What is C?

C is a general-purpose programming language created by Dennis Ritchie at the Bell laboratories in 1972.
It is a very popular language, despite being old.

C is strongly associated with UNIX, as it was developed to write the UNIX operating system.

Why learn C?

- It is one of the most popular programming languages in the world.
- If we know C, we will have no problem learning other programming languages such as Java, Python, C++, C# etc as the syntax is similar.
- C is very fast, compared to other programming languages like Java & Python.
- C is very versatile, it can be used in both technologies & applications.

Difference between c & C++

- C++ was developed as an extension of C, & both languages have almost same syntax.
- The main difference between C & C++ is that C++ support classes & objects, while C does not.

Get Started With C

To start using C, we will 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 compiler 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.
We 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):

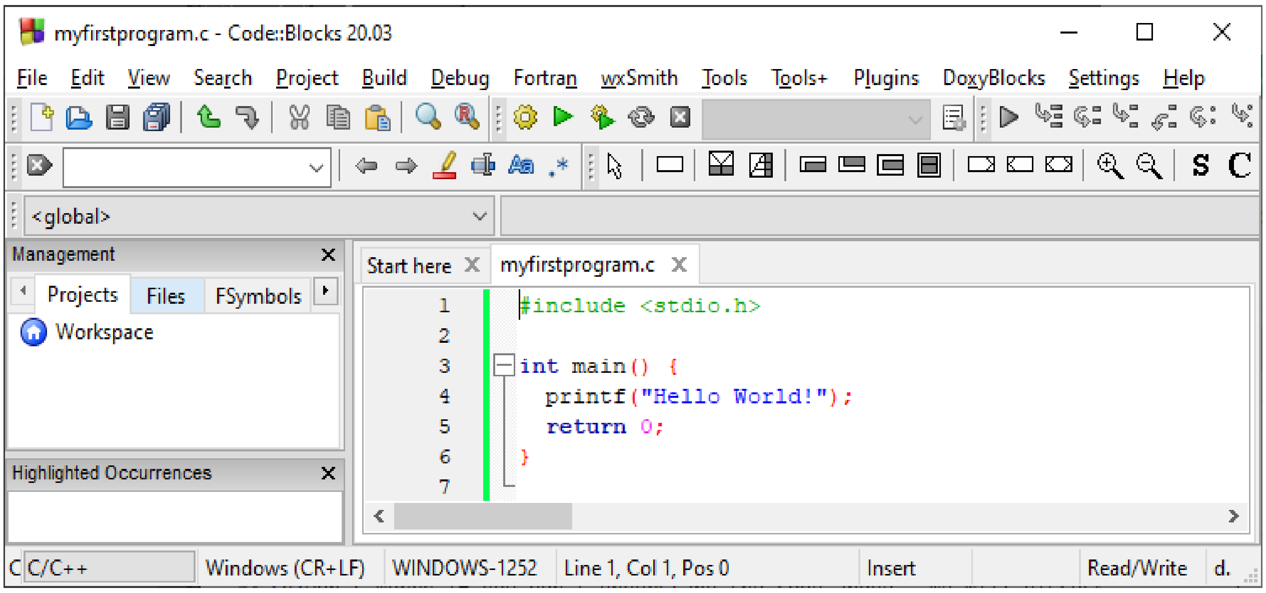
myfirstprogram.c

```
#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;
}
```

myfirstprogram.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FSymbols
Workspace
Highlighted Occurrences
Start here
myfirstprogram.c
C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write d.

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.
```

C Syntax

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

Example

```c
#include <stdio.h>

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

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.

Note: 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.

Output (Print Text)

To output values or print text in C, we can use the printf() function

Example

```
#include <stdio.h>

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

We can use as many printf() functions as you want. However, note that it does not insert a new line at the end of the output:

Example

```
#include <stdio.h>

int main() {
  printf("Hello World!");
  printf("I am learning C.");
  return 0;
}
```

New Lines

To insert a new line, we can use the \n character:

Example

```
#include <stdio.h>

int main() {
  printf("Hello World!\n");
  printf("I am learning C.");
  return 0;
}
```

We can also output multiple lines with a single printf() function. However, this could make the code harder to read:

Example

```
#include <stdio.h>

int main() {
  printf("Hello World!\nI am learning C.\nAnd it is awesome!");
  return 0;
}
```

Tip: Two \n characters after each other will create a blank line:

Example

```
#include <stdio.h>

int main() {
  printf("Hello World!\n\n");
  printf("I am learning C.");
  return 0;
}
```

What is \n exactly?

The newline character (\n) is called an **escape sequence**, and it forces the cursor to change its position to the beginning of the next line on the screen. This results in a new line.

Examples of other valid escape sequences are:

Escape Sequence	**Description**
\t	Creates a horizontal tab
\\	Inserts a backslash character (\)
\”	Inserts a double quote character

Comments in C

Comments can be used to explain code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

Comments can be **singled-lined** or **multi-lined**.

Single-line Comments

Single-line comments start with two forward slashes (//).
Any text between // and the end of the line is ignored by the compiler (will not be executed).
This example uses a single-line comment before a line of code:

Example
```
printf("Hello World!"); // This is a comment
```

C Multi-Line Comments

Multi-line comments start with /* and ends with */.
Any text between /* and */ will be ignored by the compiler:

Example
```
/* The code below will print the words Hello World!
to the screen, and it is amazing */
printf("Hello World!");
```

Single or multi-line comments?

It is up to us which we want to use. Normally, we use // for short comments, and /* */ for longer.

Good to know: Before version **C99** (released in 1999), we could only use multi-line comments in C.

C Variables

Variables are containers for storing data values, like numbers and characters. In C, there are different types of variables (defined with different keywords), for example:

- Int - stores integers (whole numbers), without decimals, such as 123 or -123.
- Float - stores floating point numbers, with decimals, such as 19.99 or -19.99
- Char - stores single characters, such as 'a' or 'B'. Char values are surrounded by **single quotes**

Declaring (Creating) Variables

To create a variable, specify the type and assign it a value:

Syntax

```
type variableName = value;
```

Where type is one of C types (such as int), and variableName is the name of the variable (such as **x** or **myName**). The **equal sign** is used to assign a value to the variable.

So, to create a variable that should **store a number**, look at the following example:

Example

Create a variable called myNum of type int and assign the value 15 to it:

```
int myNum = 15;
```

We can also declare a variable without assigning the value, and assign the value later:

Example

```
// Declare a variable
int myNum;

// Assign a value to the variable
myNum = 15;
```

Output Variables

We learned from the output chapter that you can output values/print text with the printf() function:

Example

```
printf("Hello World!");
```

In many other programming languages (like Python, Java, and C++), we would normally use a print function to display the value of a variable. However, this is not possible in C:

Example

```
int myNum = 15;
printf(myNum);  // Nothing happens
```

To output variables in C, we must get familiar with something called "format specifiers".

Format Specifiers

Format specifiers are used together with the printf() function to tell the compiler what type of data the variable is storing. It is basically a placeholder for the variable value.

A format specifier starts with a percentage sign %, followed by a character.

For example, to output the value of an int variable, we must use the format specifier %d or %i surrounded by double quotes, inside the printf() function:

Example

```
int myNum = 15;
printf("%d", myNum);  // Outputs 15
```

To print other types, use %c for char and %f for float:

Example

```
// Create variables
int myNum = 15;          // Integer (whole number)
float myFloatNum = 5.99;   // Floating point number
char myLetter = 'D';      // Character

// Print variables
printf("%d\n", myNum);
printf("%f\n", myFloatNum);
printf("%c\n", myLetter);
```

To combine both text and a variable, separate them with a comma inside the printf() function:

Example

```
int myNum = 15;
printf("My favorite number is: %d", myNum);
```

To print different types in a single printf() function, you can use the following:

Example

```
int myNum = 15;
char myLetter = 'D';
printf("My number is %d and my letter is %c", myNum, myLetter);
```

Change Variable Names

Note: If we assign a new value to an existing variable, it will overwrite the previous value:

Example

```
int myNum = 15;  // myNum is 15
myNum = 10;  // Now myNum is 10
```

We can also assign the value of one variable to another:

Example

```
int myNum = 15;

int myOtherNum = 23;

// Assign the value of myOtherNum (23) to myNum
myNum = myOtherNum;

// myNum is now 23, instead of 15
printf("%d", myNum);
```