1. What is the purpose of a constructor in C++?

a) To deallocate memory

b) To initialize an object

c) To perform arithmetic operations

d) To define class methods

2. Which access specifier allows members to be accessed from outside the class?

a) public

b) private

c) protected

d) static

3. What is the difference between a class and a structure in C++?

a) Classes can have member functions, while structures cannot.

b) Structures are used for object-oriented programming, while classes are not.

c) Structures have public access by default, while classes have private access by default.

d) There is no difference; class and structure are interchangeable.

4. How can private members of a class be accessed from within the same class?

a) Using a friend function

b) Using the scope resolution operator (::)

c) Using the dot (.) operator

d) Using the arrow (->) operator

5. What is the purpose of the keyword "virtual" in C++?

a) It defines a virtual constructor for a class.

b) It allows a member function to be overridden in derived classes.

c) It provides access to private members of a class.

d) It indicates dynamic memory allocation.

6. What does the std::move() function do in C++?

a) Copies an object to another object

b) Initializes an object with default values

c) Moves the resources from one object to another

d) Deletes an object from memory

7. Which type of constructor is used to create an object without any arguments?

a) Default constructor

b) Parameterized constructor

c) Copy constructor

d) Destructor

8. What is the purpose of the destructor in C++?

a) To allocate memory for an object

b) To initialize the data members of a class

c) To deallocate memory and clean up resources

d) To perform arithmetic operations on objects

9. Which access specifier is used by default for members of a class in C++?

a) public

b) private

c) protected

d) static

10. When is the copy constructor called in C++?

a) When an object is assigned a new value using the assignment operator (=)

b) When an object is created by copying another object

c) When an object goes out of scope

d) When an object is passed by value to a function

11. What does the keyword "const" indicate in C++?

a) The object cannot be modified

b) The object cannot be accessed

c) The object is a constant reference

d) The object is a constant pointer

12. Which keyword is used to access the base class members in a derived class?

a) base

b) super

c) parent

d) derived

13. Which function is automatically called by the compiler when an object is destroyed?

a) Constructor

b) Destructor

c) Copy constructor

d) Move constructor

14. What is the purpose of the keyword "this" in C++?

a) It refers to the current object within a member function

b) It refers to the base class object in a derived class

c) It refers to the derived class object in a base class

d) It refers to the global object within a member function

Certainly! Here are 15 multiple-choice questions based on s related to C++ programming:

1:

```
#include <iostream>

int main() {
    int x = 5;
    int y = 10;
```

```
    int& ref = x;
    ref = y;

    std::cout << x << std::endl;

    return 0;
}
```

1. What will be the output of the above code?

   a) 5

   b) 10

   c) Compiler error

   d) Runtime error

2:

```
#include <iostream>

void swap(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x = 5;
    int y = 10;
    swap(x, y);

    std::cout << x << ", " << y << std::endl;
```

```
    return 0;
}
```

2. What will be the output of the above code?

a) 5, 10

b) 10, 5

c) Compiler error

d) Runtime error

3:

```
#include <iostream>

class Rectangle {
public:
    Rectangle(int w, int h) : width(w), height(h) {}
    int calculateArea() const {
        return width * height;
    }

private:
    int width;
    int height;
};

int main() {
    Rectangle rect(5, 7);
    int area = rect.calculateArea();
```

```
    std::cout << "Area: " << area << std::endl;

    return 0;
}
```

3. What is the purpose of the constructor in the Rectangle class?

a) To calculate the area of the rectangle

b) To initialize the width and height of the rectangle

c) To print the area of the rectangle

d) To set the width and height of the rectangle to 5 and 7, respectively

4:

```
#include <iostream>

class MyClass {
public:
    MyClass(int x) {
        value = new int;
        *value = x;
    }

    ~MyClass() {
        delete value;
    }

private:
    int* value;
};
```

```
int main() {
    MyClass obj(5);

    std::cout << "Value: " << *obj.value << std::endl;

    return 0;
}
```

4. What is the purpose of the destructor in the MyClass class?

a) To allocate memory for the "value" variable

b) To initialize the "value" variable to 5

c) To deallocate memory for the "value" variable

d) To print the value of the "value" variable

5:

```
#include <iostream>

class Base {
public:
    virtual void print() {
        std::cout << "Base class" << std::endl;
    }
};

class Derived : public Base {
public:
    void print() override {
        std::cout << "Derived class" << std::endl;
    }
```

```
};

int main() {
    Base* ptr = new Derived;
    ptr->print();

    delete ptr;

    return 0;
}
```

5. What will be the output of the above code?

a) Base class

b) Derived class

c) Compiler error

d) Runtime error

6:

```
#include <iostream>

class A {
public:
    virtual void print() {
        std::cout << "Class A" << std::endl;


 }
};
```

```
class B : public A {
public:
    void print() {
        std::cout << "Class B" << std::endl;
    }
};

class C : public A {
public:
    void print() {
        std::cout << "Class C" << std::endl;
    }
};

int main() {
    A* ptr = new B;
    B* derivedPtr = dynamic_cast<B*>(ptr);
    derivedPtr->print();

    delete ptr;

    return 0;
}
```

6. What will be the output of the above code?

a) Class A

b) Class B

c) Class C

d) Compiler error

7:

```
#include <iostream>

class Base {
public:
    Base() {
        std::cout << "Base constructor" << std::endl;
    }

    Base(const Base& other) {
        std::cout << "Base copy constructor" << std::endl;
    }
};

class Derived : public Base {
public:
    Derived() {
        std::cout << "Derived constructor" << std::endl;
    }

    Derived(const Derived& other) {
        std::cout << "Derived copy constructor" << std::endl;
    }
};

int main() {
    Derived obj1;
    Derived obj2 = obj1;

    return 0;
```

```
}
```

7. What will be the output of the above code?

a) Base constructor
   Derived constructor
   Derived copy constructor

b) Derived constructor
   Base constructor
   Derived copy constructor

c) Derived constructor
   Base constructor
   Base copy constructor

d) Compiler error

8:

```
#include <iostream>

class MyClass {
public:
    MyClass() {
        std::cout << "Default constructor" << std::endl;
    }

    MyClass(int x) {
        std::cout << "Parameterized constructor" << std::endl;
    }

    MyClass(const MyClass& other) {
        std::cout << "Copy constructor" << std::endl;
```

```
    }

    ~MyClass() {
        std::cout << "Destructor" << std::endl;
    }
};

int main() {
    MyClass obj1;
    MyClass obj2 = obj1;
    MyClass obj3(5);

    return 0;
}
```

8. How many constructors and destructors will be called in the above code?

a) 1 constructor, 1 destructor

b) 2 constructors, 1 destructor

c) 2 constructors, 2 destructors

d) 3 constructors, 2 destructors

9:

```
#include <iostream>

class Parent {
public:
    virtual void print() {
        std::cout << "Parent class" << std::endl;
    }
```

```
};

class Child : public Parent {
public:
    void print() {
        std::cout << "Child class" << std::endl;
    }
};

int main() {
    Parent parent;
    Child child;
    Parent* ptr = &child;

    parent.print();
    child.print();
    ptr->print();

    return 0;
}
```

9. What will be the output of the above code?

a) Parent class
   Child class
   Parent class

b) Parent class
   Child class
   Child class

c) Child class
   Parent class

Parent class

d) Compiler error

10:

```
#include <iostream>

class Base {
public:
    virtual void print() {
        std::cout << "Base class" << std::endl;
    }
};

class Derived : public Base {
public:


    void print() {
        std::cout << "Derived class" << std::endl;
    }
};

int main() {
    Base obj1;
    Derived obj2;

    Base* ptr1 = &obj1;
    Base* ptr2 = &obj2;

    ptr1->print();
```

```
    ptr2->print();

    return 0;
}
```

10. What will be the output of the above code?

a) Base class
   Derived class

b) Derived class
   Base class

c) Base class
   Base class

d) Compiler error

11:

```
#include <iostream>

class Base {
public:
    virtual void print() {
        std::cout << "Base class" << std::endl;
    }
};

class Derived : public Base {
public:
    void print() {
        std::cout << "Derived class" << std::endl;
    }
```

```
};

int main() {
    Derived obj;
    Base& ref = obj;

    ref.print();

    return 0;
}
```

11. What will be the output of the above code?

a) Base class

b) Derived class

c) Compiler error

d) Runtime error

12:

```
#include <iostream>

class MyClass {
public:
    MyClass() {
        std::cout << "Constructor" << std::endl;
    }

    ~MyClass() {
        std::cout << "Destructor" << std::endl;
    }
```

```
};

int main() {
    MyClass* obj = new MyClass();
    delete obj;

    return 0;
}
```

12. How many constructors and destructors will be called in the above code?

a) 1 constructor, 1 destructor

b) 1 constructor, 2 destructors

c) 2 constructors, 1 destructor

d) 2 constructors, 2 destructors

13:

```
#include <iostream>

class MyClass {
public:
    MyClass() {
        std::cout << "Constructor" << std::endl;
    }

    MyClass(const MyClass& other) {
        std::cout << "Copy constructor" << std::endl;
    }

    ~MyClass() {
```

```
        std::cout << "Destructor" << std::endl;
    }
};

int main() {
    MyClass obj1;
    MyClass obj2 = obj1;

    return 0;
}
```

13. How many constructors and destructors will be called in the above code?

    a) 1 constructor, 1 destructor

    b) 1 constructor, 2 destructors

    c) 2 constructors, 1 destructor

    d) 2 constructors, 2 destructors

14:

```
#include <iostream>

class Base {
public:
    virtual void print() {
        std::cout << "Base class" << std::endl;
    }
};

class Derived : public Base {
public:
```

```
    void print() override {
        std::cout << "Derived class" << std::endl;
    }
};

void printObject(Base obj) {
    obj.print();
}

int main() {
    Derived obj;
    printObject(obj);

    return 0;
}
```

14. What will be the output of the above code?

a) Base class

b) Derived class

c) Compiler error

d) Runtime error

15:

```
#include <iostream>

class Parent {
public:
    virtual void print() {
        std::cout << "Parent class" << std::endl;
```

```
    }
};

class Child : public Parent {
public:
    void print() {


        std::cout << "Child class" << std::endl;
    }
};

int main() {
    Parent* ptr = new Child;
    Child* derivedPtr = dynamic_cast<Child*>(ptr);
    derivedPtr->print();

    delete ptr;

    return 0;
}
```

15. What will be the output of the above code?

a) Parent class

b) Child class

c) Compiler error

d) Runtime error