C PROGRAMMING

Chapter 01 - Introduction

- ✓ C is a powerful **general-purpose** programming language that was initially developed to **rewrite the UNIX operating system**.
- ✓ Apart from that, modern web browsers like Google's Chrome and Firefox, database management systems like MySQL and hundreds of other applications use C.
- ✓ **Basic Structure** of a C code:

- ✓ Things to remember about **printf():**
  - Everything required for the printing operation is kept inside **parentheses** ()
  - The text to be printed is **enclosed within double quotes** " "
  - Each printf statement **ends with a semicolon** ;
- ✓ **Not following** these rules **will result in errors** and your code will **not run successfully**.
- ✓ In C programming, the code inside the curly braces `{}` are called **statements**.
- ✓ Statements are the **fundamental building blocks of a program**.
- ✓ In C, all statements **need to be terminated** with a semicolon `;`.
- ✓ Comments in C language are given by `//` .

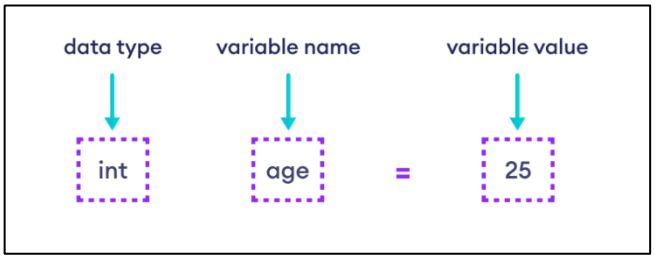
- ✓ All the fixed data (values) that we can use directly in our program are called **literals**.
- ✓ Integer literals are positive and negative numbers without decimal, such as, 0, 25, -25, 1234 etc…
- ✓ Floating-Point literals are positive and negative numbers with decimal, such as, 30.213, -25.0, 9.0, 1234.1234 etc…
- ✓ Character literals consist of a **single character** that is enclosed within **single quotation marks** ' ' ; For example, 's', '}', 'N', '*', '5', '8', etc.
- ✓ Strings are **text / group of characters** that are enclosed within **double quotation marks** " " ; For example, "Hello" "comp@c+", "100 degrees", "a", "5", "88.8", "*", etc.
- ✓ The following are all **invalid characters** in C**:**
  - '58'
  - '58.95'
  - '-58.95'
  - 'abc'
- ✓ The variables are **containers** to store data like numbers and characters.
- ✓ Syntax for declaring a variable in C :

```
data_type  variable_name;
```

,for example, **int num1;** is a declaration of a variable named **num1** of the datatype **integer**.

- ✓ Once a variable is declared, we can assign value to it, either in a separate statement or in the same statement. For example,

```
int age = 25;
```



- ✓ Format specifiers are **placeholders** that will be replaced by the value of the variable. (%d - integer, %c - character, %f – float, %lf – double, %.2f / %.2lf – with specified decimal places, %zu – sizeof() etc..)
- ✓ In C programming, we can use the **letters f or F** at the end of a **floating-point** number to **signify** that it's a float.
- ✓ Some **operation**s using variables:
    - Print multiple variables
    - Change the value of a variable
    - Assign the value of one variable to another
    - Select proper variable names
    - Create multiple variables in a single statement
    - Use double type variables
- ✓ When we **change** the value of a variable, the type of value and variable must match. For example, age is an int variable, so **we cannot store** floating-point numbers to it.
- ✓ In assigning the value of one variable to another, for example, number1 = number2, the **value of the number2 variable is assigned to the number1 variable**.
- ✓ A good variable name has the following features:
    - A variable name should be **clear** and **concise**.
    - If the name consists of **two or more words**, use **camelCase** formatting to separate them.
- ✓ Illegal variable names: You **cannot** use spaces or other symbols (except alphabets, numbers and underscore) as variable names. Also, you **cannot start** your variable name with a **number**.
- ✓ Floating point numbers have approximately **7** decimal digits of precision. For some scientific and financial calculations, that may not be enough. Thus, we

use double precision which provides approximately **15** decimal digits of precision.

- ✓ C is a **case sensitive language** so variable names age and Age will be treated differently.
- ✓ **\n** – new line, **\t** – new tab.
- ✓ An operator is a **special symbol** that is used to perform **operations** on values and variables.
- ✓ Some operators frequently used:
    - **=** -> Assignment operator
    - **+** -> Addition operator
    - **-** -> Subtraction operator
    - ***** -> Multiplication operator
    - **/** -> Division operator
    - **%** -> Remainder operator / **Modulo** operator
- ✓ When used with integers, the division operator returns the **quotient** after division and when used with floating-point numbers, the division operator returns the **exact result**.
- ✓ The ++ (**increment**) operator **increases** the value of a variable by **1**, and the -- (**decrement**) operator **decreases** the value of a variable by **1**.
- ✓ The ++ and -- operators can only be used with integers.
- ✓ **Precedence and associativity** in C are **DMAS** whereas in Python, it is **PEMDAS**. (D – Division, M – Multiplication, A – Addition, S – Subtraction, P – Parenthesis and E – Exponential)
- ✓ Different data types occupy **different amounts of memory** in the computer.
- ✓ The **sizeof()** operator finds the **size** of values and variables.
- ✓ And to print the result returned by this operator, we can use the **%zu** format specifier.
- ✓ The result returned by the sizeof() operator is in **Bytes** (**1 Byte = 8 Bits**).

- ✓ The process of converting the value of one data type to another is known as **type conversion**.
- ✓ **Implicit Type Conversion** is a phenomenon where, sometimes one data type is **automatically converted** to another type.
- ✓ When there is conversion from larger data to smaller, we will face **loss in data**, i.e., int = 35.87; when printed will return only 35 and the remaining .87 is **lost**.
- ✓ The situations when we want to **manually convert** one data type to another is called **Explicit Type Conversion**.
    - For example,

```
int number1 = 10;
int number2 = 4;
double result;
// (double) convert number1 to double
result = (double) number1 / number2;
printf("%.2lf", result)
>> 2.50
```

- ✓ The **scanf()** function asks the user for input and assigns it to a variable.

```
scanf("%d", &age);
```

,here %d - **format specifiers** (suggest the type of data) and &age represents the **memory location** of the age variable.