Chapter 1: Getting started with C Language

Version	Standard	Publication Date
K&R	n/a	1978-02-22
C89	ANSI X3.159-1989	1989-12-14
C90	ISO/IEC 9899:1990	1990-12-20
C95	ISO/IEC 9899/AMD1:1995	1995-03-30
C99	ISO/IEC 9899:1999	1999-12-16
C11	ISO/IEC 9899:2011	2011-12-15

Section 1.1: Hello World

To create a simple C program which prints *"Hello, World"* on the screen, use a text editor to create a new file (e.g. `hello.c` — the file extension must be `.c`) containing the following source code:

hello.c

```
#include <stdio.h>

int main(void)
{
    puts("Hello, World");
    return 0;
}
```

Live demo on Coliru

Let's look at this simple program line by line

```
#include <stdio.h>
```

This line tells the compiler to include the contents of the standard library header file `stdio.h` in the program. Headers are usually files containing function declarations, macros and data types, and you must include the header file before you use them. This line includes `stdio.h` so it can call the function `puts()`.

See more about headers.

```
int main(void)
```

This line starts the definition of a function. It states the name of the function (`main`), the type and number of arguments it expects (`void`, meaning none), and the type of value that this function returns (`int`). Program execution starts in the `main()` function.

```
{
    ...
}
```

The curly braces are used in pairs to indicate where a block of code begins and ends. They can be used in a lot of ways, but in this case they indicate where the function begins and ends.

```
    puts("Hello, World");
```

This line calls the `puts()` function to output text to standard output (the screen, by default), followed by a newline.

The string to be output is included within the parentheses.

`"Hello, World"` is the string that will be written to the screen. In C, every string literal value must be inside the double quotes `"..."`.

See more about strings.

In C programs, every statement needs to be terminated by a semi-colon (i.e. `;`).

```
    return 0;
```

When we defined `main()`, we declared it as a function returning an `int`, meaning it needs to return an integer. In this example, we are returning the integer value 0, which is used to indicate that the program exited successfully. After the `return 0;` statement, the execution process will terminate.

Editing the program

Simple text editors include `vim` or `gedit` on Linux, or `Notepad` on Windows. Cross-platform editors also include `Visual Studio Code` or `Sublime Text`.

The editor must create plain text files, not RTF or other any other format.

Compiling and running the program

To run the program, this source file (`hello.c`) first needs to be compiled into an executable file (e.g. `hello` on Unix/Linux system or `hello.exe` on Windows). This is done using a compiler for the C language.

See more about compiling

Compile using GCC

GCC (GNU Compiler Collection) is a widely used C compiler. To use it, open a terminal, use the command line to navigate to the source file's location and then run:

```
gcc hello.c -o hello
```

If no errors are found in the the source code (`hello.c`), the compiler will create a **binary file**, the name of which is given by the argument to the `-o` command line option (`hello`). This is the final executable file.

We can also use the warning options `-Wall -Wextra -Werror`, that help to identify problems that can cause the program to fail or produce unexpected results. They are not necessary for this simple program but this is way of adding them:

```
gcc -Wall -Wextra -Werror -o hello hello.c
```

Using the clang compiler

To compile the program using `clang` you can use:

```
clang -Wall -Wextra -Werror -o hello hello.c
```

By design, the `clang` command line options are similar to those of GCC.

Using the Microsoft C compiler from the command line

If using the Microsoft `cl.exe` compiler on a Windows system which supports Visual Studio and if all environment variables are set, this C example may be compiled using the following command which will produce an executable `hello.exe` within the directory the command is executed in (There are warning options such as `/W3` for `cl`, roughly analogous to `-Wall` etc for GCC or clang).

```
cl hello.c
```

Executing the program

Once compiled, the binary file may then be executed by typing `./hello` in the terminal. Upon execution, the compiled program will print `Hello, World`, followed by a newline, to the command prompt.

Section 1.2: Original "Hello, World!" in K&R C

The following is the original "Hello, World!" program from the book The C Programming Language by Brian Kernighan and Dennis Ritchie (Ritchie was the original developer of the C programming language at Bell Labs), referred to as "K&R":

```
Version = K&R
```

```
#include <stdio.h>

main()
{
    printf("hello, world\n");
}
```

Notice that the C programming language was not standardized at the time of writing the first edition of this book (1978), and that this program will probably not compile on most modern compilers unless they are instructed to accept C90 code.

This very first example in the K&R book is now considered poor quality, in part because it lacks an explicit return type for `main()` and in part because it lacks a `return` statement. The 2nd edition of the book was written for the old C89 standard. In C89, the type of `main` would default to `int`, but the K&R example does not return a defined value to the environment. In C99 and later standards, the return type is required, but it is safe to leave out the `return` statement of `main` (and only `main`), because of a special case introduced with C99 5.1.2.2.3 — it is equivalent to returning 0, which indicates success.

The recommended and most portable form of `main` for hosted systems is `int main (void)` when the program does not use any command line arguments, or `int main(int argc, char **argv)` when the program does use the command line arguments.

C90 §5.1.2.2.3 **Program termination**

> A return from the initial call to the `main` function is equivalent to calling the `exit` function with the value returned by the `main` function as its argument. If the `main` function executes a return that specifies no value, the termination status returned to the host environment is undefined.

C90 §6.6.6.4 **The `return` statement**

> If a `return` statement without an expression is executed, and the value of the function call is used by the

> caller, the behavior is undefined. Reaching the } that terminates a function is equivalent to executing a `return` statement without an expression.

C99 §5.1.2.2.3 **Program termination**

> If the return type of the `main` function is a type compatible with `int`, a return from the initial call to the `main` function is equivalent to calling the `exit` function with the value returned by the `main` function as its argument; reaching the } that terminates the `main` function returns a value of 0. If the return type is not compatible with `int`, the termination status returned to the host environment is unspecified.