Module 2 – Introduction to Programming

Topic:- Overview of C Programming

LAB Practical

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

Ans :- 1️⃣ Operating Systems

C is used to make operating systems like Windows, Linux, and UNIX.
It helps in writing fast and powerful system software.

📝 **Example: Linux OS is mostly written in C.**

2️⃣ Embedded Systems

C is used in small machines and devices like washing machines, microwave ovens, and medical devices.
It helps control hardware directly.

📝 **Example: C is used to program microcontrollers.**

3️⃣ Game Development

Old and some new games are made using C.
It helps games run fast and work well with computer hardware.

📝 **Example: The old Mario game was made using C.**

✅ **Simple Conclusion:**

C is used in:

- **Operating Systems**
- **Embedded Devices**

- **Games**

Because it is fast, powerful, and works close to the hardware.

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

Ans :- **step 1-install a C Compiler**

For Windows:

- **Install MinGW from this link:**
  https://sourceforge.net/projects/mingw/
- **After installing, go to Environment Variables > add this path:**
  C:\MinGW\bin to System Path

Step 2- Install an IDE (Code Editor)

You can use:

- **Code::Blocks → Best for beginners**
- **Visual Studio Code (VS Code) → Lightweight and powerful**

👉 **Download VS Code: https://code.visualstudio.com**
👉 **Install C/C++ extension in VS Code**

Srep -3 Write Your First C Program

Create a file named hello.c

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

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

Ans :- Variable: A storage that holds data (example: age, marks)

Constant: A fixed value that does not change (example: PI = 3.14)

Data Types:

- **int → for numbers (like 10, 25)**
- **float → for decimal values (like 3.14, 99.5)**
- **char → for single characters (like 'A', 'B')**

Code Output:

```
Value of constant PI = 3.14
Age = 20
Weight = 58.5 kg
Grade = A
```

C Code with Comments:

```c
#include <stdio.h>   // Preprocessor directive

// This program shows use of variables, constants, and different data types

int main() {
    // Declare a constant
    const float PI = 3.14;

    // Declare variables
    int age = 20;           // Integer variable
    float weight = 58.5;    // Float variable
    char grade = 'A';       // Character variable

    // Print all values
    printf("Value of constant PI = %.2f\n", PI);
    printf("Age = %d\n", age);
    printf("Weight = %.1f kg\n", weight);
    printf("Grade = %c\n", grade);

    return 0;
}
```

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

   Ans :-

```c
#include <stdio.h>  // Standard input-output library

int main() {
    int a, b;  // Declare two integer variables

    // Take input from user
    printf("Enter first number: ");
    scanf("%d", &a);  // Store first number in variable 'a'

    printf("Enter second number: ");
    scanf("%d", &b);  // Store second number in variable 'b'

    // Arithmetic Operations
    printf("\n--- Arithmetic Operations ---\n");
    printf("Addition: %d + %d = %d\n", a, b, a + b);       // Addition
    printf("Subtraction: %d - %d = %d\n", a, b, a - b);    // Subtraction
    printf("Multiplication: %d * %d = %d\n", a, b, a * b); // Multiplication
    printf("Modulus: %d %% %d = %d\n", a, b, (b != 0) ? a % b : 0); // Modulus (remainder)

    // Relational Operations
    printf("\n--- Relational Operations ---\n");
    printf("a == b : %d\n", a == b);  // Check if a is equal to b
    printf("a != b : %d\n", a != b);  // Check if a is not equal to b
    printf("a > b  : %d\n", a > b);   // Check if a is greater than b
    printf("a < b  : %d\n", a < b);   // Check if a is less than b
    printf("a >= b : %d\n", a >= b);  // Check if a is greater than or equal to b
    printf("a <= b : %d\n", a <= b);  // Check if a is less than or equal to b

    // Logical Operations
    printf("\n--- Logical Operations ---\n");
    printf("a && b : %d\n", a && b);  // Logical AND: true if both a and b are non-zero
    printf("a || b : %d\n", a || b);  // Logical OR: true if any one of a or b is non-zero
    printf("!a     : %d\n", !a);      // Logical NOT: true if a is zero

    return 0;  // End of program
}
```

Out put

```
--- Arithmetic Operations ---
Addition: 10 + 5 = 15
Subtraction: 10 - 5 = 5
Multiplication: 10 * 5 = 50
Modulus: 10 % 5 = 0

--- Relational Operations ---
a == b : 0
a != b : 1
a > b  : 1
a < b  : 0
a >= b : 1
a <= b : 0

--- Logical Operations ---
a && b : 1
a || b : 1
!a     : 0
```

5. **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 :- code

```
#include <stdio.h>  // Standard input-output library

int main() {
   int number;  // Variable to store user input number
   int month;   // Variable to store month number

   // --- Part 1: Check Even or Odd ---
   printf("Enter a number to check even or odd: ");
   scanf("%d", &number);  // Take number input from user

   if (number % 2 == 0) {
      printf("%d is Even\n", number);  // If number divisible by 2
   } else {
      printf("%d is Odd\n", number);   // If not divisible by 2
   }

   // --- Part 2: Switch Case for Month ---
   printf("\nEnter a number (1 to 12) to get month name: ");
   scanf("%d", &month);  // Take month input

   switch (month) {
      case 1:
         printf("Month: January\n");
         break;
      case 2:
         printf("Month: February\n");
         break;
      case 3:
         printf("Month: March\n");
         break;
      case 4:
         printf("Month: April\n");
         break;
      case 5:
         printf("Month: May\n");
         break;
      case 6:
         printf("Month: June\n");
```

```
            break;
        case 7:
            printf("Month: July\n");
            break;
        case 8:
            printf("Month: August\n");
            break;
        case 9:
            printf("Month: September\n");
            break;
        case 10:
            printf("Month: October\n");
            break;
        case 11:
            printf("Month: November\n");
            break;
        case 12:
            printf("Month: December\n");
            break;
        default:
            printf("Invalid month number! Please enter 1 to 12.\n");
    }

    return 0;  // End of program
}
```

- **Output**

```
Enter a number to check even or odd: 7
7 is Odd

Enter a number (1 to 12) to get month name: 3
Month: March
```

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

Ans. :-

```c
#include <stdio.h>  // Standard Input Output library

int main() {
    int i;  // Variable for looping

    // ---------- WHILE LOOP ----------
    printf("Using while loop:\n");
    i = 1;  // Initialize i
    while (i <= 10) {           // Loop runs while i is less than or equal to 10
        printf("%d ", i);       // Print the current value of i
        i++;                    // Increase i by 1
    }

    printf("\n\n");  // Print a newline for spacing

    // ---------- FOR LOOP ----------
    printf("Using for loop:\n");
    for (i = 1; i <= 10; i++) {  // Initialization; condition; increment
        printf("%d ", i);        // Print numbers 1 to 10
    }

    printf("\n\n");

    // ---------- DO-WHILE LOOP ----------
    printf("Using do-while loop:\n");
    i = 1;  // Initialize i
    do {
        printf("%d ", i);       // Print the current value of i
        i++;                    // Increase i by 1
    } while (i <= 10);          // Check condition after running the loop at least once

    return 0;  // End of program
}
```

Output :-

```
Using while loop:
1 2 3 4 5 6 7 8 9 10

Using for loop:
1 2 3 4 5 6 7 8 9 10

Using do-while loop:
1 2 3 4 5 6 7 8 9 10
```

7. **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 :-

C Code with Explanation:-

```c
#include <stdio.h>  // Include standard input-output library

int main() {
    int i;  // Loop variable

    // ----------- Part 1: Using break -----------
    printf("Using break statement (stop at 5):\n");
    for (i = 1; i <= 10; i++) {
        if (i == 5) {
            break;  // Stop the loop when i is 5
        }
        printf("%d ", i);  // Print the number
    }

    printf("\n\n");  // Print newline for spacing

    // ----------- Part 2: Using continue -----------
    printf("Using continue statement (skip 3):\n");
    for (i = 1; i <= 10; i++) {
        if (i == 3) {
            continue;  // Skip the rest of loop when i is 3
        }
        printf("%d ", i);  // Print the number
    }

    return 0;  // End of program
}
```

Output:-

```
Using break statement (stop at 5):
1 2 3 4

Using continue statement (skip 3):
1 2 4 5 6 7 8 9 10
```

8. **Write a C program that calculates the factorial of a number using a function. Include function declaration, definition, and call.**

Ans. :-

```c
#include <stdio.h>  // For input and output functions

// Function Declaration
int factorial(int n);

int main() {
    int num, result;

    // Ask the user to enter a number
    printf("Enter a positive number: ");
    scanf("%d", &num);

    // Function Call
    result = factorial(num);

    // Display the result
    printf("Factorial of %d is %d\n", num, result);

    return 0;
}

// Function Definition
int factorial(int n) {
    int i, fact = 1;

    // Multiply numbers from 1 to n
    for (i = 1; i <= n; i++) {
        fact = fact * i;
    }

    return fact;  // Return the final result
}
```

Output :-

```
Enter a positive number: 5
Factorial of 5 is 120
```

9. **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 :-

```c
#include <stdio.h>  // For input and output

int main() {
    // ---------- Part 1: One-dimensional array ----------
    int arr[5];  // Declare 1D array of size 5
    int i;

    printf("Enter 5 integers:\n");
    for (i = 0; i < 5; i++) {
        scanf("%d", &arr[i]);  // Take input in array
    }

    printf("You entered:\n");
    for (i = 0; i < 5; i++) {
        printf("%d ", arr[i]);  // Print array elements
    }

    printf("\n\n");

    // ---------- Part 2: Two-dimensional array (3x3) ----------
    int matrix[3][3];  // Declare 2D array (matrix)
    int row, col, sum = 0;

    printf("Enter 9 elements for 3x3 matrix:\n");
    for (row = 0; row < 3; row++) {
        for (col = 0; col < 3; col++) {
            scanf("%d", &matrix[row][col]);  // Input elements
        }
    }

    printf("Matrix is:\n");
    for (row = 0; row < 3; row++) {
        for (col = 0; col < 3; col++) {
            printf("%d ", matrix[row][col]);  // Print matrix
            sum += matrix[row][col];          // Add to sum
        }
        printf("\n");
    }

    printf("Sum of all elements in matrix = %d\n", sum);

    return 0;
}
```

SIMPLE INPUT

```
1 2 3
4 5 6
7 8 9
```

Output:

```
Matrix is:
1 2 3
4 5 6
7 8 9
Sum of all elements in matrix = 45
```

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

Ans :-

```c
#include <stdio.h>

int main() {
    int num = 10;       // A normal integer variable
    int *ptr;           // Pointer declaration

    ptr = &num;         // Store the address of num in pointer ptr

    printf("Before changing value:\n");
    printf("num = %d\n", num);        // Print original value
    printf("Address of num = %p\n", ptr);  // Print address stored in pointer
    printf("Value at ptr = %d\n", *ptr);   // Print value at that address

    *ptr = 20;          // Modify value of num using pointer

    printf("\nAfter changing value using pointer:\n");
    printf("num = %d\n", num);        // num is now changed to 20
    printf("Value at ptr = %d\n", *ptr);   // Also shows 20

    return 0;
}
```

Output :-

```
Before changing value:
num = 10
Address of num = 0x7ffeeb1f2a9c  (this will vary)
Value at ptr = 10

After changing value using pointer:
num = 20
Value at ptr = 20
```

Term	Meaning
int *ptr;	Declares a pointer to int
ptr = #	Stores address of num in pointer
*ptr	Means “value at the address” (dereference)
*ptr = 20;	Changes value of num to 20 using pointer

11. 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 :-

```c
#include <stdio.h>
#include <string.h>  // For strcat() and strlen()

int main() {
    char str1[100], str2[100];  // Declare two character arrays
    char result[200];           // Final string after concatenation

    // Input first string
    printf("Enter first string: ");
    gets(str1);  // Take input for str1 (can also use fgets)

    // Input second string
    printf("Enter second string: ");
    gets(str2);  // Take input for str2

    // Copy str1 into result to start
    strcpy(result, str1);      // Copy str1 into result
    strcat(result, str2);      // Add str2 at the end of result

    // Display the final concatenated string
    printf("Concatenated String: %s\n", result);

    // Display the length of the final string
    printf("Length of Concatenated String: %lu\n", strlen(result));

    return 0;
}
```

Output :-

```
Enter first string: Hello
Enter second string: World

Concatenated String: HelloWorld
Length of Concatenated String: 10
```

12. **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 :-

```c
#include <stdio.h>

// Structure definition
struct Student {
    char name[50];
    int roll;
    float marks;
};

int main() {
    struct Student s[3];  // Array of 3 structure variables
    int i;

    // Input details for 3 students
    printf("Enter details of 3 students:\n");
    for (i = 0; i < 3; i++) {
        printf("\nStudent %d:\n", i + 1);

        printf("Enter name: ");
        scanf("%s", s[i].name);  // Get student's name

        printf("Enter roll number: ");
        scanf("%d", &s[i].roll);  // Get roll number

        printf("Enter marks: ");
        scanf("%f", &s[i].marks);  // Get marks
    }

    // Print all details
    printf("\nStudent Details:\n");
    for (i = 0; i < 3; i++) {
        printf("\nStudent %d:\n", i + 1);
        printf("Name: %s\n", s[i].name);
        printf("Roll Number: %d\n", s[i].roll);
        printf("Marks: %.2f\n", s[i].marks);
    }

    return 0;
}
```

Input :-

```
Enter name: Jay
Enter roll number: 1
Enter marks: 85.5

Enter name: Ravi
Enter roll number: 2
Enter marks: 90

Enter name: Neha
Enter roll number: 3
Enter marks: 88
```

Output :-

```
Student Details:

Student 1:
Name: Jay
Roll Number: 1
Marks: 85.50

Student 2:
Name: Ravi
Roll Number: 2
Marks: 90.00

Student 3:
Name: Neha
Roll Number: 3
Marks: 88.00
```

13. 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:-

```c
#include <stdio.h>

int main() {
    FILE *fp;                // Declare file pointer
    char str[100];           // String to write and read

    // ---------- Step 1: Write to File ----------
    fp = fopen("myfile.txt", "w");  // Open file in write mode

    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    printf("Enter a string to write into the file:\n");
    gets(str);                      // Get input from user

    fprintf(fp, "%s", str);         // Write string to file
    fclose(fp);                     // Close the file

    printf("Data written to file successfully.\n");

    // ---------- Step 2: Read from File ----------
    fp = fopen("myfile.txt", "r");  // Open file in read mode

    if (fp == NULL) {
        printf("Error opening file for reading!\n");
        return 1;
    }

    printf("\nReading from file:\n");
    fgets(str, 100, fp);            // Read from file
    printf("File content: %s\n", str);  // Display content

    fclose(fp);                     // Close the file
    return 0;
}
```

Output:-

```
Enter a string to write into the file:
Hello, this is Jay!

Data written to file successfully.

Reading from file:
File content: Hello, this is Jay!
```