CS304
Object Oriented Programming

Important mcqs

Lec 23 - ACCESSING BASE CLASS MEMBER FUNCTIONS IN DERIVED CLASS

1. **In object-oriented programming, a derived class can access base class member functions using:**
   a) The dot operator (.)
   b) The arrow operator (->)
   c) The scope resolution operator (::)
   d) None of the above
   Answer: c) The scope resolution operator (::)

   If a base class member function is declared as private, it can be accessed directly by the derived class.
   a) True
   b) False
   Answer: b) False

   In C++, if a base class member function is declared as protected, it can be accessed by:
   a) The derived class and any other class
   b) The derived class only
   c) Any class within the same namespace
   d) None of the above
   Answer: b) The derived class only

   When accessing a base class member function from a derived class, the derived class can modify the base class member function.
   a) True
   b) False
   Answer: b) False

   When a derived class defines a member function with the same name as a member function in the base class, it is called:
   a) Method overloading
   b) Method overriding
   c) Method shadowing
   d) None of the above
   Answer: b) Method overriding

   In C++, if a base class member function is virtual, it can be overridden by a member function in the derived class.
   a) True
   b) False
   Answer: a) True

   If a derived class inherits from multiple base classes, and both base classes have

member functions with the same name, the derived class can access both functions using the scope resolution operator.

a) True
b) False

Answer: a) True

A derived class can access the private member variables of the base class using the scope resolution operator.

a) True
b) False

Answer: b) False

If a base class has a constructor with arguments, the derived class must call the base class constructor explicitly in its own constructor.

a) True
b) False

Answer: a) True

In C++, the order in which base classes are specified in a derived class declaration affects the order in which their constructors are called.

a) True
b) False

Answer: a) True

Lec 24 - MODIFIED DEFAULT CONSTRUCTOR

1. **What is a modified default constructor?**
   a) A constructor that takes no arguments
   b) A constructor that is defined by the programmer
   c) A default constructor that has been customized to initialize the class data members to specific values
   d) None of the above
   Answer: c

   When is a default constructor provided by the compiler?
   a) When no other constructors are defined for the class
   b) When a constructor is defined by the programmer
   c) When a class has no data members
   d) None of the above
   Answer: a

   How can a modified default constructor be useful?
   a) It allows the programmer to provide a default state for the class when it is created
   b) It allows the user to provide initialization values for the class data members
   c) It allows the programmer to create multiple instances of the class with different initial values
   d) None of the above
   Answer: a

   Can a modified default constructor take arguments?
   a) Yes
   b) No
   Answer: b

   What happens if a modified default constructor is not defined for a class?
   a) The compiler will provide a default constructor that initializes the class data members to default values
   b) The program will not compile
   c) The class data members will be left uninitialized
   d) None of the above
   Answer: a

   What is the difference between a default constructor and a modified default constructor?
   a) A default constructor initializes the class data members to default values, while a modified default constructor initializes them to specific values
   b) There is no difference, they are the same thing
   c) A default constructor is provided by the compiler, while a modified default constructor is defined by the programmer
   d) None of the above
   Answer: a

   Can a modified default constructor be overloaded?
   a) Yes
   b) No
   Answer: a

   What is the syntax for defining a modified default constructor?
   a) ClassName();

b) ClassName::ClassName();
c) ClassName::ModifiedDefaultConstructor();
d) None of the above
Answer: b

How many constructors can a class have?
a) Only one
b) As many as the programmer wants to define
c) Only one of each type (default, copy, etc.)
d) None of the above
Answer: b

What is the purpose of a constructor?
a) To initialize the class data members
b) To allocate memory for the class object
c) To define the behavior of the class object
d) All of the above
Answer: a

Lec 25 - OVERLOADING VS. OVERRIDING

1. **What is overloading in object-oriented programming?**
   a. Creating multiple functions with the same name but different parameters
   b. Redefining a function in a subclass that was originally defined in a superclass
   c. Both a and b
   Answer: a

   What is overriding in object-oriented programming?
   a. Creating multiple functions with the same name but different parameters
   b. Redefining a function in a subclass that was originally defined in a superclass
   c. Both a and b
   Answer: b

   Can overloaded functions have the same number of parameters?
   a. Yes
   b. No
   Answer: Yes

   Can overloaded functions have the same name and parameters?
   a. Yes
   b. No
   Answer: No

   Can overridden functions have the same name and parameters?
   a. Yes
   b. No
   Answer: Yes

   Is overloading static or dynamic polymorphism?
   a. Static
   b. Dynamic
   Answer: Static

   Is overriding static or dynamic polymorphism?
   a. Static
   b. Dynamic
   Answer: Dynamic

   Can overloading be done in the same class?
   a. Yes
   b. No
   Answer: Yes

   Can overriding be done in the same class?
   a. Yes
   b. No
   Answer: No

   What happens if an overridden function is called on an object of the subclass?
   a. The function defined in the superclass is called

b. The function defined in the subclass is called
c. Both functions are called
Answer: b

Lec 26 - BASE INITIALIZATION

1. **What is base initialization in object-oriented programming?**
   a. A mechanism to initialize the derived class data members before the base class constructor is called
   b. A mechanism to initialize the base class data members before the derived class constructor is called
   c. A mechanism to initialize both the base class and derived class data members together
   d. A mechanism to dynamically allocate memory for the base class and derived class data members
   Answer: b. A mechanism to initialize the base class data members before the derived class constructor is called

   Why is base initialization important in C++?
   a. It ensures that the derived class data members are initialized properly
   b. It avoids unnecessary default constructor calls
   c. It allows the programmer to explicitly call the constructor of the base class
   d. All of the above
   Answer: d. All of the above

   Where is the base initialization list typically located in a C++ constructor?
   a. At the beginning of the constructor body
   b. After the constructor body
   c. Before the constructor declaration
   d. None of the above
   Answer: a. At the beginning of the constructor body

   Which of the following is an advantage of using base initialization?
   a. It can improve performance by avoiding unnecessary default constructor calls
   b. It can make code more readable and maintainable
   c. It can ensure that base class data members are initialized properly
   d. All of the above
   Answer: d. All of the above

   What is the syntax for using base initialization in C++?
   a. baseClassName(argumentList), constructorBody
   b. constructorBody, baseClassName(argumentList)
   c. baseClassName(argumentList) : constructorBody
   d. None of the above
   Answer: c. baseClassName(argumentList) : constructorBody

   When should base initialization be used in C++?
   a. When the base class has a parameterized constructor
   b. When the derived class needs to initialize data members that are dependent on the values of the base class data members
   c. When the base class has const data members that cannot be initialized in the derived class constructor
   d. All of the above
   Answer: d. All of the above

   Which of the following is an example of base initialization in C++?
   a. A derived class constructor that initializes the base class data members using default values

b. A derived class constructor that initializes the base class data members using constructor arguments
c. A derived class constructor that initializes the derived class data members using constructor arguments
d. None of the above
Answer: b. A derived class constructor that initializes the base class data members using constructor arguments

Can base initialization be used to initialize data members of both the base class and derived class?
a. Yes, it can be used to initialize both the base class and derived class data members
b. No, it can only be used to initialize the base class data members
c. It depends on the constructor arguments passed in
d. None of the above
Answer: b. No, it can only be used to initialize the base class data members

Which of the following is an example of a scenario where base initialization is useful?
a. When a derived class needs to allocate memory for the base class data members
b. When a derived class needs to call the default constructor of the base class
c. When a derived class needs to initialize const data members in the base class
d. None of the above
Answer: c. When a derived class needs to initialize const data members in the base class

What is the difference between base initialization and default initialization in C++?
a. Base initialization initializes the base class data members, while default initialization initializes the derived class data members

Lec 27 - SPECIALIZATION (RESTRICTION)

1. **Which of the following is a restriction of specialization in C++?**
   A) You can partially specialize function templates
   B) You can specialize function templates for built-in types
   C) You can specialize function templates for any type
   D) Specialization can lead to code duplication and maintenance issues
   Answer: B

   What is specialization in C++?
   A) A mechanism that allows programmers to define a different implementation of a template or function for a specific set of arguments
   B) A way to restrict access to certain parts of a program
   C) A technique used to improve the performance of a program
   D) None of the above
   Answer: A

   Which of the following is not a restriction of specialization in C++?
   A) You cannot partially specialize function templates
   B) You cannot specialize function templates for built-in types
   C) You cannot specialize function templates for any type
   D) None of the above
   Answer: D

   When is specialization useful in C++?
   A) When the default behavior of a template or function is not suitable for a particular data type or value
   B) When you want to restrict access to certain parts of a program
   C) When you want to improve the performance of a program
   D) None of the above
   Answer: A

   Can you partially specialize function templates in C++?
   A) Yes
   B) No
   Answer: B

   Can you specialize function templates for a built-in type such as int or double in C++?
   A) Yes
   B) No
   Answer: B

   What are some restrictions of specialization in C++?
   A) You can partially specialize function templates
   B) You can specialize function templates for any type
   C) Specialization can lead to code duplication and maintenance issues
   D) None of the above
   Answer: C

   How does specialization help in C++?
   A) By allowing programmers to define a different implementation of a template or function for a

specific set of arguments
B) By restricting access to certain parts of a program
C) By improving the performance of a program
D) None of the above
Answer: A

What is the syntax for specialization in C++?
A) template <> function_name<>(){}
B) template <> function_name<>{}()
C) template <typename T> function_name<T>(){}
D) None of the above
Answer: A

Is overuse of specialization a good practice in C++?
A) Yes
B) No
Answer: B

Lec 28 - VIRTUAL FUNCTIONS

1. **What is the purpose of virtual functions in C++?**
   A. To achieve static polymorphism
   B. To achieve dynamic polymorphism
   C. To improve code readability
   D. To increase program performance
   Answer: B

   Which keyword is used to declare a function as virtual in C++?
   A. static
   B. virtual
   C. dynamic
   D. polymorphic
   Answer: B

   In which class are virtual functions declared in C++?
   A. Base class
   B. Derived class
   C. Abstract class
   D. Static class
   Answer: A

   Which function is called when a virtual function is invoked through a base class pointer?
   A. Base class function
   B. Derived class function
   C. Default function
   D. Static function
   Answer: B

   What is a virtual function table (vtable) in C++?
   A. A table that stores the addresses of all virtual functions in a class hierarchy
   B. A table that stores the names of all virtual functions in a class hierarchy
   C. A table that stores the values of all virtual functions in a class hierarchy
   D. A table that stores the types of all virtual functions in a class hierarchy
   Answer: A

   Can a derived class override a non-virtual function of its base class in C++?
   A. Yes
   B. No
   Answer: A

   What is the syntax for providing a default implementation of a virtual function in C++?
   A. virtual void functionName() { ... }
   B. virtual void functionName() = 0;
   C. virtual void functionName() default;
   D. virtual void functionName() { ... } default;
   Answer: D

   What is the difference between a pure virtual function and a virtual function with a default

implementation in C++?
A. A pure virtual function has no implementation, while a virtual function with a default implementation does
B. A pure virtual function cannot be called, while a virtual function with a default implementation can be
C. A pure virtual function is declared with the = 0 syntax, while a virtual function with a default implementation is declared with the = default syntax
D. There is no difference between the two
Answer: A

Can virtual functions be defined as private in a C++ class?
A. Yes
B. No
Answer: A

What is the purpose of a virtual destructor in C++?
A. To improve program performance
B. To allow objects to be destroyed properly in a class hierarchy
C. To prevent memory leaks
D. To allow objects to be cloned easily
Answer: B

Lec 29 - ABSTRACT CLASSES

1. **Which keyword is used to define an abstract class in C++?**
   a) virtual
   b) abstract
   c) interface
   d) class
   Answer: b) abstract

   Which of the following is true about abstract classes?
   a) They can be instantiated
   b) They cannot be inherited
   c) They provide a common interface for derived classes
   d) They do not allow any data members
   Answer: c) They provide a common interface for derived classes

   Which of the following is true about pure virtual functions?
   a) They have an implementation in the base class
   b) They can be called from the base class
   c) They must be overridden in the derived class
   d) They cannot be overridden in the derived class
   Answer: c) They must be overridden in the derived class

   Can an abstract class have concrete (non-virtual) functions?
   a) Yes
   b) No
   Answer: a) Yes

   Which of the following is a correct syntax for declaring a pure virtual function?
   a) virtual void func() const = 0;
   b) pure virtual void func() = 0;
   c) void virtual func() = 0;
   d) void func() const = 0;
   Answer: a) virtual void func() const = 0;

   Which of the following is a correct way to create an instance of an abstract class?
   a) Shape s;
   b) Shape* s = new Shape();
   c) Circle c;
   d) None of the above
   Answer: d) None of the above

   Which of the following is true about abstract classes and interfaces?
   a) They are the same thing
   b) Interfaces cannot have data members
   c) Abstract classes cannot have pure virtual functions
   d) None of the above
   Answer: b) Interfaces cannot have data members

   Which of the following is an advantage of using abstract classes?
   a) They allow multiple inheritance

b) They allow for runtime polymorphism
c) They provide a mechanism for code reuse
d) They can be instantiated
Answer: c) They provide a mechanism for code reuse

Which of the following is not an example of an abstract class?
a) Shape
b) Animal
c) Car
d) Vehicle
Answer: c) Car

Which of the following statements is true about abstract classes?
a) All member functions must be pure virtual functions
b) Abstract classes cannot have constructors
c) Abstract classes cannot have concrete functions
d) Abstract classes cannot have data members
Answer: c) Abstract classes cannot have concrete functions

Lec 30 - POLYMORPHISM – CASE STUDY: A SIMPLE PAYROLL APPLICATION

1. **What is polymorphism?**
   A) The ability of an object to take on many forms
   B) The ability of an object to change its type at runtime
   C) The ability of an object to inherit multiple classes
   D) The ability of an object to override its superclass methods
   Answer: A

   What is a common use case for polymorphism?
   A) Implementing different payment methods
   B) Creating complex mathematical algorithms
   C) Defining new data types
   D) All of the above
   Answer: A

   In a payroll application, how might polymorphism be used?
   A) To implement a common interface for calculating employee pay
   B) To ensure that all employees are paid the same amount
   C) To limit the types of employees that can be added to the system
   D) To prevent employees from accessing each other's pay information
   Answer: A

   What is an interface in Java?
   A) A concrete implementation of a class
   B) A template for defining a class
   C) A set of methods and constants that a class must implement
   D) A way to declare private methods in a class
   Answer: C

   What is method overloading?
   A) Defining multiple methods with the same name but different parameters
   B) Defining multiple methods with the same name and same parameters
   C) Overriding a superclass method with a subclass method
   D) None of the above
   Answer: A

   What is method overriding?
   A) Defining multiple methods with the same name but different parameters
   B) Defining multiple methods with the same name and same parameters
   C) Overriding a superclass method with a subclass method
   D) None of the above
   Answer: C

   What is the difference between method overloading and method overriding?
   A) Method overloading is done at compile time, while method overriding is done at runtime
   B) Method overloading is done by the superclass, while method overriding is done by the subclass
   C) Method overloading is used to define new methods, while method overriding is used to modify existing methods

D) Method overloading is based on the number and type of parameters, while method overriding is based on the method name and parameters
Answer: D

Can a subclass access private methods and fields of its superclass?
A) Yes, always
B) No, never
C) Only if the subclass is in the same package as the superclass
D) Only if the superclass declares the methods and fields as protected
Answer: B

What is an abstract class in Java?
A) A class that cannot be instantiated
B) A class that does not have any methods
C) A class that only has private methods
D) A class that can only be used as a superclass
Answer: A

Can an abstract class have non-abstract methods?
A) Yes, but only if the class also has at least one abstract method
B) Yes, but only if the class has no abstract methods
C) No, an abstract class can only have abstract methods
D) None of the above
Answer: B

Lec 31 - MULTIPLE INHERITANCE

1. **What is multiple inheritance in object-oriented programming?**
   A) Inheriting from multiple subclasses
   B) Inheriting from multiple parent classes
   C) Inheriting from multiple sibling classes
   D) Inheriting from multiple child classes
   Answer: B

   Which programming languages support multiple inheritance?
   A) Java
   B) Python
   C) C++
   D) All of the above
   Answer: D

   What is the main advantage of multiple inheritance?
   A) Code reusability
   B) Improved modularity
   C) Enhanced flexibility
   D) Reduced complexity
   Answer: C

   What is the potential downside of using multiple inheritance?
   A) Code duplication
   B) Ambiguity in method and attribute resolution
   C) Inability to access parent class attributes and methods
   D) Reduced code readability
   Answer: B

   What is the diamond problem in the context of multiple inheritance?
   A) A conflict that arises when two classes define the same attribute or method
   B) A conflict that arises when two classes inherit from the same parent class
   C) A conflict that arises when a class inherits from two or more classes that have a common ancestor
   D) A conflict that arises when a class has multiple methods with the same name and arguments
   Answer: C

   How can the diamond problem be resolved?
   A) By renaming conflicting methods and attributes
   B) By removing one of the conflicting parent classes
   C) By using virtual inheritance
   D) By using multiple inheritance only when necessary
   Answer: C

   What is the syntax for implementing multiple inheritance in Python?
   A) class ChildClass(ParentClass1, ParentClass2):
   B) class ChildClass(ParentClass1 and ParentClass2):
   C) class ChildClass(ParentClass1 or ParentClass2):
   D) class ChildClass(ParentClass1 ParentClass2):
   Answer: A

   What is the order of method resolution in multiple inheritance?

B) Breadth-first search
C) Random order
D) Alphabetical order
Answer: A

What is the role of the super() function in multiple inheritance?
A) It calls the constructor of the parent class
B) It calls the method of the parent class
C) It resolves conflicts in method and attribute names
D) It prevents ambiguity in method resolution
Answer: B

What is the difference between multiple inheritance and interface inheritance?
A) Multiple inheritance allows a class to inherit from multiple classes, while interface inheritance allows a class to inherit only method signatures.
B) Multiple inheritance allows a class to inherit only method signatures, while interface inheritance allows a class to inherit from multiple classes.
C) Multiple inheritance and interface inheritance are the same thing.
D) Multiple inheritance and interface inheritance are not related concepts.
Answer: A

Lec 32 - GENERIC PROGRAMMING

1. **What is the purpose of generic programming?**
   a) To improve code readability
   b) To increase code efficiency
   c) To write reusable code
   d) To decrease code maintainability

   Answer: c) To write reusable code

   Which programming paradigm is most commonly associated with generic programming?
   a) Object-oriented programming
   b) Procedural programming
   c) Functional programming
   d) Event-driven programming

   Answer: a) Object-oriented programming

   In C++, what is the primary mechanism for achieving generic programming?
   a) Templates
   b) Polymorphism
   c) Inheritance
   d) Encapsulation

   Answer: a) Templates

   What is the advantage of using templates in C++?
   a) Templates reduce code complexity and improve code readability
   b) Templates allow for more efficient code execution
   c) Templates enable code to be reused with different data types
   d) Templates make it easier to write object-oriented code

   Answer: c) Templates enable code to be reused with different data types

   In Java, what is the primary mechanism for achieving generic programming?
   a) Templates
   b) Polymorphism
   c) Inheritance
   d) Generics

   Answer: d) Generics

   What is the difference between templates in C++ and generics in Java?
   a) Templates are more efficient than generics
   b) Templates are a more powerful mechanism for achieving generic programming than generics
   c) Templates are more difficult to use than generics
   d) Templates require explicit type parameterization, while generics do not

   Answer: d) Templates require explicit type parameterization, while generics do not

   Which of the following is an example of a generic algorithm?
   a) Bubble sort

b) Quick sort
c) Binary search
d) All of the above

Answer: d) All of the above

Which of the following is an advantage of generic algorithms?
a) Generic algorithms are more efficient than non-generic algorithms
b) Generic algorithms can be used with any data type
c) Generic algorithms can only be used with primitive data types
d) Generic algorithms are easier to debug than non-generic algorithms

Answer: b) Generic algorithms can be used with any data type

Which of the following is a disadvantage of using generics in Java?
a) Generics can lead to code bloat
b) Generics can be slower than non-generic code
c) Generics can make code harder to read
d) Generics can lead to type erasure

Answer: d) Generics can lead to type erasure

Which of the following is an example of a generic class in C++?
a) std::vector
b) std::map
c) std::pair
d) All of the above

Answer: d) All of the above

Lec 33 - MULTIPLE TYPE ARGUMENTS

1. **What are multiple type arguments?**
   a) A type of argument that can only be used with a specific data type
   b) The ability to define multiple data types for use with a generic class or function
   c) A type of argument that is only used in functional programming
   Answer: b

   Which programming languages support multiple type arguments?
   a) C++
   b) Java
   c) Python
   d) All of the above
   Answer: d

   What is the advantage of using multiple type arguments?
   a) Increased flexibility in the use of generic programming
   b) Reduced code complexity
   c) Improved code efficiency
   Answer: a

   Which keyword is used to define multiple type arguments in Java?
   a) class
   b) template
   c) < >
   Answer: c

   Which programming paradigm uses multiple type arguments extensively?
   a) Object-oriented programming
   b) Functional programming
   c) Imperative programming
   Answer: a

   Can multiple type arguments be used with functions in C++?
   a) Yes
   b) No
   Answer: a

   What is the syntax for defining multiple type arguments in C++?
   a) template<typename T, U>
   b) template<class T, class U>
   c) template<class T, U>
   Answer: b

   What is the default type argument in Java?
   a) Object
   b) Integer
   c) Double
   Answer: a

   How many type arguments can be defined for a generic class in C++?
   a) One

b) Two
c) Any number
Answer: c

What is the difference between single type arguments and multiple type arguments?
a) Single type arguments can only be used with one data type, while multiple type arguments can be used with multiple data types
b) Single type arguments are used in functional programming, while multiple type arguments are used in object-oriented programming
c) There is no difference between single type arguments and multiple type arguments
Answer: a

Lec 34 - GENERIC ALGORITHMS

1. **What are generic algorithms in programming?**
   a. Algorithms that work with a specific data type only
   b. Algorithms that work with any data type
   c. Algorithms that are optimized for performance
   Answer: b

   In which programming paradigm are generic algorithms commonly used?
   a. Object-oriented programming
   b. Procedural programming
   c. Functional programming
   Answer: a

   What is the main advantage of using generic algorithms?
   a. Improved performance
   b. Increased code complexity
   c. Reusability and adaptability of code
   Answer: c

   Which programming languages support generic algorithms?
   a. C++
   b. Java
   c. Python
   d. All of the above
   Answer: d

   Can generic algorithms be used with user-defined data types?
   a. Yes
   b. No
   Answer: a

   What is the syntax for using generic algorithms in C++?
   a. < >
   b. { }
   c. ( )
   Answer: a

   Which standard library in C++ provides support for generic algorithms?
   a. stdio.h
   b. iostream
   c. algorithm
   Answer: c

   What is the purpose of the std::sort algorithm in C++?
   a. To sort elements in ascending order
   b. To sort elements in descending order
   c. To remove duplicate elements
   Answer: a

   Which of the following is an example of a generic algorithm?
   a. Bubble sort

b. Quick sort
c. Binary search
Answer: c

What is the main disadvantage of using generic algorithms?
a. Limited applicability to specific data types
b. Reduced performance compared to specialized algorithms
c. Increased code complexity
Answer: b

Lec 35 - MEMBER TEMPLATES

1. **What are member templates in C++?**
   A. Templates defined inside a class or struct
   B. Templates defined outside a class or struct
   C. Templates that cannot be used with any data type
   D. Templates that can only be used with integer data types
   Answer: A. Templates defined inside a class or struct.

   What is the advantage of using member templates?
   A. Increased reusability and adaptability of code
   B. Improved code performance
   C. Reduced code complexity
   D. All of the above
   Answer: D. All of the above.

   Can member templates access the data members and methods of the class they are defined in?
   A. Yes
   B. No
   Answer: A. Yes.

   Can member templates be used to provide generic constructors?
   A. Yes
   B. No
   Answer: A. Yes.

   What is the syntax for defining a member template?
   A. template<class T> void myFunction(T arg);
   B. template<typename T> void myFunction(T arg);
   C. template<class T> struct MyClass {...};
   D. All of the above
   Answer: D. All of the above.

   Can member templates be specialized for specific data types?
   A. Yes
   B. No
   Answer: A. Yes.

   What is the purpose of a member function template?
   A. To provide a generic member function that can work with any data type
   B. To provide a specialized member function for a specific data type
   C. To provide a constructor for a class or struct
   D. None of the above
   Answer: A. To provide a generic member function that can work with any data type.

   What is the advantage of using a member function template over a regular member function?
   A. Increased reusability and adaptability of code
   B. Improved code performance

C. Reduced code complexity
D. All of the above
Answer: D. All of the above.

What is the difference between a member function template and a regular member function?
A. A member function template can work with any data type, whereas a regular member function can only work with specific data types
B. A member function template is defined inside a class or struct, whereas a regular member function is defined outside the class or struct
C. A member function template cannot access the data members and methods of the class it is defined in, whereas a regular member function can
D. None of the above
Answer: A. A member function template can work with any data type, whereas a regular member function can only work with specific data types.

What is the purpose of template specialization?
A. To provide a generic template that can work with any data type
B. To provide a specialized template for a specific data type
C. To provide a constructor for a class or struct
D. None of the above
Answer: B. To provide a specialized template for a specific data type.

Lec 36 - MEMBER TEMPLATES REVISITED

1. **What is member templates revisited in C++?**
   a) A way to create specialized versions of member function templates
   b) A technique to simplify the syntax of member function templates
   c) A type of member function template that only works with specific data types
   d) A way to access private data members of a class using member templates
   Answer: b) A technique to simplify the syntax of member function templates

   What is the benefit of using member templates revisited?
   a) Reduced code duplication
   b) Improved performance
   c) Simplified syntax
   d) All of the above
   Answer: d) All of the above

   What is template argument deduction?
   a) The process of providing explicit template arguments to a function template
   b) The process of inferring template arguments from function arguments
   c) The process of specializing a template for a specific data type
   d) The process of defining a template inside a class
   Answer: b) The process of inferring template arguments from function arguments

   What is the syntax for using member templates revisited?
   a) auto func(args...)
   b) template auto func(args...)
   c) template<typename T> auto Class::func(T arg)
   d) template<typename T> auto Class<T>::func(args...)
   Answer: d) template<typename T> auto Class<T>::func(args...)

   Can member templates revisited be used with constructors?
   a) Yes
   b) No
   Answer: a) Yes

   What is the difference between regular member function templates and member function templates revisited?
   a) Member function templates revisited use template argument deduction
   b) Member function templates revisited can only be used with specific data types
   c) Regular member function templates have a simpler syntax
   d) Regular member function templates cannot be specialized
   Answer: a) Member function templates revisited use template argument deduction

   What is the purpose of template argument deduction in member templates revisited?
   a) To simplify the syntax of member function templates
   b) To reduce code duplication
   c) To allow for specialization of member function templates
   d) To infer the template arguments from the function arguments
   Answer: d) To infer the template arguments from the function arguments

   Can member templates revisited be used with non-static member functions?
   a) Yes

b) No
Answer: a) Yes

What is the advantage of using member templates revisited over regular member function templates?
a) Reduced code duplication
b) Improved performance
c) More concise and readable code
d) All of the above
Answer: d) All of the above

What is the disadvantage of using member templates revisited?
a) Limited support for certain compilers
b) Increased complexity
c) Slower compile times
d) None of the above
Answer: d) None of the above

Lec 37 - RESOLUTION ORDER

1. **What is resolution order in programming?**
   a) The order in which legal disputes are resolved
   b) The order in which software applications resolve conflicts
   c) The order in which functions are defined
   d) The order in which variables are declared
   Answer: b

   Which of the following statements is true about resolution order?
   a) It is only relevant in legal disputes
   b) It determines the order in which variables are initialized
   c) It can be defined by explicit rules or by default
   d) It is only used in object-oriented programming
   Answer: c

   What is the default resolution order for Python classes?
   a) Depth-first search
   b) Breadth-first search
   c) Left-to-right
   d) Right-to-left
   Answer: a

   In Java, which keyword is used to explicitly specify the resolution order for method calls?
   a) super
   b) this
   c) extends
   d) implements
   Answer: a

   In C++, what is the resolution order for overloaded operators?
   a) Left-to-right
   b) Right-to-left
   c) Undefined
   d) Depends on the operator being overloaded
   Answer: c

   What is the resolution order for CSS styles?
   a) Inline styles, internal styles, external styles
   b) External styles, internal styles, inline styles
   c) Internal styles, inline styles, external styles
   d) Inline styles, external styles, internal styles
   Answer: d

   In Perl, which operator is used to explicitly specify the resolution order for method calls?
   a) ->>
   b) <<
   c) ::
   d) //
   Answer: c

   What is the resolution order for conflicting domain name records?
   a) MX, CNAME, A

b) CNAME, A, MX
c) A, CNAME, MX
d) MX, A, CNAME
Answer: b

In Python, what is the resolution order for multiple inheritance?
a) Left-to-right
b) Right-to-left
c) Depth-first search
d) Breadth-first search
Answer: c

In Ruby, what is the resolution order for method calls?
a) Right-to-left
b) Left-to-right
c) Depth-first search
d) Breadth-first search
Answer: c

Lec 38 - FUNCTION TEMPLATE OVERLOADING

1. **What is function template overloading in C++?**
   A) Creating multiple functions with the same name and argument types
   B) Creating multiple functions with the same name but different argument types
   C) Creating a single function that can be used with different data types
   D) Creating a single function with multiple return types
   Answer: B

   How is function template overloading achieved in C++?
   A) By defining multiple functions with different names
   B) By defining multiple functions with different return types
   C) By defining a single function with multiple argument types
   D) By defining a function template with placeholders for argument types
   Answer: D

   What is the purpose of function template overloading in C++?
   A) To reduce the number of functions in a program
   B) To create more complex functions
   C) To improve code flexibility and reusability
   D) To improve program performance
   Answer: C

   Can function templates be overloaded based on the return type?
   A) Yes
   B) No
   Answer: B

   What is the difference between function overloading and function template overloading?
   A) Function overloading is limited to a single data type, while function template overloading allows for multiple data types.
   B) Function template overloading is limited to a single data type, while function overloading allows for multiple data types.
   C) Function overloading creates multiple functions with the same name and argument types, while function template overloading creates multiple functions with the same name but different argument types.
   D) There is no difference between function overloading and function template overloading.
   Answer: C

   Can function templates be overloaded based on the number of arguments?
   A) Yes
   B) No
   Answer: A

   Which of the following is an advantage of function template overloading?
   A) It makes the code more complex
   B) It makes the code less flexible
   C) It improves code flexibility and reusability
   D) It improves program performance
   Answer: C

   Can function templates be overloaded based on the constness of the arguments?
   A) Yes

B) No
Answer: A

Which of the following is true regarding function template overloading in C++?
A) Only one function template can be defined for a given set of argument types
B) Multiple function templates can be defined for a given set of argument types
C) Function template overloading is not supported in C++
D) Function template overloading is only supported for built-in data types
Answer: B

Can function templates be overloaded based on the type of argument?
A) Yes
B) No
Answer: A

Lec 39 - TEMPLATES & STATIC MEMBERS

1. **Which keyword is used to define a template in C++?**
   a) template
   b) class
   c) typename
   d) all of the above
   Answer: a) template

   Which of the following is not a benefit of using templates in C++?
   a) Code reusability
   b) Improved efficiency
   c) Flexibility
   d) Simplified syntax
   Answer: d) Simplified syntax

   What is the purpose of static members in C++?
   a) To create class objects
   b) To provide a single instance of a variable for all class objects
   c) To define functions that can be accessed without creating an object
   d) To provide a way to create objects dynamically
   Answer: b) To provide a single instance of a variable for all class objects

   Which keyword is used to declare a static member in a class definition?
   a) static
   b) const
   c) friend
   d) virtual
   Answer: a) static

   Which of the following is true about static member functions in C++?
   a) They can be called using an object of the class.
   b) They cannot access non-static members of the class.
   c) They can only be declared in the private section of a class.
   d) They cannot be overloaded.
   Answer: b) They cannot access non-static members of the class.

   Which of the following is not a valid template parameter type in C++?
   a) int
   b) float
   c) void
   d) char*
   Answer: b) float

   What is the purpose of the typename keyword in template definitions?
   a) To indicate a class type
   b) To indicate a function type
   c) To indicate a pointer type
   d) To indicate a void type
   Answer: a) To indicate a class type

   Can a static member of a class be accessed using the class name and the scope

resolution operator?
a) Yes
b) No
Answer: a) Yes

Can a template function be defined outside the class definition?
a) Yes
b) No
Answer: a) Yes

Which of the following is not a valid way to specialize a template function?
a) Explicit specialization
b) Partial specialization
c) Function overloading
d) None of the above
Answer: c) Function overloading

Lec 40 - CURSORS

1. **What is a cursor in a database management system?**
   A) A database object that stores data
   B) A pointer that enables traversal over a set of rows in a result set
   C) A file that stores database information
   D) A function that returns a value
   Answer: B

   What is the purpose of a cursor in SQL?
   A) To insert data into a database
   B) To retrieve data from a database
   C) To update data in a database
   D) To delete data from a database
   Answer: B

   What are the types of cursors in SQL?
   A) Static, dynamic, and keyset-driven
   B) Primary, secondary, and tertiary
   C) Logical, physical, and virtual
   D) Simple, complex, and compound
   Answer: A

   Which SQL keyword is used to define a cursor?
   A) DECLARE
   B) CREATE
   C) INSERT
   D) SELECT
   Answer: A

   What is the purpose of the OPEN statement in SQL cursors?
   A) To define the cursor
   B) To fetch the next row from the cursor
   C) To close the cursor
   D) To execute a stored procedure
   Answer: B

   Which SQL keyword is used to fetch the next row from a cursor?
   A) FETCH
   B) SELECT
   C) UPDATE
   D) DELETE
   Answer: A

   What is the purpose of the CLOSE statement in SQL cursors?
   A) To define the cursor
   B) To fetch the next row from the cursor
   C) To close the cursor
   D) To execute a stored procedure
   Answer: C

   What is the purpose of the DEALLOCATE statement in SQL cursors?
   A) To define the cursor

B) To fetch the next row from the cursor
C) To close the cursor
D) To free up memory used by the cursor
Answer: D

Which type of cursor is more efficient in terms of performance?
A) Static cursor
B) Dynamic cursor
C) Keyset-driven cursor
D) There is no difference in performance between cursor types
Answer: A

Can cursors be used in all programming languages?
A) Yes
B) No, only in SQL and related languages
C) No, only in procedural languages
D) No, only in object-oriented languages
Answer: B

Lec 41 - STANDARD TEMPLATE LIBRARY

1. **What is the purpose of the Standard Template Library (STL) in C++?**
   a) To provide a library of generic algorithms, data structures, and iterators
   b) To provide a library of graphics functions
   c) To provide a library of mathematical functions
   d) To provide a library of input/output functions
   Answer: a

   Which of the following is a container class in STL?
   a) Vector
   b) Set
   c) List
   d) All of the above
   Answer: d

   Which of the following is not a type of iterator in STL?
   a) Forward iterator
   b) Reverse iterator
   c) Bidirectional iterator
   d) Parallel iterator
   Answer: d

   What is the complexity of finding an element in a set in STL?
   a) O(1)
   b) O(log n)
   c) O(n)
   d) O(n^2)
   Answer: b

   Which algorithm in STL is used to sort a range of elements in ascending order?
   a) sort
   b) reverse
   c) merge
   d) unique
   Answer: a

   Which container class in STL stores unique elements in sorted order?
   a) Vector
   b) Map
   c) Set
   d) Queue
   Answer: c

   What is the difference between a stack and a queue in STL?
   a) A stack is a LIFO structure, while a queue is a FIFO structure.
   b) A stack is a FIFO structure, while a queue is a LIFO structure.
   c) Both are LIFO structures.
   d) Both are FIFO structures.
   Answer: a

   Which algorithm in STL is used to copy a range of elements from one container to

another?
a) find
b) copy
c) sort
d) replace
Answer: b

Which of the following is not a header file in STL?
a) <vector>
b) <set>
c) <list>
d) <iostream>
Answer: d

Which container class in STL is used to implement a priority queue?
a) Map
b) Queue
c) Stack
d) Heap
Answer: d

Lec 42 - ITERATORS

1. **Which of the following is NOT a type of iterator in C++?**
   a) Forward iterator
   b) Backward iterator
   c) Input iterator
   d) Random access iterator
   Answer: b) Backward iterator

   Which of the following is a feature of a forward iterator?
   a) Can move in both directions
   b) Can be used to modify the elements in a container
   c) Can access the elements of a container multiple times
   d) Can only move forward in a container
   Answer: d) Can only move forward in a container

   Which of the following is an example of a container that supports random access iterators?
   a) Linked list
   b) Queue
   c) Array
   d) Set
   Answer: c) Array

   What is the purpose of an output iterator?
   a) To iterate through a container in reverse order
   b) To modify the elements in a container
   c) To read the elements in a container
   d) To write data to a container
   Answer: d) To write data to a container

   Which of the following is an example of a bidirectional iterator?
   a) Stack
   b) Deque
   c) Forward list
   d) Map
   Answer: b) Deque

   What is the complexity of the operator++() function for a random access iterator?
   a) O(n)
   b) O(log n)
   c) O(1)
   d) O(n^2)
   Answer: c) O(1)

   Which of the following algorithms require a random access iterator?
   a) std::sort()
   b) std::transform()
   c) std::reverse()
   d) std::unique()
   Answer: a) std::sort()

   Which of the following is a characteristic of an input iterator?

b) Can only access the elements of a container once
c) Can move in both directions
d) Can skip elements in a container
Answer: b) Can only access the elements of a container once

Which of the following is an example of a container that supports bidirectional iterators?
a) Hash table
b) Binary search tree
c) Vector
d) Queue
Answer: b) Binary search tree

Which of the following is an example of a container that supports forward iterators?
a) Stack
b) Map
c) Linked list
d) Set
Answer: c) Linked list

Lec 43 - EXAMPLE – ABNORMAL TERMINATION

1. **What is abnormal termination?**
   A. A program that runs indefinitely
   B. A program that terminates normally
   C. A program that terminates unexpectedly due to an error or exception
   D. A program that never compiles
   Answer: C

   What can cause abnormal termination?
   A. Memory access violations
   B. Stack overflow
   C. Division by zero
   D. All of the above
   Answer: D

   What is a segmentation fault?
   A. A type of error that occurs when a program attempts to access memory that has already been freed or is outside of its allocated range
   B. A type of error that occurs when a program runs out of stack space
   C. A type of error that occurs when a program attempts to divide by zero
   D. A type of error that occurs when a program fails to compile
   Answer: A

   What is an access violation?
   A. A type of error that occurs when a program attempts to access memory that has already been freed or is outside of its allocated range
   B. A type of error that occurs when a program runs out of stack space
   C. A type of error that occurs when a program attempts to divide by zero
   D. A type of error that occurs when a program fails to compile
   Answer: A

   What is a floating-point exception?
   A. A type of error that occurs when a program attempts to divide a number by zero
   B. A type of error that occurs when a program attempts to access memory that has already been freed or is outside of its allocated range
   C. A type of error that occurs when a program runs out of stack space
   D. A type of error that occurs when a program fails to compile
   Answer: A

   What is integer division by zero?
   A. A type of error that occurs when a program attempts to divide an integer by zero
   B. A type of error that occurs when a program attempts to access memory that has already been freed or is outside of its allocated range
   C. A type of error that occurs when a program runs out of stack space
   D. A type of error that occurs when a program fails to compile
   Answer: A

   How can abnormal termination be prevented?
   A. Implementing error handling and exception handling mechanisms

B. Proper testing and debugging practices
C. Both A and B
D. None of the above
Answer: C

What is error handling?
A. A mechanism to catch and handle errors or exceptions
B. A way to prevent programs from crashing
C. A way to write more efficient code
D. A way to increase program security
Answer: A

What is exception handling?
A. A mechanism to catch and handle errors or exceptions
B. A way to prevent programs from crashing
C. A way to write more efficient code
D. A way to increase program security
Answer: A

What is debugging?
A. The process of identifying and fixing errors in a program
B. The process of optimizing a program for performance
C. The process of writing code
D. The process of designing a program
Answer: A

Lec 44 - STACK UNWINDING

1. **What is stack unwinding?**
   a) A process of releasing memory from the heap
   b) A process of releasing memory from the stack
   c) A process of clearing the call stack in response to an exception being thrown
   d) A process of allocating memory on the stack
   Answer: c

   What happens during stack unwinding?
   a) All functions that were called before the exception occurred are popped off the stack
   b) All functions that were called after the exception occurred are popped off the stack
   c) All functions in the program are popped off the stack
   d) All variables in the program are popped off the stack
   Answer: a

   Why is stack unwinding important?
   a) It prevents resource leaks in C++ programs
   b) It allows programs to allocate memory on the stack
   c) It improves program performance
   d) It allows programs to release memory from the heap
   Answer: a

   What is the purpose of calling destructors during stack unwinding?
   a) To release any dynamically allocated memory or resources
   b) To allocate memory on the stack
   c) To improve program performance
   d) To initialize variables on the stack
   Answer: a

   What happens if an exception is thrown but not caught?
   a) The program terminates immediately
   b) The program continues to execute normally
   c) The program enters an infinite loop
   d) The program enters a state of undefined behavior
   Answer: a

   What is the role of the try block in stack unwinding?
   a) It contains the code that may throw an exception
   b) It contains the code that is executed if an exception is caught
   c) It contains the code that is executed if an exception is not caught
   d) It contains the code that is executed before stack unwinding begins
   Answer: a

   What is the role of the catch block in stack unwinding?
   a) It catches and handles exceptions
   b) It releases memory from the heap
   c) It initializes variables on the stack
   d) It allocates memory on the stack
   Answer: a

   What happens if a function throws an exception but does not have a catch block?
   a) The program terminates immediately

b) The program continues to execute normally
c) The exception is caught by a catch block in a higher function on the call stack
d) The program enters a state of undefined behavior
Answer: c

When are destructors called during stack unwinding?
a) In reverse order of construction
b) In the order of construction
c) Randomly
d) It depends on the implementation
Answer: a

What is the difference between stack unwinding and stack overflow?
a) Stack unwinding is intentional, while stack overflow is unintentional
b) Stack unwinding releases memory from the stack, while stack overflow overwrites memory on the stack
c) Stack unwinding occurs during exception handling, while stack overflow occurs when the stack becomes too full
d) Stack unwinding only occurs in C++, while stack overflow can occur in any programming language
Answer: c

Lec 45 - RESOURCE MANAGEMENT

1. **Which of the following is not a common system resource that requires efficient management?**
   a. Memory
   b. CPU cycles
   c. Input devices
   d. Network connections
   Answer: c

   Which of the following is not a technique used for efficient resource management?
   a. Resource pooling
   b. Resource caching
   c. Resource sharing
   d. Resource wasting
   Answer: d

   What is resource pooling?
   a. The process of allocating resources to specific tasks
   b. The process of sharing resources among multiple tasks
   c. The process of caching resources for future use
   d. The process of deallocating unused resources
   Answer: b

   Which of the following is an example of a resource leak?
   a. Using a caching mechanism for database queries
   b. Releasing memory after it has been used
   c. Failing to close a database connection
   d. Allocating resources dynamically
   Answer: c

   What is garbage collection?
   a. The process of deallocating unused memory
   b. The process of deallocating unused network connections
   c. The process of deallocating unused file handles
   d. The process of deallocating unused CPU cycles
   Answer: a

   What is resource caching?
   a. The process of allocating resources to specific tasks
   b. The process of sharing resources among multiple tasks
   c. The process of caching resources for future use
   d. The process of deallocating unused resources
   Answer: c

   What is resource sharing?
   a. The process of allocating resources to specific tasks
   b. The process of sharing resources among multiple tasks
   c. The process of caching resources for future use
   d. The process of deallocating unused resources
   Answer: b

   Which of the following is not a benefit of efficient resource management?

b. Increased reliability
c. Reduced resource usage
d. Increased memory leaks
Answer: d

Which of the following is an example of resource pooling?
a. Sharing a database connection among multiple threads
b. Allocating a fixed amount of memory for a process
c. Caching query results for future use
d. Deallocating memory when it is no longer needed
Answer: a

What is resource caching used for?
a. To allocate resources to specific tasks
b. To share resources among multiple tasks
c. To cache resources for future use
d. To deallocate unused resources
Answer: c