Introduction to C Programming

What is C Language?

C is a **general-purpose, procedural programming language** developed by **Dennis Ritchie** at **Bell Labs in 1972**. It is widely used for **system programming**, including operating systems, compilers, and embedded systems.

Features of C:

- **Simple and efficient**
- **Fast execution**
- **Rich set of built-in functions and operators**
- **Low-level memory access using pointers**
- **Modularity** (can divide code into functions)
- **Portable** (can run on different machines with minimal changes)

Why Learn C?

- Foundation for other languages like C++, Java, Python
- Helps understand how memory works (with pointers, arrays)
- Used in operating system and compiler development
- Still used in embedded systems and microcontroller programming

Applications of C:

- Operating systems (e.g., Unix)
- Compilers
- Text editors
- Database systems

Structure of a C Program

A C program is made up of **sections** that follow a specific order.

General Structure:

```
#include <stdio.h>     // Preprocessor directive

// Global declarations (if any)

int main()             // Main function
{
    // Variable declarations
    // Executable statements
    return 0;
}
```

Sections of a C Program:

Section	Description
1. Documentation Section	Optional comment section describing the program, author, date, etc.
2. Preprocessor Directives	Includes files and macros (e.g., `#include <stdio.h>`)
3. Global Declaration	Declare global variables or functions before `main()` if needed
4. main() Function	Entry point of every C program
5. Variable Declarations	Inside `main()` to define variables used
6. Executable Statements	Actual code logic: input/output, loops, conditions, etc.
7. Return Statement	Returns a value from `main()` to the OS (`return 0;`)

What are Tokens?

In C programming, **tokens** are the **smallest elements** or **building blocks** of a program that the compiler can understand. Every C program is made up of tokens.

Types of Tokens in C

There are **6 main types** of tokens:

- #include, int, float, return → **Keywords**
- main, a, b → **Identifiers**
- 10, 5.5 → **Constants**
- ,, ;, (), {} → **Symbols**
- +,-,*,/ → **Operators**
- "The value of a is..." → **String**

What is a Variable?

A **variable** in C is a **named storage location** used to store data that can be changed during program execution.

Rules (Must follow)

1. Name must **start with a letter (A–Z or a–z)** or an **underscore (_)**
2. Can include **letters, digits (0–9), and underscores**
3. Cannot use **spaces, symbols** (`@`, `#`, `%`, etc.)
4. **Cannot be a C keyword** (like `int`, `return`, etc.)
5. **Example→** int marks; float percentage;

What are Data Types?

In C, **data types define** the type of data a variable can hold — like integer, float, character, etc.

Categories of Data Types: There are 3 types of data type mainly in C programming

1. Primitive (Basic / Built-in) Data Types

These are **predefined** in C.

Data Type	Size (Typical)	Description	Example
`int`	2 or 4 bytes	Stores integers	`int age = 25;`
`float`	4 bytes	Stores decimal (single precision)	`float pi = 3.14;`
`double`	8 bytes	Stores decimal (double precision)	`double d = 3.14159;`
`char`	1 byte	Stores single character	`char grade = 'A';`
`void`	0 byte	No value (used in functions)	`void main()`

2. Derived Data Types

Formed using primitive types.

Type	Description	Example
Array	Collection of similar data types	`int marks[5];`
Pointer	Stores address of variable	`int *ptr;`
Function	Block of reusable code	`int sum(int a, int b);`
Structure	Group of variables of different types	`struct student { ... }`

3. User-defined Data Types

Created by the programmer using keywords like `struct`, `union`, `typedef`, `enum`.

Type	Description	Example
`struct`	Group of different data types	`struct car { char name[10]; int speed; };`
`union`	Memory shared between members	`union data { int i; float f; };`
`enum`	Set of named integer constants	`enum week {Mon, Tue, Wed};`
`typedef`	Alias name for existing data types	`typedef int age;`

Typical Size Chart of Primitive Data Types

Data Type	Size	Range
`char`	1 byte	-128 to 127 or 0 to 255 (unsigned)
`int`	2 or 4 bytes	-32,768 to 32,767 (2 bytes) or more (4 bytes)
`float`	4 bytes	3.4e-38 to 3.4e+38 (6 decimal places)
`double`	8 bytes	1.7e-308 to 1.7e+308 (15 decimal places)
`void`	0 bytes	No data

Note: Size may vary slightly depending on compiler and system architecture (32-bit / 64-bit)**.**

Operators in C Programming

Operators are special symbols that perform operations on variables and values.

1. Types of Operators

A. Arithmetic Operators

Operator	Description	Example
`+`	Addition	`a + b`
`-`	Subtraction	`a - b`

Operator	Description	Example
*	Multiplication	a * b
/	Division	a / b
%	Modulus (remainder)	a % b

B. Relational (Comparison) Operators

Used to compare two values.

Operator	Meaning	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

C. Logical Operators

Used to combine conditions.

Operator	Meaning	Example
&&	Logical AND	(a > 10) && (b < 5)
||	Logical OR	(a > 10) || (b < 5)
!	Logical NOT	!(a > 10)

D. Assignment Operators

Used to assign values to variables.

Operator	Meaning	Example
=	Assign	a = 10;
+=	Add and assign	a += 5;
-=	Subtract and assign	a -= 3;
*=	Multiply and assign	a *= 2;
/=	Divide and assign	a /= 4;
%=	Modulus and assign	a %= 3;

E. Increment and Decrement Operators

Operator	Meaning	Example
++	Increment by 1	a++ or ++a
--	Decrement by 1	a-- or --a

- a++ → Post-increment
- ++a → Pre-increment

F. Bitwise Operators

Works on bits (binary values).

Operator	Meaning	Example
&	Bitwise AND	a & b
`	`	Bitwise OR
^	Bitwise XOR	a ^ b
~	Bitwise NOT	~a
<<	Left shift	a << 2
>>	Right shift	a >> 2

G. Conditional (Ternary) Operator

```
int result = (a > b) ? a : b;
```

H. Special Operators

Operator	Use
`sizeof`	Returns size of a variable/type
`&`	Address of variable
`*`	Pointer (value at address)
`->`	Access structure member using pointer
`.`	Access structure member