Assignment-3

Module-3 Introduction to OOPS Programming

Theory answers

1. **What are the key differences between Procedural Programming and ObjectOrientedProgramming (OOP)?**

   Key differences between Procedural Programming and Object-Oriented Programming (OOP):

- Approach:
  Procedural focuses on *functions and procedures*, while OOP focuses on *objects and classes*.
- Data Handling:
  In Procedural, data is *global and unprotected*; in OOP, data is *encapsulated* within classes.
- Reusability:
  Procedural code has less reusability, while OOP supports *inheritance* and *polymorphism* for code reuse.

2. **List and explain the main advantages of OOP over POP.**
   Main advantages of OOP over POP:
- Encapsulation: Protects data by hiding implementation details.
- Reusability: Inheritance allows reuse of existing code.
- Maintainability and Modularity: Easier to update, debug, and manage complex systems.

3. **Explain the steps involved in setting up a C++ development environment.**
   Steps involved in setting up a C++ development environment:

1. Install a compiler (e.g., GCC or MinGW for Windows).
2. Install an IDE or editor (e.g., Code::Blocks, Dev-C++, or Visual Studio Code).
3. Write, compile, and run a simple C++ program to verify setup.

4. **What are the main input/output operations in C++? Provide examples.**

   Input: `cin` → reads data from the user.

   ```
   int age; cin >> age;
   ```

   Output: `cout` → displays data on the screen.

   ```
   cout << "Age: " << age;
   ```

5. **What are the different data types available in C++? Explain with examples.**

   C++ supports the following main data types:

1. Basic Data Types:
   - int → stores integers (e.g., int age = 20;)
   - float, double → store decimal values (e.g., float pi = 3.14;)
   - char → stores single character (e.g., char grade = 'A';)
   - bool → stores true/false values (e.g., bool pass = true;)
2. Derived Data Types:
   - Arrays, Functions, Pointers, References
3. User-defined Data Types:
   - struct, class, enum, union

   These types help store and manipulate different kinds of data efficiently.

6. **Explain the difference between implicit and explicit type conversion in C++.**

1. Implicit Type Conversion (Type Promotion):
   - Done automatically by the compiler when different data types are used in an expression.
   - Example:
   - int a = 5;
   - float b = a + 2.5; // int → float automatically
2. Explicit Type Conversion (Type Casting):
   - Done manually by the programmer using casting.
   - Example:
   - float x = 5.7;
   - int y = (int)x; // float → int by user

   Difference: Implicit is automatic, while explicit requires user-defined conversion.

7. **What are the different types of operators in C++? Provide examples of each.**

C++ provides several types of operators:

1. Arithmetic Operators: +, -, *, /, %
   → Example: int c = a + b;
2. Relational Operators: ==, !=, >, <, >=, <=
   → Example: if (a > b)
3. Logical Operators: &&, ||, !
   → Example: if (a > 0 && b > 0)
4. Assignment Operators: =, +=, -=, *=, /=
   → Example: x += 5;
5. Bitwise Operators: &, |, ^, ~, <<, >>
   → Example: c = a & b;
6. Increment/Decrement Operators: ++, --
   → Example: i++;
7. Conditional Operator: ?:
   → Example: max = (a > b) ? a : b;

8. **Explain the purpose and use of constants and literals in C++.**

- Constants are fixed values that do not change during program execution. Declared using const keyword or #define.
  Example:
- const float PI = 3.14;
- #define MAX 100
- Literals are the actual constant values used directly in code.
  Example: 10, 'A', 3.14, "Hello"
  Purpose: They make programs more readable, prevent accidental changes, and improve code reliability.

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

1. Conditional Statements (3 Marks):
Conditional statements control the flow based on conditions.

- if-else: Executes one block if the condition is true, else another.
- if (x > 0) cout << "Positive";
- else cout << "Negative";

- switch: Used for multiple choices based on a variable's value.
- switch(ch) {
- case 1: cout << "One"; break;
- default: cout << "Other";
  }

9. 2. What is the difference between for, while, and do-while loops in C++?

- for loop: Used when number of iterations is known.
- for(int i=0; i<5; i++) cout << i;
- while loop: Condition checked before execution.
- while(i<5) { cout << i; i++; }
- do-while loop: Executes at least once since condition is checked later.
  do { cout << i; i++; } while(i<5);

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

- break: Exits loop immediately.
- for(int i=1; i<=5; i++){ if(i==3) break; cout<<i; }
- continue: Skips current iteration and moves to next.
  for(int i=1; i<=5; i++){ if(i==3) continue; cout<<i; }

11. Explain nested control structures with an example.

12. **4. Nested Control Structures (3 Marks):**
    When one control structure is placed inside another.
    Example:

```
13.  for(int i=1; i<=3; i++){
14.    for(int j=1; j<=i; j++)
15.      cout << "* ";
16.    cout << endl;
17.  }
```

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

A function is a block of code that performs a specific task and can be reused.

- Declaration: Tells the compiler about the function's name, return type, and parameters.

- int add(int, int);
- Definition: Contains the actual code.
- int add(int a, int b){ return a + b; }
- Calling: Executes the function.

int sum = add(5, 3);

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

Scope defines where a variable can be accessed.

- Local Scope: Declared inside a function or block; accessible only there.
- void func(){ int x = 10; }
- Global Scope: Declared outside all functions; accessible everywhere.

int x = 10; void func(){ cout << x; }

14. Explain recursion in C++ with an example.

Recursion is when a function calls itself to solve a smaller subproblem.
Example:

```
int fact(int n){
  if(n==0) return 1;
  else return n * fact(n-1);
}
```

Here, fact() calls itself until n becomes 0.

15. What are function prototypes in C++? Why are they used?

A function prototype declares the function before its use, helping the compiler check for correct usage.
Example:

int add(int, int); // prototype

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

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

- 1D Array: Single row of elements.
- int arr[5] = {1,2,3,4,5};
- 2D Array: Elements in rows and columns.
- int mat[2][3] = {{1,2,3},{4,5,6}};

17. Explain string handling in C++ with examples.

C++ supports C-style strings (char arrays) and string class.

- C-style:
- char str[] = "Hello";
- string class:
- #include <string>
- string s = "Hello";

cout << s.length();

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

- 1D Array:
- int arr[5] = {1,2,3,4,5};
- 2D Array:
  int mat[2][3] = {{1,2,3},{4,5,6}};

19. Explain string operations and functions in C++.

Common operations using string class:

- Concatenation: s1 + s2

- Length: s.length()
- Access Character: s[i]
- Substring: s.substr(0,3)
- Comparison: s1 == s2
  Example:
  string s1 = "Hello", s2 = "World";
  string s3 = s1 + " " + s2; // "Hello World"

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

 Encapsulation: Hiding data and providing access through functions.
 Inheritance: Reusing properties of an existing class in a new class.
 Polymorphism: Same function behaves differently (overloading/overriding).
 Abstraction: Showing essential features while hiding details.
 Classes & Objects: Blueprint and instance of a class.

21. What are classes and objectsin C++? Provide an example.

- **Class:** Blueprint defining attributes and methods.
- **Object:** Instance of a class.
  Example:

```
class Car {
  public: string brand;
  void display() { cout << brand; }
};
Car c1;
c1.brand = "Toyota";
c1.display();
```

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

Mechanism to create a new class from an existing class.
Example:

```
class Person { public: string name; };
class Student : public Person { public: int roll; };
Student s; s.name="Rushi"; s.roll=101;
```

23. What is encapsulation in C++? How isit achieved in classes?

Data is hidden inside a class using private members and accessed via public methods.

Example:

```
class Bank { private: double balance;
public: void deposit(double amt){ balance += amt; }
      double getBalance(){ return balance; } };
```