Getting Started with C Programming (Part 2): My First C Program

icodemag.com/getting-started-with-c-programming-part-2-my-first-c-program/

Bhupendra Singh

April 30, 2019

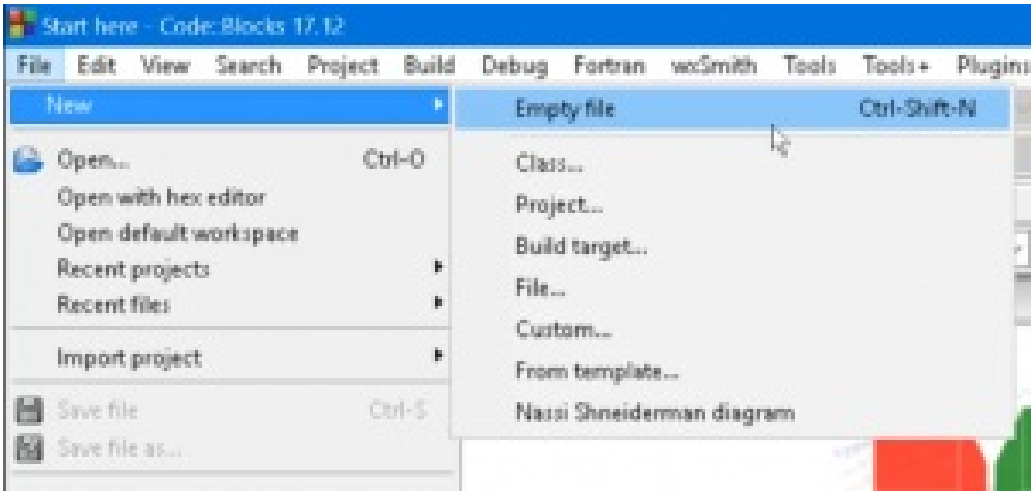
In the part 1 of this series, several key factors were addressed. In this part 2 of the series, practical stuff will be addressed. Before making the first program, we need to set up the programming environment.

Installing an IDE

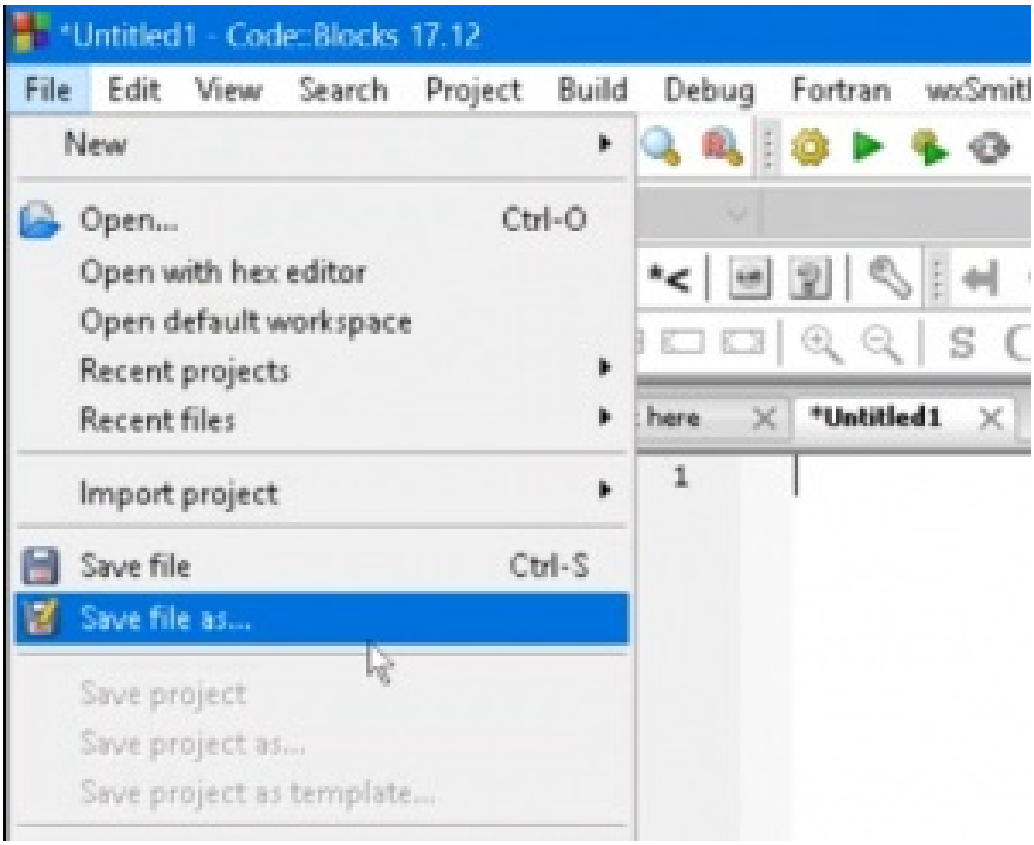
IDE's are explained in the part 1 of this series. Please refer to part 1 to have a clearer understanding of an IDE. Here, for C programming, we are going to use Code::Blocks IDE. Code::Blocks is a free, open-source (licensed under GNU GPLv3) and cross-platform IDE. In order to install, visit official website's download page. Select "**Download the binary release**" option and then download the setup package according to the platform. Once the setup file has been downloaded, install it selecting the default options during installation. In this tutorial, the Windows environment is being used.

Making a .c File

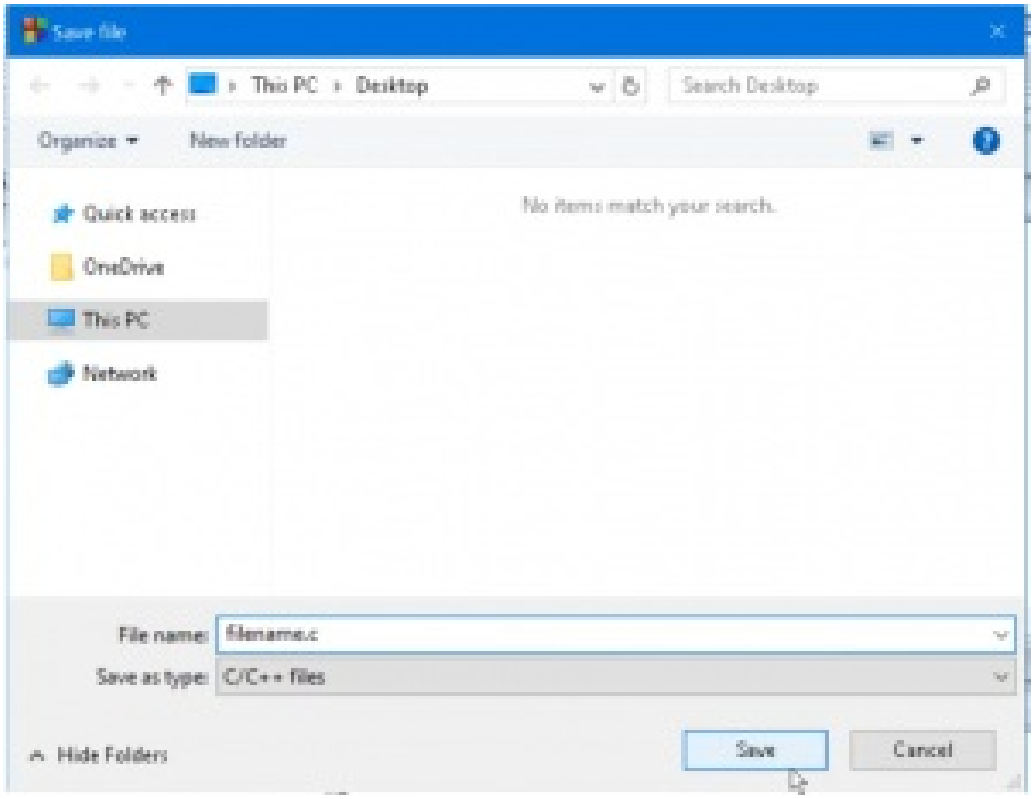
Programming files written in C language end with a .c extension. In order to make one in Code::Blocks, go to : **File -> New -> Empty File** .



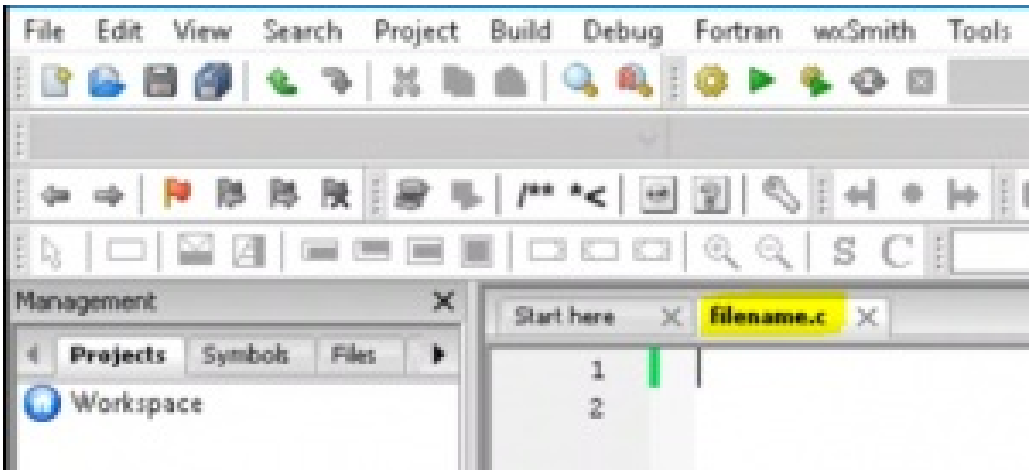
Since the newly created file doesn't have a name and extension, in order to give them to it, go to **File -> Save file as...**.



Enter the file name and select the location where the file should be created. Also, the file type should be selected as "**C/C++ files**".



Once the steps above are completed, the file "**fileName.c**" should be viewed in the current working tab.



Hello World in C Programming Language

In the area provided to type the code, paste the below code there.

```
1  #include <stdio.h>
2  int main(){
3  printf("Hello World");
4  return 0;
5  }
```

Build and Run the Code

Once the above steps are finished, it's now time to run the code and see the output. In order to do so, go to **Build -> Build and run**. After doing so, "**Hello World**" should appear in a command prompt (windows) screen.

Explanation

The above lines of code cannot be explained in one go. Every keyword, component, part etc. has its explanation on its own. But still, let's break everything and understand what it actually means:

Every line of code is called as a statement.

#include <stdio.h>

1. Statements starting with "#" are called as pre-processor directives. These statements are executed before the processing of the actual program. They instruct to include the required files for further process, define macros etc.
2. stdio.h : This is a header file. Header files have the extension .h . Here, stdio stands for standard input-output. The prototype of printf() [a standard output function] is declared in stdio.h .

int main(){ }

1. main() is the only function which gets executed at first. If there is no main, there is no working of program at all.
2. "int" is the return type of main. Every function can either return a value or not.
3. ( ) : Every function name ends with the parentheses. These parenthesis can also contain arguments in it.
4. { } : Every statement written under these curly braces belongs to that function (here: main()).

printf("Hello World");

1. printf() is also a function which is called within the main() function.
2. The definition of printf() is defined in the inbuilt libraries.
3. The string "Hello World" is its arguments

return 0;

1. Functions can also return a value. The returned value is of the type mentioned in the function header.
2. Here, since main() has to return an int type value, 0 is returned. It means that our program has been run successfully and return terminates the main function.