

Introduction to Computers and Programming

Why Program ?

Every Profession has tools that make its job easier to do

- Carpenters use hammers – Mechanics use wrenches .. etc.

The computer is a **tool** that is used by many professionals and it can perform so many jobs

- Accountant – Computers balance book analyzes profits and losses .. etc.
- Factory Workers – Control Manufacturing machines and track production

Why Computers ??

Computer Programmers ??

Components of a Computer System

- A computer system consists of both hardware and information stored on hardware. Information stored on computer hardware is often called *software*.
- The **hardware** components of a computer system are the electronic and mechanical parts.
- The **software** components of a computer system are the data and the computer programs.

- The major hardware components of a computer system are:
- **Processor - CPU / Central Processing Unit-electronic** : Controls the computer and how it works. It consists of two basic parts: the **ALU** and the **Control Unit**.
 - **ALU / Arithmetic and Logic Unit**
addition (+), subtraction (-), multiplication (*), division (/),
comparisons (<, <=, >, >=, =, <>)
 - **Control Unit** - Controls the instruction cycle of the program
 - **ROM** - Read Only Memory is found in the CPU.
- **Main Memory/Core Memory (RAM Random Access Memory)**
 - Random Access Memory is **fast**, "**expensive**" and provides more power.
 - Main Memory is **discrete**. (Each space in main memory has its own address...like post office boxes. Addresses are unique, unsigned numbers that are stored in binary form.)
 - Main memory is **volatile** in that it is easily lost when the computer is turned off and the information has not been saved.
- **Secondary Memory** - Slow, cheap, long-lasting (Auxiliary Memory / Savable, infinite memory, non-volatile, external storage). e.g. floppy disks, hard drives, CD, DVD, flash drives, etc.

- **Registers**

- Special purpose storage locations in processor
- Program Counter (PC) : Stores the memory address of the next instruction to be executed
- Instruction Register (IR) : Stores the current instruction being executed or decoded. In simple processors each instruction to be executed is loaded into the instruction register which holds it while it is decoded, prepared and ultimately executed, which can take several steps.
- Accumulator (ACC) : Where the results of all arithmetic operations and loads is stored.

- **Input/Output Devices**

- **Input:** . Input devices take data and converts into information such as keyboard, mouse, scanners, light pens, optical mark readers, voice recognizers, digital cameras, microphones, etc.
- **Output:** monitors, printers, speakers, disk drives, voice synthesizers

For typical desktop computers, the processor, main memory, secondary memory, power supply, and supporting hardware are housed in a metal case. Many of the components are connected to the main circuit board of the computer, called the *motherboard*. The *power supply* supplies power for most of the components. Various input devices (such as the keyboard) and output devices (such as the monitor) are attached through connectors at the rear of the case.



Program or an algorithm

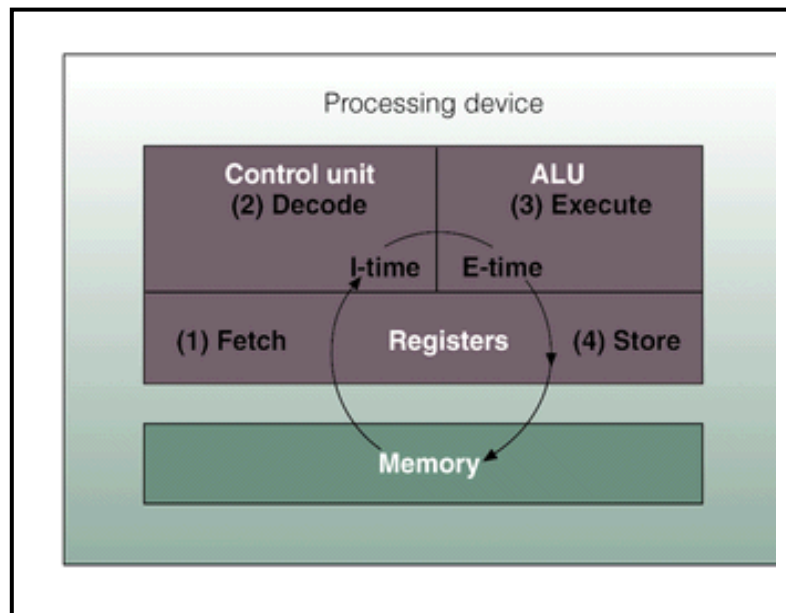
- Program is a set of instructions to perform a specific task
- Stored in main memory
- Instructions are stored sequentially
- Instructions are in machine language (binary)

Example (Program or an algorithm)

- Display on screen: “how many hours did you work?”
- Wait for user to enter number, store in memory
- Display on screen: “what is your pay rate (per hour)?”
- Wait for user to enter rate, store in memory
- Multiply hours by rate, store in memory
- Display on screen: “you have earned \$xx.xx” where xx.xx is result of previous step

Instruction Cycle or Execution Cycle - How does the computer execute a program?

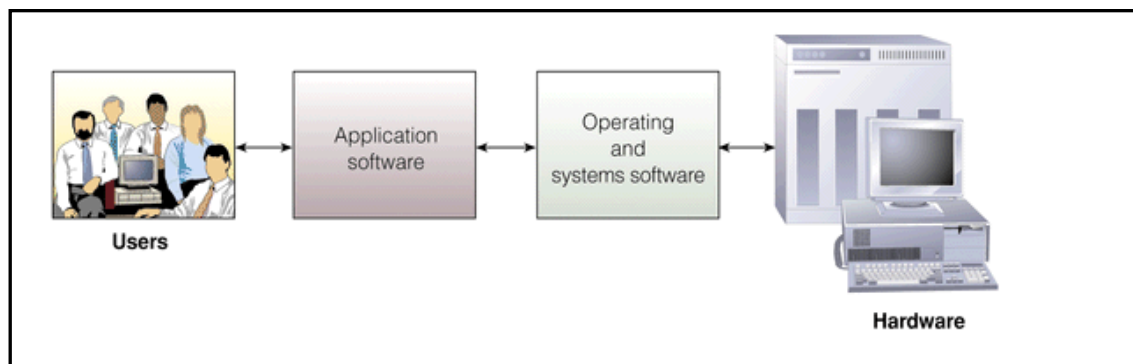
- Fetch the next instruction from memory
 - then increment the program counter
- Decode the instruction
 - interpret components of the instruction
- Execute
 - set things up and send command to appropriate components (ALU, memory, etc.)
- Repeat



Software

- Programs that run on the hardware :-
 - **System Software** (operating systems - a set of programs that allow you to interface with the hardware responsible for processing user requests or user programs e.g. WINDOWS, UNIX .. etc.).
 - **Single tasking** – running one program at a time (MS-DOS)
 - **Multitasking** – running multiple programs at once – called time sharing – Unix – Windows
 - **Application Software** (application programs - e.g. spreadsheets, word processors, database management systems).

A computer system is a collection of hardware (physical components) and software (programs). Together, hardware and software define the computing environment.



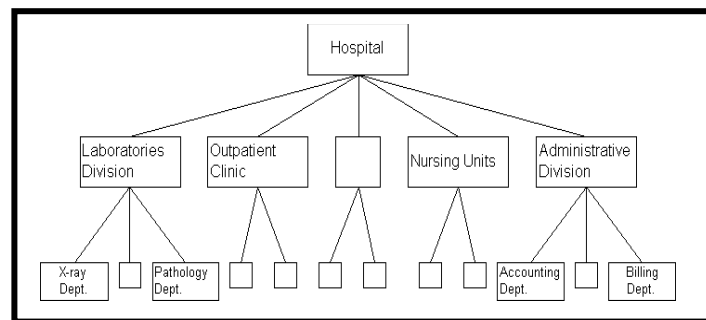
Algorithm

- An algorithm is A step by step ordered procedure that solves a problem in a finite number of **precise** steps

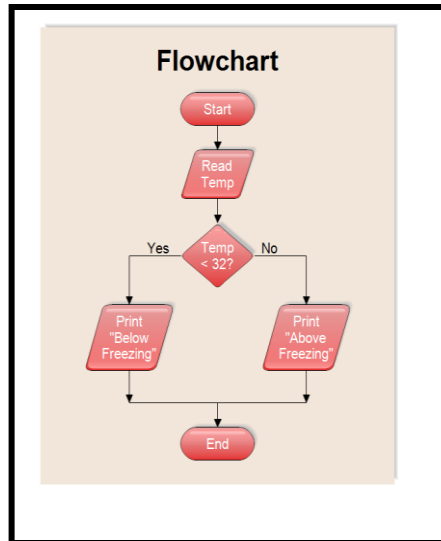
Examples:

- Top-down design (hierarchy chart) : start with the overall task, then break down into progressively smaller tasks (Divide and Conquer Approach).
- Natural Language (English-like statements) .
- Flowchart - a diagram that shows the logical flow of a program.
- Pseudocode - a cross between natural language and a programming language
 - Control Structures (decisions, loops).
 - Style: indentation.

Top-down design



Flowchart



Pseudocode

Example :-

Set total to zero

Set grade counter to one

While grade counter is less than or equal to ten

 Input the next grade

 Add the grade into the total

Set the class average to the total divided by ten

Print the class average.

Programming Languages :-

▪ **Machine Language:** binary (1's and 0's), bits. They are machine specific.

▪ **Low Level Language:** Assembly Language - closer to the numeric machine language of the computer than to natural language. Consist of letters and digits. Disadvantages:

- Machine dependent.
- Not close enough to natural language to be easily learned and understood.
- Require technical background (understanding of computer architecture).

▪ **High Level Languages :-**

Consist of Words, symbols, numbers - Easier for humans to read and use - Must be translated to Machine Code.

- BASIC - met the need for simplicity; often used in an interactive environment
- FORTRAN - designed in the late 1950s to meet the needs of the scientific and engineering communities
- COBOL - designed for business applications
- Pascal - introduced the concept of structured programming & special data types; a teaching language
- C - a structured language developed at Bell Laboratories that allows low-level programming while using a high-level style language
- C++ - a spin off of the C language also developed at Bell Labs that offers object oriented features not found in C; portable
- Java - object-oriented language developed at Sun Microsystems used to develop programs that run over the internet in a Web browser

- Visual Basic - a software development environment by Microsoft that allows programmers to create Windows-based applications
- Python – general purpose languages created in early 1990's. it has become popular in both business and academic applications.
- Ruby – general purpose languages created in early 1990's. it has become popular in language for program that run on Web Servers.

Language Processors or Translators : -

- Assembler - software that translates assembly language programs to machine language
instructions to be executed (later) on a computer
- Interpreter - software that translates one statement at a time of a program into machine
language and executes the statement immediately before going on to process the next
statement ([Dartmouth BASIC](#))
- Compiler - software that translates a program written in a high-level language into binary
machine language instructions so that the program can be executed (later) on a computer.
(C , C++, Java , Cobol)

Translation Process

- Source Code File → [Preprocessor] →
- Modified Source Code → [Compiler] →
- Object Code → [Linker] →
- Executable Code File
- Usually don't see intermediate files
- Using an “Integrated Development Environment” (like Eclipse / Code::Blocks , Visual Studio) you may only see the source, and
- result of running the executable file.

Example

```
#include <iostream>
using namespace std;
int main()
{
    double hours, rate, pay;
    // Get the number of hours worked
    cout << "How many hours did you work? ";
    cin >> hours;
    // Get the hourly pay rate
    cout << "How much do you get paid per hour? ";
    cin >> rate;
    // Calculate the pay
    pay = hours * rate;
    // Display the pay
    cout << "You have earned $" << pay << endl;
    return 0;
}
```

Language Elements

- Reserve Words or Key Words** : have special meaning (lowercase)
- Programmer Defined Identifiers** : names made by programmer
- Operators** : instruction to manipulate data (* , + ...)
- Punctuation** : special meaning to compiler (; \ ...)
- Statement** : complete instruction to computer to perform an action.
- Variables** : named storage location in memory for holding a piece of information.
- Variable Definition** : instruction to set up variable requires :
 - data type
 - information (numbers, characters , ...)

Categories of Instructions

- Input
 - `cin >> hours`
 - gathers info from “outside world”
- Processing
 - `pay = hours * rate;`
 - computation
- Output
 - `cout << “How many hours did you work? “;`
 - sends info to “outside world”

Programming Process

1. Clearly define the problem
2. Visualize output of program
3. Make a model of the program
 - # hierarchy chart
 - # flowcharts
 - # pseudocode
4. Translate to C++ code (type it into a file)
5. Compile, fix syntax errors, repeat
6. Test the program (execute it with data)
7. Correct errors, go to step 5. If no errors, quit.

What is Software Engineering?

Software Engineering **is** a branch of computer science. It deals with entire process of developing and maintaining computer software :-

- a. Analysis
- b. Designing
- c. Writing Code
- d. Testing
- e. Debugging
- f. Documenting
- g. Modifying (updating)
- h. Maintaining (fixing bugs reported by users)

- **Problem Analysis and Specification**

- Analyze the problem - Specify precisely what the solution requires - list **data objects**.

- a. **input values**
- b. **constant values**
- c. **output values**

- **Design**

1. **Describe** the data objects.
 - a. **kind** (variable or constant)
 - b. **type** (integer, real, etc.)
 - c. **name** (How will the program refer to the data object?)
2. **Describe** the operations using English like statements.
3. **Construct** an algorithm using pseudocode--a step by step ordered procedure that describes the solution to a given problem in a finite number of precise steps.

- **Coding** (implement pseudocode in language of choice)
- **Verification and Validation** (testing - debugging) - **Error types:**
 - a. **Compile time** or **syntax** errors --- ex. missing semicolon
 - b. **Linking** errors - ex. `#include <cmath>` missing when using `sqrt()` function
 - c. **Run-time errors** (occur when program is running) ex. attempt to divide by zero
 - d. **Logic errors** (errors in reasoning)
- **Documentation**
- **Modifying – Maintenance**
- **Obsolescence**

C++

What is C++

- C++ is a programming language designed to making programming more enjoyable for the serious programmer.
- In the 1990s it became one of the most popular programming languages in the world.
-

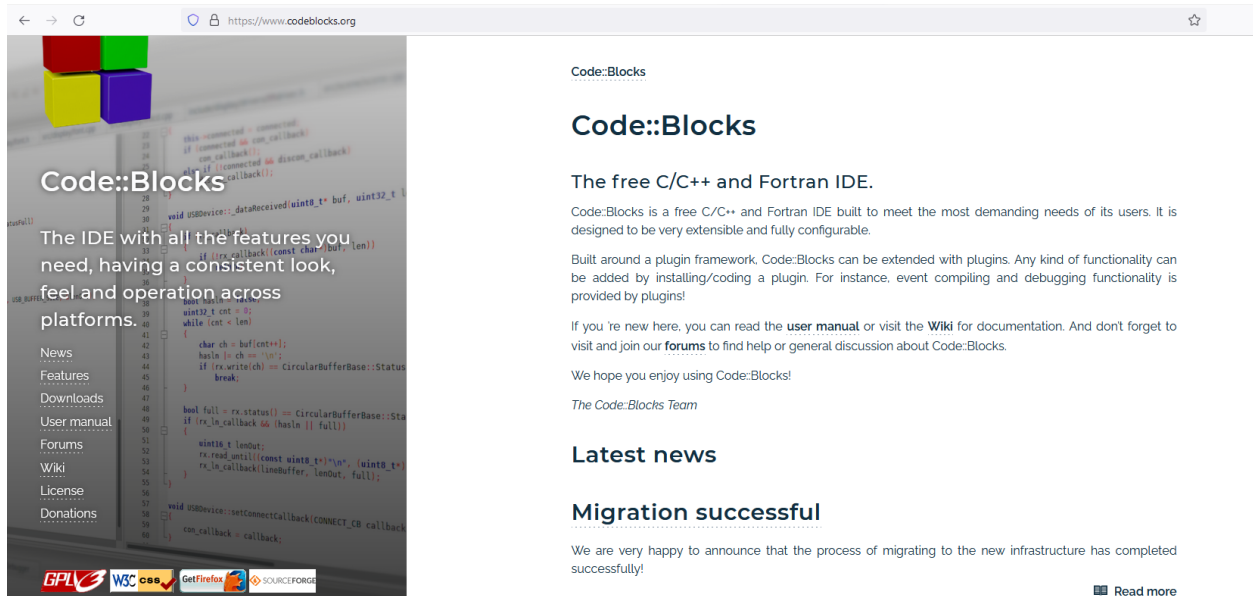
Development

- C++, designed by AT&T Bell Lab's Bjarne Stroustrup, was developed as an enhancement to the C programming language.
- Enhancements Included the addition of classes followed by many features such as:
 - Virtual functions
 - Operator overloading
 - Multiple Inheritance
 - Templates
 - Exception handling

CodeBlocks – Installation

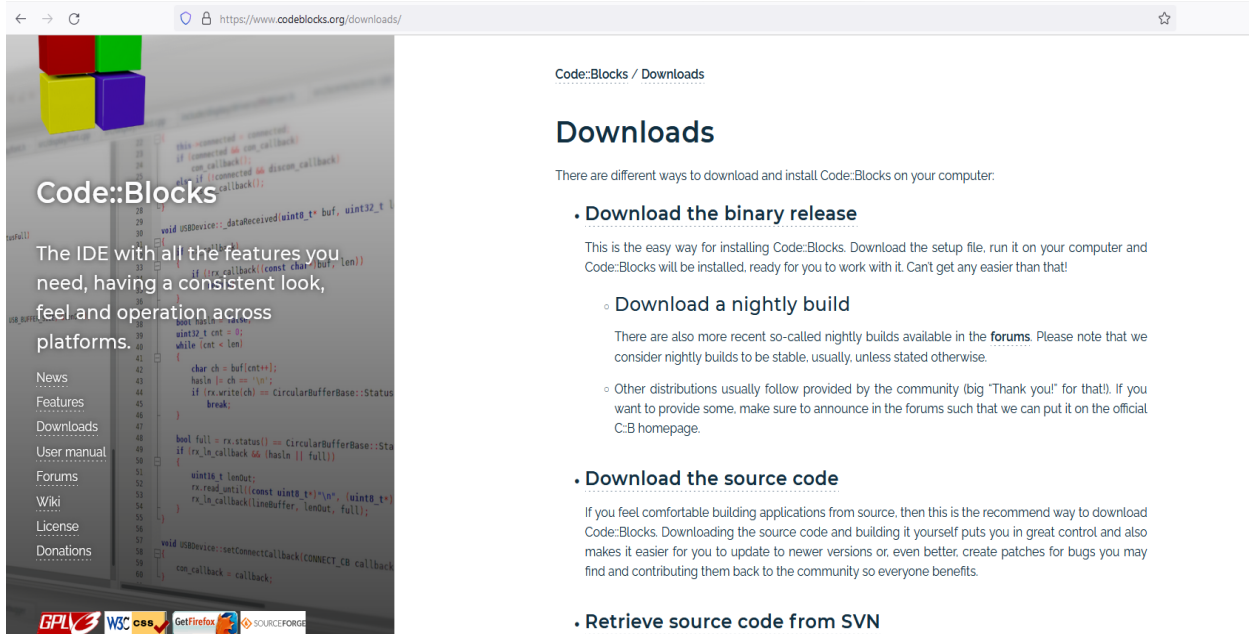
To install CodeBlocks Version 20.03 on [Windows](#)

Go to <http://codeblocks.org>



Click on Downloads

<http://codeblocks.org/downloads/>



Code::Blocks / Downloads

Downloads

There are different ways to download and install Code::Blocks on your computer.

- **Download the binary release**

This is the easy way for installing Code::Blocks. Download the setup file, run it on your computer and Code::Blocks will be installed, ready for you to work with it. Can't get any easier than that!

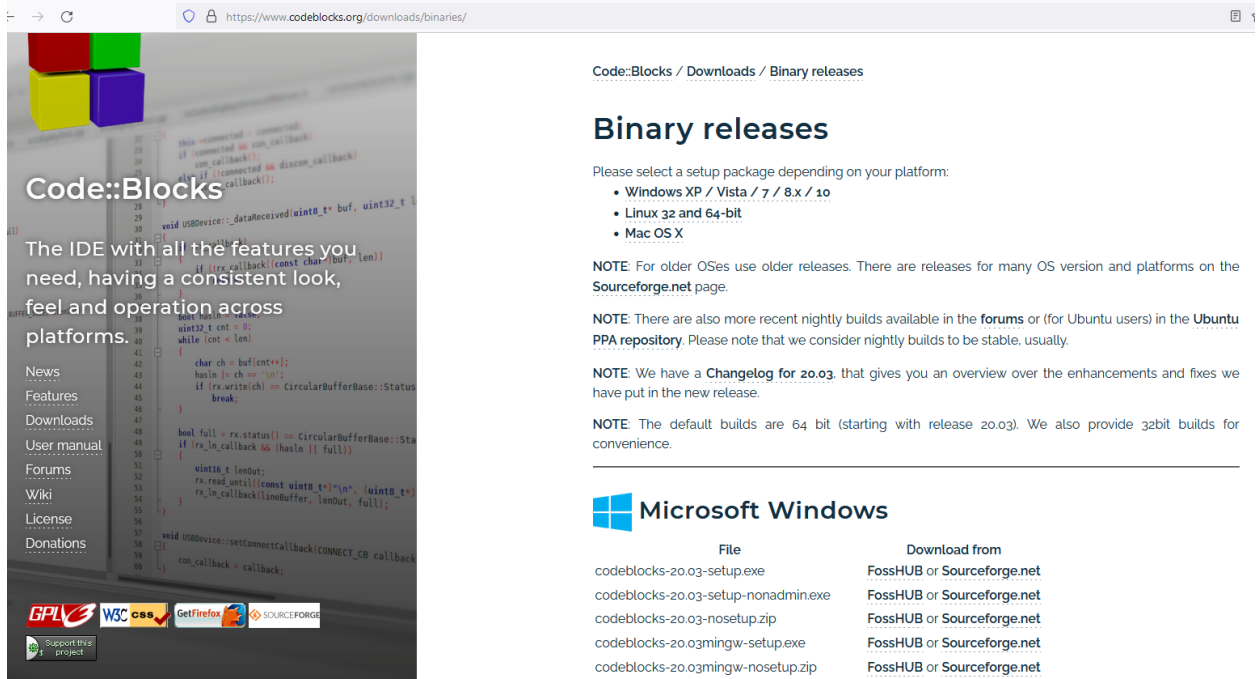
 - **Download a nightly build**

There are also more recent so-called nightly builds available in the **forums**. Please note that we consider nightly builds to be stable, usually, unless stated otherwise.
 - Other distributions usually follow provided by the community (big "Thank you!" for that!). If you want to provide some, make sure to announce in the forums such that we can put it on the official C.B homepage.
- **Download the source code**

If you feel comfortable building applications from source, then this is the recommend way to download Code::Blocks. Downloading the source code and building it yourself puts you in great control and also makes it easier for you to update to newer versions or, even better, create patches for bugs you may find and contributing them back to the community so everyone benefits.
- **Retrieve source code from SVN**

Click on Binaries

Click on <http://codeblocks.org/downloads/binaries>



Code::Blocks

The IDE with all the features you need, having a consistent look, feel and operation across platforms.

News
Features
Downloads
User manual
Forums
Wiki
License
Donations

GPL W3C CSS Get Firefox SOURCEFORGE

Support this project

Code::Blocks / Downloads / Binary releases

Binary releases

Please select a setup package depending on your platform:

- Windows XP / Vista / 7 / 8.x / 10
- Linux 32 and 64-bit
- Mac OS X

NOTE: For older OSes use older releases. There are releases for many OS version and platforms on the Sourceforge.net page.

NOTE: There are also more recent nightly builds available in the forums or (for Ubuntu users) in the Ubuntu PPA repository. Please note that we consider nightly builds to be stable, usually.

NOTE: We have a Changelog for 20.03, that gives you an overview over the enhancements and fixes we have put in the new release.

NOTE: The default builds are 64 bit (starting with release 20.03). We also provide 32bit builds for convenience.

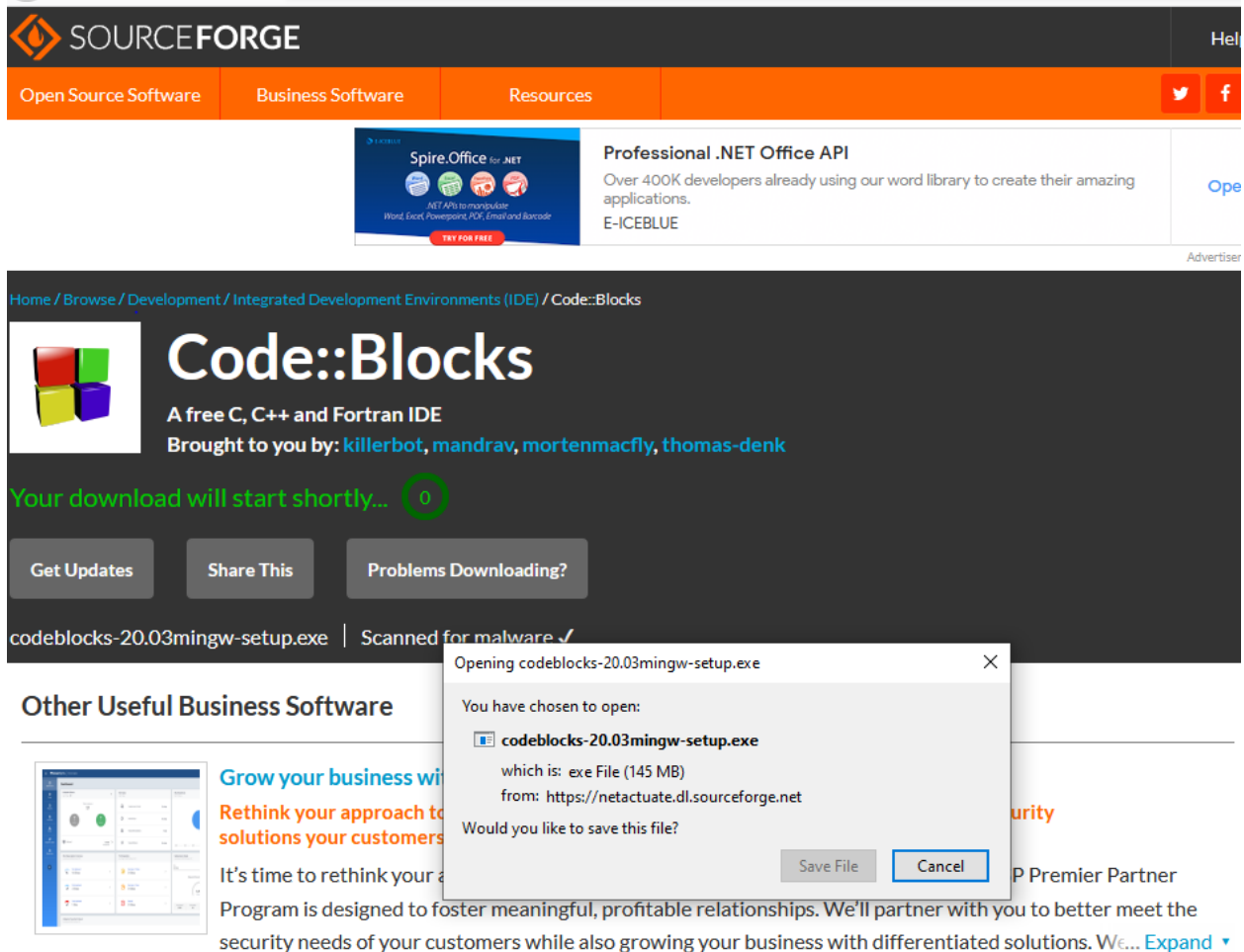
Microsoft Windows

| File | Download from |
|-------------------------------------|---|
| codeblocks-20.03-setup.exe | FossHUB or Sourceforge.net |
| codeblocks-20.03-setup-nonadmin.exe | FossHUB or Sourceforge.net |
| codeblocks-20.03-nosetup.zip | FossHUB or Sourceforge.net |
| codeblocks-20.03mingw-setup.exe | FossHUB or Sourceforge.net |
| codeblocks-20.03mingw-nosetup.zip | FossHUB or Sourceforge.net |

Select and download

codeblocks-20.03mingw-setup

Download from Sourceforge.net



The screenshot shows the SourceForge website for Code::Blocks. The page header includes the SourceForge logo and navigation links for Open Source Software, Business Software, and Resources. A banner for Spire.Office for .NET is visible. The main content area features the Code::Blocks logo and a download progress bar that says "Your download will start shortly..." with a green circle containing the number 0. Below the progress bar are buttons for "Get Updates", "Share This", and "Problems Downloading?". A Windows file dialog box is open, titled "Opening codeblocks-20.03mingw-setup.exe". The dialog box contains the text: "You have chosen to open:", "codeblocks-20.03mingw-setup.exe", "which is: exe File (145 MB)", "from: https://netactuate.dl.sourceforge.net", and "Would you like to save this file?". The "Save File" button is highlighted with a blue border.

Home / Browse / Development / Integrated Development Environments (IDE) / Code::Blocks

Code::Blocks

A free C, C++ and Fortran IDE
Brought to you by: [killerbot](#), [mandrav](#), [mortenmacfly](#), [thomas-denk](#)

Your download will start shortly... 0

[Get Updates](#) [Share This](#) [Problems Downloading?](#)

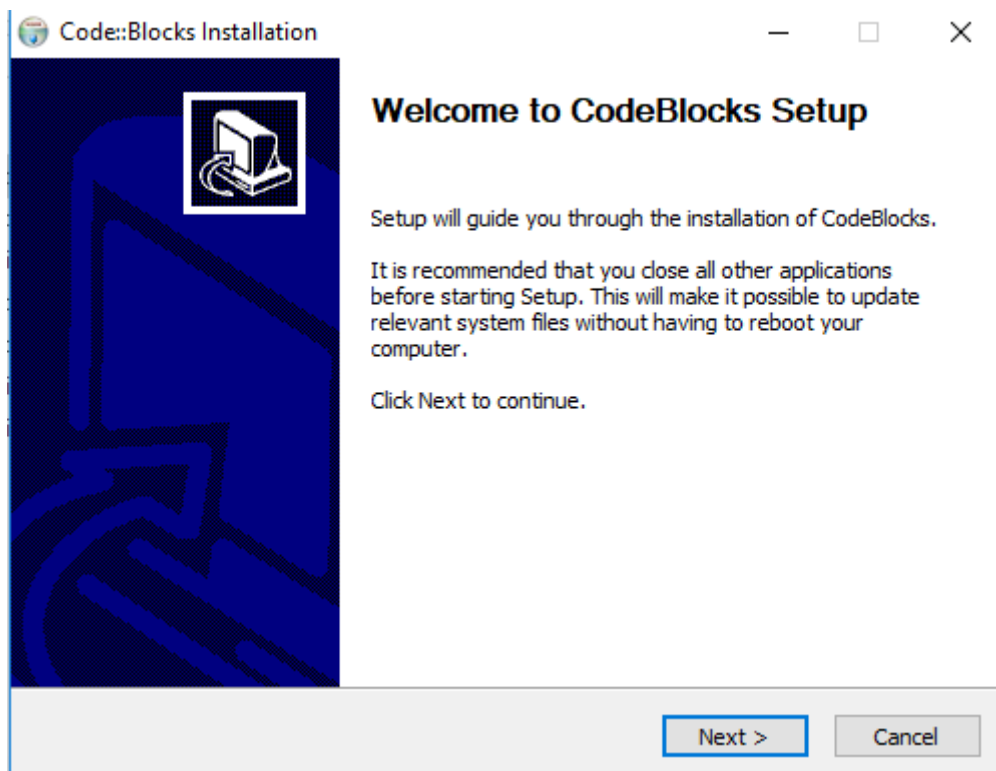
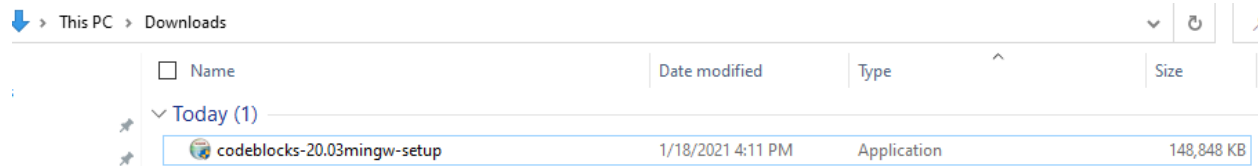
codeblocks-20.03mingw-setup.exe | Scanned for malware ✓

Other Useful Business Software

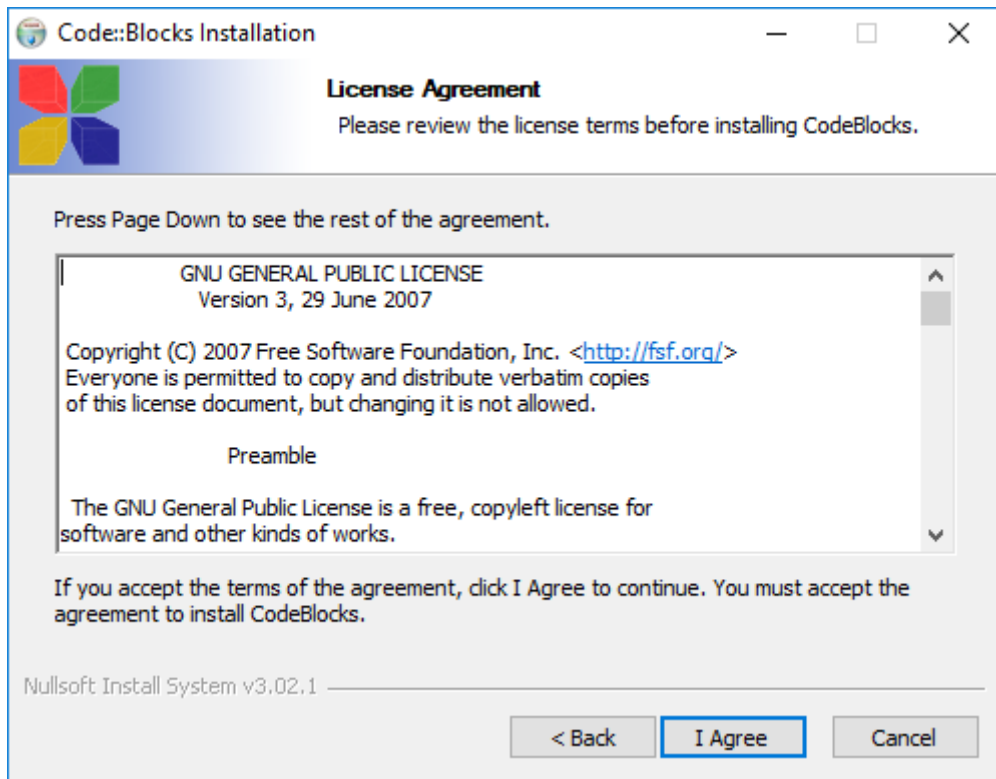
Grow your business with...
Rethink your approach to...
solutions your customers...
It's time to rethink your...
Program is designed to foster meaningful, profitable relationships. We'll partner with you to better meet the security needs of your customers while also growing your business with differentiated solutions. We... [Expand](#)

Click on **Save File**

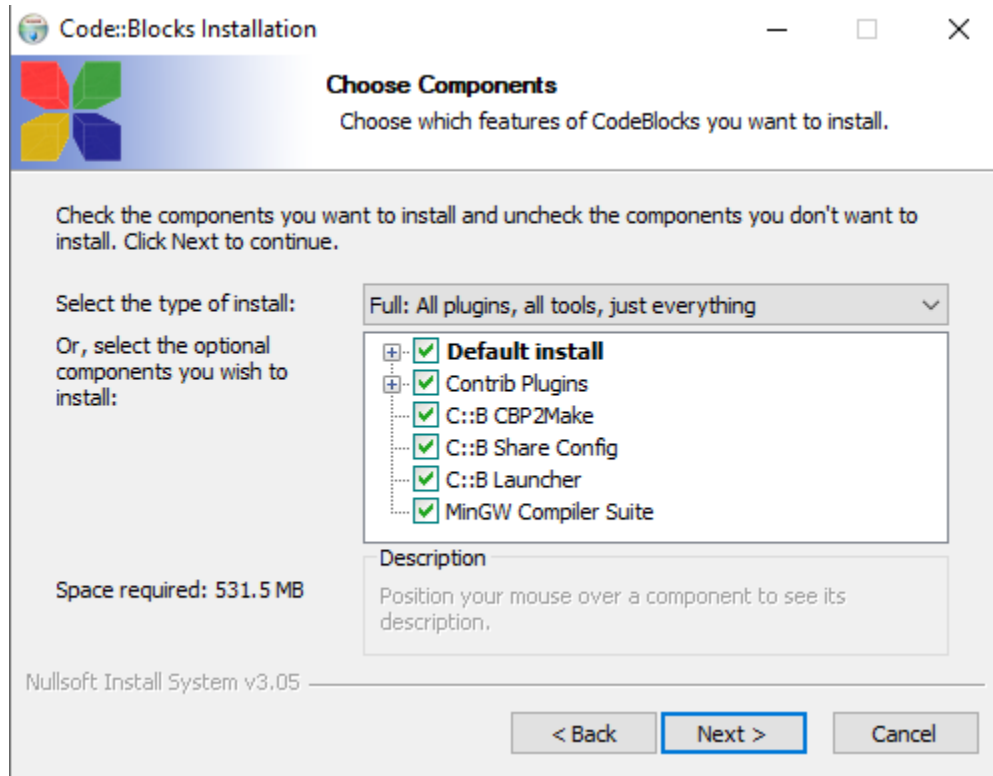
From your download folder , Double click on the downloaded file



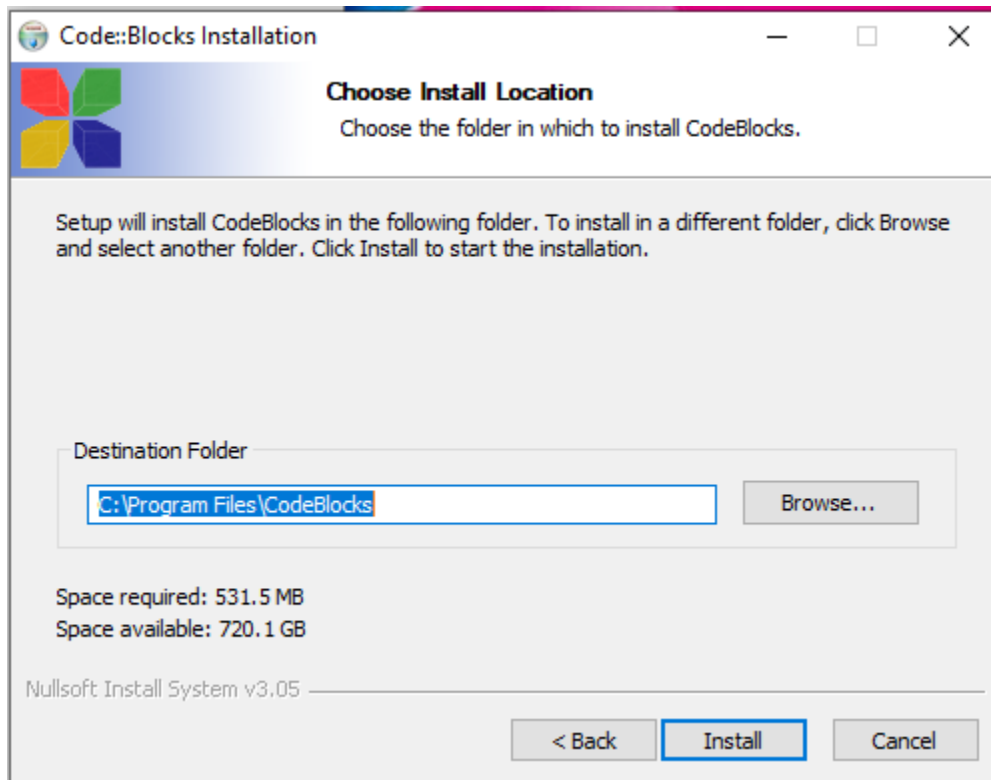
Click on **Next**



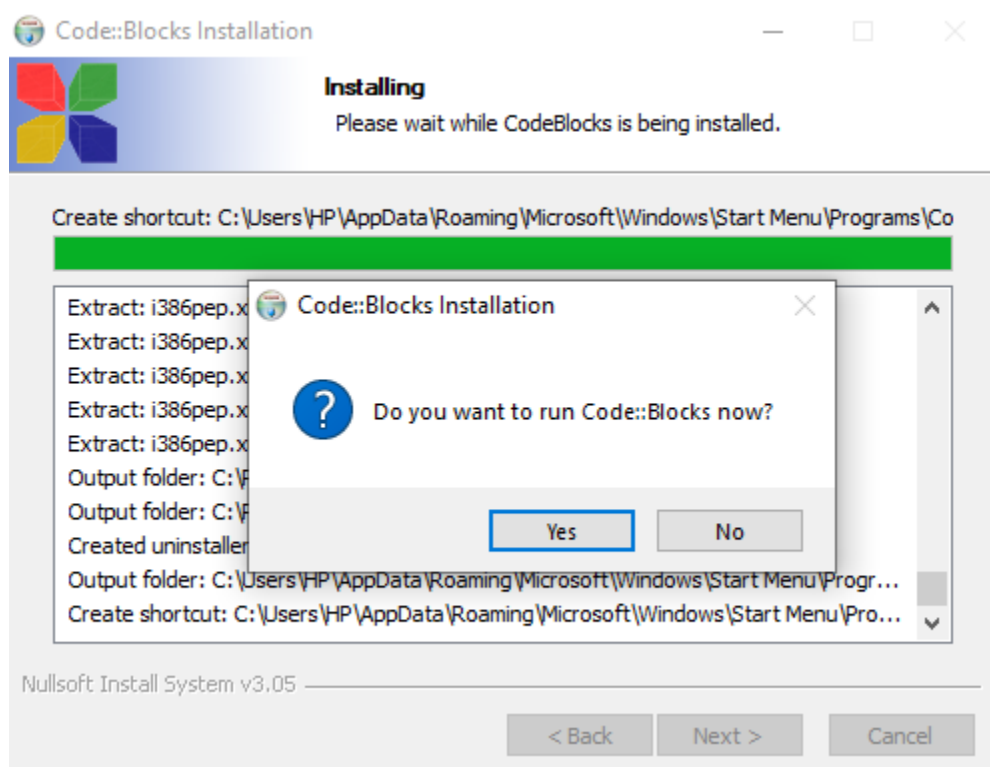
Click on **I Agree**



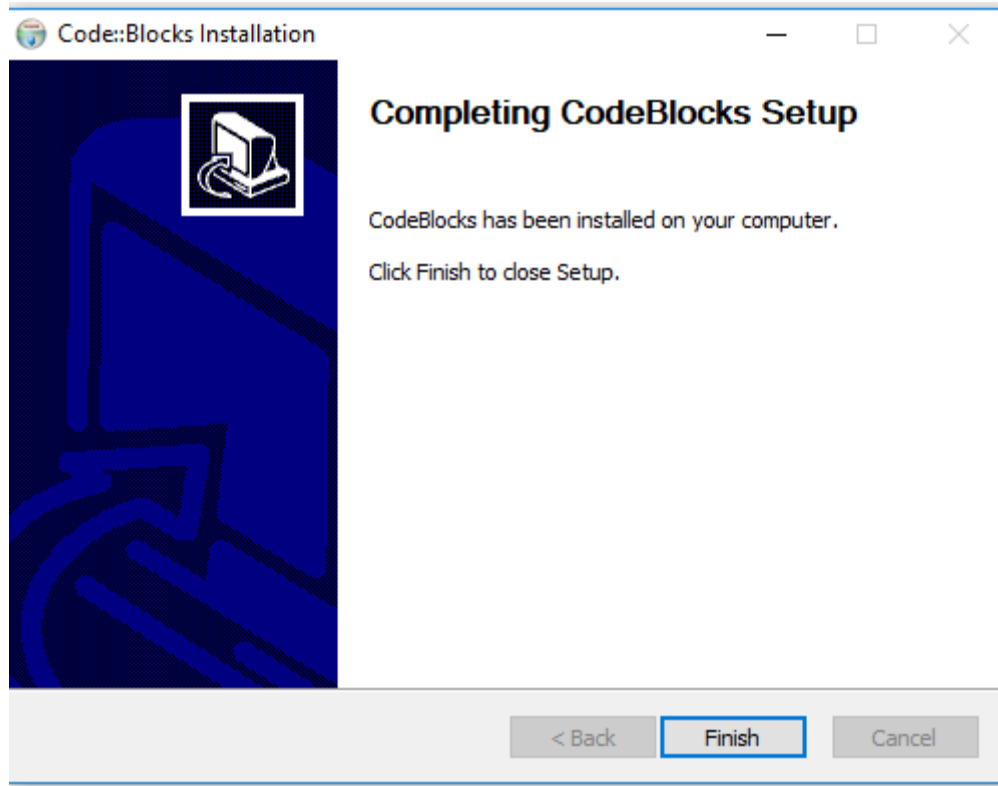
Click on Next



Click on Install



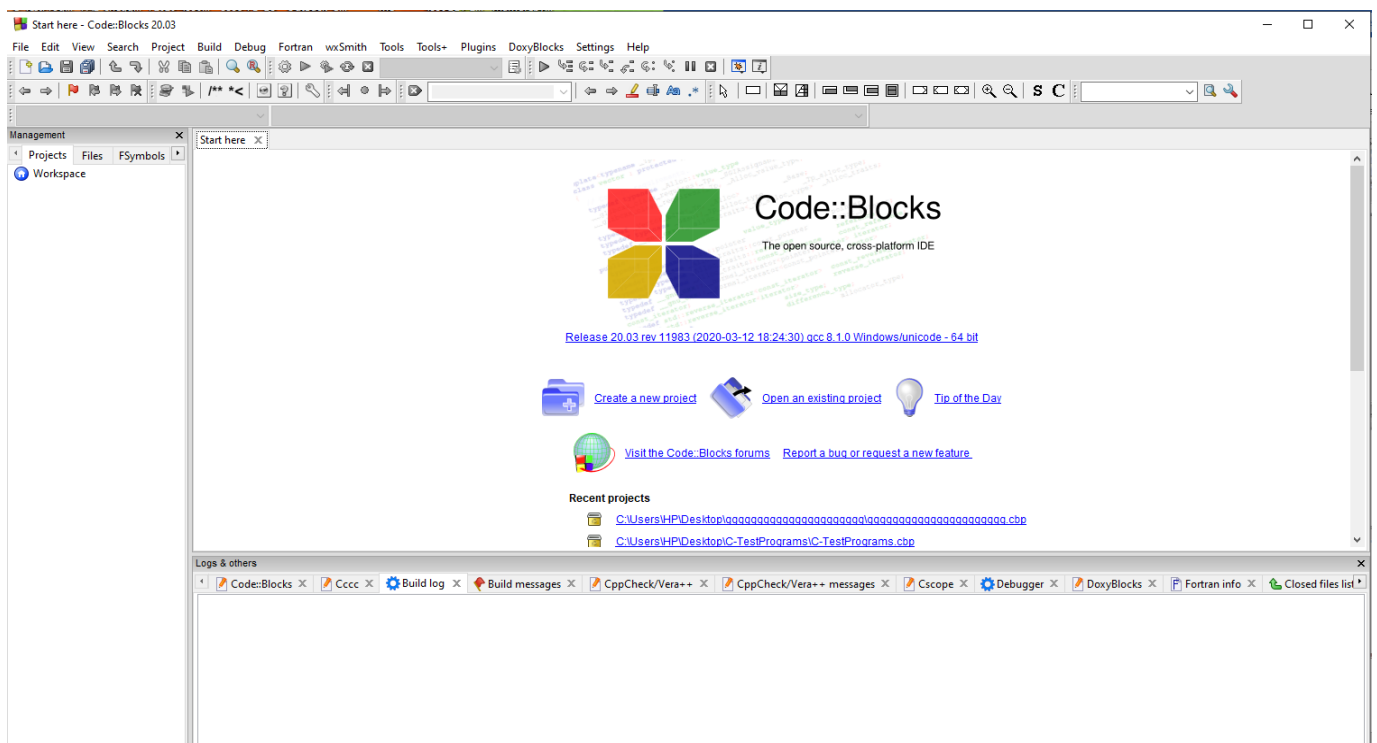
Click on Yes



Select Finish

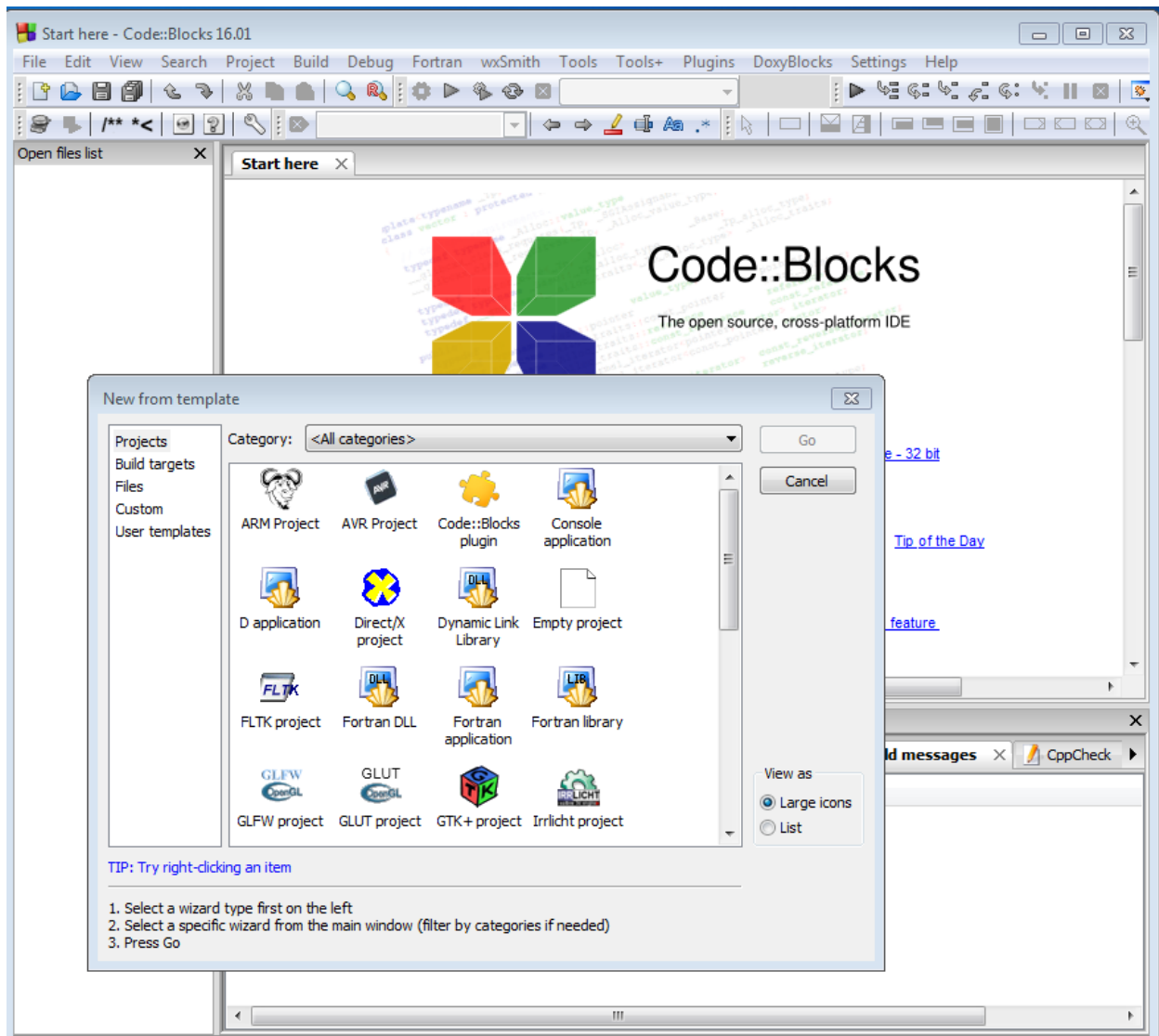
How to create projects and write programs using CodeBlocks

Double click on the CodeBlocks short cut



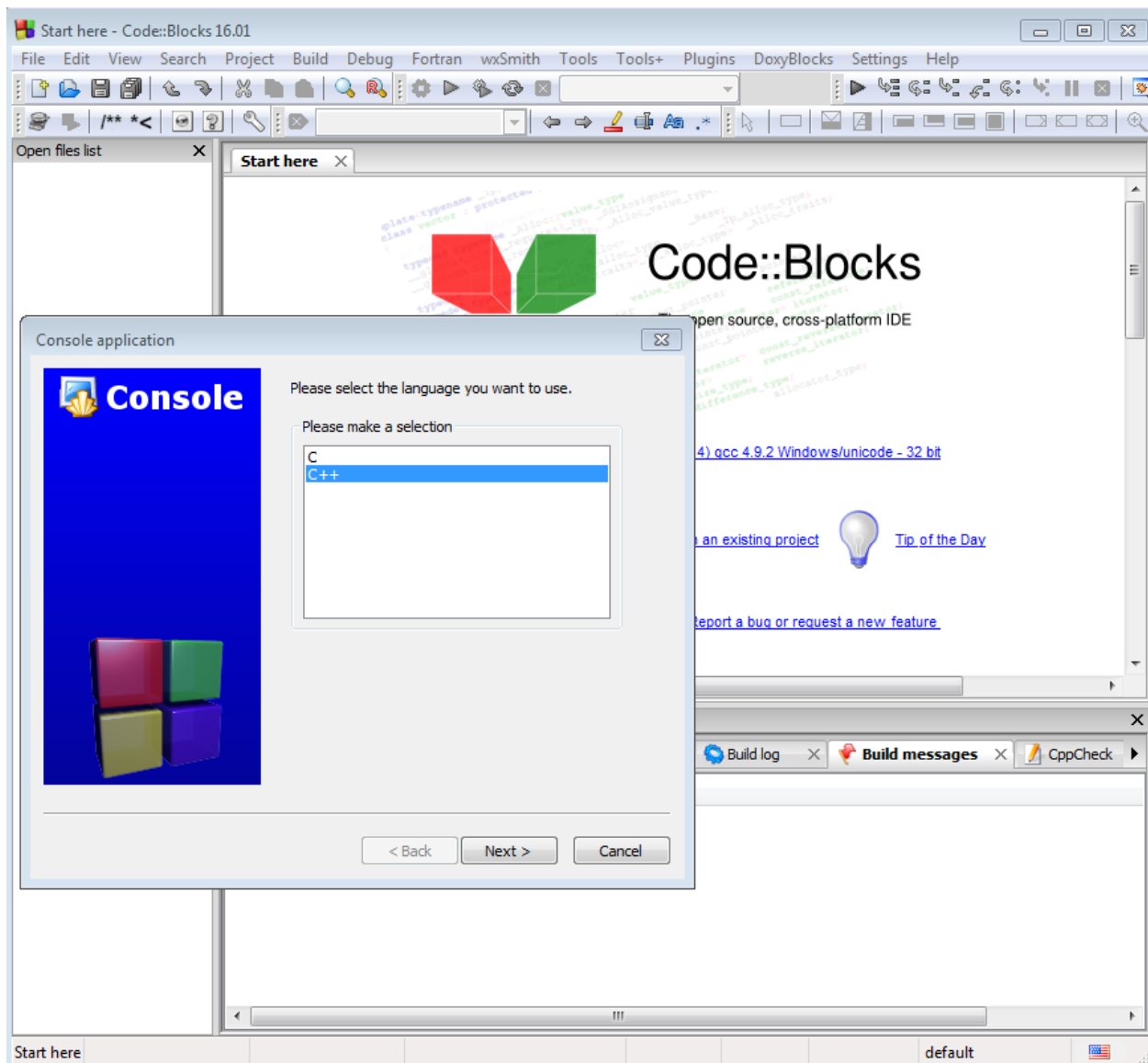
First , you need to create a project as follows :-

File -----> New -----> Project ---> enter

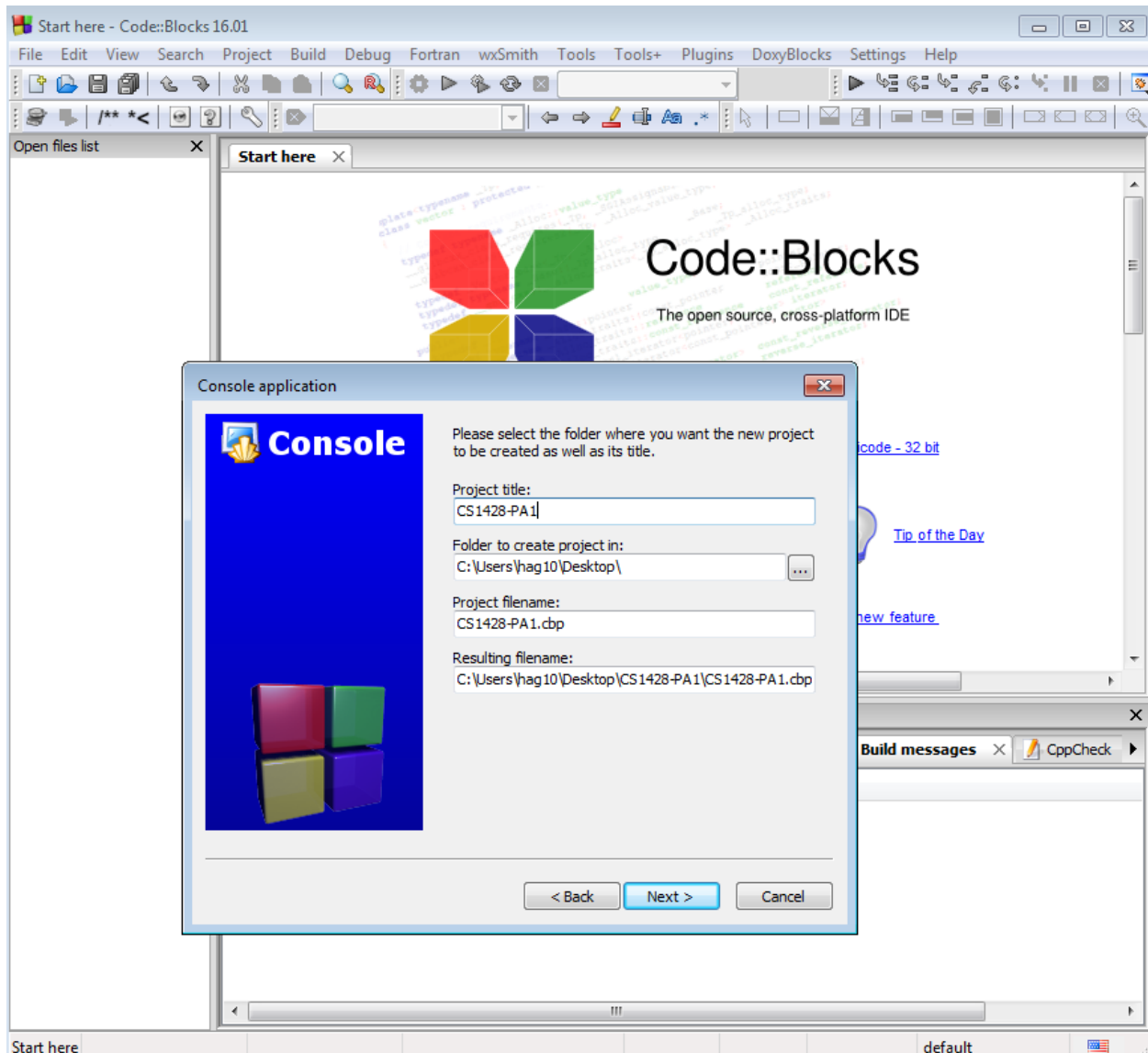


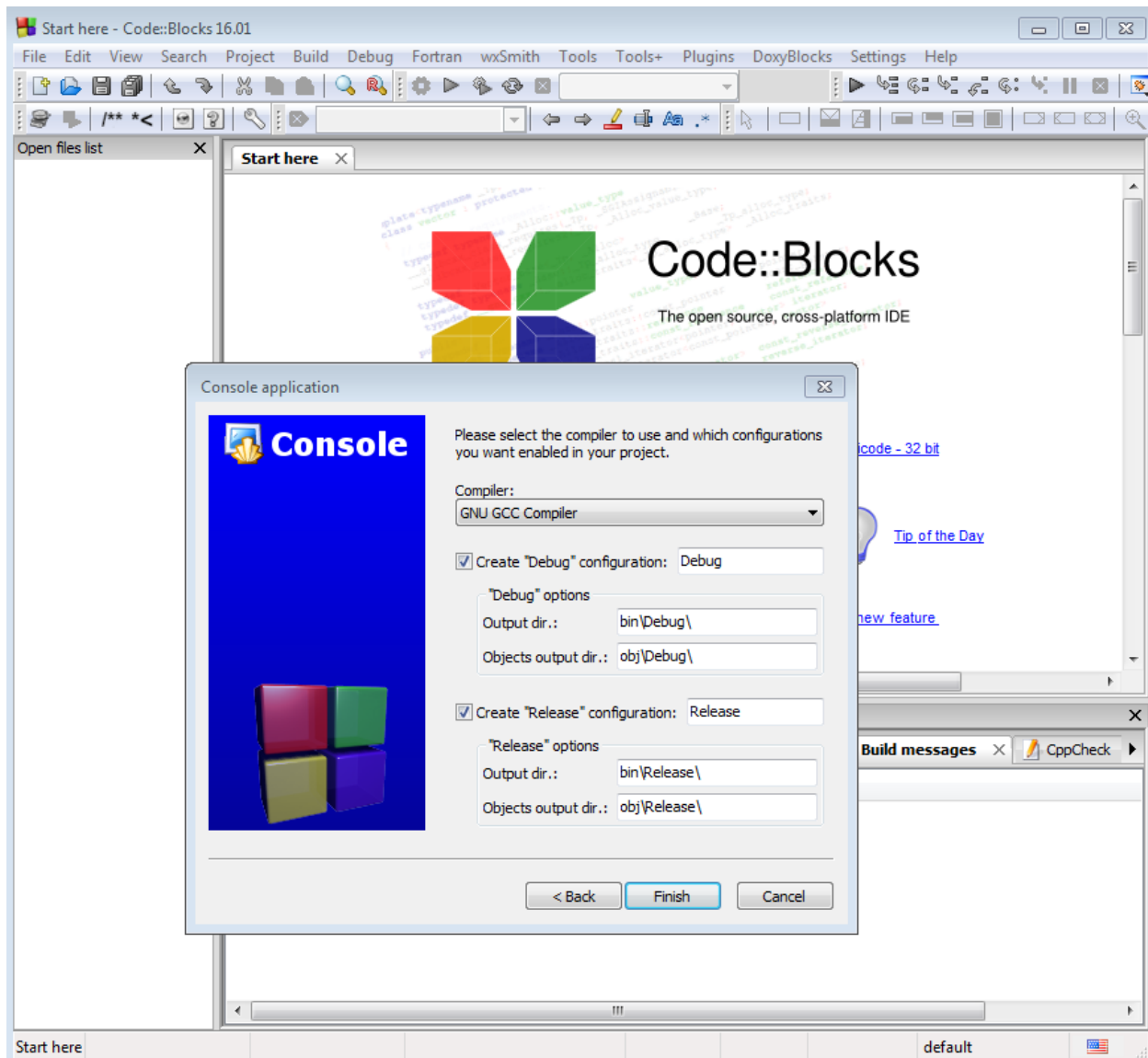
Select Console application

Then Select C++ --- > next



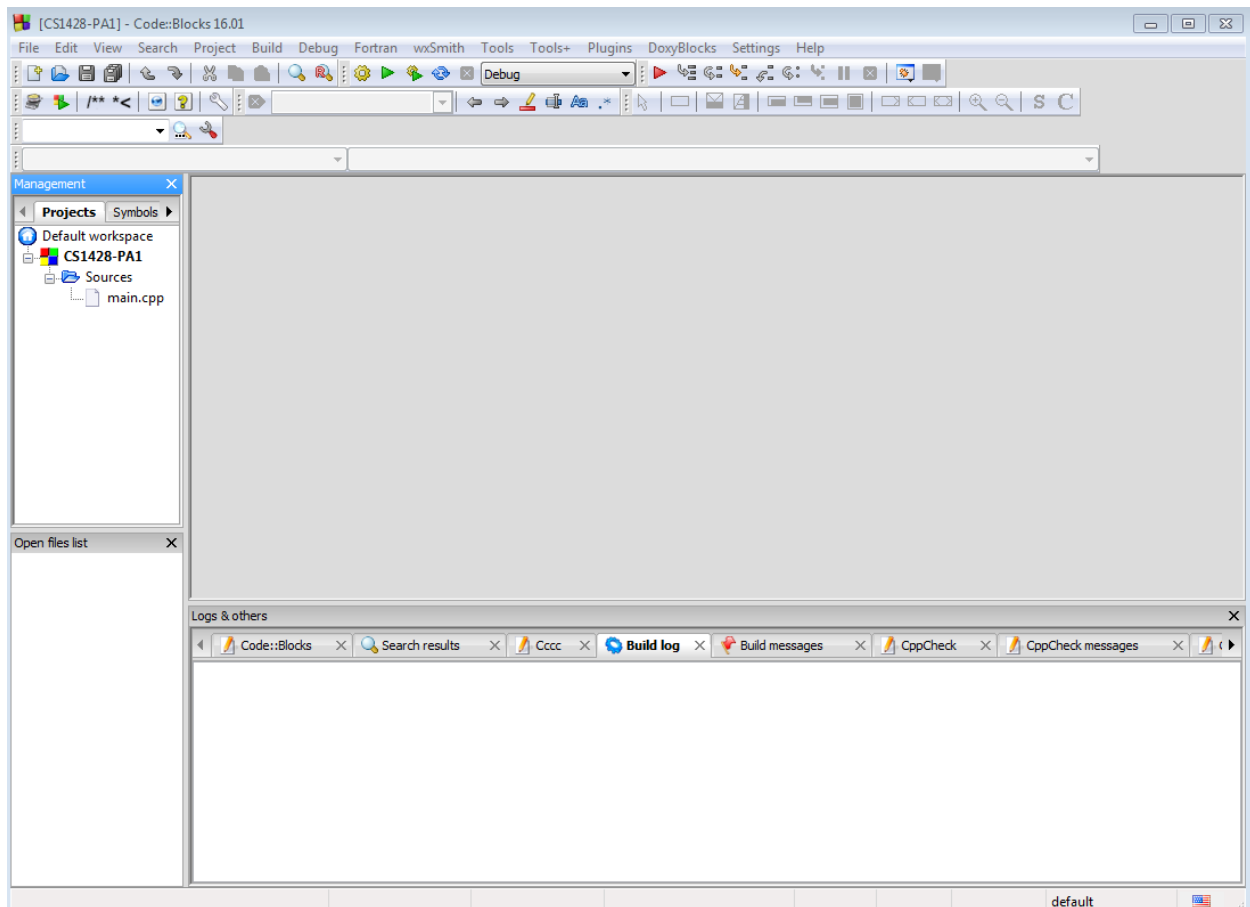
Type in Project title and chose to folder to create the project in then --- > next





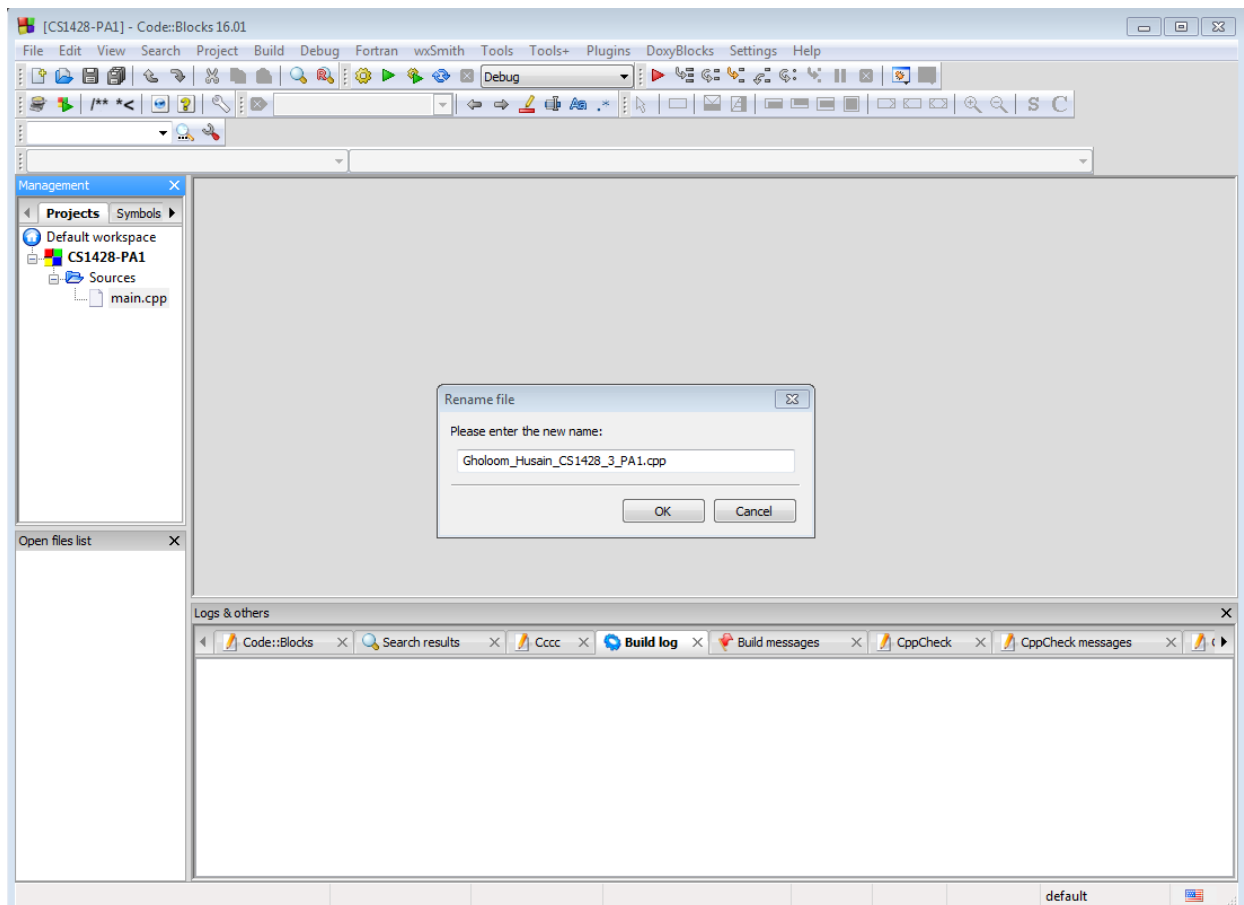
Press on Finish

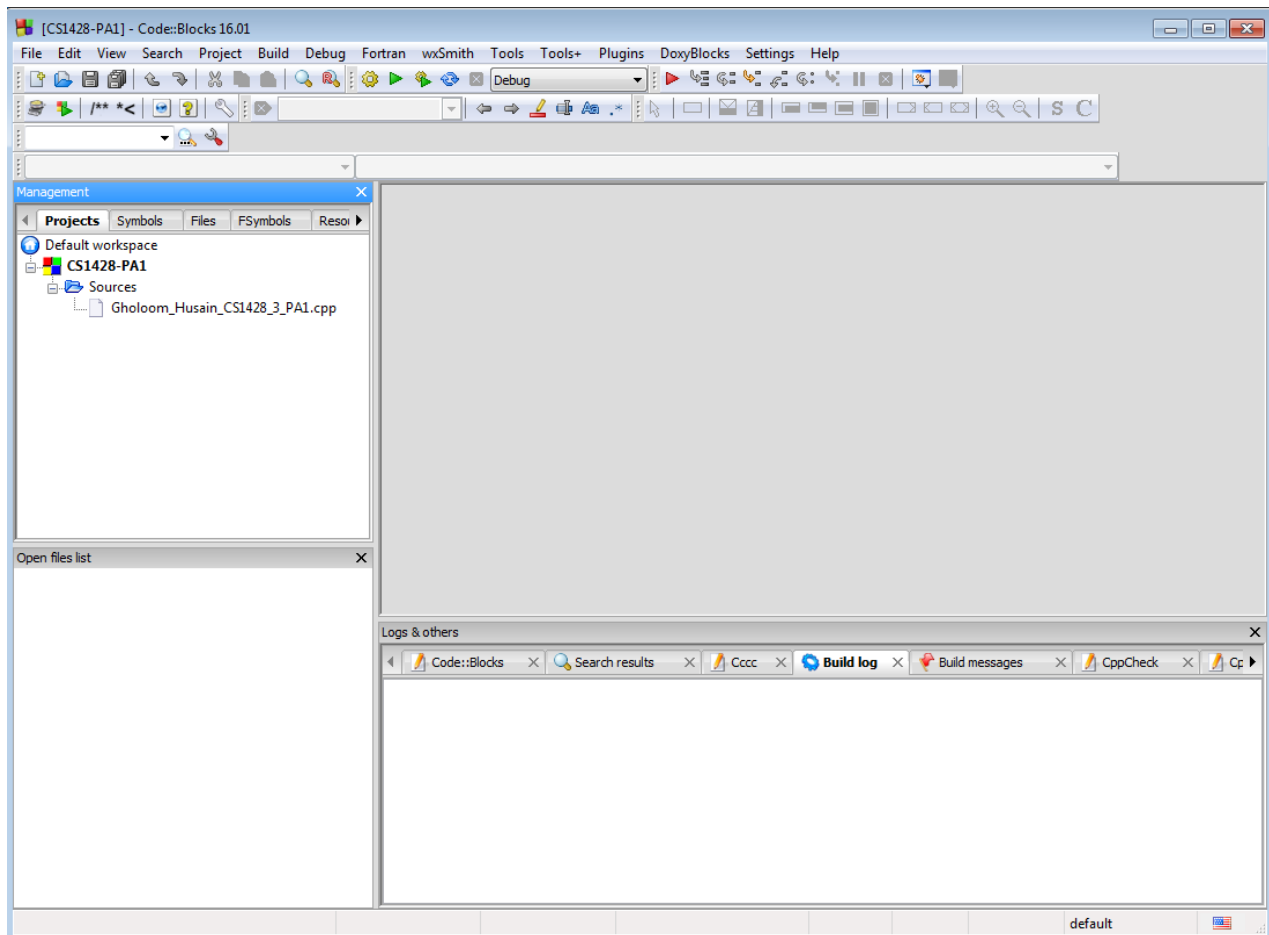
A projects and its main program will be created as it is displayed :



First , make sure that you rename the .cpp file from main.cpp to the name that is given in the assignment then click ok. High light main.cpp and change it be

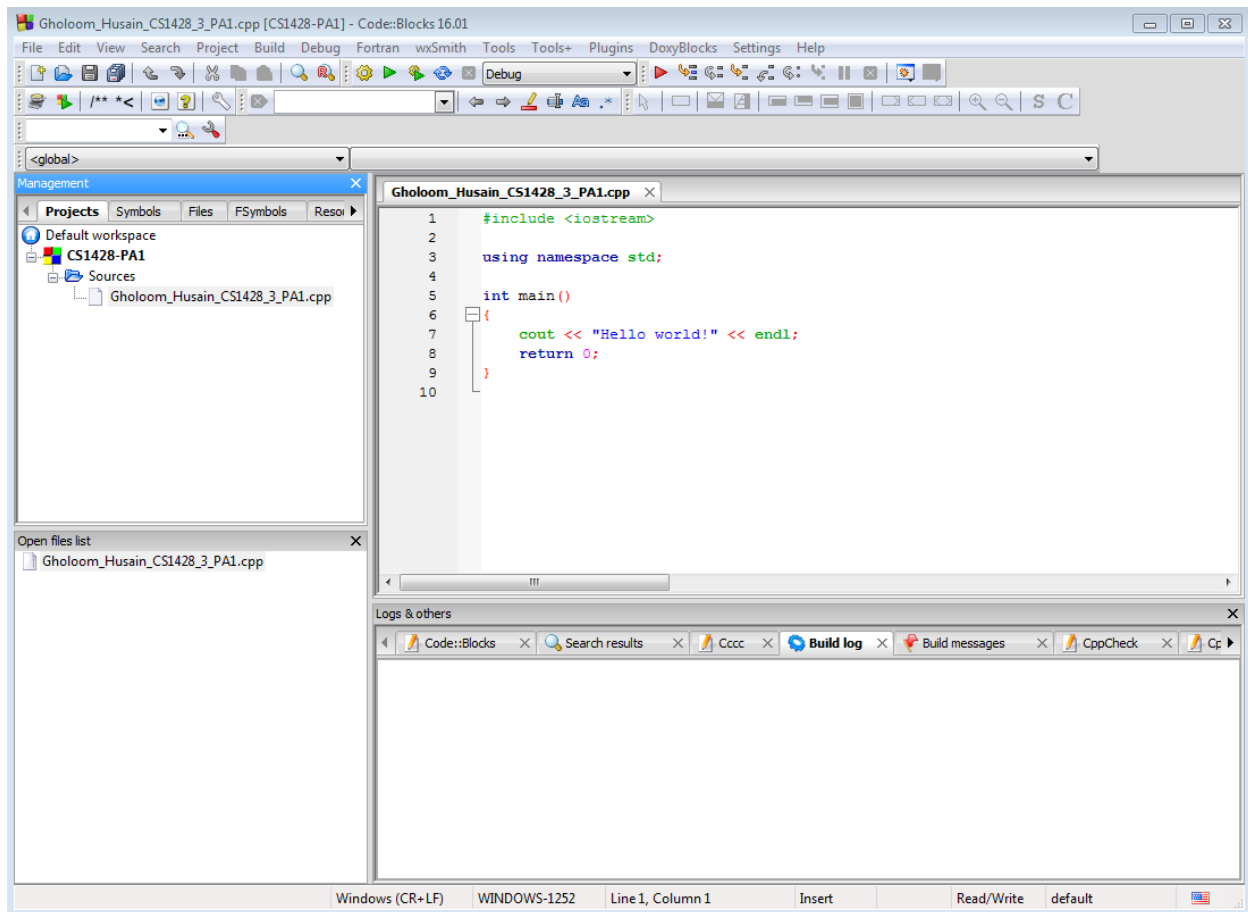
LastName_FirstName_CS1428_PA1.cpp



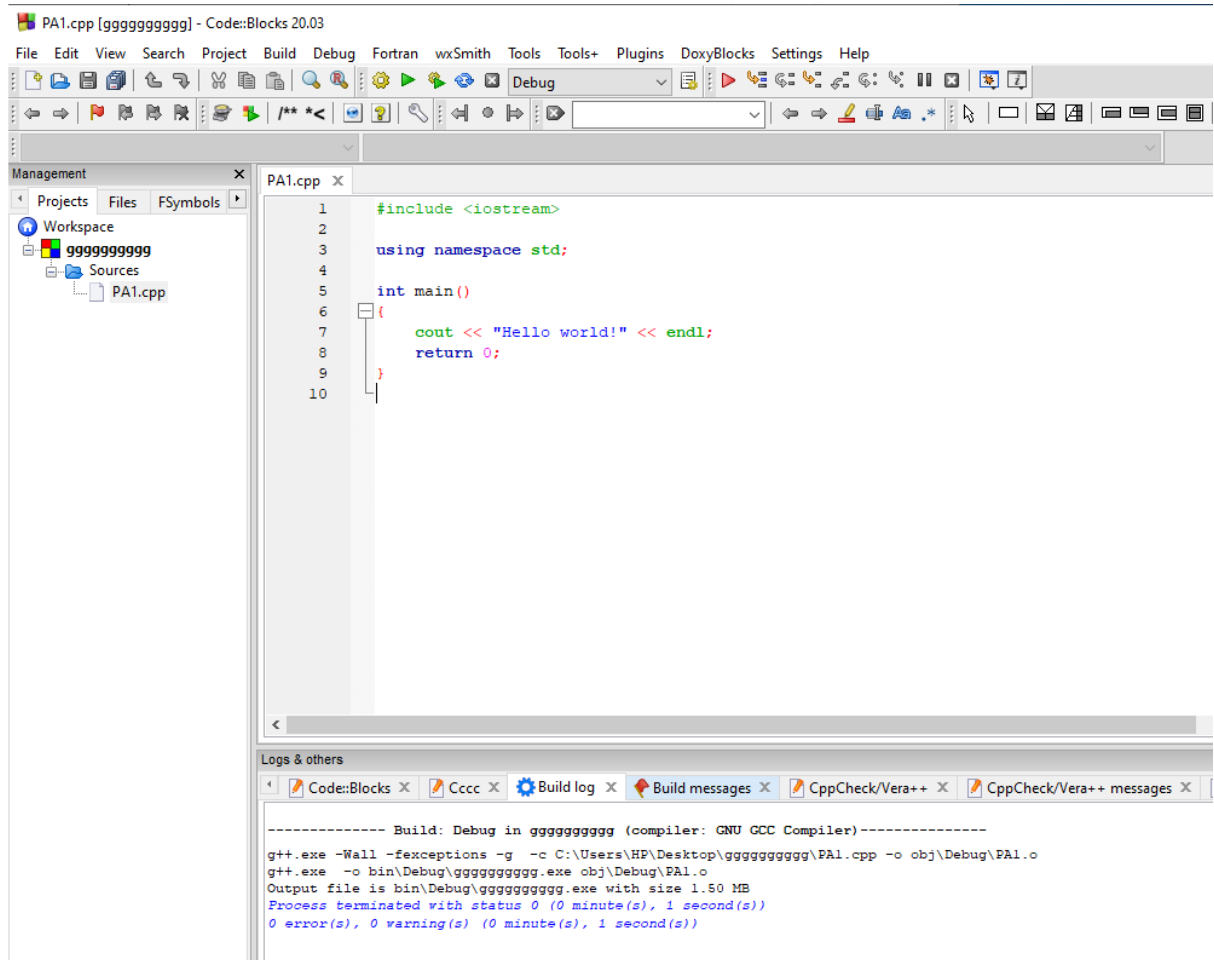


Press the filename.cpp

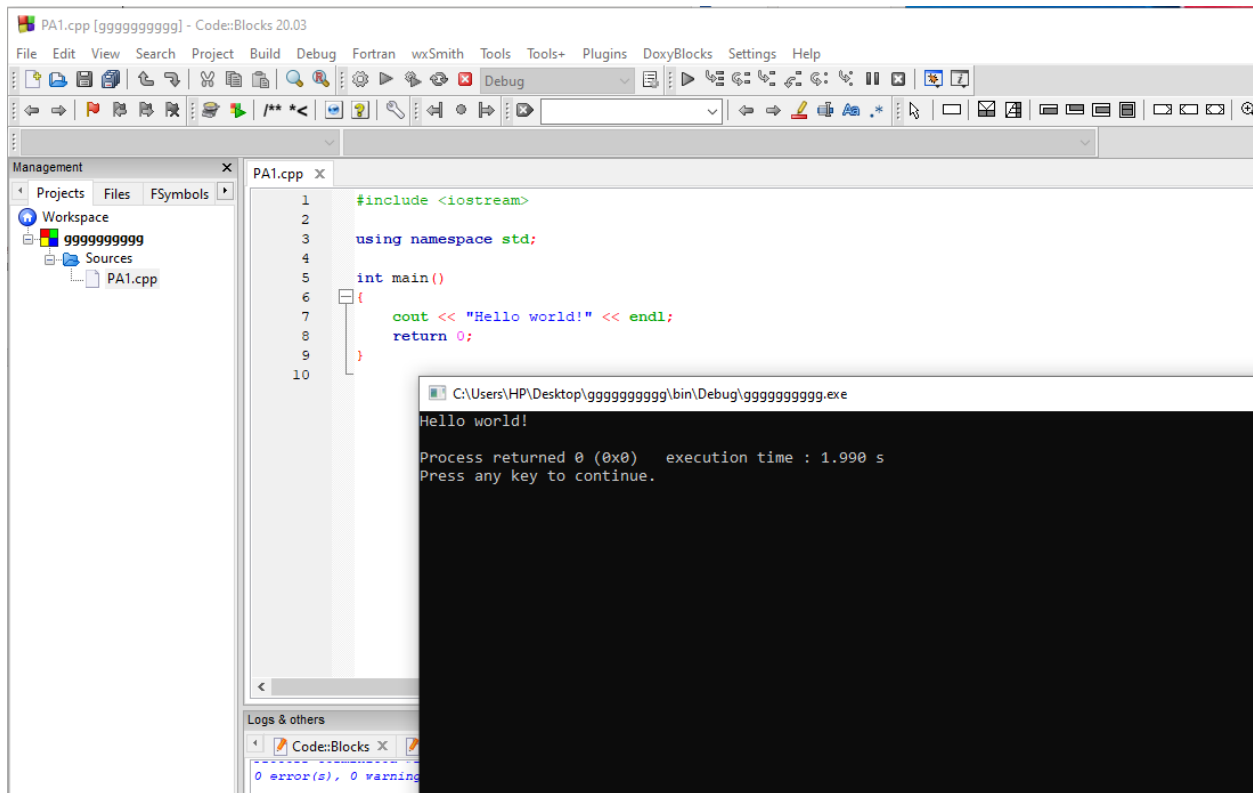
The following final Screen will be displayed:



Select Build:



Fix all syntax errors if any then re-build.

Select Run:

Fix all logical errors if any then re-build then re-run