Name: Paramprakash Makwana

Module #3 Introduction to OOPS Programming(Theory)

3.1) What are the key differences between Procedural Programming and Object-Oriented Programming (OOP)?

Ans:- POP follows a step-by-step, function-based approach, while OOP is based on objects and classes.

POP focuses on functions, OOP focuses on data security, reusability, and modularity.

3.2) List and explain the main advantages of OOP over POP.

Ans:- OOP provides data hiding, modularity, and code reusability using classes and objects.

Supports inheritance and polymorphism, which reduce code duplication and improve flexibility.

3.3) Explain the steps involved in setting up a C++ development environment.

Ans:- Install a compiler like GCC/MinGW and an IDE like CodeBlocks/VS Code.

Configure compiler paths in the IDE and create a new C++ project or file.

3.4) What are the main input/output operations in C++? Provide examples.

Ans:- Input**:** cin >> variable; → takes user input.

Output**:** cout << "Message"; → displays output on screen.

3.5) What are the different data types available in C++? Explain with examples.

Ans:- Basic types**:** int, float, char, bool → int a = 10;

Derived/User-defined**:** arrays, pointers, structures, classes → struct A { int x; };

3.6) Explain the difference between implicit and explicit type conversion in C++.

Ans:- Implicit: Done automatically by the compiler → int a = 5; float b = a;

Explicit: Manually done by the programmer → float b = (float)5;

3.7) What are the different types of operators in C++? Provide examples of each.

Ans:- Arithmetic: + - * / % → a + b

Relational: == != > < → a > b

Logical: && || ! → a && b

Assignment: = += -= → a += 5

Unary: ++ -- → ++a

Ternary: ?: → (a>b ? a : b)

Bitwise: & | ^ << >> → a & b

3.8) Explain the purpose and use of constants and literals in C++.

Ans:- Constants: Fixed values that cannot change during execution → const int x = 10;

Literals: Actual data values written directly in code → 20, 'A', 3.14.

3.9) What are conditional statements in C++? Explain the if-else and switch statements.

Ans:- Conditional statements control program flow based on conditions.

if-else checks conditions one by one, switch selects a case based on a single value.

3.10) What is the difference between for, while, and do-while loops in C++?

Ans:- for loop is used when the number of iterations is known.

while/do-while run until a condition becomes false; do-while executes at least once.

3.11) How are break and continue statements used in loops? Provide examples.

Ans:- break exits the loop immediately → if(x==5) break;

continue skips the remaining code of the current iteration → if(x==3) continue;

3.12) Explain nested control structures with an example.

Ans:- A loop or condition inside another loop/condition is called nested.

Example: a loop inside a loop → printing a pattern using for(i) inside for(j).

3.13) 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 specific task.

Declaration tells the compiler about the function, definition contains the code, and calling executes it.

3.14) What is the scope of variables in C++? Differentiate between local and global scope.

Ans:- Local variables exist inside a function/block and are accessible only there.

Global variables are declared outside all functions and can be accessed anywhere in the program.

3.15) Explain recursion in C++ with an example.

Ans:- Recursion is when a function calls itself to solve smaller parts of a problem.

Example: factorial using recursion → fact(n) = n * fact(n-1);

3.16) What are function prototypes in C++? Why are they used?

Ans:- A function prototype is a declaration that specifies the function's name, return type, and parameters.

It is used so the compiler knows the function before it is called.

3.17) What are arrays in C++? Explain the difference between single-dimensional and multi- dimensional arrays.

Ans:- An array is a collection of elements of the same type stored in contiguous memory.

1D array is a single list, while multi-dimensional (like 2D) represents rows and columns.

3.18) Explain string handling in C++ with examples.

Ans:- C++ handles strings using char arrays or the string class.

Example: string s = "Hello"; or char a[] = "Hi";

3.19) How are arrays initialized in C++? Provide examples of both 1D and 2D arrays.

Ans:- 1D**:** int a[3] = {1, 2, 3};

2D**:** int b[2][2] = {{1, 2}, {3, 4}};

3.20) Explain string operations and functions in C++.

Ans:- Common operations: concatenation, length, compare, uppercase, lowercase, reverse, copy.

Useful functions: cat(), length(), compare(), toupper(), tolower(), reverse(), copy().

3.21) Explain the key concepts of Object-Oriented Programming (OOP).

Ans:- OOP is based on Abstraction, Encapsulation, Inheritance, and Polymorphism.

These concepts help in organizing code, reusing it, and securing data.

3.22) What are classes and objects in C++? Provide an example.

Ans:- A class is a blueprint, and an object is an instance of that class.

Example: class Car { }; Car c1;

3.23) What is inheritance in C++? Explain with an example.

Ans:- Inheritance allows a class to acquire properties of another class.

Example: class B : public A { }; (B inherits A)

3.24) What is encapsulation in C++? How is it achieved in classes?

Ans:- Encapsulation means binding data and functions together and hiding data from outside.

Achieved using private/protected members and public functions in a class.