Module 2 – Introduction to Programming

Overview of C Programming

Q-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.

Ans-

GCC Installation (Windows):

- Download MinGW → Install → Add C:\MinGW\bin to system Path → Run gcc --version in CMD to check.

DevC++:

- Download from SourceForge → Install → Open → Write code → Compile & Run.

Code:Blocks:

- Download version with MinGW → Install → New Project → Write code → Build & Run.

VS Code:

- Install VS Code → Add C/C++ extension → Install GCC → Write code → Use terminal to compile with gcc.

Q-2-Explain the basic structure of a C program, including headers, main function, comments, data types, and variables. Provide examples.

Ans-

1. Header File
#include <stdio.h> – Includes built-in functions like printf() and scanf().

2. **Main Function**
int main() { ... } – Every C program starts from the main() function.

3. Comments
// single-line or /* multi-line */

4. **Data Types**
Define type of data:

- int for integers
- float for decimal numbers
- char for characters

5. Variables
Named memory locations to store data.
Example: int age = 20;

Exmaple-

```
#include <stdio.h>   // Header file

   int main()// Main function

  {
```

```
    int age = 20;     // Variable (int type)
    float pi = 3.14;  // Variable (float type)
    char grade = 'A'; // Variable (char type)
      printf("Age: %d", age);  // Output
    return 0;              // End of program
}
```

Q-3-Write notes explaining each type of operator in C: arithmetic, relational, logical, assignment, increment/decrement, bitwise, and conditional operators.

Ans-

1.Arithmetic Operators:

Used for mathematical calculations.

Operator	Meaning	Example
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
%	Modulus	a % b (remainder)

2. Relational Operators

Used to compare two values.

Operator	Meaning	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater or equal	a >= b
<=	Less or equal	a <= b

3. Logical Operators

Used to combine multiple conditions.

Operator	Meaning	Example
&&	Logical AND	a > 0 && b < 10
||	Logucal OR	a>0 || b<10
!	Logical NOT	!(a > b)

4. Assignment Operators

Used to assign values to variables.

Operator	Meaning	Example
=	Assign value	a = 10
+=	Add and assign	a += 5 → a = a + 5
-=	Subtract and assign	a -= 2
*=	Multiply and assign	a *= 3
/=	Divide and assign	a /= 2

5. Increment/Decrement Operators

Used to increase or decrease value by 1.

Operator	Meaning	Example
++	Increment	a++ or ++a
--	Decrement	a-- or --a

6. Bitwise Operators

Operate on bits (binary level).

Operator	Meaning	Example
&	AND	a & b

Operator	Meaning	Example
|	OR	a|b
^	XOR	a ^ b
~	NOT	~a
<<	Left Shift	a << 1
>>	Right Shift	a >> 1

7. Conditional (Ternary) Operator

Short form of if...else.

(condition) ? true_value : false_value

Example:

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

Q-4- Explain decision-making statements in C (if, else, nested if-else, switch). Provide examples of each.

Ans-

1. if Statement

Used to execute code **only if** a condition is true.

Syntax:

```
if (condition) {
   // Code runs if condition is true
```

}

Example:

```
int age = 18;
if (age >= 18) {
    printf("You are eligible to vote.");
}
```

2. if-else Statement

Used to choose between two options.

Syntax:

```
if (condition) {
    // Code if condition is true
} else {
    // Code if condition is false
}
```

Example:

```
int marks = 40;
if (marks >= 35) {
    printf("Pass");
} else {
    printf("Fail");
}
```

3. Nested if-else

An if or else block contains another if-else.

Syntax:

```
if (condition1) {
    if (condition2) {
        // Code if both are true
    } else {
        // Code if condition1 true but condition2 false
    }
} else {
    // Code if condition1 is false
}
```

Example:

```
int marks = 85;
if (marks >= 35) {
    if (marks >= 75) {
        printf("Distinction");
    } else {
        printf("Pass");
    }
} else {
    printf("Fail");
}
```

4. switch Statement

Used when you want to choose between multiple fixed options (like menu).

Syntax:

```
switch (expression) {
    case value1:
        // Code
        break;
    case value2:
        // Code
        break;
    default:
        // Code if no match
}
```

Example:

```
int day = 3;
switch (day) {
    case 1: printf("Monday"); break;
    case 2: printf("Tuesday"); break;
    case 3: printf("Wednesday"); break;
    default: printf("Invalid day");
}
```

Q-5-What are functions in C? Explain function declaration, definition, and how to call a function. Provide examples.

Ans-

What Are Functions in C?

A function in C is a block of code that performs a specific task. It helps to divide a large program into smaller, manageable parts and supports code reuse.

1. Function Declaration

It tells the compiler about the function's name, return type, and parameters before it is used.

Example:

```
int add(int a, int b);
```

2. Function Definition

It contains the actual code or body of the function.

Example:

```
int add(int a, int b) {
   return a + b;
}
```

3. Function Call

Used to **execute** the function from main() or another function.

Example:

```
int result = add(5, 3);  // Function call
```

Complete Example:

```
#include <stdio.h>
int add(int a, int b);
int add(int a, int b) {
    return a + b;
}
int main() {
    int result = add(10, 20);
    printf("Sum: %d", result);
    return 0;
}
```

Q-6-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 is a collection of similar data types stored in contiguous memory locations. It allows storing multiple values using a single variable name and accessing them using index numbers.

One-Dimensional Array:

A 1D array stores a list of elements in a single row or linear form.

Example:

int arr[3] = {10, 20, 30};

Here, arr[0] = 10, arr[1] = 20, and arr[2] = 30

Multi-Dimensional Array:

A multi-dimensional array is an array of arrays. The most common is a 2D array, like a table (rows × columns).

Example:

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

Here, matrix[0][0] = 1, matrix[1][1] = 4

Q-7-Explain the importance of file handling in C. Discuss how to perform file operations like opening, closing, reading, and writing files.

Ans-

Importance of File Handling in C:

File handling in C allows a program to store and retrieve data permanently from files on disk. Without file handling, data is lost once the program ends. It is essential for tasks like saving user data, logs, reports, etc.

File Operations in C:

1. Opening a File

Use the fopen() function to open a file in a specific mode (read, write, append, etc.).

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

2. Closing a File

Use the fclose() function to close the file and free resources.

fclose(fp);

3. Reading from a File

Functions like fgetc(), fgets(), or fread() are used to read data from a file.

char ch = fgetc(fp);

4. Writing to a File

Use functions like fprintf(), fputc(), or fwrite() to write data to a file.

fprintf(fp, "Hello, world!");

	Common Mode:
"r"	Read (file must exist)
"w"	Write (creates new or overwrites)
"a"	Append (adds to end, if exists)
"r+"	Read and write
"w+"	Write and read