Module -3

(Introduction to OOPS Programming)

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

Feature	Procedural Programming (POP)	Object-Oriented Programming (OOP)
Structure	Program divided into functions/procedures.	Program organized around objects and classes.
Data Handling	Data and functions are separate. Global data often used.	Data and methods are encapsulated within objects.
Approach	Follows a Top-Down approach.	Follows a Bottom-Up approach.
Modularity	Achieved through functions.	Achieved through classes and objects.
Reusability	Limited code reusability.	High code reusability through inheritance.
Data Security	Less secure; data can be easily accessed.	More secure due to data hiding (encapsulation).
Real-world Modeling	Less emphasis on real-world modeling.	Excellent for modeling real-world entities.
Key Concepts	Focus on procedures and sequence of actions.	Focus on objects, classes, inheritance, polymorphism, encapsulation, abstraction.

2. List and explain the main advantages of OOP over POP.

Advantages of OOP:

1. Code Reusability: Through inheritance.
2. Data Security: Encapsulation protects data.
3. Easy Maintenance: Modular structure makes debugging simple.

4. Polymorphism: Same function name can be used for different tasks.
5. Real-World Mapping: Objects represent real-world entities.

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

Steps:
1. Install a C++ compiler (e.g., GCC, MinGW, Turbo C++).
2. Install an IDE (e.g., Code::Blocks, Visual Studio, Dev-C++).
3. Configure the compiler in the IDE.
4. Create a new project or file.
5. Write, compile, and run the program.

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

In C++, input is handled by `cin` and output by `cout`.
Example:
```cpp
#include <iostream>
using namespace std;
int main() {
    int num;
    cout << "Enter a number: ";
    cin >> num;
    cout << "You entered: " << num;
    return 0;
}
```

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

Types of Data:
- Basic: int, char, float, double, bool
- Derived: array, pointer, reference
- User-defined: class, struct, enum, typedef

Example:
```cpp
int age = 20;
float pi = 3.14;
char grade = 'A';
bool isTrue = true;
```

6. Explain the difference between implicit and explicit type conversion in C++.
Implicit Conversion (Type Casting):
- Done automatically by the compiler.
Example: `int a = 10; double b = a;`

Explicit Conversion (Type Casting):
- Done manually by the programmer using cast operators.
Example: `double pi = 3.14; int x = (int)pi;`

7. What are the different types of operators in C++? Provide examples of each.
Types of Operators:
- Arithmetic: +, -, *, /, % → `a + b`
- Relational: ==, !=, <, >, <=, >= → `a > b`
- Logical: &&, ||, ! → `(a > b && b > c)`
- Assignment: =, +=, -=, *= → `a += 5`
- Increment/Decrement: ++, -- → `a++`
- Bitwise: &, |, ^, <<, >> → `a & b`
- Conditional (Ternary): `condition ? trueVal : falseVal`
- Special: sizeof, comma, pointer operators.

8. Explain the purpose and use of constants and literals in C++.
Constants: Fixed values that cannot be changed during program execution.
- Declared using `const` or `#define`.
Example: `const int MAX = 100;`

Literals: Actual fixed values used in the code.
- Example: `10`, `'A'`, `3.14`, `true`.

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

- if-else:
```cpp
if (x > 0) cout << "Positive";
else cout << "Non-positive";
```

- switch:
```cpp
```

```cpp
switch(day) {
  case 1: cout << "Monday"; break;
  case 2: cout << "Tuesday"; break;
  default: cout << "Invalid";
}
```

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

- for: Entry-controlled, used when number of iterations is known.
- while: Entry-controlled, used when number of iterations is not known.
- do-while: Exit-controlled, executes at least once.

Example:

```cpp
for(int i=0;i<5;i++) cout<<i;
while(i<5) cout<<i;
do{cout<<i;}while(i<5);
```

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

- break: Terminates the loop immediately.
- continue: Skips current iteration and moves to the next.

Example:

```cpp
for(int i=1;i<=5;i++) {
   if(i==3) continue;
   if(i==5) break;
   cout<<i<<" ";
}
```

12. Explain nested control structures with an example.

Nested control structures are placing one control statement inside another.
Example:

```cpp
for(int i=1;i<=3;i++) {
   for(int j=1;j<=2;j++) {
      cout<<"i="<<i<<", j="<<j<<endl;
   }
}
```

13. What is a function in C++? Explain function declaration, definition, and calling.

- Function: A block of code that performs a task.
- Declaration: `int add(int,int);`
- Definition:

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

- Calling: `cout<<add(2,3);`

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

- Local Scope: Declared inside a function, accessible only within it.
- Global Scope: Declared outside functions, accessible everywhere.

Example:

```cpp
int g=10; //global
void test(){int x=5;} //local
```

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

Recursion: A function calling itself.
Example:

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

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

A function prototype is a declaration of a function before its use.
It ensures the compiler knows the return type and parameters.

Example:
`int add(int,int);`

17. What are arrays in C++? Explain 1D and 2D arrays.

- Array: Collection of elements of the same type.
- 1D Array: `int arr[5] = {1,2,3,4,5};`
- 2D Array: `int mat[2][3] = {{1,2,3},{4,5,6}};`

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

Strings can be handled using character arrays or `string` class.
Example:
```cpp
string s = "Hello";
cout<<s.length();
```

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

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

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

String functions: length(), substr(), append(), compare().
Example:
```cpp
string a="Hello", b="World";
cout<<a+b;
cout<<a.length();
```

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

Key Concepts:
1. Encapsulation
2. Inheritance
3. Polymorphism
4. Abstraction

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

Class: Blueprint of objects.
Object: Instance of a class.
Example:
```cpp
```

```cpp
class Car {
  public:
   string brand;
   void show(){cout<<brand;}Car c1; c1.brand="BMW"; c1.show();
};
```

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

Inheritance: One class deriving properties from another.
Example:

```cpp
class A{public: void show(){cout<<"Base";}};
class B: public A{};
B obj; obj.show();
```

24. What is encapsulation in C++? How is it achieved in classes?

Encapsulation: Wrapping data and functions together.
Achieved using private data members and public functions.
Example:

```cpp
class Student {
  private: int marks;
  public: void set(int m){marks=m;}
          int get(){return marks;}
};
```