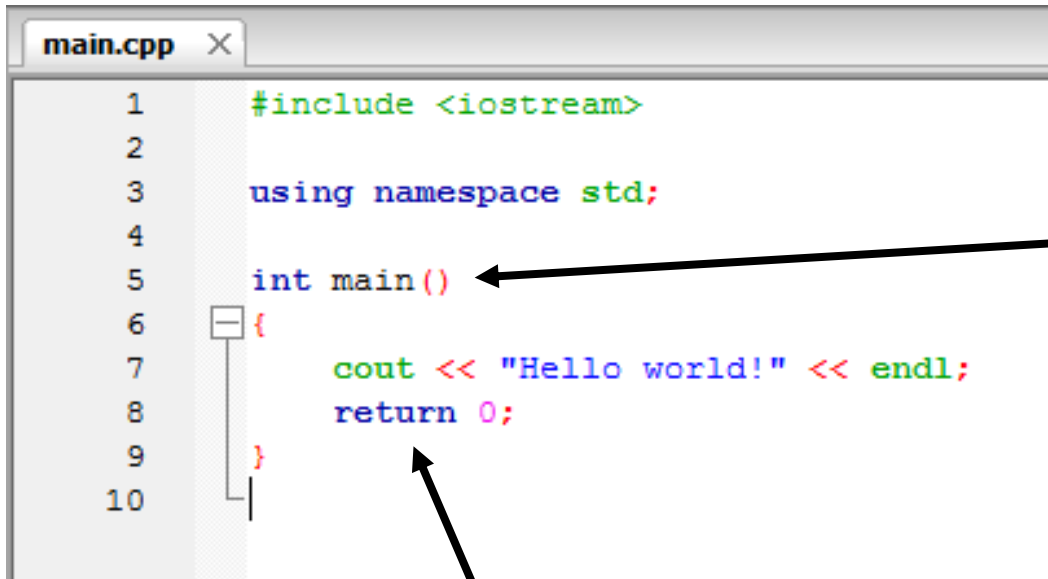
----- Build: Debug in HelloWorld (compiler: GNU GCC Compiler)-----
x86_64-w64-mingw32-g++.exe -Wall -fexceptions -g -c C:\Projects\Sandbox\codeblocks\HelloWorld\main.cpp -o obj\Debug\main.o
x86_64-w64-mingw32-g++.exe -o bin\Debug\HelloWorld.exe obj\Debug\main.o
Output file is bin\Debug\HelloWorld.exe with size 2.59 MB
Process terminated with status 0 (0 minute(s), 2 second(s))
0 error(s), 0 warning(s) (0 minute(s), 2 second(s))

----- Run: Debug in HelloWorld (compiler: GNU GCC Compiler)-----
Checking for existence: C:\Projects\Sandbox\codeblocks\HelloWorld\bin\Debug\HelloWorld.exe
Executing: "C:\Program Files (x86)\CodeBlocks\cb_console_runner.exe" "C:\Projects\Sandbox\codeblocks\HelloWorld\bin\Debug\HelloWorld.exe" (in C:\Projects\Sandbox\codeblocks\HelloWorld\.)
```

Behind the Scenes: The Compilation Process



Hello, World! Explained (boilerplate code: the code which **MUST** be written in any program in a particular programming language)



```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
10
```

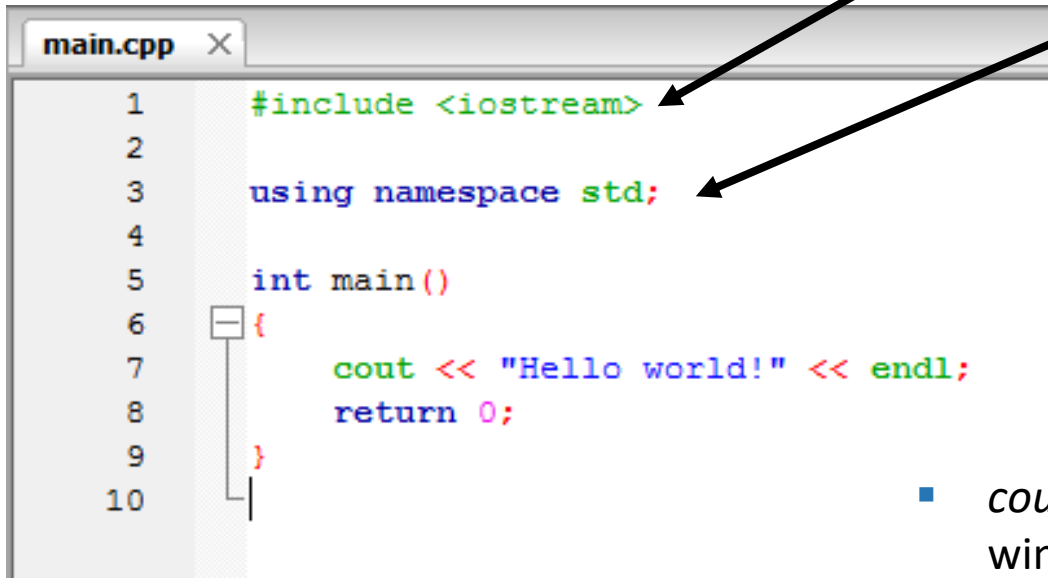
The *main* function – the start of **every** C++ program! It returns an integer value to the operating system and (in this case) takes no arguments: `main()`

The **return** statement returns an integer value to the operating system after completion. 0 means “no error”. C++ programs **must** return an integer value.

Hello, World! explained

loads a *header* file containing function and class definitions

Loads a *namespace* called *std*. Namespaces are used to separate sections of code for programmer convenience. To save typing we'll always use this line in this tutorial.



```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
10
```

- *cout* is the *object* that writes to the stdout device, i.e. the console window.
- Without the “using namespace std;” line this would have been called as *std::cout*. It is defined in the *iostream* header file.
- << is the C++ *insertion operator*.
 - It is used to pass characters from the right to the object on the left.
- *endl* is the C++ newline character.

Header Files

C++ (along with C) uses *header files* as to hold definitions for the compiler to use while compiling.

- `<iostream>` provides definitions for I/O functions, including the *cout* function.
- C header files end with `'h'` suffix, while C++ language headers do not end with the `'h'`.

A source file (`file.cpp`) contains the code that is compiled into an object file (`file.o`).

The header (`file.h`) is used to tell the compiler what to expect when it assembles the program in the linking stage from the object files.

Source files and header files can refer to any number of other header files.