.....Module-2, :- Theory.....

Q.1 what is c programing.History and Evolution ?

Ans:- C Programming :-

- **C** is a **high-level language** made in the **1970** by **Dennis Ritchie**.
- It is used to make system software like operating systems and embedded systems.
- C is fast and works on many devices.
- It can directly access memory and control hardware.
- Still used today because it is powerful and efficient.

➡ **History and Evolution :-**

C programming was created by **Dennis Ritchie in 1972 at Bell Labs**. It was made for system programming. especially for the Unix operating system. Later, it became a standard language called ANSI C. New versions like C99 and C11 added more features. C is still important today because it is fast**,** powerful, and is the base for many other languages.

Q .2 Describe the steps to install a C compiler (e.g., GCC) Choosing an IDE (Integrated Development Environment) DevC++. VS Code, Code blocks ?

Ans:- To install a C compiler :-

- Linux: Run `sudo apt-get install gcc` in the terminal.
- Windows: Install MinGW to get GCC.

After installing, make sure GCC is added to your system PATH so you can run it from the command line.

➡ Choose an IDE to write C programs:-

- **DevC++** – Simple and beginner-friendly
- **VS Code** – Modern, with many features and extensions
- **Code Blocks** – Good for C/C++ with built-in compiler support

Pick one based on your comfort and system.

Q .3 Explain the basic structure of a C program. comments data types and variables. Provide example?

Ans:- A C program has three main parts:

1. **Preprocessor Directives** – Like #include, used to add libraries before compiling.
2. **Main Function** – int main() is where the program starts.
3. **Functions and Statements** – Code that performs tasks, ending with return 0; in main.

1. **Comments** – Used to explain code.

```
// This is a comment
```

2. **Data Types** – Define type of data.

- int → numbers
- float → decimal
- char → character

3. **Variables** – Store values.

```
int age = 18;
float pi = 3.14;
char grade = 'A';
```

Q .4 Write notes explaining each type of operator in C: arithmetic, relational, logical, assignment, increment/decrement, bitwise, and conditional operators.

Ans:- **1. Arithmetic**

+, -, *, /, % → Math operations.

2. Relational

==, !=, >, <, >=, <= → Compare values.

3. Logical

&&, ||, ! → Combine conditions (true/false).

4. Assignment

=, +=, -=, *= → Assign values.

5. Increment/Decrement

++, -- → Increase or decrease by 1.

6. Bitwise

&, |, ^, ~, <<, >> → Work on bits.

7. Conditional

? : → Short form of if-else.

Q5. Explain decision-making statements in C (if, else, nested if-else, switch). Provide examples of each.

Ans:-

C uses condition statements to make decisions.

1. **if:**

```c
int x = 10;
if (x > 5) {
    printf("x is greater than 5");
}
```

2.if-else:

```c
if (x % 2 == 0) {
    printf("Even");
} else {
    printf("Odd");
}
```

3.Nested if-else:

```c
if (x >= 0) {
    if (x == 0)
        printf("Zero");
    else
        printf("Positive");
} else {
    printf("Negative");
}
```

4 switch:

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

Q6. Compare and contrast while loops, for loops, and do-while loops. Explain the scenarios in which each loop is most appropriate.

Ans:-

Loop Type	When to Use	Runs At Least Once	Example
for	Known number of times	No	for(int i=0; i<5; i++)
while	Unknown condition, check before start	No	while(i<5)
do-while	Must run at least once	Yes	do { } while(i<5);

➡ **Example for all:**

```c
// for loop
for(int i=0; i<5; i++) {
    printf("%d ", i);
}

// while loop
int i = 0;
while(i < 5) {
    printf("%d ", i);
    i++;
}

// do-while loop
int j = 0;
do {
    printf("%d ", j);
    j++;
} while(j < 5);
```

Q7. Explain loop control statements (break, continue, and goto) in C with examples.

Ans:-

1. **break: stops the loop**

```c
for(int i=0; i<10; i++) {
    if(i == 5) break;
    printf("%d ", i);
}
```

2. **continue: skips current iteration**

```c
for(int i=0; i<5; i++) {
    if(i == 2) continue;
    printf("%d ", i);
}
```

3. **goto: jumps to a label**

```c
int x = 1;
goto skip;
printf("This won't print");
skip:
printf("Jumped here");
```

Q8. What is a function in C? Explain the concept of function declaration, definition, and calling.

Ans:-

A function is a block of code that performs a task.

- **Declaration – tells the compiler about the function.**
- **Definition – actual code inside the function.**
- **Calling – running the function.**

➡ **Example:**

```c
// Declaration
int add(int, int);

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

// Calling
int result = add(5, 10);
printf("Sum is %d", result);
```

Q9. What are arrays in C? Explain with examples of single and multi-dimensional arrays.

Ans:-

An array stores multiple values of the same data type in one variable.

1D Array Example:

```c
int numbers[5] = {1, 2, 3, 4, 5};
printf("%d", numbers[0]);
```

2D Array Example:

```c
int matrix[2][2] = {{1, 2}, {3, 4}};
printf("%d", matrix[0][1]);  // Output: 2
```

Q10. Explain pointers in C and their importance. Provide a simple example.

Ans:-

A pointer stores the address of another variable.
They are used in dynamic memory, arrays, functions, etc.

➡ **Example:**

```c
int a = 10;
int *p = &a;
printf("Value = %d", *p);   // Output: 10
```

Q11. Describe string handling in C using common string functions.

Ans:-

Strings are arrays of characters ending with \0.

Common string functions:

- **strlen() – returns length**
- **strcpy() – copies one string into another**
- **strcat() – joins two strings**
- **strcmp() – compares two strings**

Example:

```c
char str1[20] = "Hello";
char str2[20] = "World";

strcat(str1, str2);
printf("%s", str1);  // Output: HelloWorld
```

Q12. What is a structure in C? How does it differ from an array?

Ans:-

A structure is used to group different data types.

Difference from Array:

Array	Structure
Same data type	Different data types
int marks[5];	struct Student { int id; char name[20]; };

➡ **Example:**

```c
struct Student {
    int id;
    char name[20];
};

struct Student s1 = {1, "Jay"};
printf("%d %s", s1.id, s1.name);
```

Q13. Explain file handling in C. Describe how to open, read, write, and close a file.

Ans:-

In C, we use FILE type and functions to work with files.

Common Functions:

- **fopen() – open file**
- **fprintf() / fputs() – write**
- **fscanf() / fgets() – read**
- **fclose() – close file**

➡ **Example:**

```
FILE *fp;
fp = fopen("data.txt", "w");
fprintf(fp, "Hello File!");
fclose(fp);
```