**C Programming**

**Introduction**

* This is the mother language of all the other languages like C++ , JAVA , Python .
* C is a structured oriented language unlike C++ , C# and JAVA .

**Reasons of studying C in the age of OOP languages**

* C is faster
* Major parts of Operating systems like Windows , UNIX , Linux and Android are written in C .
* All the device driver programs are exclusively written in C
* Embedded programs in Consumer devices like microwaves , Ovens are written in C .
* Popular gaming Frameworks like DirectX are written in C .
* C provides various language elements that make one to have very close interaction with hardware devices .
* It provides good base for learning other modern languages like C++ , C# , JAVA etc.

**Specialities :-**

* Time and memory efficient.
* Can Interact with hardwares very closely .

**Important terms and definitions : -**

* Keywords -> words which carry special/fixed meaning . Its meaning is already been explained to the compiler/computer .
* There are 32 keywords available in C .
* Semicolon ‘;’ acts as the statement terminator .
* In main() function returning 0 means success .
* **#include** is a **preprocessor directive**
* %d , %c , %fare the **format specifiers**
* **Compiler** converts the the high level language to **machine language**
* **&(ampersand)** is ‘ Address of ’ operator . It gives location number(address) used by the variable in memory
* Variables are the **names given to memory locations .**
* Every compiler is **Platform specific** .
* A platform is a combination of specific OS and microprocessor(i.e **OS + Micropreocessor = Platform**)

**C Instructions**

* **Program** is a set of instructions in order to achieve some certain tasks .
* Types of instructions : -
  + Type declaration instructions – used to declare the type of variables .

ex:- int a , float b etc

* + Arithmetic Instruction – used to perform arithmetic instructions on constants and variables .

s = a + b + 32 , t = 32 \* / a etc

* + Control Instruction – used to control the sequence of execution of various statements in a program
* Arithmetic operators are ‘+’ , ‘-’ , ‘\*’ , ‘/’
* Modulus operator(%) is not used on float
* Using % , sign of the remainder is always same as the sign of numerator
* **ASCII codes** are used to represent any character in the memory , ex:- ASCII code for ‘a’ and ‘b’ are 01100001 and 01100010 respectively
* **ASCII value**s are the decimal equivalent of the **ASCII codes ,** ex:- ASCII values of ‘a’ and ‘b’ are 97 and 98 respectively .
* **‘b’ + ‘a’** , here , addition is performed on the ASCII values of the characters i.e , **97+98**
* **pow(a,b)** is a standard **library function** declared in **<math.h>** header file , used for exponentiation (i.e a raised to power b) .
* **pow()** only works with real numbers .
* **sqrt()** function is used for getting square-root.
* **#include** is a preprocessor directive
* In an operation between **real and integer** always yield **real** . To achieve this integer is first promoted to real .
* **Hierarchy of operators** (priority / precedence):-
  + Parantheses ‘()’
  + \* / %
  + + -
  + =
* **Priority can be changed using ()**
* **Associativity of Operators**
  + When an expression contains two operators of equal priorioty/precedence , the tie between them is settled using associativity of operators
  + \* / follows **left to right** associativity , ex:- 3/2\*5 , / is in left of \* , hence / will be processed first the \*
  + = follows **right to left** associativity , ex:- a=b=2 , rightmost ‘=’ will be performed first and then left one , i.e **at first b = 2 , then a = b**
* **Control Instruction**
  + Specify the order in which the various instructions are to be executed in a program .
  + Types:-
    1. Sequence Control Instruction – It ensures that the instructions are executed in the same order in which they appear in the program
    2. Decision Control Instruction – Takes the decision as to which instruction is to be executed next
    3. Repetition or Loop Control – helps to execute group of statements repeatedly
    4. Case Control Instruction – It allows us to perform instructions as per our choice . OR , It executes only specific **case statements** based on the **switch expression** .

**Decision Control**

- **if-else** and **switch-case** statement are the two main decision control instructions

if(i==5) **;**  ----------- 1

printf(“You entered 5”) **;** ----------- 2

- Statement 1 and 2 will act as **two different statements** , since null operator ( ; ) after if() statement gets the printf() operator out of the if block

- Hence , printf() is bound to get executed no matter if() gives true or false . And there **won’t be any error**

xcxc

- format of if-else :-

if(**condition** is true)

statement 1;

else

statement 2 ;

- We use **relational operators(< , > , <= , >= , == , !=)** to express the conditions in if-else statement

- ‘=’ is used for assignment whereas ‘==’ is used to compare two quantities

- We can use if() without else , but else must always be associated with an if()

- Any non-zero number is true , 0 is false .

if(i=5)

statement ;

- **this statement won’t throw error**

- ‘;’ is a null statement . It doesn’t do anything on execution