Get Started With C

To start using C, you need two things:

- A text editor, like Notepad, to write C code
- A compiler, like GCC, to translate the C code into a language that the computer will understand

There are many text editors and compilers to choose from. In this tutorial, we will use an ***IDE*** (see below).

C Install IDE

An IDE (Integrated Development Environment) is used to edit AND compile the code.

Popular IDE's include Code::Blocks, Eclipse, and Visual Studio. These are all free, and they can be used to both edit and debug C code.

Note: Web-based IDE's can work as well, but functionality is limited.

We will use **Code::Blocks** in our tutorial, which we believe is a good place to start.

You can find the latest version of Codeblocks at http://www.codeblocks.org/. Download the `mingw-setup.exe` file, which will install the text editor with a compiler.

C Quickstart

Let's create our first C file.

Open Codeblocks and go to **File > New > Empty File**.

Write the following C code and save the file as `myfirstprogram.c` (**File > Save File as**):

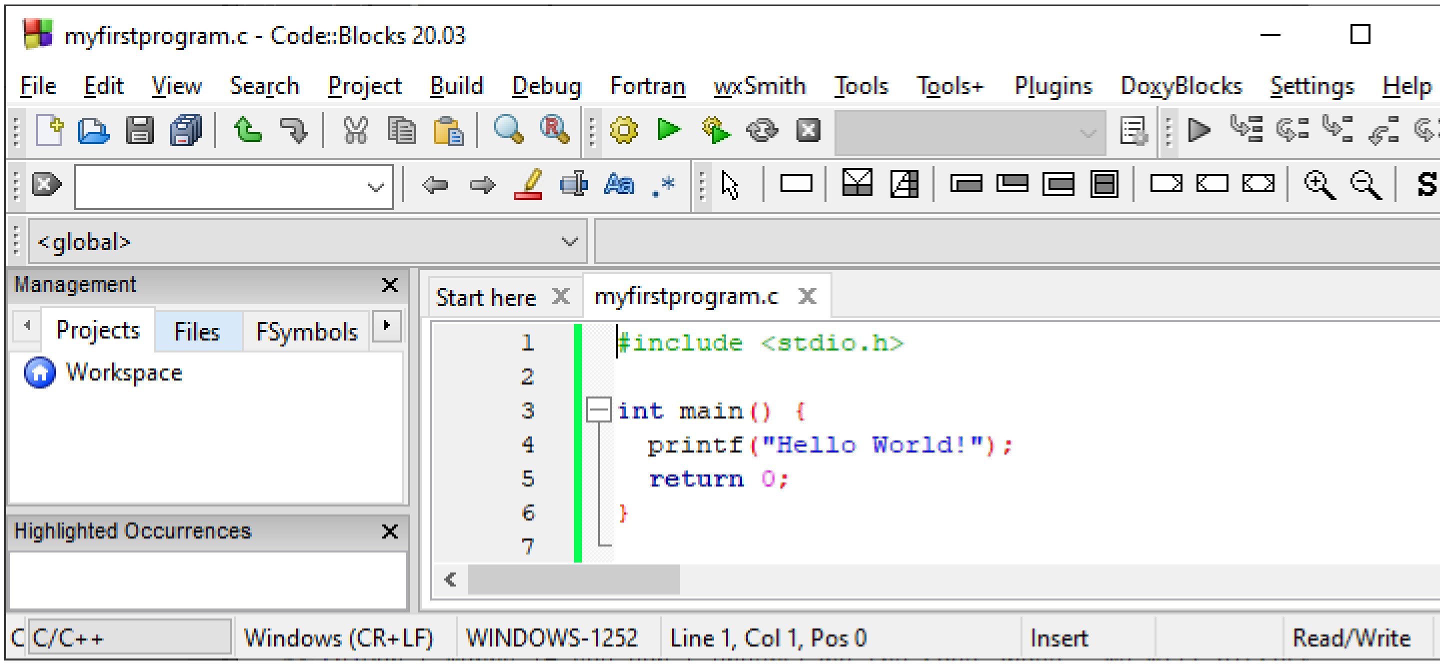
myfirstprogram.c

```
#include <stdio.h>

int main() {
  printf("Hello World!");
  return 0;
}
```

Don't worry if you don't understand the code above - we will discuss it in detail in later chapters. For now, focus on how to run the code.

In Codeblocks, it should look like this:



```
#include <stdio.h>

int main() {
  printf("Hello World!");
  return 0;
}
```

Then, go to **Build > Build and Run** to run (execute) the program. The result will look something to this:

```
Hello World!
Process returned 0 (0x0) execution time : 0.011 s
Press any key to continue.
```

Congratulations! You have now written and executed your first C program.

When learning C you can use our "Try it Yourself" tool, which shows both the code and the result. This will make it easier for you to understand every part as we move forward:

myfirstprogram.c

Code:

```
#include <stdio.h>

int main() {
  printf("Hello World!");
  return 0;
}
```

Result:

```
Hello World!
```

Syntax

You have already seen the following code a couple of times in the first chapters. Let's break it down to understand it better:

Example

```
#include <stdio.h>

int main() {
  printf("Hello World!");
  return 0;
}
```

Try it Yourself »

Example explained

Line 1: `#include <stdio.h>` is a **header file library** that lets us work with input and output functions, such as `printf()` (used in line 4). Header files add functionality to C programs.

Don't worry if you don't understand how `#include <stdio.h>` works. Just think of it as something that (almost) always appears in your program.

Line 2: A blank line. C ignores white space. But we use it to make the code more readable.

Line 3: Another thing that always appear in a C program is `main()`. This is called a **function**. Any code inside its curly brackets `{}` will be executed.

Line 4: `printf()` is a **function** used to output/print text to the screen. In our example, it will output "Hello World!".

Note that: Every C statement ends with a semicolon `;`

Note: The body of `int main()` could also been written as:

```
int main(){printf("Hello World!");return 0;}
```

Remember: The compiler ignores white spaces. However, multiple lines makes the code more readable.

Line 5: `return 0` ends the `main()` function.

Line 6: Do not forget to add the closing curly bracket `}` to actually end the main function.

C Exercises

Exercise:

Insert the missing part of the code below to output "Hello World!":

```
int  () {
```

```
            ("Hello World!");
   return 0;
}
```