Interview Questions

1. **What is Encapsulation in C++? Why is it called Data hiding?**
   The process of binding data and corresponding methods (behavior) into a single unit is called encapsulation in C++.
   In other words, encapsulation is a programming technique that binds the class members (variables and methods) together and prevents them from accessing other classes. Thereby we can keep variables and methods safes from outside interference and misuse.
   If a field is declared private in the class, it cannot be accessed by anyone outside the class and hides the fields. Therefore, Encapsulation is also called data hiding.

2. **What is the difference between Abstraction and Encapsulation?**

Abstraction	Encapsulation
Abstraction is the method of hiding unnecessary details from the necessary ones.	Encapsulation is the process of binding data members and methods of a program together to do a specific job without revealing unnecessary details.
Achieved through encapsulation.	You can implement encapsulation using Access Modifiers (Public, Protected & Private.)
Abstraction allows you to focus on what the object does instead of how it does it.	Encapsulation enables you to hide the code and data into a single unit to secure the data from the outside world.
In abstraction, problems are solved at the design or interface level.	While in encapsulation, problems are solved at the implementation level.

3. **How much memory does a class occupy?**
   Classes do not consume any memory. They are just a blueprint based on which objects are created. When objects are created, they initialize the class members and methods and therefore consume memory.

4. **Are there any limitations of Inheritance?**
   Yes, with more powers comes more complications. Inheritance is a very powerful feature in OOPs, but it also has limitations. Inheritance needs more time to process, as it needs to navigate through multiple classes for its implementation. Also, the classes involved in Inheritance - the base class and the child class, are very tightly coupled together. So if one needs to make some changes, they might need to do nested changes in both classes. Inheritance might be complex for implementation, as well. So if not correctly implemented, this might lead to unexpected errors or incorrect outputs.

5. **What is the difference between overloading and overriding?**
   Overloading is a compile-time polymorphism feature in which an entity has multiple implementations with the same name—for example, Method overloading and Operator overloading.
   Whereas Overriding is a runtime polymorphism feature in which an entity has the same name, but its implementation changes during execution. For example, Method overriding.

6. **What are the various types of inheritance?**
   The various types of inheritance include:
   - Single inheritance
   - Multiple inheritances
   - Multi-level inheritance
   - Hierarchical inheritance
   - Hybrid inheritance

7. **What are the advantages of Polymorphism?**
   There are the following advantages of polymorphism in C++:
   a. Using polymorphism, we can achieve flexibility in our code because we can perform various operations by using methods with the same names according to requirements.
   b. The main benefit of using polymorphism is when we can provide implementation to an abstract base class or an interface.
8. **What are the differences between Polymorphism and Inheritance in C++?**
   The differences between polymorphism and inheritance in C++ are as follows:

a. Inheritance represents the parent-child relationship between two classes. On the other hand, polymorphism takes advantage of that relationship to make the program more dynamic.

b. Inheritance helps in code reusability in child class by inheriting behavior from the parent class. On the other hand, polymorphism enables child class to redefine already defined behavior inside parent class.
Without polymorphism, a child class can’t execute its own behavior.