EE 441 Supplementary Notes

Project Creation, Code Writing and Formatting

Contents

1. What is Code::Blocks? 1
2. Installation 1
   2.1. Installation in Windows 1
3. Project Creation 4
4. Example Project 8
5. How to Write a Clear Source Code? 8
   5.1. Formatting In Your Code 9
   5.2. Commenting 9
6. Homework Format 9
7. Differences between C and C++ Syntax 10
   7.1. Key Difference 10
   7.2. Data Types 11
   7.3. Overloading 11
   7.4. Struct vs Class 11
   7.5. Reference Variable 11
   7.6. Time of Declaring Functions 11
   7.7. Time of Defining Variables 11
   7.8. Functions for Standard Input and Output 12
   7.9. Memory Allocation and Deallocation 12

1. What is Code::Blocks?

Code::Blocks is an open-source, cross-platform Integrated Development Environment (IDE). An IDE is a software application that generally consists of a source code editor, software building tools (compiler, linker, and build automation tools), and a debugger. This creates an easy and ready-to-use environment for software development.

Code::Blocks will be used for the assignments because it is open-source and cross-platform (Windows, Linux, and Mac OS X). For the same reasons, GCC (GNU Compiler Collection) will be used as the compiler.

2. Installation

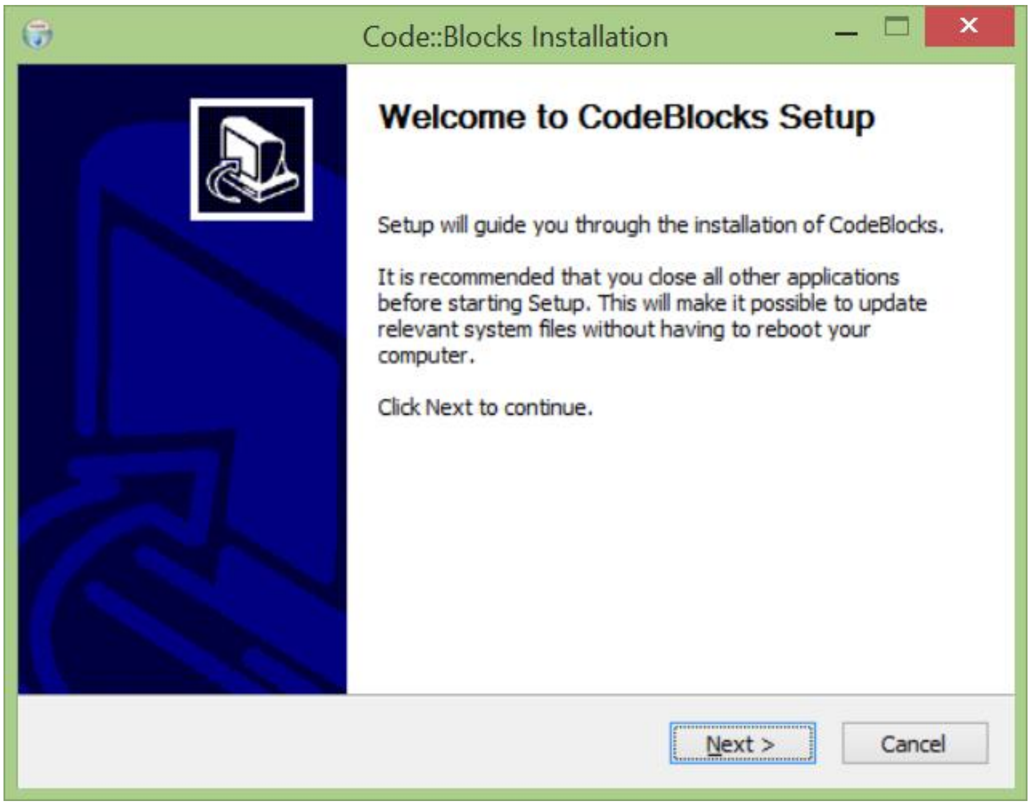
First of all, you need to have the installation file. To download the latest official release, visit this page. Select a setup depending on your OS.

IMPORTANT FOR WINDOWS USERS: Download the file ending with "mingw-setup.exe". You can download directly with this link.

For more information about installation, you can visit here.

2.1. Installation in Windows

1. Run the setup file (codeblocks-17.12mingw-setup.exe).
2. Click "Next". (Fig. 1)



Code::Blocks Installation

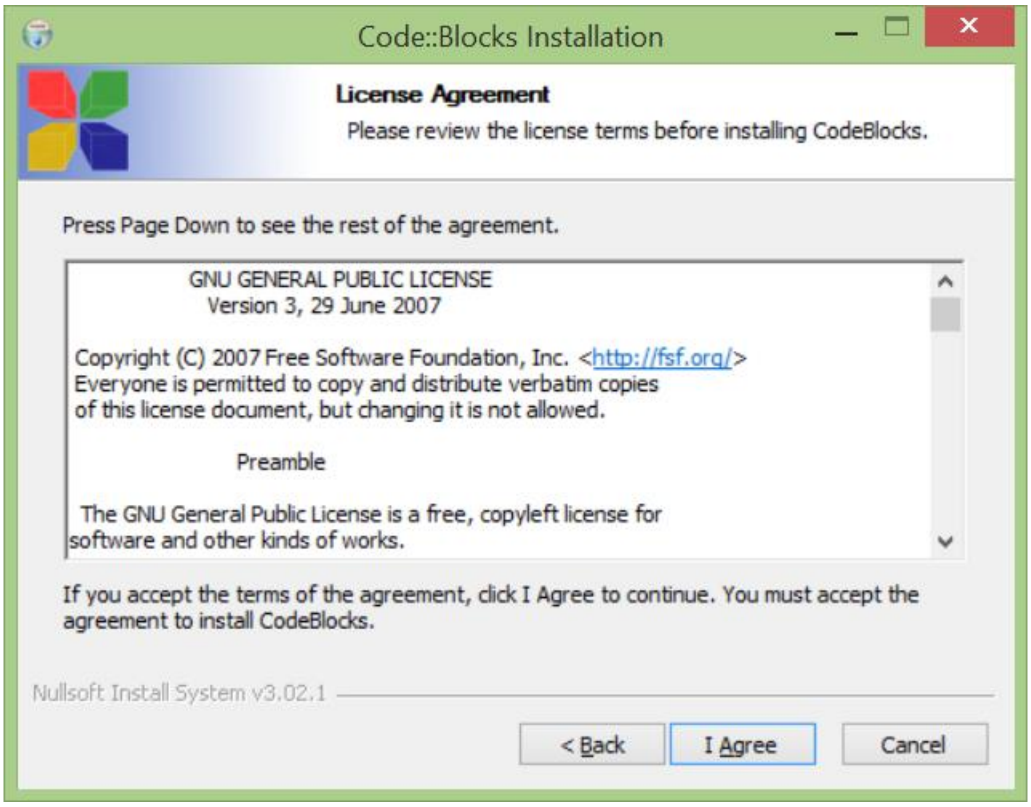
Welcome to CodeBlocks Setup

Setup will guide you through the installation of CodeBlocks.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

3. Click "I Agree". (Fig. 2)



License Agreement

Please review the license terms before installing CodeBlocks.

Press Page Down to see the rest of the agreement.

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

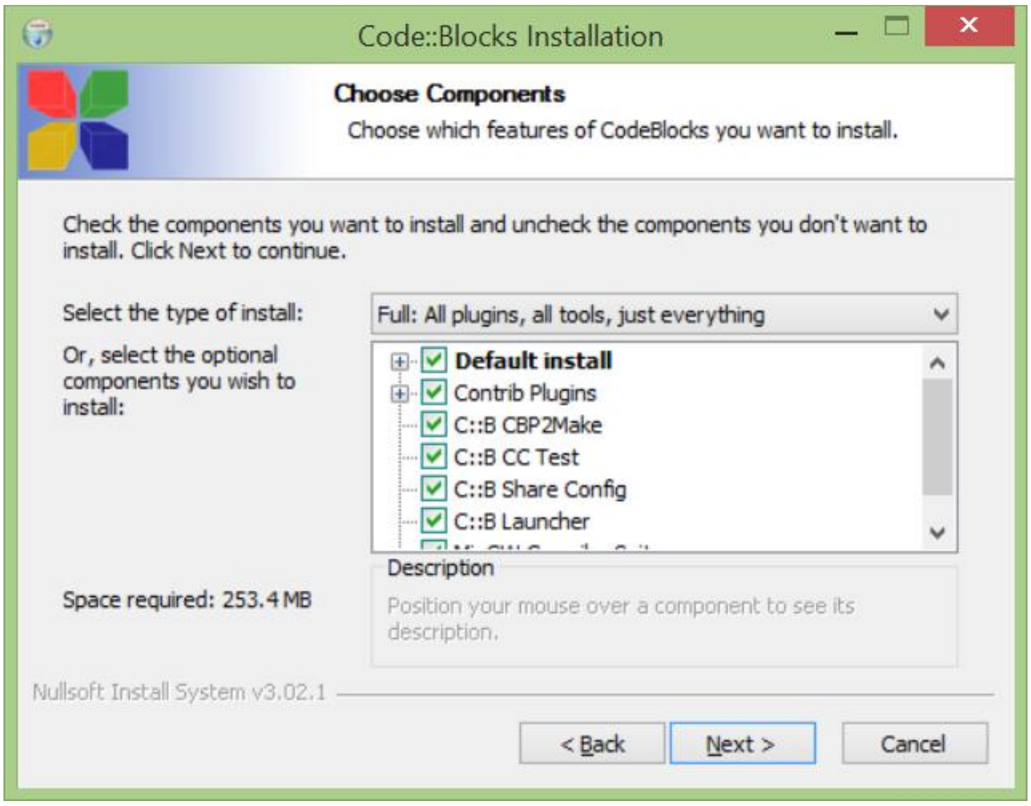
Preamble

The GNU General Public License is a free, copyleft license for
software and other kinds of works.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install CodeBlocks.

Nullsoft Install System v3.02.1

4. Choose "Full: All plugins, all tools, just everything". Click "Next". (Fig. 3)



Choose Components

Choose which features of CodeBlocks you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Full: All plugins, all tools, just everything

Or, select the optional components you wish to install:

- ✓ **Default install**
- ✓ Contrib Plugins
  - ✓ C::B CBP2Make
  - ✓ C::B CC Test
  - ✓ C::B Share Config
  - ✓ C::B Launcher

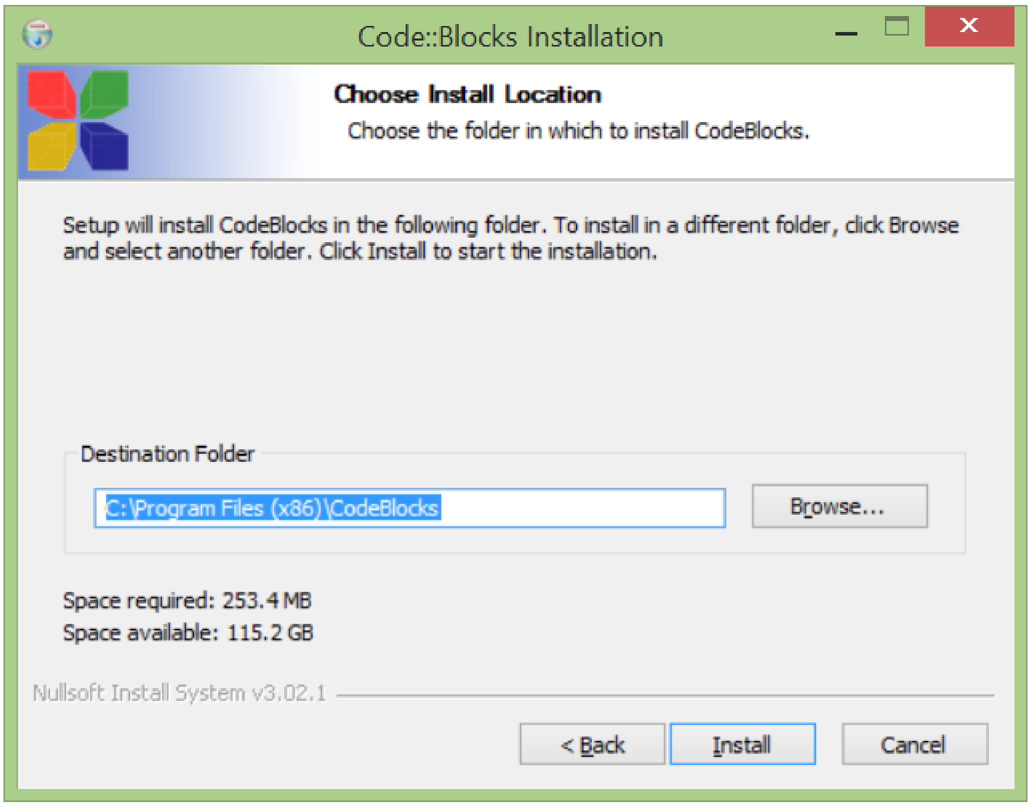
Space required: 253.4 MB

Description

Position your mouse over a component to see its description.

Nullsoft Install System v3.02.1

5. If you want, change the destination folder. Click "Install". (Fig. 4)



Code::Blocks Installation

Choose Install Location

Choose the folder in which to install CodeBlocks.

Setup will install CodeBlocks in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

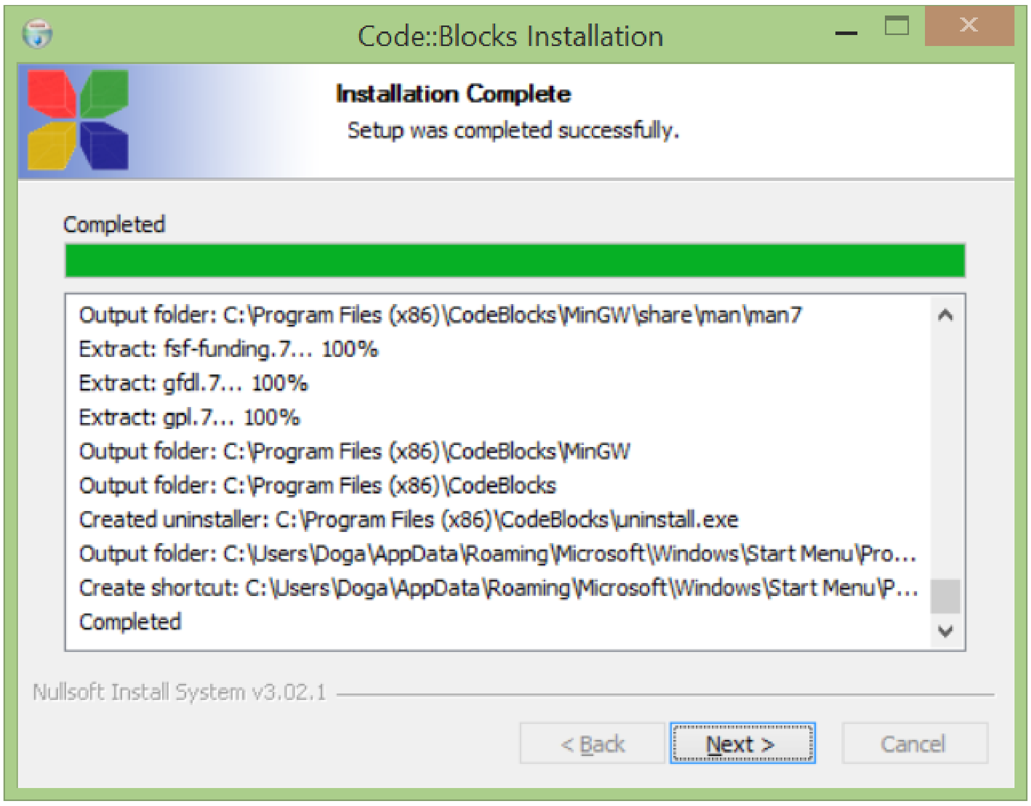
Destination Folder

C:\Program Files (x86)\CodeBlocks

Space required: 253.4 MB
Space available: 115.2 GB

Nullsoft Install System v3.02.1

6. Click "Next. (Fig. 5)



Code::Blocks Installation

Installation Complete

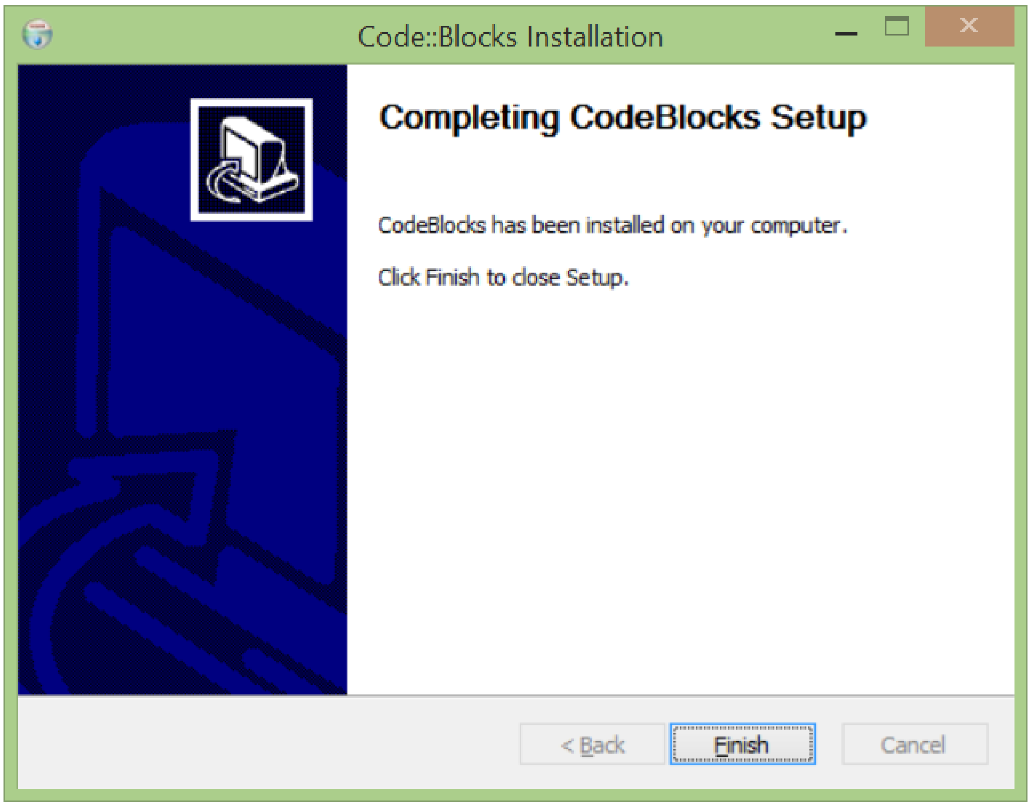
Setup was completed successfully.

Completed

```
Output folder: C:\Program Files (x86)\CodeBlocks\MinGW\share\man\man7
Extract: fsf-funding.7... 100%
Extract: gfdl.7... 100%
Extract: gpl.7... 100%
Output folder: C:\Program Files (x86)\CodeBlocks\MinGW
Output folder: C:\Program Files (x86)\CodeBlocks
Created uninstaller: C:\Program Files (x86)\CodeBlocks\uninstall.exe
Output folder: C:\Users\Doga\AppData\Roaming\Microsoft\Windows\Start Menu\Pro...
Create shortcut: C:\Users\Doga\AppData\Roaming\Microsoft\Windows\Start Menu\P...
Completed
```

Nullsoft Install System v3.02.1

7. Click “Finish”. (Fig. 6)


Code::Blocks Installation

Completing CodeBlocks Setup

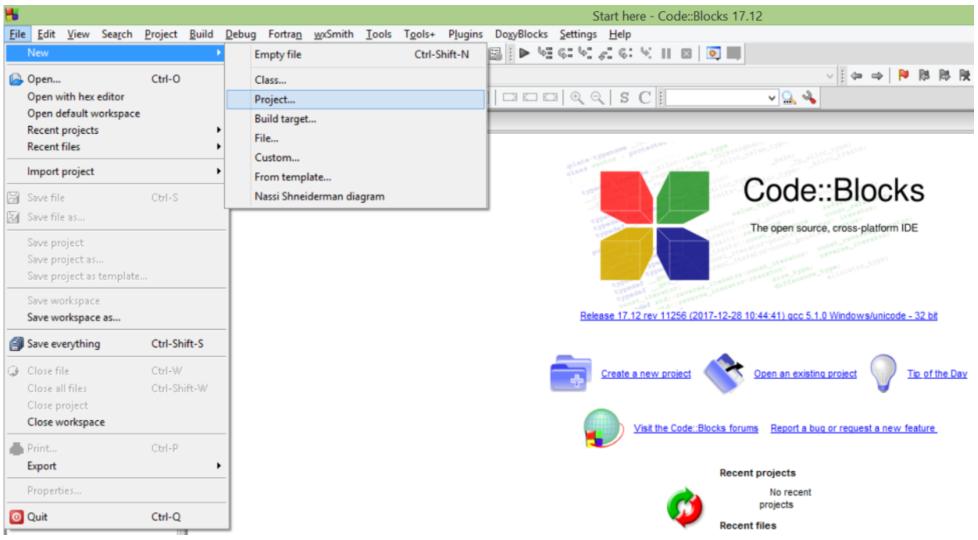
CodeBlocks has been installed on your computer.

Click Finish to close Setup.

3. Project Creation

After installing the Code::Blocks to your system, a project can be created in order to start creating a program. A project is a complete collection of the headers, code parts, and some additive elements such as different wrapped up .dll or .lib files or even media files such as image and audio. All code writing exercises should be completed in a project structure in order for IDE to compile and link all parts of the software properly.

1. A new project can be created by following the “File -> New -> Project…” or clicking “Create a new project”.


Start here - Code::Blocks 17.12

Code::Blocks

The open source, cross-platform IDE

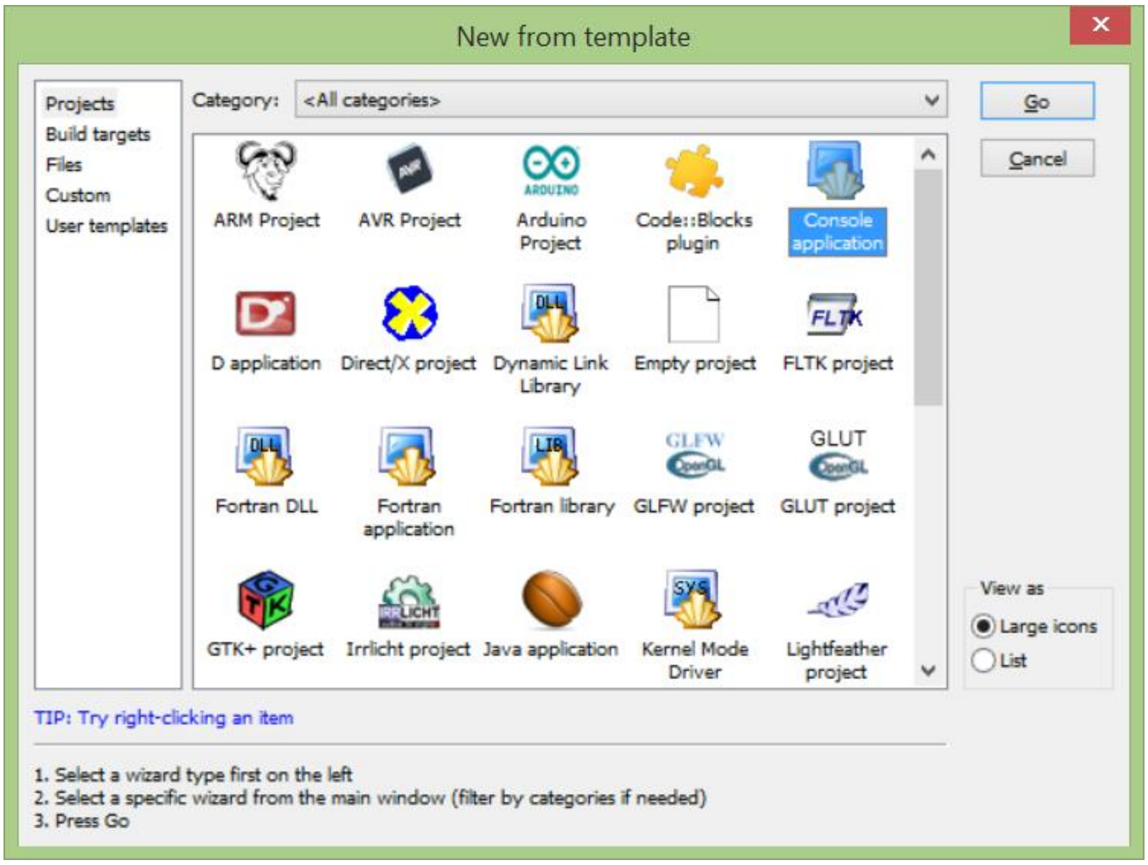
Release 17.12 rev 11256 (2017-12-28 10:44:41) gcc 5.1.0 Windows/unicode - 32 bit

Recent projects

No recent projects

Recent files

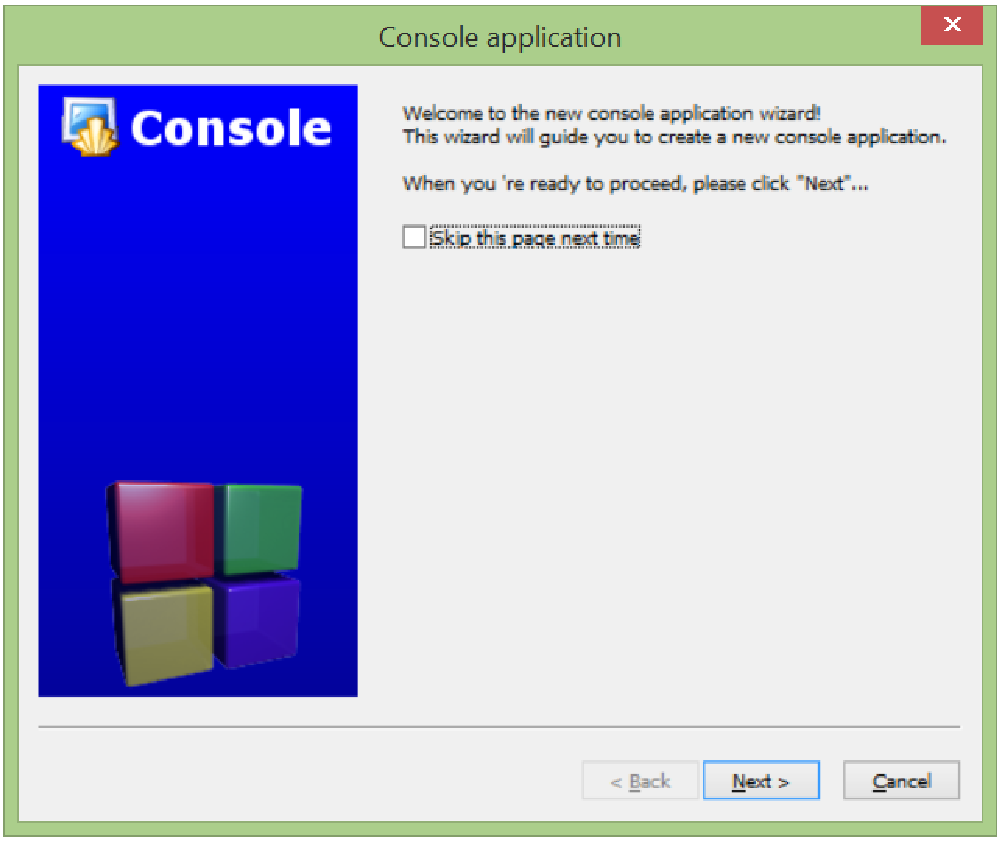
2. Choose “Console application” and click “Go”.



TIP: Try right-clicking an item

1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

3. Click “Next”.

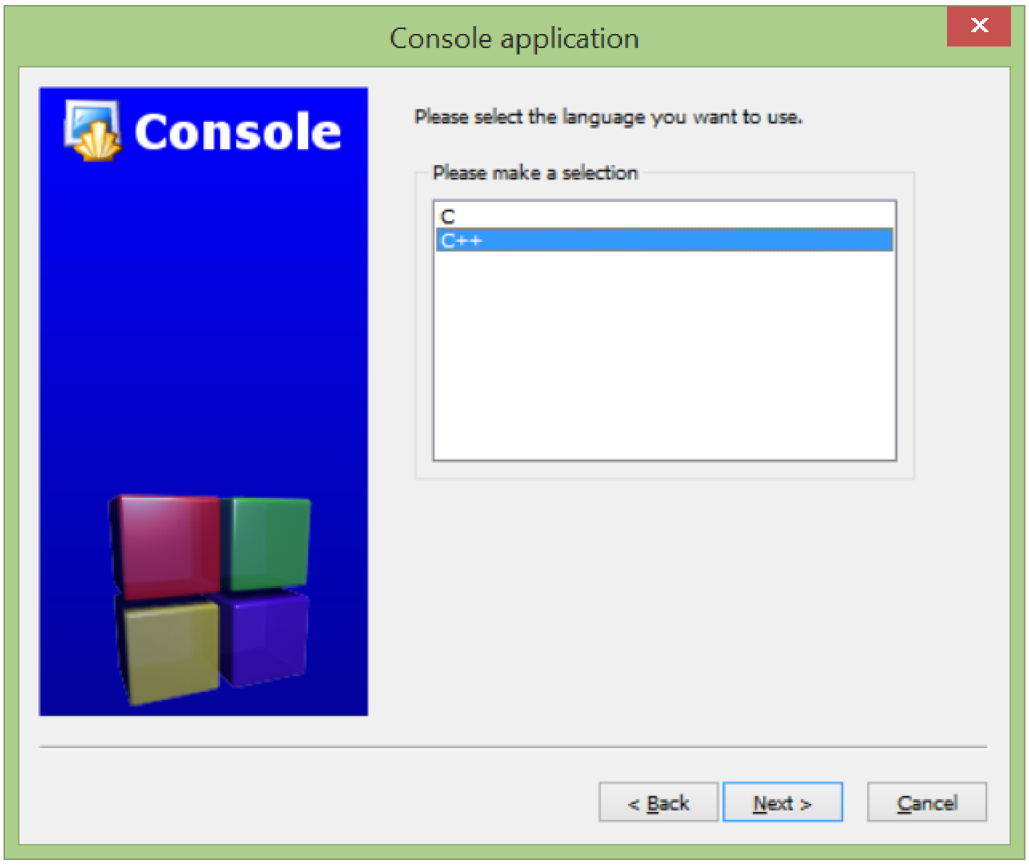


Welcome to the new console application wizard!
This wizard will guide you to create a new console application.

When you 're ready to proceed, please click "Next"...

☐ Skip this page next time

4. Select “C++” and click “Next”.

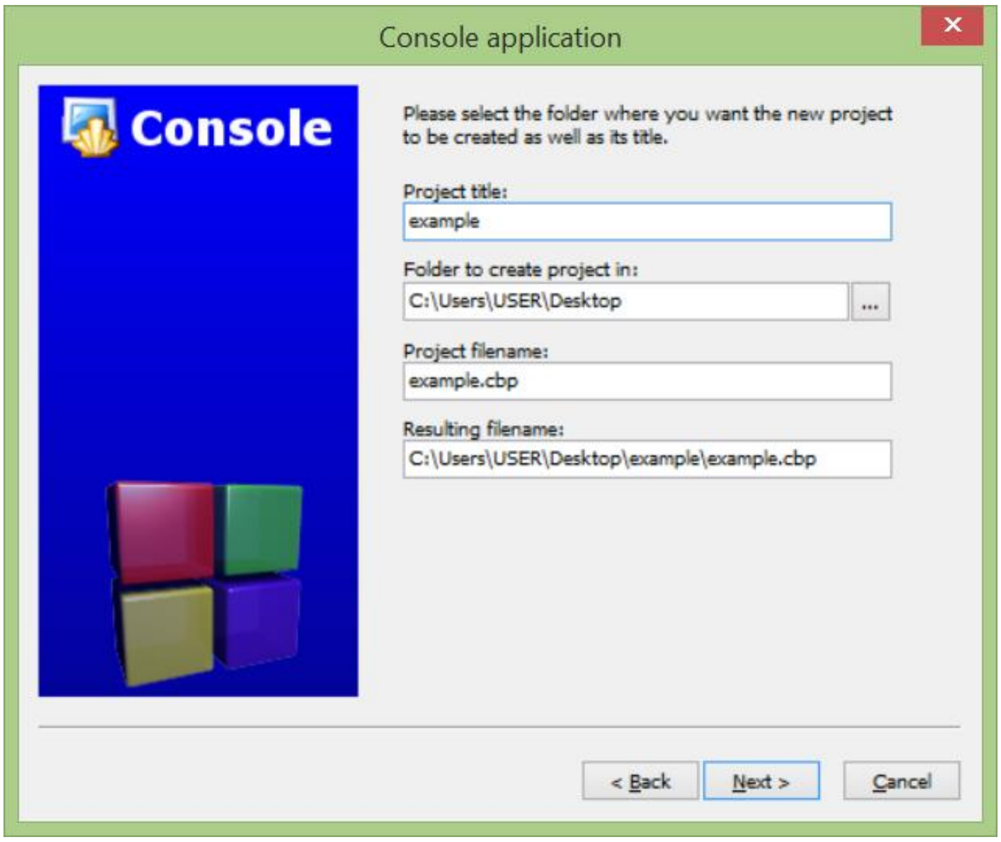


Please select the language you want to use.

Please make a selection

C
C++

5. Choose project title and specify the project folder. Click “Next”.



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

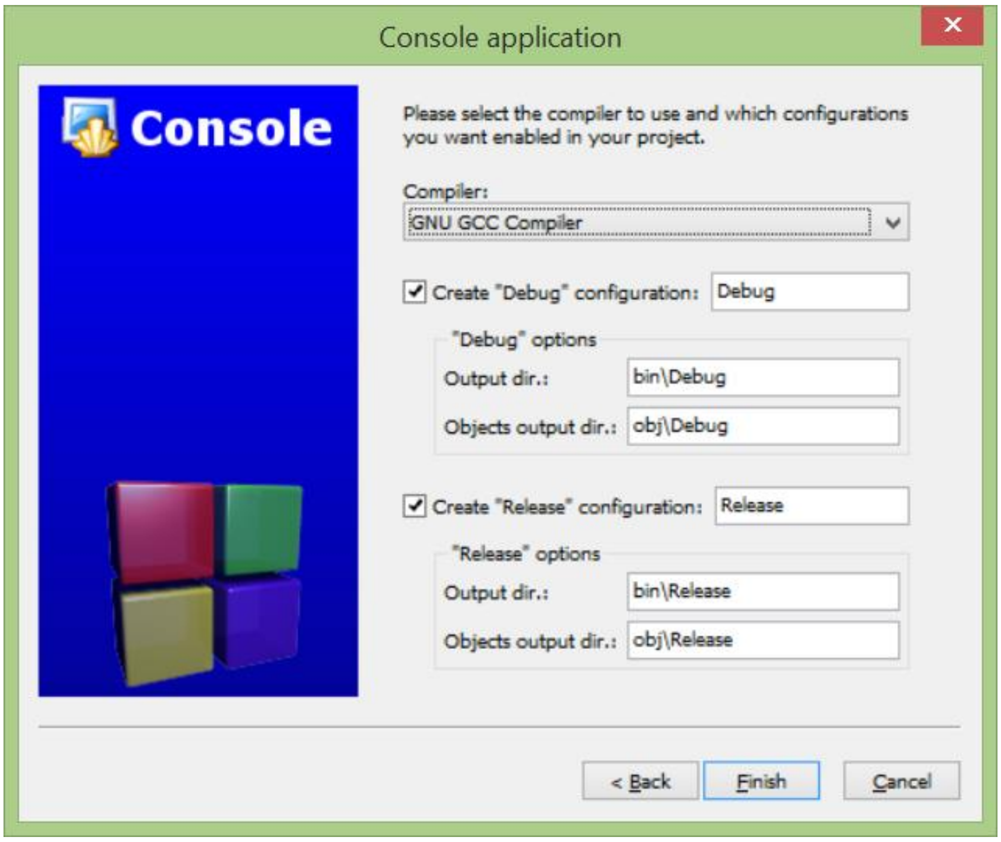
Project title: example

Folder to create project in: C:\Users\USER\Desktop

Project filename: example.cbp

Resulting filename: C:\Users\USER\Desktop\example\example.cbp

6. Choose the compiler as "GNU GCC Compiler", check both debug and release checkboxes. Click "Finish".



Please select the compiler to use and which configurations you want enabled in your project.

Compiler: GNU GCC Compiler

[✓] Create "Debug" configuration: Debug

"Debug" options
Output dir.: bin\Debug
Objects output dir.: obj\Debug

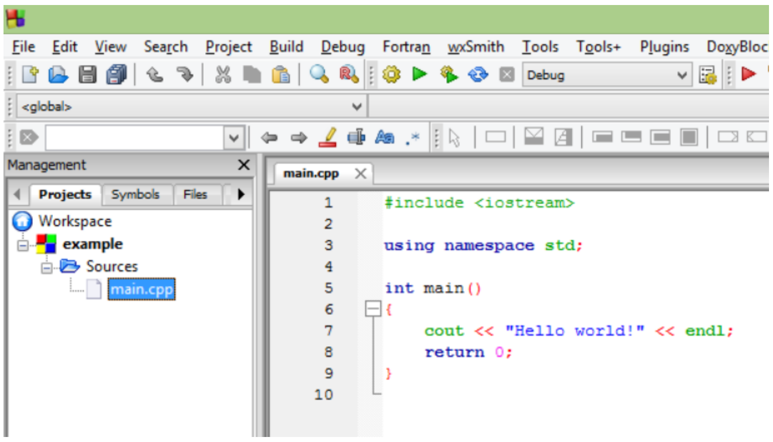
[✓] Create "Release" configuration: Release

"Release" options
Output dir.: bin\Release
Objects output dir.: obj\Release

If you need to add (or create) different files such as headers or additional .cpp files, you should create or add one by choosing "File -> New -> Empty File" option and confirm for addition to the project for creation. You may also right click the project icon on the left pane and choose "Add files" to add external files to your project.

In the Management pane, expand Sources folder. There will be a "main.cpp" file, in which you will write your codes. In the Compiler Toolbar, there are three buttons that you will use frequently:

- Build: Compile the code and generate an executable.
- Run: Run the generated executable.
- Build and run: Compile the code and run the resulting executable.

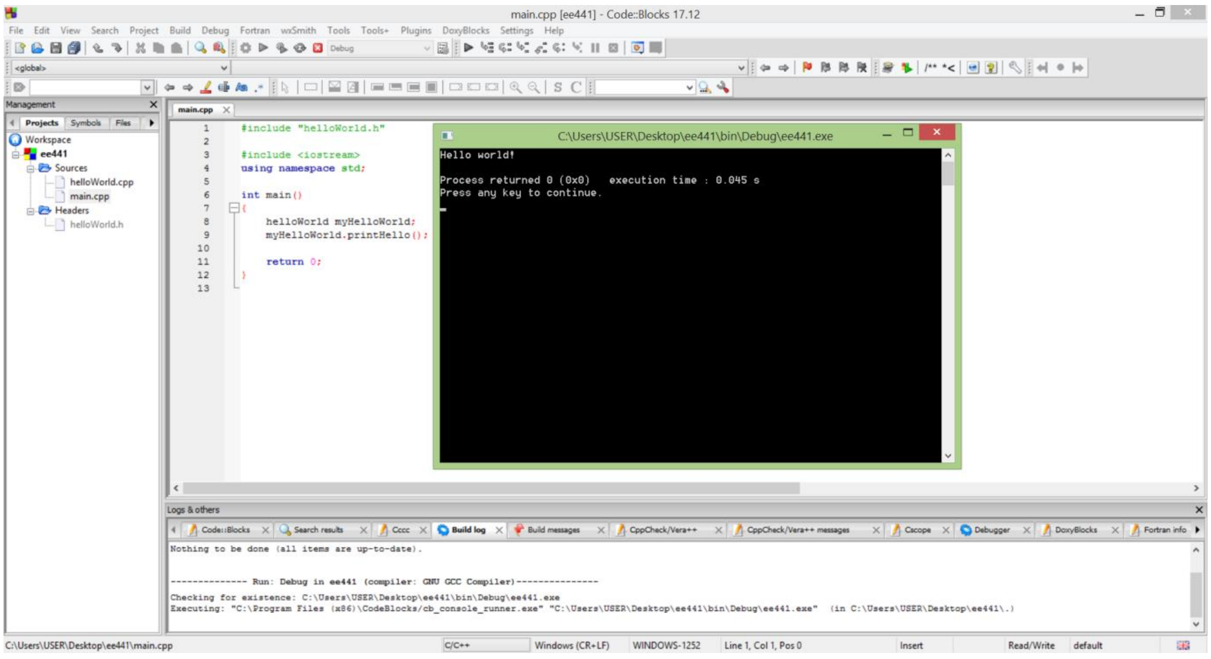

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBloc
Debug
<global>
Management
Projects Symbols Files
Workspace
example
Sources
main.cpp

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

4. Example Project

You are going to implement your classes as separate modules. These modules generally consist of a header file, which has the extension .h and contains the class declaration, and an implementation file, which has the extension .cpp. Please refer to the example project distributed via ODTÜClass. In order to use the class in the project:

- Right click on the project name (ee441) in the Management pane. Choose “Add files”. Find your .cpp file. The file will be added in the Sources folder.
- If the class is in the same directory as the project, #include the .h file in the beginning of the main file. You should use double quotes (“”) instead of angle brackets (<>) when including custom headers. The header file will not be listed under the Sources folder.


main.cpp [ee441] - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
ee441
Sources
helloWorld.cpp
main.cpp
Headers
helloWorld.h
main.cpp
C:\Users\USER\Desktop\ee441\bin\Debug\ee441.exe
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info
C:\Users\USER\Desktop\ee441\main.cpp C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default

```cpp
#include "helloWorld.h"

#include <iostream>
using namespace std;

int main()
{
    helloWorld myHelloWorld;
    myHelloWorld.printHello();

    return 0;
}
```

```
Hello world!

Process returned 0 (0x0)   execution time : 0.045 s
Press any key to continue.
```

```
Nothing to be done (all items are up-to-date).


-------------- Run: Debug in ee441 (compiler: GNU GCC Compiler)---------------

Checking for existence: C:\Users\USER\Desktop\ee441\bin\Debug\ee441.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\USER\Desktop\ee441\bin\Debug\ee441.exe"  (in C:\Users\USER\Desktop\ee441\.)
```

5. How to Write a Clear Source Code?

In order to write a good source code, you need to understand the problem and create an algorithm to solve the standing problem. After an algorithm is created, each step of algorithm is taken as different parts or mostly one or more separate functions in order to support both readability and flexibility.

What is the importance of writing a clean and understandable source code if it does its job well? As many engineering disciplines, we need to use programming for our needs. In this endeavour, there will be different and numerous colleagues to work with. While working with some other team on a certain software development problem, the communication between the developers carry a great importance for the well-being of the work. Even for a single developer, the understandability is needed. You may need your previously written programs and you may not remember the exact way of thinking. Thus, writing clean and understandable software is an essential behaviour.

There are different opinions and debates about the good source writing. To start with, here there are some very basics to keep your source code both understandable and flexible for other applications or additions.

5.1. Formatting In Your Code

In order to keep your source code understandable, first you must start with understandable variable names and class names. Setting your variable names clear will help you when your source code gets more and more complex. You may create or choose your own style of formatting and commenting.

There are different conventions in software development society, which you are NOT asked to learn within the scope of this course. Each of these is created to ease the understandability. If you are interested in formatting your code in a better way, you may look at Google's C++ Guide, which is used in many open-source projects, and different code styling tips and tricks such as naming convention for your functions and parameters.

5.2. Commenting

Commenting is a necessary and helpful act of notation in software development works. Through the supplied comments, a programmer can understand the previous work without analysing the whole software line by line. Without the proper comments, any source code becomes very hard to follow and understand. The commenting should not be overly done (i.e. comments in every line) but it should be done to explain main functions of the blocks or functions. The code given in Appendix part shows an example of both formatting and commenting in a smaller scale.

Writing a clear source code for your software may be a dull thing to do; however, it is necessary to be a good programmer. Please keep those in your mind while doing your work.

6. Homework Format

Even though the required format for the homework will be mentioned in each homework announcement, the format in general requires a few basic principles. These principles can be mentioned as follows:

Implement a completed work of homework requirements.

Each homework will be given by its fully covered requirements. Each of these requirements is important and will affect the performance of your work, and also your grade. Please pay attention to the requirements and try to implement those as your best.

Try to provide a clear output formatting when you are working on Command Line Interface.

On this course, you will be writing console (i.e. command line) applications. It is a good practice for your application to give its outputs in a formatted form in console. That way, your work is easy to follow, and you can understand much easy only by looking to your output and tell where the mistake is when you are debugging your problem.

Send all documents you are asked for as they are.

Please send all extra documents that you are asked to send besides your project in the form as they are asked. You may be asked to write an extra report or comment to your work besides your source code.

Always clear your build before compressing and sending your work.

Because we all may use different operating systems, the executables generated due to your build is not required. In addition to that, it is essential to be your source code to be buildable at any platform. Hence your source code will be built after your homework is downloaded.

Because of these reasons, we do not require executables on your homework. To clean your project, please choose “Build->Clean” option before you compress your project folder.

Always send the project folder (and extra documents) in its compressed form with zip file format.

In order to evaluate your homework, we need a complete project contents with their proper relations. So, sending the project folder after compressing that folder in a zip file is required. Find your project folder in your computer and compress the whole folder. If there are extra documents needed, copy those to the same location of your project folder, then compress all of them in a single zip file.

Name your compressed folder as your student id in e123456 form.

Name your compressed zip file as it is asked in the homework. The naming convention is “e123456_hwX.zip”, where your student number is 123456-7.

Why do we do all these restrictions? Can’t we just write some code and upload it to the Online?

Each one of those which are asked from you is a part of readying you to submit a good, proper, and well documented source code as it is needed to be submitted. Even though you did a perfect job, without proper formatting and documentation, that job will remain as a code piece not software. You may want to be a programmer or not, either way the importance of delivering a good wrapped holds still.

7. Differences between C and C++ Syntax

You learned C language in CENG230/CENG229 course. C and C++ have different syntaxes and have different properties. This part is prepared to show some differences between them.

7.1. Key Difference

- C++ was developed from the C; however they are quite different in nature.
- The most obvious difference is that C is a procedure oriented language; whereas C++ is both procedure and object oriented programming language.

7.2. Data Types

- C does not provide String or Boolean data types. It supports primitive & built-in data types.
- C++ provides String and Boolean data types. It supports both user-defined and built-in data types.

7.3. Overloading

- C does not support function overloading.
- C++ supports function overloading, which means you can have same name of a function with different parameter types.

7.4. Struct vs Class

- C++ structure (named as class) supports private, public and protected member variables as well as functions but C structure (named as struct) does not have private and protected variables.
- C struct does not provide functions but the C++ class provides functions in it.

7.5. Reference Variable

C++ allows pointers and reference variables whereas C only supports pointers.

7.6. Time of Declaring Functions

Apart from main function all C++ functions must be declared before they can be used whereas in C you can use the function before it is declared.

Following code is valid for C, but it is not valid for C++ :

```
#include <stdio.h>
int main()
{
    foo();
    return 0;
}
int foo()
{
    printf( "Hello world" );
}
```

7.7. Time of Defining Variables

- In C, variables has to be defined at the beginning in the function.

```
int i;
for (i=5; i<10; i++)
```

- In C++, variables can be defined anywhere in the function.

```
for (int i=5; i<10; i++)
```

7.8. Functions for Standard Input and Output

- scanf and printf functions are used in C.

```
char X;

printf(“Enter a character: \n”);

scanf(“%c”, &X);
```

- cin and cout functions are used in C++.

```
char X;

cout << “Enter a character: \n“;

cin >> X;
```

7.9. Memory Allocation and Deallocation

- In C programming language, “malloc” and “free” operators are used to memory allocation and deallocation respectively.

```
/* MEMORY ALLOCATION  */

int *Y = malloc( sizeof(int) );
int *ARRAY = malloc( sizeof(int)*10 );

/* MEMORY DEALLOCATION */

free( Y );
free( ARRAY );
```

- “new” and “delete” operators are used for memory allocation and deallocation in C++.

```
/* MEMORY ALLOCATION */

int *Y = new int;
int *ARRAY = new int[10];

/* MEMORY DEALLOCATION */

delete Y;
delete[] ARRAY;
```