Tutorial 20 - Object-Oriented Programming (OOP) in C++

Why Object-Oriented Programming?

1. **Limitations of Procedural Programming**:
   - Focused on functions and procedures, neglecting data.
   - Data security is easily compromised.
   - Increased program size reduces readability, maintainability, and reliability.
   - Higher chances of not addressing complex problems effectively.
2. **Advantages of OOP**:
   - Models programs based on real-world scenarios.
   - Focuses on **classes** and **objects** to encapsulate data and functionality.

Difference Between Procedural and Object-Oriented Programming

Aspect	Procedural Programming	Object-Oriented Programming
Focus	Functions and instructions	Classes and objects
Data Flow	Data moves openly between functions	Data is encapsulated and treated as critical
Data Access	Uses local and global data	Controlled through encapsulation
Problem-Solving Approach	Decomposes problem into functions	Decomposes problem into objects

---

Basic Concepts in OOP

1. **Classes**: Templates for creating objects.
2. **Objects**: Basic runtime entities created using classes.
3. **Data Abstraction & Encapsulation**:
   - Wrapping data and methods into a single unit.
   - Provides controlled data access.
4. **Inheritance**: Reuse properties and methods of one class in another.
5. **Polymorphism**: Ability to take multiple forms (e.g., function overloading, overriding).
6. **Dynamic Binding**: Code executed is determined at runtime.
7. **Message Passing**: Communication between objects using message calls.

Benefits of Object-Oriented Programming

1. **Code Reusability**: Through inheritance and objects.
2. **Data Security**: Encapsulation and data hiding ensure secure systems.
3. **Complexity Management**: Simplifies the handling of large, complex software.

4. **Modularity**: Multiple objects can coexist without interference.

Short Notes

1. **Why OOP?**:
   - Procedural programming lacks data security and maintainability.
   - OOP models real-world problems using classes and objects.
2. **Core Concepts**:
   - **Class**: Template for objects.
   - **Object**: Instance of a class.
   - **Encapsulation**: Wrapping data & methods together.
   - **Inheritance**: Reusing properties of one class in another.
   - **Polymorphism**: One interface, multiple functionalities.
   - **Dynamic Binding**: Code decided at runtime.
   - **Message Passing**: Object communication.
3. **Benefits**:
   - Secure data handling.
   - Better code reuse and modularity.
   - Simplifies complex software management.