1. What is the purpose of a virtual destructor in C++?
   1. To deallocate memory allocated using new
   2. **To allow derived class destructors to be called through a base class pointer**
   3. To enable multiple inheritance
   4. To prevent memory leaks
2. Which keyword is used to declare a virtual function in C++?
   1. **virtual**
   2. polymorphic
   3. abstract
   4. dynamic
3. Virtual functions are used to achieve which type of polymorphism?
   1. Compile-time polymorphism
   2. Static polymorphism
   3. **Run-time polymorphism**
   4. Overload polymorphism
4. In C++, what is the correct way to define a virtual destructor in a class?
   1. **virtual ~MyClass() {}**
   2. virtual MyClass::~MyClass() {}
   3. ~virtual MyClass() {}
   4. ~MyClass() virtual {}
5. When is the memory for an object's destructor called?
   1. **When the object goes out of scope or is explicitly deleted**
   2. When the object is created
   3. When the program exits
   4. When the constructor is called
6. What is the purpose of a pure virtual function in C++?
   1. To prevent instantiation of a class
   2. To make the function const
   3. To provide a default implementation
   4. **To make the class abstract and act as an interface**
7. Can a class have both a virtual and a pure virtual destructor?
   1. **Yes, it is common practice**
   2. No, it is not allowed
   3. Only if the class is marked as final
   4. Only if the class is marked as abstract
8. Which type of polymorphism is achieved through function overloading in C++?
   1. **Compile-time polymorphism**
   2. Runtime polymorphism
   3. Static polymorphism
   4. Dynamic polymorphism
9. What happens if a class with a virtual destructor does not have any derived classes?
   1. The program will compile but throw an error at runtime
   2. The program will not compile
   3. **It will cause a memory leak**
   4. It will automatically become an abstract class
10. Which type of inheritance allows a class to inherit from multiple base classes?
    1. Hierarchical inheritance
    2. **Multiple inheritance**
    3. Single inheritance
    4. Hybrid inheritance
11. The process of defining multiple functions with the same name but different parameters in a class is called:
    1. **Overloading**
    2. Overriding
    3. Inheriting
    4. Polymorphism
12. In C++, what is the access specifier used to define a base class function that can be overridden by derived classes?
    1. override
    2. base
    3. virtual
    4. **protected**
13. When does dynamic polymorphism occur in C++?
    1. During compile-time
    2. During link-time
    3. **During runtime**
    4. During preprocessing
14. What is the purpose of the 'final' keyword in C++11 and later versions?
    1. To indicate that a class cannot be instantiated
    2. **To indicate that a function cannot be overridden**
    3. To indicate that a function is pure virtual
    4. To indicate that a class cannot be inherited from
15. What is the size of an empty class with a virtual destructor in C++?
    1. 0 bytes
    2. 1 byte
    3. 4 bytes
    4. **It depends on the compiler**
16. Which of the following is true regarding virtual base classes in C++?
    1. They are always abstract classes
    2. They cannot have member variables
    3. **They are used to avoid the "diamond problem" in multiple inheritance**
    4. They cannot have member functions
17. In C++, can you have a virtual constructor?
    1. Yes, but only if the class is also an abstract class
    2. Yes, virtual constructors are allowed
    3. **No, virtual constructors are not allowed**
    4. Only if the class is marked as final
18. Which of the following is true about the order of destruction in a class hierarchy with virtual destructors?
    1. Derived class destructor is called first, followed by the base class destructor
    2. **Base class destructor is called first, followed by the derived class destructor**
    3. Destructors are called randomly
    4. Destructor order is undefined
19. Can a class have multiple virtual functions with the same name but different parameters?
    1. **Yes, it is allowed**
    2. No, it is not allowed
    3. Only if the class is marked as abstract
    4. Only if the class is marked as final
20. Which of the following statements about virtual functions is correct?
    1. Virtual functions are automatically inlined by the compiler
    2. Virtual functions cannot be overridden in derived classes
    3. Virtual functions must have a return type of 'void'
    4. **Virtual functions can be defined in the base class and redefined in the derived class**
21. When should you declare a destructor in a class as 'protected'?
    1. When you want to prevent any instance of the class from being created
    2. **When you want to allow the class to be inherited, but not destructible**
    3. When you want to enforce the use of the 'delete' keyword to destroy objects
    4. When you want to prevent any other class from inheriting the class
22. Which type of polymorphism is achieved through function overriding in C++?
    1. Compile-time polymorphism
    2. **Runtime polymorphism**
    3. Static polymorphism
    4. Dynamic polymorphism
23. What is the meaning of the 'override' keyword in C++11 and later versions?
    1. It indicates that a function is pure virtual
    2. It indicates that a function is automatically called
    3. **It indicates that a function overrides a base class function**
    4. It indicates that a function is a virtual constructor
24. In C++, can a base class pointer point to a derived class object?
    1. **Yes, but only if the base class is marked as 'virtual'**
    2. No, it is not allowed in C++
    3. Yes, as long as the derived class does not have any virtual functions
    4. Yes, it is always allowed
25. Which of the following is true about polymorphism in C++?
    1. Polymorphism can only be achieved through virtual functions
    2. Polymorphism can only be achieved through function overloading
    3. **Polymorphism allows a single function to have multiple forms**
    4. Polymorphism is not supported in C++
26. When should you declare a class destructor as virtual?
    1. When the class has any virtual functions
    2. **When the class is a base class and is intended to be inherited**
    3. When the class is intended to be an abstract class
    4. When the class has private member variables
27. Which type of polymorphism is resolved at compile-time in C++?
    1. Compile-time polymorphism
    2. Runtime polymorphism
    3. **Static polymorphism**
    4. Dynamic polymorphism
28. If a derived class does not provide its own destructor, what happens when the object is deleted?
    1. The program will throw a compilation error
    2. **The base class destructor will be called automatically**
    3. The program will crash at runtime
    4. The compiler will provide a default destructor
29. Can a class have a virtual constructor and a virtual destructor at the same time?
    1. Yes, it is allowed
    2. **No, a class cannot have both at the same time**
    3. Only if the class is marked as final
    4. Only if the class is marked as abstract
30. What is the output of the following code?

#include <iostream>

class Base {

public:

virtual ~Base() { std::cout << "Base destructor\n"; }

};

class Derived : public Base {

public:

~Derived() { std::cout << "Derived destructor\n"; }

};

int main() {

Base\* ptr = new Derived;

delete ptr;

return 0;

}

* 1. Base destructor
  2. Derived destructor
  3. **Both Base and Derived destructors**
  4. Neither Base nor Derived destructor

1. Which of the following statements about function overriding is true?
   1. **The access specifier for the overridden function in the derived class must be the same as the base class**
   2. Overridden functions cannot have the same name as the base class function
   3. Overridden functions cannot have different parameters from the base class function
   4. Overridden functions cannot have a different return type from the base class function
2. What is the output of the following code?

#include <iostream>

class Animal {

public:

virtual void speak() const { std::cout << "Animal sound\n"; }

};

class Dog : public Animal {

public:

void speak() const override { std::cout << "Bark!\n"; }

};

int main() {

Dog\* d = new Dog;

Animal\* a = d;

a->speak();

return 0;

}

* 1. Animal sound
  2. **Bark!**
  3. Compile error
  4. Runtime error

1. In C++, what is the purpose of the 'typeid' operator?
   1. To determine the type of an object at compile-time
   2. **To determine the type of an object at runtime**
   3. To convert between different types
   4. To get the memory address of an object