26 Lecture - CS304

Important Subjective

1. **What is base initialization and how is it used in C++?**
   Answer: Base initialization is a mechanism used to initialize the data members of a base class in a derived class. It is done by calling the constructor of the base class in the initialization list of the derived class constructor.

   How is base initialization different from default initialization?
   Answer: Base initialization initializes the data members of the base class, while default initialization initializes the data members of the derived class.

   Why is base initialization important in C++?
   Answer: Base initialization is important because it ensures that the data members of the base class are properly initialized before the derived class constructor is called.

   Can base initialization be used to initialize data members of both the base class and derived class?
   Answer: No, base initialization can only be used to initialize the data members of the base class.

   What is the syntax for using base initialization in C++?
   Answer: The syntax for using base initialization in C++ is: derived_class::derived_class(arg1, arg2, ...): base_class(arg1, arg2, ...){...}

   What is the order of execution for constructors when base initialization is used?
   Answer: The order of execution for constructors when base initialization is used is: base class constructor(s) followed by derived class constructor.

   Can base initialization be used to initialize const data members in the base class?
   Answer: Yes, base initialization can be used to initialize const data members in the base class.

   When should base initialization be used in C++?
   Answer: Base initialization should be used in C++ when the base class has const data members that cannot be initialized in the derived class constructor, or when the derived class needs to initialize data members that are dependent on the values of the base class data members.

   How does base initialization improve performance in C++?
   Answer: Base initialization can improve performance in C++ by avoiding unnecessary default constructor calls for base class data members.

   What happens if a derived class constructor does not explicitly call the base class constructor using base initialization?
   Answer: If a derived class constructor does not explicitly call the base class constructor using base initialization, the default constructor of the base class is called.