C

UNIT 1

C is a general-purpose procedural programming language initially developed by **Dennis Ritchie** in **1972** at Bell Laboratories of AT&T Labs.

Advantages:-

Procedural Language:

One of the most important features of C language is that it is a procedural language. C Language focuses on procedures or functions. It encourages developers to break down difficult issues into small, organized modules by encouraging clean, linear coding.

Rich Library Support: C has a rich set of built-in functions and libraries that facilitate various programming tasks, from mathematical computations to string, manipulation and file handling.

Modularity: C supports modular programming through the use of functions. This allows developers to break down complex problems into smaller, manageable pieces, making code easier to read, maintain, and debug.

Foundation for Other Languages: Many modern programming languages, such as C++, C#, and Java, are influenced by C.

Strong Typing: C enforces strong typing, which helps catch errors at compile. Time rather than at runtime. This can lead to more reliable and maintainable code.

Pointers: Using pointers in C, you can directly interact with memory. As the name suggests, pointers point to a specific location in the memory and interact directly with it. You can use the C pointers to operate with memory, arrays, functions, and structures.

Recursion: Recursion is another important feature of C language. Recursion means that you can create a function that can call itself multiple times until a given condition is true, just like the loops. Recursion in C programming provides code reusability and backtracking functionality**.**

Tokens: - Tokens in C include identifiers, keywords, constants, operators, and punctuation symbols. The C compiler reads the source code and breaks it down into these tokens, which it uses to understand the program's structure and functionality. Tokens serve as the foundation upon which C programs are built and executed.

Classification of Tokens in C:-

While there are 6 primary types of tokens in C language, we do have other classifications. Listed below are all the different types of tokens in C:

- Keywords (Reserved words with special meaning that are used for specific purposes only.)

- Identifiers (Names used to identify variables, functions, classes, etc., components.)
- Operators (Symbols used to perform manipulations and modifications on data.)
- Literals
- Special Symbols
- Constants
- Strings (A sequence of characters/ character array.)
- Character Set

Keyword Tokens In C:-

Keywords are reserved (or pre-defined) words in C that have predefined meanings and are used to show actions or operations in the code. They are an integral part of the C language syntax and cannot be used as identifiers (variable names or function names) in the code. There are a total of 32 keyword tokens in C language.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	Volatile
const	float	short	Unsigned

Variables in C: - A variable in C **is a named piece of memory which is used to store data and access it whenever required.**

Syntax for Creating Variables: - datatype name;

Rules for Naming Variables in C

We can assign any name to a C variable as long as it follows the following rules:

- A variable name must only contain **letters**, **digits**, and **underscores**.
- It must **start with an alphabet** or an **underscore** only. It cannot start with a digit.
- **No white space** is allowed within the variable name.
- A variable name must **not** be any reserved word or **keyword**.
- The name must be unique in the program.

C Variable Initialization

Once the variable is declared, we can store useful values in it. The first value we store is called initial value and the process is called **Initialization**. It is done using assignment operator **(=)**.

Example:-
In numb = 3;
Data types in C:-

Each variable in C has an associated data type. It specifies the type of data that the variable can store like integer, character, floating, double, etc.

Basic (or Primitive) Data Types:

- int: Stores whole numbers (integers). Size typically 2 or 4 bytes, depending on the system.
- char: Stores a single character. Size is 1 byte.
- float: Stores single-precision floating-point numbers (numbers with decimal points). Size is typically 4 bytes.
- double: Stores double-precision floating-point numbers, offering higher precision and a larger range than float. Size is typically 8 bytes.
- void: Represents the absence of a type, often used for functions that don't return a value or for generic pointers.

Derived Data Types:

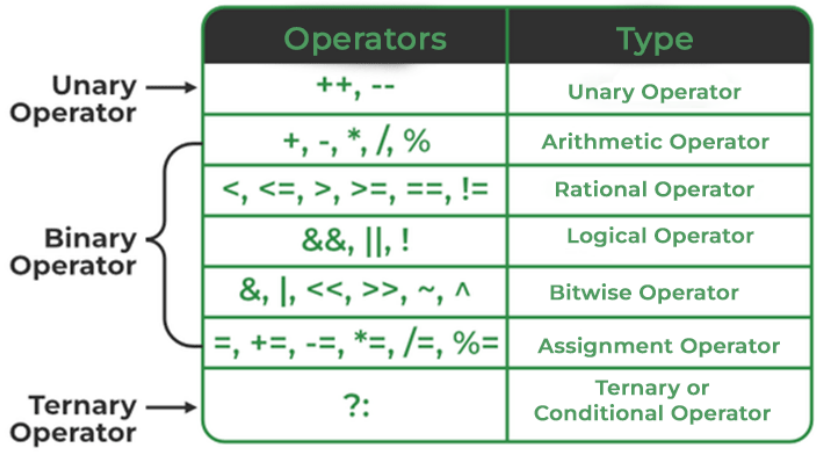
- **Arrays**: Collections of elements of the same data type.
- **Pointers**: Variables that store memory addresses.

Data type	Size(bytes)	Range	Format String
char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
short	2	-32,768 to 32,767	%d
unsigned short	2	0 to 65535	%u
int	2	32,768 to 32,767	%d
unsigned int	2	0 to 65535	%u
long	4	-2147483648 to +2147483647	%ld
Unsinged long	4	0 to 4294967295	%lu
float	4	-3.4e-38 to +3.4e-38	%f
double	8	1.7 e-308 to 1.7 e+308	% lf
long double	10	3.4 e-4932 to 1.1 e+4932	%lf

Basic syntax of a C program:-

```
#include <stdio.h>

int main(void)

{

   //body of the program

   return 0;

}
```

OPERATORS IN C:-



Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values.

C Increment and Decrement Operators

C programming has two operators increment `++` and decrement `--` to change the value of an operand (constant or variable) by 1.

C Assignment Operators

An assignment operator is used for assigning a value to a variable. The most common assignment operator is `=`

C Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

C Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	Meaning	Example
&&	Logical AND. True only if all operands are true	If c = 5 and d = 2 then, expression `((c==5) && (d>5))` equals to 0.
||	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression `((c==5) || (d>5))` equals to 1.
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression `!(c==5)` equals to 0.

C Bitwise Operators

During computation, mathematical operations like: addition, subtraction, multiplication, division, etc are converted to bit-level which makes processing faster and saves power.

Precedence of operators:- The precedence of operators determines which operator is executed first if there is more than one operator in an expression.

Associativity of Operators:- The associativity of operators determines the direction in which an expression is evaluated.

[when two operators of the same precedence appear in an expression. Associativity can be either from **Left to Right or Right to Left.** Let's evaluate the following expression,
100 / 5 % 2

The division (/) and modulus (%) operators have the same precedence, so the order in which they are evaluated depends on their left-to-right associativity. This means the division is performed first, followed by the modulus operation. After the calculations, the result of the modulus operation is determined.**]**

OPERATOR	TYPE	ASSOCIAVITY
( ) [ ] . ->		left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Unary Operator	right-to-left
* / %	Arithmetic Operator	left-to-right
+ -	Arithmetic Operator	left-to-right
<< >>	Shift Operator	left-to-right
< <= > >=	Relational Operator	left-to-right
== !=	Relational Operator	left-to-right
&	Bitwise AND Operator	left-to-right
^	Bitwise EX-OR Operator	left-to-right
|	Bitwise OR Operator	left-to-right
&&	Logical AND Operator	left-to-right
||	Logical OR Operator	left-to-right
? :	Ternary Conditional Operator	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operator	right-to-left
,	Comma	left-to-right

What is Type Casting in C?

Type casting is the process in which the compiler automatically converts one data type in a program to another one.

In typecasting, the destination data type may be smaller than the source data type when converting the data type to another data type, that's why it is also called **narrowing conversion.**

For instance, if a programmer wants to store a long variable value into some simple integer in a program, then they can type cast this long into the int. Thus, the method of type casting lets users convert the values present in any data type into another data type with the help of the cast operator, like:

Syntax:- (new_data_type) expression
For example:

```
int num = 10;

double result = (double) num;   // Casting 'num' to a double
```

Types of Type Casting in C:

- **Implicit Casting** (automatically) - Done automatically by the compiler.
- Occurs when converting a smaller data type to a larger data type.

[char -> int -> long -> float -> double]

Example:

```
int a = 10;

float b = a;
```

Typecasting

- **Explicit Casting** (manually) - converting a larger type to a smaller size type
- Done manually by the programmer using the **cast operator**.

Also known as Typecasting.

[double -> float -> long -> int -> char]

Example:

```
float a = 10.5;
int b = (int) a;
```

No of difference	TYPE CASTING (Explicit type casting)	TYPE CONVERSION (Implicit Type casting)
1)	When a user can convert the one data type into other then it is called as the typecasting.	Type Conversion is that which automatically converts the one data type into another.
2)	Implemented on two 'incompatible' data types.	Implemented only when two data types are 'compatible'.
3)	For casting a data type to another, a casting operator '()' is required.	No operator required.
4)	It is done during program designing.	It is done explicitly while compiling.
5)	It is a narrowing conversion.	It is a widening conversion.