**C PROGRAMMING MODULE- 2**

1. **Write an essay covering the history and evolution of C programming. Explain its importance and why it is still used today.**

* **Short Essay: History and Importance of C Programming**

C programming was developed in the early 1970s by **Dennis Ritchie** at **Bell Labs**. It was created to help build the Unix operating system and quickly became popular because of its speed and power.

* C is important because it allows programmers to write fast and efficient code. It is used in system programming, operating systems, and embedded systems. Even today, C is still widely used because it is simple, portable, and helps us understand how computers work.
* In short, C is a strong and reliable language that has helped shape modern programming and continues to be useful even after many years.

1. **Describe the steps to install a C compiler (e.g., GCC) and set up an Integrated Development Environment (IDE) like DevC++, VS Code, or CodeBlocks.**

**1. Install GCC Compiler**

* Windows: Install MinGW
* Mac: Run xcode-select --install
* Linux: Run sudo apt install gcc

**2. Install an IDE**

* DevC++ or Code::Blocks: Download and install (comes with compiler)
* VS Code: Install VS Code and C/C++ extension. Make sure GCC is installed.

1. **Explain the basic structure of a C program, including headers, main function, comments, data types, and variables. Provide examples.**
2. **Header Files – Include libraries**

**#include <stdio.h> // for input/output**

**#include <conio.h> // for getch() and clrscr()**

1. **Main Function – Program starts here**

**void main() {**

**// code goes here**

**}**

1. **Comments – Used to explain code**

**// Single-line comment**

**/\* Multi-line comment \*/**

1. **Data Types – Define data**

* **int, float, char**

1. **Variables – Store values**

**int a = 10;**

**float b = 5.5;**

**char c = 'A';**

1. **Write notes explaining each type of operator in C: arithmetic, relational, logical, assignment, increment/decrement, bitwise, and conditional operators.**
2. **Arithmetic Operators**

**Examples: +, -, \*, /, %**

**a + b**

**a - b**

**a \* b**

**a / b**

**a % b**

**2. Relational Operators**

**Examples: ==, !=, >, <, >=, <=**

**a == b**

**a != b**

**a > b**

**3. Logical Operators**

**Examples: && (AND), || (OR), ! (NOT)**

**(a > 0) && (b < 10)**

**4. Assignment Operators**

**Examples: =, +=, -=, \*=, /=**

**a = 10;**

**a += 5;**

**5. Increment/Decrement Operators**

**Examples: ++, --**

**a++;**

**b--;**

**6. Bitwise Operators**

**Examples: &, |, ^, ~, <<, >>**

**a & b**

**a | b**

**7. Conditional (Ternary) Operator**

**Syntax: condition ? true : false**

**int max = (a > b) ? a : b;**

1. **Explain decision-making statements in C (if, else, nested if-else, switch). Provide examples of each.**

**1. if Statement**

**Checks a condition. If true, code runs.**

**if (a > 0) {**

**printf("Positive number");**

**}**

**2. if-else Statement**

**Runs one block if condition is true, another if false.**

**if (a % 2 == 0) {**

**printf("Even");**

**} else {**

**printf("Odd");**

**}**

**3. Nested if-else**

**if-else inside another if-else.**

**if (a > 0) {**

**printf("Positive");**

**} else {**

**if (a < 0) {**

**printf("Negative");**

**} else {**

**printf("Zero");**

**}**

**}**

**4. switch Statement**

**Used to choose from many options.**

**int choice = 2;**

**switch (choice) {**

**case 1:**

**printf("One");**

**break;**

**case 2:**

**printf("Two");**

**break;**

**default:**

**printf("Invalid");**

**}**

1. **Compare and contrast while loops, for loops, and do-while loops. Explain the scenarios in which each loop is most appropriate.**

**while Loop**

* **Checks condition before running the code.**
* **Runs zero or more times.**

**Syntax :-**

**while (condition) {**

**// code**

**}**

**for Loop**

* **Best when you know exactly how many times to repeat.**
* **Has initialization, condition, and update in one line.**

**Syntax:-**

**for (int i = 0; i < 5; i++) {**

**// code**

**}**

**do-while Loop**

* **Runs the code first, then checks condition.**
* **Runs at least once.**
* **Use when code must run at least once.**

**Syntax:-**

**do {**

**// code**

**} while (condition);**

1. **Explain the use of break, continue, and statements in C. Provide examples of each.**

**break**

**Stops the loop or switch immediately.**

**Example:-**

**for (int i = 0; i < 5; i++) {**

**if (i == 3)**

**break;**

**printf("%d ", i);**

**}**

**continue**

**Skips the current loop iteration and goes to the next one.**

**Example:-**

**for (int i = 0; i < 5; i++) {**

**if (i == 2)**

**continue;**

**printf("%d ", i);**

**}**

1. **What are functions in C? Explain function declaration, definition, and how to call a function. Provide examples.**

**What are Functions :-**

**Functions are blocks of code that perform a specific task. They help make programs organized and reusable.**

**Function Declaration:-**

**Example :-**

**int add(int, int);**

**Function Definition:-**

**Example :-**

**int add(int a, int b) {**

**return a + b;**

**}**

**Function Call :-**

**Example:-**

**int result = add(5, 3);**

**printf("Sum = %d", result);**

1. **Explain the concept of arrays in C. Differentiate between one-dimensional and multi-dimensional arrays with examples.**

**Array :-**

**An array is a collection of similar data types stored in a single variable with multiple values.**

**1. One-Dimensional Array:-**

**int num[5] = {10, 20, 30, 40, 50};**

**printf("%d", num[2]);**

**2. Multi-Dimensional Array**

**int mat[2][2] = {{1, 2}, {3, 4}};**

**printf("%d", mat[1][0]);**

1. **Explain what pointers are in C and how they are declared and initialized. Why are pointers important in C?**

**Pointers :-**

**Pointers are variables that store the address of another variable.**

**Declaration and Initialization**

**int a = 10;**

**int \*p;**

**p = &a;**

**Why Are Pointers Important?**

* **Help in memory management**
* **Used in arrays, functions, and dynamic memory**
* **Allow direct access to memory**
* **Needed for efficient programming**

1. **Explain string handling functions like strlen(), strcpy(), strcat(), strcmp(), and strchr(). Provide examples of when these functions are useful.**
2. **strlen()**

**Returns the length of a string.**

**Example:-**

**int len = strlen("Hello");**

1. **strcpy()**

**Copies one string to another.**

**Example:-**

**char a[20];**

**strcpy(a, "Hello");**

1. **strcat()**

**Joins (concatenates) two strings.**

**Example :-**

**char a[20] = "Hello ";**

**strcat(a, "World");**

1. **strcmp()**

**Compares two strings. Returns 0 if equal.**

**Example:-**

**strcmp("apple", "apple");**

**strcmp("apple", "banana");**

1. **strchr()**

**Finds the first occurrence of a character in a string.**

**Example:-**

**char \*ptr = strchr("hello", 'e'); // ptr points to "ello"**

1. **Explain the concept of structures in C. Describe how to declare, initialize, and access structure members.**

**Structure :-**

**A structure is a user-defined data type that groups different types of data together.**

**Declaration :-**

**struct Student {**

**int id;**

**char name[20];**

**float marks;**

**};**

**Initialization :-**

**struct Student s1 = {101, "Meet", 85.5};**

**Access Members :-**

**printf("%d", s1.id);**

**printf("%s", s1.name);**

**printf("%.2f", s1.marks);**

1. **Explain the importance of file handling in C. Discuss how to perform file operations like opening, closing, reading, and writing files.**

**Importance of File Handling :-**

**File handling lets you store data permanently in files (like .txt). It allows programs to read/write data even after the program ends.**

**Basic File Operations**

1. **Opening a File:-**

**FILE \*fp;**

**fp = fopen("meet.txt", "w");**

1. **Writing to a File:-**

**fprintf(fp, "Hello World");**

1. **Reading from a File :-**

**fp = fopen("meet.txt", "r");**

**char ch = fgetc(fp);**

1. **Closing a File**

**fclose(fp);**