# Hello, World!

**Introduction**

The C programming language is a general purpose programming language, which relates closely to the way machines work. Understanding how computer memory works is an important aspect of the C programming language. Although C can be considered as "hard to learn", C is in fact a very simple language, with very powerful capabilities.

C is a very common language, and it is the language of many applications such as Windows, the Python interpreter, Git, and many many more.

C is a compiled language - which means that in order to run it, the compiler (for example, GCC or Visual Studio) must take the code that we wrote, process it, and then create an executable file. This file can then be executed, and will do what we intended for the program to do.

**Our first program**

#include <stdio.h>

int main() {

printf("Hello, World!");

return 0;

}

Every C program uses libraries, which give the ability to execute necessary functions. For example, the most basic function called printf, which prints to the screen, is defined in the stdio.h header file.

To add the ability to run the printf command to our program, we must add the following include directive to our first line of the code:

#include <stdio.h>

The second part of the code is the actual code which we are going to write. The first code which will run will always reside in the mainfunction.

int main() {

... our code goes here

}

The int keyword indicates that the function main will return an integer - a simple number. The number which will be returned by the function indicates whether the program that we wrote worked correctly. If we want to say that our code was run successfully, we will return the number 0. A number greater than 0 will mean that the program that we wrote failed.

For this tutorial, we will return 0 to indicate that our program was successful:

return 0;

Notice that every line in C must end with a semicolon, so that the compiler knows that a new line has started.

Last but not least, we will need to call the function printf to print our sentence.

# Variables and Types

### Data types

C has several types of variables, but there are a few basic types:

* Integers - whole numbers which can be either positive or negative. Defined using char, int, short, long or long long.
* Unsigned integers - whole numbers which can only be positive. Defined using unsigned char, unsigned int, unsigned short, unsignedlong or unsigned long long.
* Floating point numbers - real numbers (numbers with fractions). Defined using float and double.
* Structures - will be explained later, in the Structures section.

The different types of variables define their bounds. A char can range only from -128 to 127, whereas a long can range from -2,147,483,648 to 2,147,483,647 (long and other numeric data types may have another range on different computers, for example - from –9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 on 64-bit computer).

Note that C does not have a boolean type. Usually, it is defined using the following notation:

#define BOOL char

#define FALSE 0

#define TRUE 1

**Personal notes:**

**Register variable:**

Not discarded until c++.

Variable stored in a CPU register can always be accessed faster than the one that is stored in memory.

**Advantages of Register variable:**  
- Access optimization and speed of program execution: The operations of these variables are faster by orders of magnitude.  
  
- It is useful when you want to refer a variable frequently.  
  
- It allocates fast memory in the form of a register.  
  
- It helps to speed up program execution.

**Languages introduction:**

All computer languages are designed to communicate with hardware at the end. But programs written in high level languages may go through many steps of translations before being executed. Programs written in C are first converted to an assembly program (designed for the underlying hardware), which then in turn is converted to the machine language, the language understood by the hardware. （There may be many steps in between. ）

Machine language instructions are simple. They typically consist of very simple instructions such as adding two numbers or moving data or jumping from one instruction to another. However, it is of course very difficult to write and debug programs in machine language.

One layer of this program translation is the assembly language. A high level language is translated into assembly language. Each CPU/processor has its own assembly language. Assembly code is then translated into the target machine code. Assembly languages are human readable and contains very simple instructions. （Eventually this assembly code is mapped into the corresponding machine language so that the underlying hardware can carry out the instructions. ）

**Definition of compiler:**

A compiler (such as GCC – GNU C compiler or lately GNU compiler collection) translates a program written in a high-level language to object code that can be interpreted and executed by the underlying system.

**The steps of complier doing:**

Compilers go through multiple levels of processing such as, syntax checking, pre-processing macros and libraries, object code generation, linking, and optimization among many other things.

**Operating system:**

Definition: An operating system is a software program that manages coordination between application programs and underlying hardware. UNIX is an operating system.

What does it do? OS manages devices such as printers, disks, monitors and manage multiple tasks such as processes.

**Data && Program Instructions**

The term "memory" when used with reference to computers generally refers to random-access memory (RAM).

All data and program instructions are stored as sequences of bytes in the memory called Random Access Memory (RAM).

To increase the speed of execution of a program, a compiler may use fast accessed memory locations such as registers and cache memory. There could be 8 registers in the machine with one called the zero register (containing the value zero for initializations).

Example: the architecture of a uni-processor machine that contains a CPU, memory and IO modules. (CPU: PC, IR, MAR, MBR, I/O; memory: Instruction, Data; I/O modules:)

**What are the types of /main memory/?**

There are two main kinds of semiconductor memory: volatile and non-volatile.

Examples of non-volatile memory:

flash memory (used as secondary memory) / ROM / PROM/ EPROM / EEPROM memory (used for storing firmware such as BIOS).

**Difference between RAM && ROM memory:**

Definition of ROM: Once data has been written onto a ROM chip, it cannot be removed and can only be read. Unlike main memory (RAM), ROM retains its contents even when the computer is turned off. ROM is referred to as being non-volatile, whereas RAM is volatile.

Is ROM a main storage? Yes. They make up the primary memory. Although one is volatile and the other contains non-volatile firmware, the similarity is in that they are both directly accessed by the CPU. ROM is NOT secondary/Auxiliary/external storage.

There is one major difference between a read-only memory (ROM) and a random-access memory (RAM) chip: ROM can hold data without power and RAM cannot. Essentially, ROM is meant for permanent storage, and RAM is for temporary storage.

Is RAM or ROM faster? Although the relative speed of RAM vs. ROM has varied over time, large RAM chips can be read faster than most ROMs. For this reason (and to allow uniform access), ROM content is sometimes copied to RAM or shadowed before its first use, and subsequently read from RAM.

**MAR, MBR, MDR**

Memory address register (MAR). the Memory Address Register (MAR) is the CPU register that either stores the memory address from which data will be fetched from the CPU, or the address to which data will be sent and stored. In other words, MAR holds the memory location of data that needs to be accessed.

A memory buffer register (MBR) or memory data register (MDR) is the register in a computer's processor, or central processing unit, CPU, that stores the data being transferred to and from the immediate access storage. It contains the copy of designated memory locations specified by the memory address register.

**The difference between register and memory:**

The primary difference between register and memory is that register holds the data that the CPU is currently processing whereas, the memory holds the data the that will be required for processing. ... On the other hands, memory is referred as the main memory of the computer which is RAM.

**Register of Computer:**

A processor register (CPU register) is one of a small set of data holding places that are part of the computer processor. A register may hold an instruction, a storage address, or any kind of data (such as a bit sequence or individual characters). Some instructions specify registers as part of the instruction.

**Different types of registers:**

There are various types of Registers those are used for various purpose. Some Mostly used Registers are Accumulator (AC), Data Register(DR), Address Register(AR), Program Counter(PC), Memory Data Register (MDR), Index Register(IR), Memory Buffer Register(MBR).

**The difference between primary (main) storage && secondary storage:**

Primary memory is the main memory (Hard disk, RAM) where the operating system resides. Secondary memory can be external devices like CD, floppy magnetic discs etc. secondary storage cannot be directly accessed by the CPU and is also external memory storage.

The key **difference between primary and secondary memory** is that **primary memory** can be directly accessed by the CPU whereas, the CPU cannot directly access the **secondary memory**. The **primary memory** of the computer is also known as the main **memory** of the computer. However, **secondary memory** is known as auxiliary **memory**.

**The difference between cache && buffer:**

A buffer is a region of memory used to temporarily hold data while it is being moved from one place to another within a computer, while a cache is a temporary storage area where frequently accessed data can be stored for rapid access.

**How much faster is cache than RAM:**

The memory is needed since the CPU is getting faster than RAM is, so the cache is much faster the RAM. The CPU cache is also known as SRAM - again it's much faster - but it takes up more space. The cache is used for storing CPU instructions and such whiles RAM stores the less frequently accessed data.

**Bytes, Bits:**

A bit is the smallest unit of storage represented by 0 or 1.

A byte is typically 8 bits. C character data type requires one byte of storage.

A file is a sequence of bytes. A size of the file is the number of bytes within the file.

Each data byte can be represented using an ASCII (or extended ASCII) value.

Standard ASCII table assigns each character to a numerical value.