INTRODUCTION TO 'C' PROGRAMMING

PROGRAMMING

Programming is the process of creating a set of instructions that a computer can follow to perform specific tasks.

These instructions, known as code, are written in a programming language, which serves as a medium of communication between the programmer and the computer.

A program is a set of instructions written in a programming language that a computer follows to perform a specific task or solve a particular problem. These instructions are executed by the computer's central processing unit (CPU) to achieve the desired outcome.

WHAT IS 'C'?

C is a general-purpose programming language that was developed in the early 1970s by Dennis Ritchie at Bell Labs.

It is one of the most influential programming languages and has had a significant impact on many other languages, such as C++, Java, and Python.

Evolved from B and BCPL (Basic Combined Programming Language).

- Initially used for system programming ← O/S
- Played a crucial role in the development of UNIX

FEATURES OF 'C'

- **Low-Level Access:** C provides low-level access to memory through pointers, making it ideal for system programming, such as developing operating systems, embedded systems, and device drivers.

- **Efficiency:** C is known for its efficiency, both in terms of speed and memory usage. This makes it suitable for resource-constrained environments.

- **Portability:** Programs written in C can be compiled and run on different types of computer systems with minimal changes, making it a portable language.

FEATURES OF ‘C’

- **Rich Library:** C comes with a standard library that provides functions for tasks like input/output operations, string manipulation, and mathematical computations.

- **Modularity:** C supports modular programming, which allows developers to break down a program into smaller, reusable functions or modules.

- **Structured Programming:** C supports structured programming, which encourages a clear and logical structure in the code, making it easier to understand, debug, and maintain.

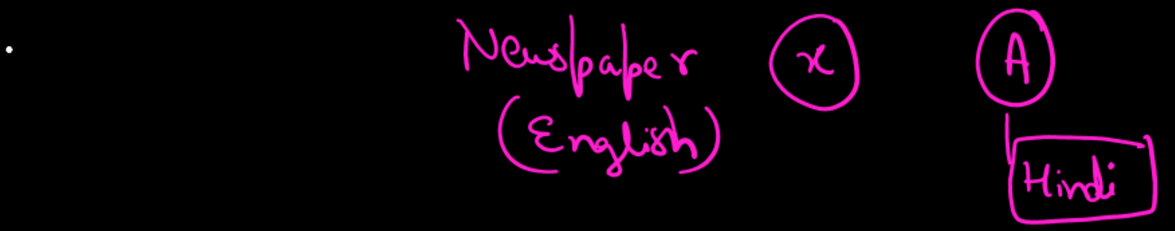
COMMON USES OF ‘C’

- **Operating Systems:** Many operating systems, including UNIX, Linux, and Windows, have been written in C.
- **Embedded Systems:** C is widely used in developing firmware and software for embedded systems.
- **System Programming:** Tasks that require direct interaction with the hardware, such as writing device drivers, often use C.
- **Application Development:** C is used in developing various applications, including text editors, compilers, and network protocols.

LANGUAGE TRANSLATOR: COMPILER AND INTERPRETER

- A compiler translates the entire source code of a program into machine code (binary code) or an intermediate code before execution. This translation happens all at once, producing an executable file.

- An interpreter translates and executes the source code line by line or statement by statement. It doesn't produce an intermediate machine code file. Instead, it directly executes the instructions.



INSTALLING THE ‘GNU GCC/G++’ COMPILER

UNDERSTANDING THE DEVELOPMENT ENVIRONMENT

[Visual Studio Code, Code Blocks, Sublime text, Notepad++ etc]

An **Integrated Development Environment (IDE)** is a software application that provides comprehensive facilities to computer programmers for software development.

An IDE typically consists of several tools that assist in writing, testing, and debugging code, all within a single interface.

IDEs are designed to streamline the development process, making it easier for developers to write code, test it, and deploy applications.

BASIC STRUCTURE OF A C PROGRAM

essential

1. Preprocessor Directives:- Begin with # (e.g., #include, #define)
2. main() Function:- The entry point of the program
3. Variable Declarations:- Variables are declared before use
4. Statements & Expressions:- The logic of the program is written here
5. Return Statement:- Indicates the end of the program (return 0;)

PREPROCESSOR DIRECTIVES

Pre - पहले

1. Preprocessor Directives:

   - Begin with # symbol (e.g., #include, #define)
   - Processed before actual compilation
   - Commonly used for including header files and defining constants

#include <header.h>

Pre - Programmed

eg PI 3.14159

Example:

header file

```
#include <stdio.h>  // Includes standard input/output library
```

THE MAIN() FUNCTION

[यहाँ से चलेगा]

2. main() Function:

- The entry point of every C program
- Typically returns an integer value (int)
- Execution starts from the main function

Example:

```
int main() {
    // Code goes here
    return 0;  // Indicates successful execution
}
```

↳ Succesful termination of program

VARIABLE DECLARATIONS

चर

int x = 10;

3. Variable Declarations:
   - Variables must be declared with a specific data type before use
   - Can be placed inside the main function or globally

Example:

```
int x;  // Declaration of an integer variable
float y;  // Declaration of a floating-point variable
```

STATEMENTS AND EXPRESSIONS

4. Statements and Expressions: & Equations
   - Core of the program where the logic is implemented
   - Includes assignments, loops, conditionals, and function calls

Example:

```
x = 10;  // Assignment statement

printf("Value of x: %d\n", x);  // Function call to print the value of x
```

RETURN STATEMENT

5. Return Statement:

- Indicates the end of the main function
- Returns a value to the operating system
- Typically, return 0; indicates successful execution

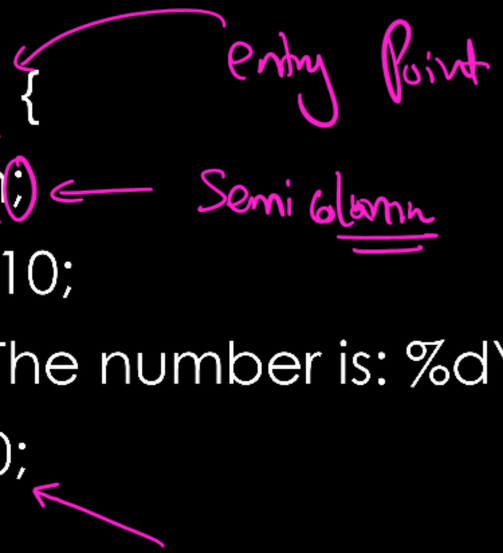
Example:

```
return 0;  // Return statement indicating successful execution
```

EXAMPLE: BASIC C PROGRAM

```c
#include <stdio.h>  // Preprocessor directive to include standard I/O functions

int main() {
    int num;
    num = 10;
    printf("The number is: %d\n", num);
    return 0;
}
```



⇒ Tokens

⇒ Identifiers