**Module 2 – Introduction to Programming**

**Q-1. Overview of C Programming**

**Theory Exercise:-**

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

History and Evolution of C Programming C was developed in the early 1970s by Dennis Ritchie at Bell Labs to create the UNIX operating system. It evolved from earlier languages like B and BCPL. C became popular due to its efficiency, portability, and flexibility. It laid the foundation for many modern languages like C++, Java, and C#. Today, C is still widely used because it allows direct hardware access and offers great performance in low-level system programming.

**• LAB EXERCISE:**

**Research and provide three real-world applications where C programming is extensively used, such as in embedded systems, operating systems, or game development.**

C programming is extensively used in various real-world applications due to its efficiency, speed, and low-level memory access. One major area is embedded systems, where C is the preferred language for programming microcontrollers in devices like washing machines, medical equipment, and automotive systems, due to its ability to interact directly with hardware. Another key application is in operating systems—many core components of popular operating systems like Windows, Linux, and UNIX are written in C, as it provides powerful system-level access and control. Additionally, C is widely used in game development, particularly for developing game engines and performance-critical modules, where speed and optimization are crucial for real-time rendering and gameplay mechanics. These applications highlight the enduring importance of C in both hardware-level and high-performance software development.

**Q-2. Setting Up Environment**

**Theory Exercise:-**

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

**Steps to Install C Compiler & IDE:**

1. Install a compiler like GCC (part of MinGW for Windows).
2. Download an IDE like Dev C++, VS Code, or Code Blocks.
3. Set environment variables (for GCC).
4. Write and compile code using the IDE.

**Q-3. Basic Structure of a C Program**

**Theory Exercise:-**

**Explain the basic structure of a C program, including headers, main function, comments, data types, and variables. Provide examples.**

**Structure Elements:**

* Header files: #include<stdio.h>
* Main () function: Starting point of the program.
* Comments: // single-line or /\* multi-line \*/
* Data types: int, char, float, etc.
* Variables: Used to store data.

**Q-4. Operators in C**

**Theory Exercise:-**

**Write notes explaining each type of operator in C: arithmetic, relational, logical, assignment, increment/decrement, bitwise, and conditional operators?**

**Types of Operators :-**

1.Arithmetic:- +, -, \*, /, %

2.Relational:- >, <, ==, !=

3.Logical:- &&, ||, !

4.Assignment:- =, +=, -=

5.Increment/Decrement:- ++, --

6.Bitwise:- &, |, ^, ~

7.Conditional (Ternary):- condition ? expr1 : expr2

**Q-5. Control Flow Statements in C**

**Theory Exercise:-**

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

**Decision-Making Statements:**

* if: Checks a condition.
* if-else: Chooses between two options.
* nested if-else: Multiple conditions.
* switch: Replaces multiple ifs for fixed values.

**Q-6. Looping in C**

**Theory Exercise:-**

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

**Ans:-**

While Loop:- Entry-controlled loop.

For Loop:- Used when number of iterations is known.

do-while Loop:- Executes at least once.

**Q-7. Loop Control Statements**

**Theory Exercise:-**

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

**Ans:-** The **break statement** is used to terminate a loop or switch statement prematurely. When the break statement is encountered, control exits from the loop or switch.

Use Case: Exit from for, while, do-while, or switch when a condition is met.

**->**The **continue statement** skips the rest of the loop body and jumps to the next iteration of the loop. It does not exit the loop, just skips the current cycle.

Use Case**:** Skip specific iterations in for, while, or do-while.

**->**The **go to** statement transfers control to a labeled statement elsewhere in the code. Although considered harmful in modern programming due to readability issues, it is still supported.

Use Case**:** Jump to a specific part of the program (rarely recommended).

**Q-8. Functions in C**

**• THEORY EXERCISE:**

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

**Ans:-** In C programming, a function is a block of code that performs a specific task. Functions help in modular programming, code reuse, and better organization of complex programs. Every C program must have a main() function, but additional functions can be created by the user.

1. **Function Declaration (Prototype)**

int add ( int a, int b ) ; // Declaration

1. **Function Definition**

int add (int a, int b) {

return a + b;

}

1. **Function Call**

int result = add (5, 3); // Calling the function

**Q-9. Arrays in C**

**• THEORY EXERCISE:**

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

**Ans:-**

**Concept of Arrays in C**

**—>**An array in C is a collection of elements of the same data type stored in contiguous memory locations. Instead of declaring multiple variables for similar data, an array allows storing and accessing values using a single name and an index. The index starts from 0. Arrays can be **one-dimensional**, **two-dimensional** (like matrices), or even **multi-dimensional.**

**Q-10 Pointers in C**

**• THEORY EXERCISE:**

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

Pointers in C are variables that store the **memory address** of another variable. Instead of holding data directly like normal variables, pointers "point to" the location where the data is stored. This allows efficient handling of memory and direct manipulation of data stored in memory.

**Importance of Pointers in C:**

* **Memory Efficiency:** Pointers allow dynamic memory allocation using functions like malloc(), calloc(), etc.
* **Function Arguments:** Enable call by reference, allowing functions to modify actual variables.
* **Arrays and Strings:** Efficient handling and manipulation of arrays and strings.
* **Data Structures:** Essential for creating complex structures like linked lists, trees, and graphs.

**Q-11 String in C**

**• THEORY EXERCISE:**

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

**Ans:-**

strlen(): This function returns the length of a string (excluding the null character \0).

strcpy():This function copies the contents of one string into another

strcat():This function appends one string to the end of another.

strcmp():This function compares two strings and returns 0 if they are equal, a negative value if the first string is less, and a positive value if greater.

**strchr():** This function returns a pointer to the first occurrence of a character in a string, or NULL if not found.

**Q-12 Structures in C**

**• THEORY EXERCISE:**

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

**Ans:-**

In C, a structure is a user-defined data type that allows grouping variables of different data types under a single name. Structures are useful for representing real-world entities like students, employees, or books, which have multiple attributes.

Structures help organize complex data efficiently and are widely used in file handling, databases, and system programming in C.

**Q-13 File handling in C**

**• THEORY EXERCISE:**

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

**Ans:-**

File handling in C is crucial for managing data beyond the lifetime of a program's execution. It allows data to be stored permanently in files, making it possible to read or write information whenever needed. This is especially useful for applications such as databases, configuration files, or logging systems. In C, file operations are done using the standard library functions provided in **<stdio.h>.** To begin file operations, a file pointer of type **FILE \*** is declared. The file is opened using the **fopen()** function, where you specify the filename and mode **("r" for reading, "w" for writing, "a" for appending, etc.).** Once the file is open, data can be written using functions like **fprintf()** or **fputs()**, and read using **fscanf()** or **fgets().** After completing the operations, the file must be closed using **fclose()** to free resources and ensure data is saved properly. Proper file handling ensures data persistence, efficient data management, and safer program execution.