

# C++: A Basic Introduction

Ben Kettle

10 Jan 2020

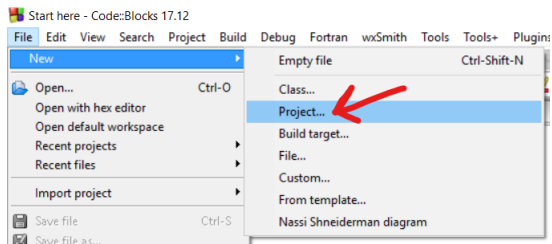
# What is Programming?

Programming allows us to tell the computer what to do. Instead of relying on software that performs certain functions, we can use programming to make the computer do exactly what we want.

# Getting Started

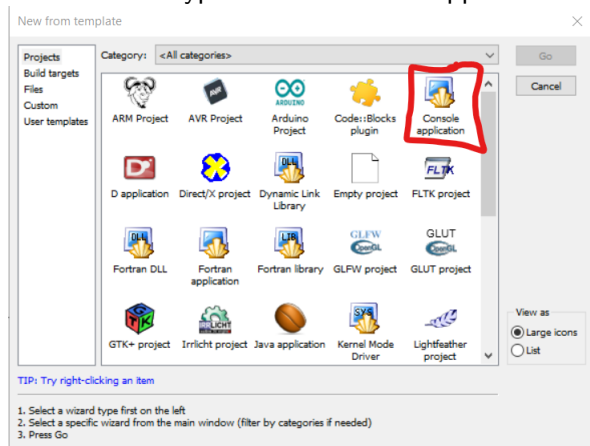
1. Open CodeBlocks 

# Create a new File



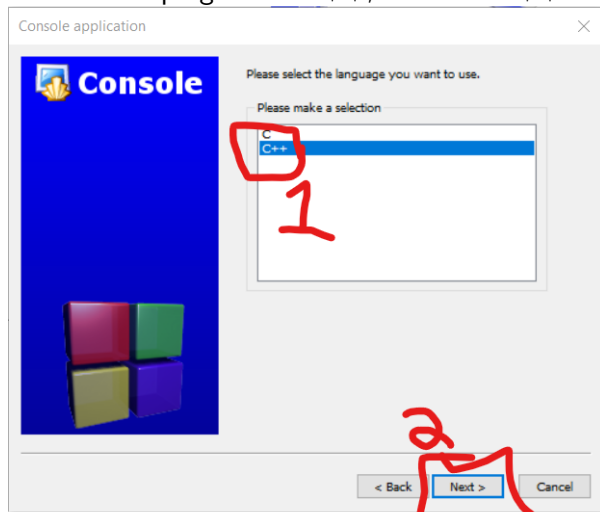
# Create a New File

We want the type to be "Console Application".



# Create the file


We want to program in C++, so click "C++" then "Next".



# Name Project

We now need to choose what to name the project. Type "hello" (or whatever you want) in the "project title" box. Then, click the "..."

Console application

 **Console**

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

Project title:  
hello

Folder to create project in:  
...

Project filename:  
hello.cbp

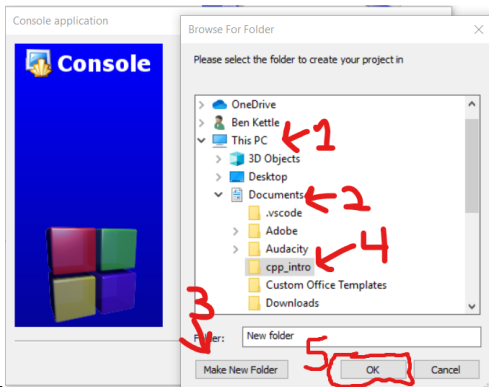
Resulting filename:  
<invalid path>

< Back Next > Cancel

# Place File

We now need to choose where to store the project files. We'll put them in a folder in My Documents, so click on "Documents" and create a new folder called

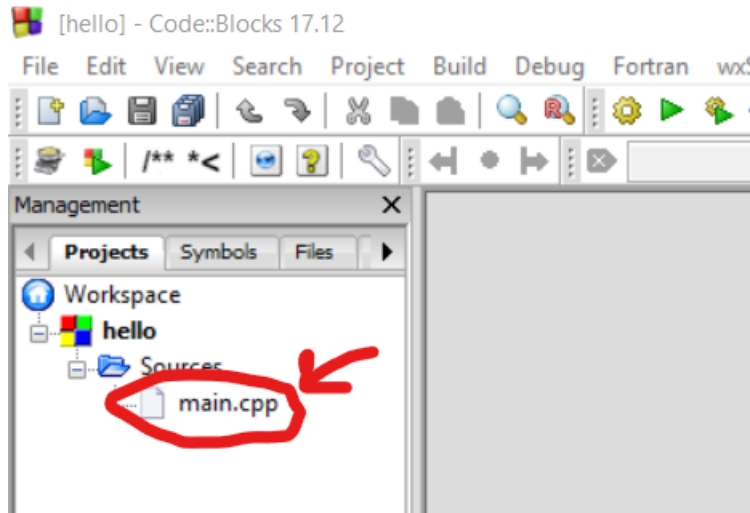
"cpp;ntro". Then, click OK.





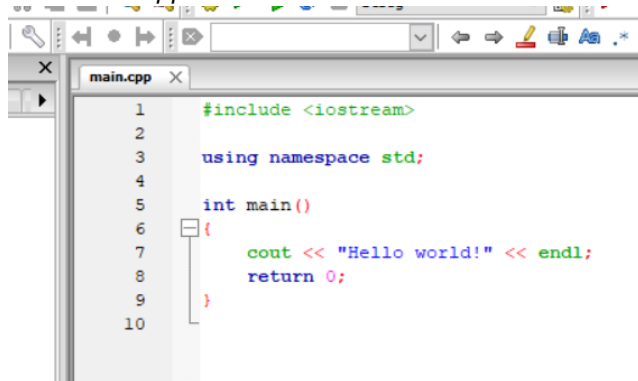
## Open main.cpp

Now we are ready to start programming! Expand the "source" section on the left side of the screen and open "main.cpp" by double clicking.



# We have code!

Your *main.cpp* file should look like this:

A screenshot of a code editor window titled 'main.cpp'. The editor shows a C++ program with the following code:

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

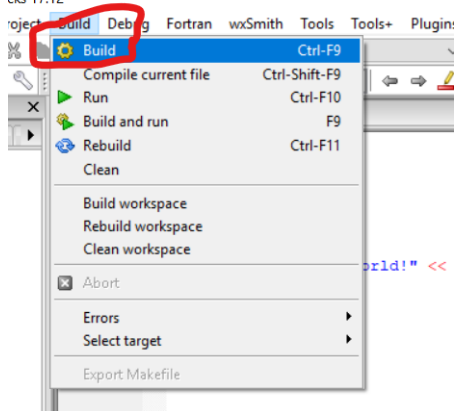
The code is color-coded: keywords like 'include', 'using', 'int', 'return', and 'cout' are in blue, string literals are in green, and operators and punctuation are in red. The editor has a toolbar at the top with various icons for file operations, editing, and running. A line number margin is visible on the left side of the code area.

We'll have the chance to change this soon, but for now we'll leave it as it is.

# Build the file

In order to run our code, we first need to **build** it. This translates your program from code that is easily readable to humans into **assembly** code that the computer can understand. To do this, click "Build" under the "Build" menu.

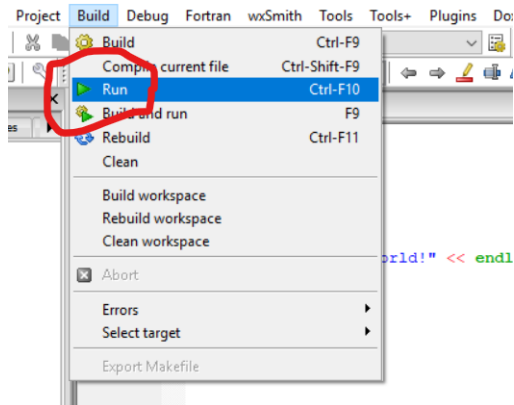
cks 17.12



# Run the file

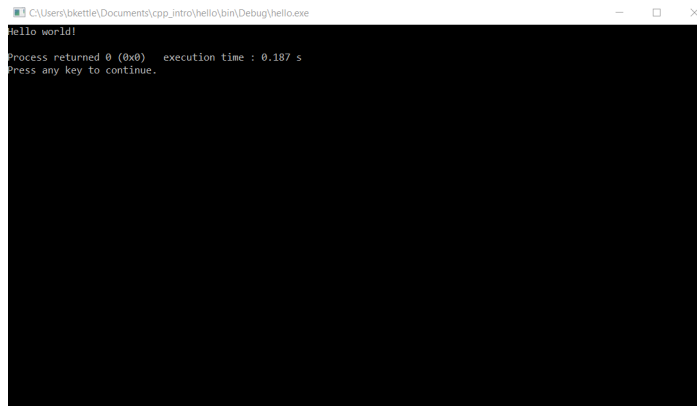
Once your program finishes building, we can run it! To do this, click "run" under the "build" menu.

Blocks 17.12



# Your first program!

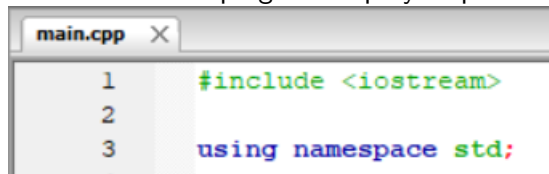
Once your program runs, your screen should look like this! This may not look like much, but we'll be able to do much more exciting things soon.



```
C:\Users\bkettle\Documents\cpp_intro\hello\bin\Debug\hello.exe
Hello world!
Process returned 0 (0x0)   execution time : 0.187 s
Press any key to continue.
```

# What's going on here?

Sure, we made it work, but *why*?  
Let's look at the program step by step.



```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
```

For now, don't worry about these too much. The first line is simply allowing us to get inputs and outputs from the user. The second line is allowing us to use all of the handy functions that C++ comes with.

# What's going on here?

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

Every C++ application must have a `main()` section. This is the part of code that is run first. The `{` and `}` symbols define what is inside this section.

The next line is what is doing most of the work here. `cout` prints to the user, and the `<<` here is writing "Hello world!" to the user. The `<< endl` is then ending the line, so that the next thing printed will be on the next line. Notice the semicolons at the end of the line: in C++, lines must end with semicolons. You'll learn more about functions soon, but basically this `"main()"` section has a value attached to it called a "return value". When a function has finished, it sets this value, and that is what the `"return 0"` line is doing.

# Try it!

So, we're supposed to make computers do things we want them to, right? Change the text inside the quotation marks to have your program print something else, then Build and Run it as we did before.



# Variables

Printing is very useful, but doesn't allow us to do much on its own. **Variables** are very important in any programming language, and allow us to store values for later.

For now, delete everything inside the `main()` section. We're going to create a variable, called `x`. Every variable in C++ must have a **type**. In this case, we want the variable to store numbers—specifically, integers. For this, we use the `"int"` type. On the first line after the `{` following `main`, type `"int x;"` and hit enter. Here, we're **declaring** a new variable called `x`, and telling the computer that it will be used to store integers.

# Variables

Printing is very useful, but doesn't allow us to do much on its own. **Variables** are very important in any programming language, and allow us to store values for later.

For now, delete everything inside the `main()` section. We're going to create a variable, called `x`. Every variable in C++ must have a **type**. In this case, we want the variable to store numbers—specifically, integers. For this, we use the `"int"` type. On the first line after the `{` following `main`, type `"int x;"` and hit enter. Here, we're **declaring** a new variable called `x`, and telling the computer that it will be used to store integers. Now, let's assign a value to the variable. To do this, we use the `=` sign. On the next line, type `"x=4;"` and press enter again. We've just stored the number 4 in this variable.

# Variables

Printing is very useful, but doesn't allow us to do much on its own. **Variables** are very important in any programming language, and allow us to store values for later.

For now, delete everything inside the `main()` section. We're going to create a variable, called `x`. Every variable in C++ must have a **type**. In this case, we want the variable to store numbers—specifically, integers. For this, we use the `"int"` type. On the first line after the `{` following `main`, type `"int x;"` and hit enter. Here, we're **declaring** a new variable called `x`, and telling the computer that it will be used to store integers. Now, let's assign a value to the variable. To do this, we use the `=` sign. On the next line, type `"x=4;"` and press enter again. We've just stored the number 4 in this variable.

Finally, let's print out the value of that variable! To do this, we'll again write to `cout`. On the line after `"x=4;"`, type `"cout << x << endl;"` and hit enter. Build and run!

# Operations

One of the things we'll make the computer do for us most often is math! So let's have it do  $4 * 4 = 8$ . On the line after the print statement we just wrote, type `"x=x*4;"`. This is **reassigning** `x` to the result of `x * 4`.

# Operations

One of the things we'll make the computer do for us most often is math! So let's have it do  $4 * 4 = 8$ . On the line after the print statement we just wrote, type `"x=x*4;"`. This is **reassigning** `x` to the result of  $x * 4$ .

Then, on the following line, copy the line we wrote to print `x` earlier.

Q: What do you expect to happen when we run?

# Operations

One of the things we'll make the computer do for us most often is math! So let's have it do  $4 * 4 = 8$ . On the line after the print statement we just wrote, type `"x=x*4;"`. This is **reassigning** `x` to the result of  $x * 4$ .

Then, on the following line, copy the line we wrote to print `x` earlier.

Q: What do you expect to happen when we run?

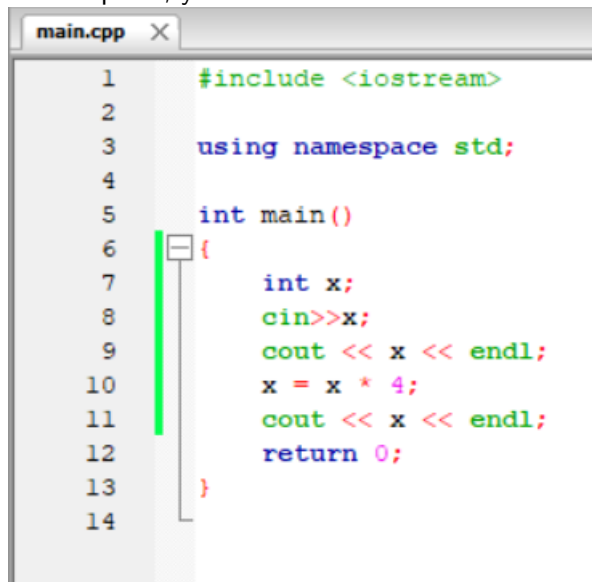
Try it! Build and run your code as before. Try changing your code to do other operation on other numbers! You can add with `+`, subtract with `-`, and divide with `/`.

# Getting Input

It can also be very useful to get input from the user. To do this, we'll do something similar to printing, but in reverse. To try this, we'll make a program that multiplies any number by 4. Remove the line where we assigned `x=4;`, and replace it with `cin >> x;`. Build and run this code! If you type a number into the console and press enter, what do you expect it to print? Does it work?

## Code so Far

At this point, your code should look like this:



```
main.cpp X
1      #include <iostream>
2
3      using namespace std;
4
5      int main()
6      {
7          int x;
8          cin>>x;
9          cout << x << endl;
10         x = x * 4;
11         cout << x << endl;
12         return 0;
13     }
14
```



# Play around

Try combining the things we've done so far to write a cool program.