**Module 2 – Introduction to Programming**

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

The C programming language was developed in the early 1970s by Dennis Ritchie at Bell Laboratory. C was created to write the UNIX operating system.

C gained popularity quickly due to its low-level access and high-level features.

C remains one of the main important language as its fast, efficient, easy to execute.

It also forms the foundation of many modern languages like C++, Java, and Python.

**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.**

Embedded Systems: medical devices, automotive systems, and home appliances.

Operating Systems: UNIX, Linux, Windows.

Game Development: Unreal Engine, Mobile games etc.

**THEORY EXERCISE: 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.**

Step 1 Download MinGW

Step 2 Install and select the C compiler component.

Step 3 Add the bin folder to your System PATH:

To download vs code visit vs code official website and download the latest version.

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

Header is used to include header files like stdio, conio.

Main is used to include the main code of the application. It is executed by the c compiler when the program Is run.

Comments are non executable part of the code which are written by programmers to make the code understandable.

Datatypes denotes the type of data being used by the variable, function, etc.

Variables are names given to data stored at the computer memory of the program.

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

Arithmetic Operators is Used for basic mathematical operations. (+ - / \* %)

Relational Operators is Used to compare values. (== != < > <= >=)

Logical Operators is Used to combine conditions. (&& || !)

Assignment Operators is Used to assign values to variables. (= += -= \*= /= %=)

Increment/Decrement OperatorsUsed to increase or decrease a variable's value by 1. (++ --)

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

Decision making statement in c is used to execute a piece of code based on the condition.

If: if is used to check the condition and if the condition is true then if block is executed.

Else: if the condition is false then the else block is executed.

Nested-if: in nested if there are two if statements inside each other. The first if is executed first and if the condition is true the control jumps in and checks the next if statement thus its called nested if.

Switch: switch is used when we have multiple options and need to execute a certain option based on the user input. Each of these options is written inside a case and the switch holds the option that the user selects.

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

In While Loop the Condition is checked first. Its Use when number of iterations are unknown. Example is Reading input until a condition is met.

In For Loop the var is first Initialized then condition is checked and then var is increment all in one line. Its Use when number of iterations known. Example is looping through an array.

In Do-While Loop the loop is Executes at least once, condition checked after. Its Used when Code must run at least once before checking the condition. Its Example is Menu-driven programs.

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

The break keyword is used to get out from a loop or conditional statement. Its is mainly used with the switch statements. Eg if we want to get out from a loop of 1 to 10 when ilterator reaches 5 then we can use break along with if.

The continue keyword is used the skip a single ilteration in the loop. Eg skipping 2 after 1 in loop.

Goto statement is used to move the control of execution from one part to another part of the program by assigning the destination part as a target element to the goto statement.

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

Function in c are executable piece of code that are created by a programmer to modularize the program and to make the program easy to read and make a piece of code reusable.

Function Declaration tells the compiler about the function’s name, return type, and parameters. Syntax: int add(int a, int b);

Function Definition contains the actual code to execute. Syntax: int add(int a, int b) {….}

Calling a Function executes the function using its name and arguments. Syntax: int result = add(3, 5);

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

An array is a collection of elements of the same data type stored in contiguous memory location. Arrays can be of multiple types.

Difference between one dimension and multidimensional array are as follows:

One dimension: stores single list of elements, elements are in one row, syntax includes only one rectangle bracket.

Multi dimension: stores elements in a grid, elements are in more than one row, syntax includes more than one rectangle bracket.

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

A pointer is a variable that stores the **memory address** of another variable.

Pointer declaration and initialization:

int x = 10;

int \*ptr; // pointer declaration.

ptr = &x; // pointer initialization.

Pointers are important in c as they allow us to directly access and modify the value of other variables, allow us to dynamically change variables, modify memory locations, etc.

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

strlen is Used to Returns the length of a string. Eg: strlen("Hello");

strcpy is Used to Copy value of one string into another. Eg: strcpy(dest, src);

strcat is Used to Append one string to the end of another and store the value of new string in the first string. Eg: strcat(a, b);

strcmp is Used to Compare value of two strings. It returns 0 if it equals. Eg: strcmp("abc", "abc");

strchr is Used to find the first occurrence of a character in a string. Eg: strchr("hello", 'e');

**THEORY EXERCISE: Explain the concept of structures in C. Describe how to declare, initialize, and access structure members.**  
A structure is a user-defined data type that is used to store values of different type together at one place, similar to array.

Structure declaration:

struct Student {

int id;

char name[50];

float marks;

};

Structure initialization:

struct Student s1 = {101, "John", 88.5};

Access structure members:

Student.id

Student.name

Student.marks

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

File handling allows user to store data permanently and accessed later, beyond the program’s runtime. It's essential for storing user data.

A file can be opened with the following syntax - FILE \*fp = fopen("filename.txt", "mode");

Here we have multiple modes like r for reading, w for writing, a for appending new text, r+ for reading and writing.

We can close the file using - fclose(fp);

We can read from the file using fgetc(fp)

And we can also write using fputc(char, fp)