Module 2 – Introduction to Programming

1.Overview of C 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.

Ans:

1. Operating Systems

- **Description:** C is the backbone of many operating systems.
- **Examples:** Linux Kernel, UNIX, Windows components.
- **Why C?** Low-level memory control, speed, portability, and direct hardware access.

2. Embedded Systems

- **Description:** C is used to program microcontrollers in small, resource-constrained devices.
- **Examples:** IoT devices, automotive ECUs, medical instruments.
- **Why C?** Lightweight, fast, close to hardware, uses minimal resources.

3. Game Development (Game Engines)

- **Description:** C is used in performance-critical parts of game engines.
- **Examples:** Doom, Quake, components of Unreal Engine.
- **Why C?** High performance, low latency, fine-tuned memory and graphics control.

2. Setting Up Environment

LAB EXERCISE: Install a C compiler on your system and configure the IDE. Write your first program to print "Hello, World!" and run it.

Ans:

1.Install a C Compiler

- **For Windows:**
  - Download and install MinGW, or use **TDM-GCC**.
- **For Linux:**
  - Use terminal to install `gcc`:
  - `sudo apt update`
  - `sudo apt install build-essential`

2. Install and Configure an IDE

- **Recommended IDEs:**
  - **Code::Blocks**
  - **Dev-C++**
  - **Visual Studio Code** (with **C/C++ extension**)

- **Configuration:**
  - Set up the compiler path (e.g., link MinGW in Code::Blocks).
  - Create a new C project or file.
  - Choose C as the language during project setup.

3. Write Your First Program

```
#include <stdio.h>
int main() {
    printf("Hello, World!\n");
    return 0;
}
```

4. Compile and Run the Program

- Click **Build and Run** in the IDE or:
  - From terminal/command prompt:

```
gcc hello.c -o hello
./hello
```

3. Basic Structure of a C Program.

LAB EXERCISE: Write a C program that includes variables, constants, and comments. Declare and use different data types (int, char, float) and display their values.

Ans:

```
#include <stdio.h>

int main() {

   // Constant declaration

   const float PI = 3.14159;


   // Variable declarations

   int age = 20;            // Integer variable

   char grade = 'A';        // Character variable

   float height = 5.9;      // Float variable


   // Output the values

   printf("Student Details:\n");

   printf("Age: %d years\n", age);

   printf("Grade: %c\n", grade);
```

```
    printf("Height: %.1f feet\n", height);
    printf("Value of PI (constant): %.5f\n", PI);
    return 0;
}
```

Explanation

const float PI = 3.14159; → A constant float that cannot be changed.

int, char, float → Show how different data types are used.

Comments (// and /* */) explain what each part does.

printf is used to display the values with formatting.

4. Operators in C

LAB EXERCISE: Write a C program that accepts two integers from the user and performs arithmetic, relational, and logical operations on them. Display the results.

Ans:

```
#include <stdio.h>
int main() {
    int a, b;
    // Get input from user
    printf("Enter first number: ");
    scanf("%d", &a);
    printf("Enter second number: ");
    scanf("%d", &b);

    // Arithmetic operations
    printf("\n--- Arithmetic ---\n");
    printf("Addition: %d + %d = %d\n", a, b, a + b);
    printf("Subtraction: %d - %d = %d\n", a, b, a - b);

    // Relational operations
```

```
    printf("\n--- Relational ---\n");
    printf("Are they equal? %d\n", a == b);
    printf("Is a greater than b? %d\n", a > b);

    // Logical operations
    printf("\n--- Logical ---\n");
    printf("a AND b (a && b): %d\n", a && b);
    printf("a OR b (a || b): %d\n", a || b);
    return 0;
}
```

5. Control Flow Statements in C

LAB EXERCISE: Write a C program to check if a number is even or odd using an if-else statement. Extend the program using a switch statement to display the month name based on the user's input (1 for January, 2 for February, etc.).

Ans:

```
#include <stdio.h>
int main() {
    int number, month;
    //  Even or Odd Check
    printf("Enter a number: ");
    scanf("%d", &number)
    if (number % 2 == 0) {
        printf("The number %d is Even.\n", number);
    } else {
        printf("The number %d is Odd.\n", number);
    }
    //Display Month Name Using Switch
    printf("\nEnter a number (1 to 12) for the month: ");
    scanf("%d", &month);
    printf("Month: ");
```

```
    switch(month) {
        case 1:  printf("January\n"); break;
        case 2:  printf("February\n"); break;
        case 3:  printf("March\n"); break;
        case 4:  printf("April\n"); break;
        case 5:  printf("May\n"); break;
        case 6:  printf("June\n"); break;
        case 7:  printf("July\n"); break;
        case 8:  printf("August\n"); break;
        case 9:  printf("September\n"); break;
        case 10: printf("October\n"); break;
        case 11: printf("November\n"); break;
        case 12: printf("December\n"); break;
        default: printf("Invalid month number.\n");
    }
    return 0;
}
```

6. Looping in C

LAB EXERCISE: Write a C program to print numbers from 1 to 10 using all three types of loops (while, for, do-while)

Ans:

```
#include <stdio.h>
int main() {
    int i;

    // Using while loop
    i = 1;
    printf("Using while loop:\n");
    while (i <= 10) {
        printf("%d ", i);
        i++;
```

```
    }

    // Using for loop
    printf("\n\nUsing for loop:\n");
    for (i = 1; i <= 10; i++) {
        printf("%d ", i);
    }

    // Using do-while loop
    i = 1;
    printf("\n\nUsing do-while loop:\n");
    do {
        printf("%d ", i);
        i++;
    } while (i <= 10);
    printf("\n");
    return 0;
}
```

7. Loop Control Statements

LAB EXERCISE: Write a C program that uses the break statement to stop printing numbers when it reaches 5. Modify the program to skip printing the number 3 using the continue statement.

Ans:

```
//break statment
#include <stdio.h>
int main() {
    int i;
    printf("Using break statement:\n");
    for (i = 1; i <= 10; i++) {
        if (i == 5) {
            break;
        }
        printf("%d ", i);
    }
```

```
    return 0;
}
//continue
#include <stdio.h>
int main() {
    int i;
    printf("Using continue statement:\n");
    for (i = 1; i <= 5; i++) {
        if (i == 3) {
            continue;  // Skip the number 3
        }
        printf("%d ", i);
    }
    return 0;
}
```

8. Functions in C

LAB EXERCISE: Write a C program that calculates the factorial of a number using a function. Include function declaration, definition, and call.

Ans:

```
#include <stdio.h>
// Function declaration
int factorial(int n);
int main() {
    int num;
    // Input from user
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    // Function call
    if (num < 0) {
```

```
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        printf("Factorial of %d is %d\n", num, factorial(num));
    }
    return 0;
}

// Function definition
int factorial(int n) {
    int fact = 1;
    for (int i = 1; i <= n; i++) {
        fact *= i;
    }
    return fact;
}
```

9. Arrays in C

LAB EXERCISE: Write a C program that stores 5 integers in a one-dimensional array and prints them. Extend this to handle a two-dimensional array (3x3 matrix) and calculate the sum of all elements.

Ans:

```
#include <stdio.h>
int main() {
    int arr[5] = {10, 20, 30, 40, 50};
    printf("One-Dimensional Array Elements:\n");
    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n\n");
//Two-Dimensional Array (3x3 Matrix) & Sum
    int matrix[3][3] = {
```

```
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
    int sum = 0;
    printf("Two-Dimensional Array (3x3 Matrix):\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matrix[i][j]);
            sum += matrix[i][j];
        }
        printf("\n");
    }
    printf("\nSum of all elements in the matrix: %d\n", sum);
    return 0;
}
```

10. Pointers in C

LAB EXERCISE: Write a C program to demonstrate pointer usage. Use a pointer to modify the value of a variable and print the result.

Ans:

```
#include <stdio.h>
int main() {
    int num = 10;       // Declare an integer variable
    int *ptr;           // Declare a pointer to int
    ptr = &num;         // Assign the address of num to the pointe
    printf("Original value of num: %d\n", num);
    *ptr = 20;          // Modify the value of num using the pointer
    printf("Modified value of num using pointer: %d\n", num);
    return 0;
```

}

Explanation

int *ptr; declares a pointer to an integer.

ptr = # stores the address of num in the pointer.

***ptr = 20; changes the value of num by dereferencing the pointer.**

11. Strings in C

LAB EXERCISE: Write a C program that takes two strings from the user and concatenates them using strcat(). Display the concatenated string and its length using strlen().

Ans:

- Uses `strcat()` to **concatenate** them
- Uses `strlen()` to display the **length** of the concatenated string

Example:

```
#include <stdio.h>

#include <string.h>

int main() {

    char str1[100], str2[100];

    // Input two strings

    printf("Enter the first string: ");

    fgets(str1, sizeof(str1), stdin);

    printf("Enter the second string: ");

    fgets(str2, sizeof(str2), stdin);

    // Remove newline characters if present

    str1[strcspn(str1, "\n")] = '\0';

    str2[strcspn(str2, "\n")] = '\0';

    // Concatenate str2 to str1

    strcat(str1, str2);
```

```
    // Display result
    printf("\nConcatenated string: %s\n", str1);
    printf("Length of concatenated string: %lu\n", strlen(str1));
    return 0;
}
```

12. Structures in C

LAB EXERCISE: Write a C program that defines a structure to store a student's details (name, roll number, and marks). Use an array of structures to store details of 3 students and print them.

Ans:

```
#include <stdio.h>
// Define structure
struct Student {
    char name[50];
    int roll;
    float marks;
};
int main() {
    struct Student students[3];  // Array of 3 students
    // Input student details
    for (int i = 0; i < 3; i++) {
        printf("Enter details for Student %d:\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]", students[i].name);  // Reads full name including spaces
        printf("Roll Number: ");
        scanf("%d", &students[i].roll);
        printf("Marks: ");
        scanf("%f", &students[i].marks);
        printf("\n");
```

```
    }
    // Display student details
    printf("Student Details:\n");
    for (int i = 0; i < 3; i++) {
        printf("Student %d:\n", i + 1);
        printf("Name: %s\n", students[i].name);
        printf("Roll Number: %d\n", students[i].roll);
        printf("Marks: %.2f\n\n", students[i].marks);
    }
    return 0;
}
```

13. File Handling in C

LAB EXERCISE: Write a C program to create a file, write a string into it, close the file, then open the file again to read and display its contents.

Ans:

```
#include <stdio.h>
int main() {
    FILE *file;
    char str[] = "Hello, this is a test string written to the file.";
    char buffer[100];
    // Step 1: Create and open file in write mode
    file = fopen("example.txt", "w");
    if (file == NULL) {
        printf("Error opening file for writing.\n");
        return 1;
    }

    // Step 2: Write string to file
    fprintf(file, "%s", str);
```

```
    // Step 3: Close the file
    fclose(file);

    // Step 4: Open file in read mode
    file = fopen("example.txt", "r");
    if (file == NULL) {
        printf("Error opening file for reading.\n");
        return 1;
    }
    // Step 5: Read and display file content
    printf("Reading from file:\n");
    while (fgets(buffer, sizeof(buffer), file) != NULL) {
        printf("%s", buffer);
    }
    // Close file again
    fclose(file);
    return 0;
}
```