**C++ Programming Questions and Answers – Class Hierarchies Introduction**

1. What will happen when introduce the interface of classes in a run-time polymorphic hierarchy?  
a) Separation of interface from implementation  
b) Merging of interface from implementation  
c) Separation of interface from debugging  
d) Merging of interface from debugging

Answer: a  
Explanation: Separation of interface from implementation introduce the interface of classes in a run-time polymorphic hierarchy.

2. Which classes are called as mixin?  
a) Represent a secondary design  
b) Classes express functionality which represents responsibilities  
c) Standard logging stream  
d) Represent a priary design

Answer: b  
Explanation: A class that expresses functionality rather than its primary design role is called a mixin.

3. What is the use of clog?  
a) Standard logging stream  
b) Error stream  
c) Input stream  
d) output stream

Answer: a  
Explanation: clog is an object of class ostream that represents the standard logging stream. It is associated with the cstdio stream stderr, like cerr.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <sstream>
3. using namespace std;
4. int main()
5. {
6. stringstream mys(ios :: in | ios :: out);
7. std :: string dat("The double value is : 74.79 .");
8. mys.str(dat);
9. mys.seekg(-7, ios :: end);
10. double val;
11. mys >> val;
12. val = val\*val;
13. mys.seekp(-7,ios::end);
14. mys << val;
15. std :: string new\_val = mys.str();
16. cout << new\_val;
17. return 0;
18. }

a) 5593.54  
b) Error  
c) Runtime error  
d) 5463.54

Answer: a  
Explanation: In this program, We have used the string hierarchy to compute the square of the number.  
Output:

$ g++ class.cpp

$ a.out

The double value is : 5593.54 .

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Base
4. {
5. public:
6. Base(){}
7. ~Base(){}
8. protected:
9. private:
10. };
11. class Derived:public Base
12. {
13. public:
14. Derived(){}
15. Derived(){}
16. private:
17. protected:
18. };
19. int main()
20. {
21. cout << "The program exceuted" << endl;
22. }

a) The program executed  
b) Error  
c) Runtime error  
d) program exceuted

Answer: b  
Explanation: We are allowed to overload constructor but in this case as both the constructor have no parameters which implies that both the constructor have same signature which is not allowed i.e. constructors can be overloaded but two overloaded constructors can not have same function signature.

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class MyException
4. {
5. public:
6. MyException(int value) : mValue(value)
7. {
8. }
9. int mValue;
10. };
11. class MyDerivedException : public MyException
12. {
13. public:
14. MyDerivedException(int value, int anotherValue) : MyException(value), mAnotherValue(anotherValue)
15. {
16. }
17. int mValue;
18. int mAnotherValue;
19. };
20. void doSomething()
21. {
22. throw MyDerivedException(10,20);
23. }
24. int main()
25. {
26. try
27. {
28. doSomething();
29. }
30. catch (MyDerivedException &exception)
31. {
32. cout << "**\n**Caught Derived Class Exception**\n**";
33. }
34. catch (MyException &exception)
35. {
36. cout << "**\n**Caught Base Class Exception**\n**";
37. }
38. return 0;
39. }

a) Caught Base Class Exception  
b) Caught Derived Class Exception  
c) Caught Base & Derived Class Exception  
d) Caught Base Class

Answer: b  
Explanation: As we are throwing the value from the derived class, it is arising an exception in derived class  
Output:

$ g++ class1.cpp

$ a.out

Caught Derived Class Exception

7. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <string>
3. using namespace std;
4. int main()
5. {
6. string s = "a long string";
7. s.insert(s.size() / 2, " \* ");
8. cout << s << endl;
9. return 0;
10. }

a) a long\* string  
b) a long st\*ring  
c) Depends on compiler  
d) a long string\*

Answer: c  
Explanation: In this program, We are placing the string based on the size of the string and it is a string hierarchy.  
Output:

$ g++ class2.cpp

$ a.out

a long\* string

8. How many types of guarantees are there in exception class can have?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are three types of guarantees in c++. They are weak, strong and no-throw.

9. Which operator is used to create the user-defined streams in c++?  
a) >>  
b) <<  
c) &  
d) Both >> & <<

Answer: d  
Explanation: We can make user-defined types with streams by overloading the insertion operator (<<) to put objects into streams and the extraction operator (>>) to read objects from streams.

10. What does the cerr represent?  
a) Standard error stream  
b) Standard logging stream  
c) Input stream  
d) Output stream

Answer: a  
Explanation: cerr is an object of class ostream that represents the standard error stream. It is associated with the cstdio stream stderr.

# C++ Programming MCQ – Multiple Inheritance

1. What is meant by multiple inheritance?  
a) Deriving a base class from derived class  
b) Deriving a derived class from base class  
c) Deriving a derived class from more than one base class  
d) Deriving a derived base class

Answer: c  
Explanation: Multiple inheritance enables a derived class to inherit members from more than one parent.

2. Which symbol is used to create multiple inheritances?  
a) Dot  
b) Comma  
c) Dollar  
d) star

Answer: b  
Explanation: For using multiple inheritance, simply specify each base class (just like in single inheritance), separated by a comma.

3. Which of the following advantages we lose by using multiple inheritances?  
a) Dynamic binding  
b) Polymorphism  
c) Both Dynamic binding & Polymorphism  
d) Constructor

Answer: c  
Explanation: The benefit of dynamic binding and polymorphism is that they help making the code easier to extend but by multiple inheritance it makes harder to track.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class polygon
4. {
5. protected:
6. int width, height;
7. public:
8. void set\_values (int a, int b)
9. {
10. width = a; height = b;}
11. };
12. class output1
13. {
14. public:
15. void output (int i);
16. };
17. void output1::output (int i)
18. {
19. cout << i << endl;
20. }
21. class rectangle: public polygon, public output1
22. {
23. public:
24. int area ()
25. {
26. return (width \* height);
27. }
28. };
29. class triangle: public polygon, public output1
30. {
31. public:
32. int area ()
33. {
34. return (width \* height / 2);
35. }
36. };
37. int main ()
38. {
39. rectangle rect;
40. triangle trgl;
41. rect.set\_values (4, 5);
42. trgl.set\_values (4, 5);
43. rect.output (rect.area());
44. trgl.output (trgl.area());
45. return 0;
46. }

a) 20  
b) 10  
c) 20

10

d) 30

Answer: c  
Explanation: We are using the multiple inheritance to find the area of rectangle and triangle.  
Output:

$ g++ mul.cpp

$ a.out

20

10

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Base
4. {
5. public:
6. virtual void print() const = 0;
7. };
8. class DerivedOne : public Base
9. {
10. public:
11. void print() const
12. {
13. cout << "DerivedOne**\n**";
14. }
15. };
16. class DerivedTwo : public Base
17. {
18. public:
19. void print() const
20. {
21. cout << "DerivedTwo**\n**";
22. }
23. };
24. class Multiple : public DerivedOne, public DerivedTwo
25. {
26. public:
27. void print() const
28. {
29. DerivedTwo :: print();
30. }
31. };
32. int main()
33. {
34. int i;
35. Multiple both;
36. DerivedOne one;
37. DerivedTwo two;
38. Base \*array[ 3 ];
39. array[ 0 ] = &both;
40. array[ 1 ] = &one;
41. array[ 2 ] = &two;
42. array[ i ] -> print();
43. return 0;
44. }

a) DerivedOne  
b) DerivedTwo  
c) Error  
d) DerivedThree

Answer: c  
Explanation: In this program, ‘Base’ is an ambiguous base of ‘Multiple’. So it is producing an error. And this program is a virtual base class.

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class student
4. {
5. public:
6. int rno , m1 , m2 ;
7. void get()
8. {
9. rno = 15, m1 = 10, m2 = 10;
10. }
11. };
12. class sports
13. {
14. public:
15. int sm;
16. void getsm()
17. {
18. sm = 10;
19. }
20. };
21. class statement:public student,public sports
22. {
23. int tot,avg;
24. public:
25. void display()
26. {
27. tot = (m1 + m2 + sm);
28. avg = tot / 3;
29. cout << tot;
30. cout << avg;
31. }
32. };
33. int main()
34. {
35. statement obj;
36. obj.get();
37. obj.getsm();
38. obj.display();
39. }

a) 3100  
b) 3010  
c) 2010  
d) 1010

Answer: b  
Explanation: In this program, We are calculating the total and average marks of a student by using multiple inheritance.  
Output:

$ g++ mul1.cpp

$ a.out

3010

7. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. struct a
4. {
5. int count;
6. };
7. struct b
8. {
9. int\* value;
10. };
11. struct c : public a, public b
12. {
13. };
14. int main()
15. {
16. c\* p = new c;
17. p->value = 0;
18. cout << "Inherited";
19. return 0;
20. }

a) Inherited  
b) Error  
c) Runtime error  
d) inherited

Answer: a  
Explanation: In this program, We apply the multiple inheritance to structure.  
Output:

$ g++ mul2.cpp

$ a.out

Inherited

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Base1
4. {
5. protected:
6. int SampleDataOne;
7. public:
8. Base1()
9. {
10. SampleDataOne = 100;
11. }
12. ~Base1()
13. {
14. }
15. int SampleFunctOne()
16. {
17. return SampleDataOne;
18. }
19. };
20. class Base2
21. {
22. protected:
23. int SampleDataTwo;
24. public:
25. Base2()
26. {
27. SampleDataTwo = 200;
28. }
29. ~Base2()
30. {
31. }
32. int SampleFunctTwo()
33. {
34. return SampleDataTwo;
35. }
36. };
37. class Derived1 : public Base1, public Base2
38. {
39. int MyData;
40. public:
41. Derived1()
42. {
43. MyData = 300;
44. }
45. ~Derived1()
46. {
47. }
48. int MyFunct()
49. {
50. return (MyData + SampleDataOne + SampleDataTwo);
51. }
52. };
53. int main()
54. {
55. Base1 SampleObjOne;
56. Base2 SampleObjTwo;
57. Derived1 SampleObjThree;
58. cout << SampleObjThree.Base1 :: SampleFunctOne() << endl;
59. cout << SampleObjThree.Base2 :: SampleFunctTwo() << endl;
60. return 0;
61. }

a) 100  
b) 200  
c) Both 100 & 200  
d) 150

Answer: c  
Explanation: In this program, We are passing the values by using multiple inheritance and printing the derived values.  
Output:

$ g++ mul4.cpp

$ a.out

100

200

9. Which design patterns benefit from the multiple inheritances?  
a) Adapter and observer pattern  
b) Code pattern  
c) Glue pattern  
d) Star pattern

Answer: a  
Explanation: Adapter and observer patterns benefit from the multiple inheritances.

10. What are the things are inherited from the base class?  
a) Constructor and its destructor  
b) Operator=() members  
c) Friends  
d) All of the mentioned

Answer: d  
Explanation: These things can provide necessary information for the base class to make a logical decision.

# C++ Programming Questions and Answers – Inheritance – 1

1. What is Inheritance in C++?  
a) Wrapping of data into a single class  
b) Deriving new classes from existing classes  
c) Overloading of classes  
d) Classes with same names

Answer: b  
Explanation: Inheritance is the concept of OOPs in which new classes are derived from existing classes in order to reuse the properties of classes defined earlier.

2. How many specifiers are used to derive a class?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are 3 specifiers used to derive a class. They are private, protected and public.

3. Which specifier makes all the data members and functions of base class inaccessible by the derived class?  
a) private  
b) protected  
c) public  
d) both private and protected

Answer: a  
Explanation: Private access specifier is used to make all the data members and functions of the base class inaccessible.

4. If a class is derived privately from a base class then \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) no members of the base class is inherited  
b) all members are accessible by the derived class  
c) all the members are inherited by the class but are hidden and cannot be accessible  
d) no derivation of the class gives an error

Answer: c  
Explanation: Whenever a class is derived, all the members of the base class is inherited by the derived class but are not accessible by the derived class.

5. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

int a, b;

float d;

public:

void change(int i){

a = i;

}

void value\_of\_a(){

cout<<a;

}

};

class B: private A

{

};

int main(int argc, char const \*argv[])

{

B b;

cout<<sizeof(B);

return 0;

}

a) 8  
b) 12  
c) Error  
d) Segmentation fault

Answer: b  
Explanation: As class B is derived from class A and class A has three members with each of 4 bytes size hence size of B equal to 3 \* 4 = 12 bytes.

6. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

float d;

public:

int a;

void change(int i){

a = i;

}

void value\_of\_a(){

cout<<a;

}

};

class B: public A

{

int a = 15;

public:

void print(){

cout<<a;

}

};

int main(int argc, char const \*argv[])

{

B b;

b.change(10);

b.print();

b.value\_of\_a();

return 0;

}

a) 1010  
b) 1510  
c) 1515  
d) 5110

Answer: b  
Explanation: When change() is called it sets parents class ‘a’ variable = 10. When print() is called then ‘a’ from class B is printed and wehn value\_of\_a() is called then ‘a’ from class A is printed.

7. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

float d;

public:

A(){

cout<<"Constructor of class A**\n**";

}

};

class B: public A

{

int a = 15;

public:

B(){

cout<<"Constructor of class B**\n**";

}

};

int main(int argc, char const \*argv[])

{

B b;

return 0;

}

a)Constructor of class A

Constructor of class B

b) Constructor of class A  
c) Constructor of class B  
d)Constructor of class B

Constructor of class A

Answer: a  
Explanation: When a derived class is declared it calls both its constructor and the base class constructor. It first calls the base class constructor and then its own constructor.

8. What is a virtual function in C++?  
a) Any member function of a class  
b) All functions that are derived from the base class  
c) All the members that are accessing base class data members  
d) All the functions which are declared in the base class and is re-defined/overridden by the derived class

Answer: d  
Explanation: Virtual function is a function that is declared inside the base class and is re-defined inside the derived class.

9. Which is the correct syntax of declaring a virtual function?  
a) virtual int func();  
b) virtual int func(){};  
c) inline virtual func();  
d) inline virtual func(){};

Answer: a  
Explanation: To make a function virtual function we just need to add virtual keyword at the starting of the function declaration.

10. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A{

float d;

public:

virtual void func(){

cout<<"Hello this is class A**\n**";

}

};

class B: public A{

int a = 15;

public:

void func(){

cout<<"Hello this is class B**\n**";

}

};

int main(int argc, char const \*argv[])

{

B b;

b.func();

return 0;

}

a) Hello this is class B  
b) Hello this is class A  
c) Error  
d) Segmentation fault

Answer: a  
Explanation: Normal execution of the program and object calls func() from class B.

11. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

float d;

public:

virtual void func(){

cout<<"Hello this is class A**\n**";

}

};

class B: public A

{

int a = 15;

public:

void func(){

cout<<"Hello this is class B**\n**";

}

};

int main(int argc, char const \*argv[])

{

A \*a;

a->func();

return 0;

}

a) Hello this is class A  
b) Hello this is class B  
c) Error  
d) Segmentation Fault

Answer: d  
Explanation: As object ‘a’ is a pointer object and we know every pointer needs to be initialised memory before use. Hence segmentation fault. Use A \*a = new A(); to initialise memory to the object.

12. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

float d;

public:

virtual void func(){

cout<<"Hello this is class A**\n**";

}

};

class B: public A

{

int a = 15;

public:

void func(){

cout<<"Hello this is class B**\n**";

}

};

int main(int argc, char const \*argv[])

{

A \*a = new A();

B b;

a = &b;

a->func();

return 0;

}

a) Hello this is class A  
b) Hello this is class B  
c) Error  
d) Segmentation Fault

Answer: b  
Explanation: As pointer object a is pointing to the object b hence the definition of virtual function defined inside the class B will be class. This is one of the use of virtual function.

13. Which statement is incorrect about virtual function.  
a) They are used to achieve runtime polymorphism  
b) They are used to hide objects  
c) Each virtual function declaration starts with the virtual keyword  
d) All of the mentioned

Answer: b  
Explanation: Virtual function are used to achieve runtime polymorphism by calling the right function during runtime. Their declaration starts with a virtual keyword.

14. The concept of deciding which function to invoke during runtime is called \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) late binding  
b) dynamic linkage  
c) static binding  
d) both late binding and dynamic linkage

Answer: d  
Explanation: The concept of deciding which function to invoke during runtime is called late binding or dynamic linkage. Late binding because function binding to the object is done during runtime. Dynamic linkage because this binding is done during runtime.

15. What is a pure virtual function?  
a) A virtual function defined inside the base class  
b) A virtual function that has no definition relative to the base class  
c) A virtual function that is defined inside the derived class  
d) Any function that is made virtual

Answer: b  
Explanation: A virtual function that has no definition relative to the base class is called a pure virtual function.

# C++ Programming Questions and Answers – Inheritance – 2

1. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class Mammal

{

public:

Mammal(){

cout<<"I'm a Mammal**\n**";

}

~Mammal(){}

};

class Human: public Mammal

{

public:

Human(){

cout<<"I'm a Human**\n**";

}

~Human(){}

};

class Male: public Human

{

public:

Male(){

cout<<"I'm a Male**\n**";

}

~Male(){}

};

class Female: public Human

{

public:

Female(){

cout<<"I'm a Female**\n**";

}

~Female(){}

};

int main(int argc, char const \*argv[])

{

Male M;

return 0;

}

a)I'm a Mammal

I'm a Human

I'm a Male

b)I'm a Mammal

I'm a Human

I'm a Female

c)I'm a Human

I'm a Male

d)I'm a Mammal

I'm a Male

Answer: a  
Explanation: As the Male class is derived from Human class and Human class is derived from the Mammal class. Therefore when an object of Male is declared then three constructors will be called namely Mammal(), Human() and Male() in the given order.

2. What is the order of Constructors call when the object of derived class B is declared, provided class B is derived from class A?  
a) Constructor of A followed by B  
b) Constructor of B followed by A  
c) Constructor of A only  
d) Constructor of B only

Answer: a  
Explanation: Firstly the Constructor of class A is called then class B because the Constructor of the base class is called before derived class.

3. What is the order of Destructors call when the object of derived class B is declared, provided class B is derived from class A?  
a) Destructor of A followed by B  
b) Destructor of B followed by A  
c) Destructor of A only  
d) Destructor of B only

Answer: b  
Explanation: Order of Destructor call is just reverse of the order of Destructors call. First, the destructor of the derived class is called then Destructor of the base class is called.

4. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class Mammal

{

public:

virtual void Define(){

cout<<"I'm a Mammal**\n**";

}

};

class Human: public Mammal

{

public:

void Define(){

cout<<"I'm a Human**\n**";

}

};

class Male: public Human

{

public:

void Define(){

cout<<"I'm a Male**\n**";

}

};

class Female: public Human

{

public:

void Define(){

cout<<"I'm a Female**\n**";

}

};

int main(int argc, char const \*argv[])

{

Mammal \*M;

Male m;

Female f;

\*M = m;

M->Define();

return 0;

}

a) I’m a Male  
b) I’m a Mammal  
c) Error  
d) Segmentation Fault

Answer: d  
Explanation: As the Mammal pointer \*M is not Initialized memory therefore program results into segmentation faults.

5. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class Mammal

{

public:

virtual void Define(){

cout<<"I'm a Mammal**\n**";

}

};

class Human: public Mammal

{

public:

void Define(){

cout<<"I'm a Human**\n**";

}

};

class Male: public Human

{

public:

void Define(){

cout<<"I'm a Male**\n**";

}

};

class Female: public Human

{

public:

void Define(){

cout<<"I'm a Female**\n**";

}

};

int main(int argc, char const \*argv[])

{

Mammal \*M = new Mammal();

Male m;

Female f;

\*M = m;

M->Define();

M = &m;

M->Define();

return 0;

}

a)I'm a Male

I'm a Male

b)I'm a Male

I'm a Mammal

c)I'm a Mammal

I'm a Male

d)I'm a Mammal

I'm a Mammal

Answer: c  
Explanation: There is a difference between pointer and references. Pointer stores the address of a variable so we need dereferencing operator to access the pointed variable whereas references are another name for that variable so we don’t need any dereferencing operator, they are dereference by compiler itself therefore when we are using pointer then Mammal class definition is called and when reference is used then Male class definition is used.

6. Virtual functions in C++ tells the compiler to perform \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ on such functions.  
a) static binding  
b) late binding  
c) compile time binding  
d) no binding

Answer: b  
Explanation: Virtual function in C++ adds the power of late binding by deciding the type of object during run-time.

7. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class Mammal

{

public:

virtual void Define(){

cout<<"I'm a Mammal**\n**";

}

};

class Human: public Mammal

{

private:

void Define(){

cout<<"I'm a Human**\n**";

}

};

int main(int argc, char const \*argv[])

{

Mammal \*M = new Mammal();

Human H;

M = &H;

M->Define();

return 0;

}

a) Error  
b) Segmnetation fault  
c) I’m a Human  
d) Garbage Value

Answer: c  
Explanation: Using base class pointer we can call private functions of derived by using virtual keyword because virtual function asks compiler performs late binding i.e. bind function at run-time and at run-time there is no checking of access specifiers. Hence it can access private members.

8. Which concept of OOPs is shown by Virtual Functions?  
a) Inheritance  
b) Encapsulation  
c) Polymorphism  
d) Abstraction

Answer: c  
Explanation: Virtual function allows us to give different definitions of the same function i.e. overloading of functions which is known as Polymorphism.

9. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

public:

virtual A(){

cout<<"A's Constructor**\n**";

}

};

class B: public A

{

public:

A(){

cout<<"Present inside the class B**\n**";

}

};

int main(int argc, char const \*argv[])

{

A a;

return 0;

}

a) A’s Constructor  
b) Present inside the class B  
c) Error  
d) Segmentation fault

Answer: c  
Explanation: C++ does not allows programmers to make constructor a virtual function.

10. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

public:

virtual static void show(){

cout<<"class A**\n**";

}

};

class B: public A

{

public:

static void show(){

cout<<"class B**\n**";

}

};

int main(int argc, char const \*argv[])

{

A \*a = new A();

B b;

a = &b;

a->show();

return 0;

}

a) class B  
b) Error  
c) Segmentation fault  
d) class A

Answer: b  
Explanation: A function cannot be made virtual and static at the same time.

11. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

int a;

public:

A(){

a = 0;

}

static void show(){

a++;

cout<<a;

}

};

class B: public A

{

public:

};

int main(int argc, char const \*argv[])

{

B b;

b.show();

return 0;

}

a) 1  
b) 0  
c) Segmentation fault  
d) Error

Answer: d  
Explanation: Non-static members of class cannot be used inside a static functions of class.

12. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

int a;

public:

A(){

a = 0;

}

void show(){

a++;

cout<<"a: "<<a<<endl;

}

};

class B: public A

{

public:

};

int main(int argc, char const \*argv[])

{

B b;

b.show();

return 0;

}

a) a: 1  
b) a: 0  
c) Error  
d) Segmentation fault

Answer: a  
Explanation: Though the constructor of class A is not called to initialize variable ‘a’ but as we know whenever we create an object of the derived class, constructors of both base and derived classes are called hence variable ‘a’ is initialized and program runs perfectly.

13. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

int a;

public:

A(){

a = 0;

}

void show(){

a++;

cout<<"a: "<<a<<endl;

}

};

class B: private A

{

public:

};

int main(int argc, char const \*argv[])

{

B b;

b.show();

return 0;

}

a) Error  
b) Segmentation fault  
c) a: 1  
d) a: 0

Answer: a  
Explanation: As class B is derived privately from A hence all the members of class A cannot be accessible by the object of class B hence the program gives error.

14. What happens if the following C++ program is compiled?

#include <iostream>

#include <string>

using namespace std;

class A

{

int a;

public:

A(){

a = 0;

}

void show(){

a++;

cout<<"a: "<<a<<endl;

}

};

class B: private A

{

public:

void show(){

show();

}

};

int main(int argc, char const \*argv[])

{

B b;

b.show();

return 0;

}

a) Error because of the conflicts between two show() function in class B  
b) Program will compile successfully  
c) Error due to self call in show() function  
d) Error because show() function from class A is derived privately

Answer: b  
Explanation: As the program is syntactically correct and as one show() function is in class A and other in class B therefore no conflicts in same name function. Therefore program compiles successfully.

15. Pick the correct statement.  
a) Virtual function can have different names in the base and derived class  
b) Virtual function cannot be applied in Multiple Inheritance classes  
c) Virtual function are different in definitions only  
d) Virtual function does early binding

Answer: c  
Explanation: Virtual functions differ in definitions only, prototype are similar. They does the late binding. They are applicable to all types of inheritance.

# C++ Programming Questions and Answers – Access Control

1. Which access specifier is used where one wants data members to be accessed by other classes but not from outside objects?  
a) private  
b) protected  
c) public  
d) both protected and public

Answer: b  
Explanation: Protected and public members are accessible from derived classes but public members can be accessed by objects of the class so protected specifier is the answer.

2. Which of the following describes the protected access specifier?  
a) The variable is visible only outside inside the block  
b) The variable is visible everywhere  
c) The variable is visible to its block and to it’s derived class  
d) The variable is not visible to its block

Answer: c  
Explanation: Protected members are visible to its block and to the derived classes and not visible to outside objects or variables.

3. To which of the following access specifiers are applicable?  
a) Member data  
b) Functions  
c) Both Member data & Functions  
d) Protected members

Answer: c  
Explanation: The access specifiers can be applicable to the member data and functions because they need to be accessed outside the block.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class student
4. {
5. public:
6. int rno , m1 , m2 ;
7. protected:
8. void get()
9. {
10. rno = 15, m1 = 10, m2 = 10;
11. }
12. };
13. class sports
14. {
15. public:
16. int sm;
17. void getsm()
18. {
19. sm = 10;
20. }
21. };
22. class statement : public student, public sports
23. {
24. int tot, avg;
25. public:
26. void display()
27. {
28. tot = (m1 + m2 + sm);
29. avg = tot / 3;
30. cout << tot;
31. cout << avg;
32. }
33. void setObject()
34. {
35. get();
36. }
37. };
38. int main()
39. {
40. statement obj;
41. obj.setObject();
42. obj.getsm();
43. obj.display();
44. }

a) 3010  
b) 1010  
c) 2100  
d) Error

Answer: a  
Explanation: In this program we setting values of m1 and m2 using obj.setObject() function derived from student class. setting calue of sm using getsm() derived from sports function and then displaying the outputs using display() function in statement class.

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. struct A
4. {
5. int i;
6. char j;
7. float f;
8. void func();
9. };
10. void A :: func() {}
11. struct B
12. {
13. public:
14. int i;
15. char j;
16. float f;
17. void func();
18. };
19. void B :: func() {}
20. int main()
21. {
22. A a; B b;
23. a.i = b.i = 1;
24. a.j = b.j = 'c';
25. a.f = b.f = 3.14159;
26. a.func();
27. b.func();
28. cout << "Allocated";
29. return 0;
30. }

a) Allocated  
b) Error  
c) 3.14159  
d) 1

Answer: a  
Explanation: In this program, We used access specifiers for structures, As we declared all methods as public, The values can be allocated.  
Output:

$ g++ acc.cpp

$ a.out

Allocated

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. struct A
4. {
5. private:
6. int i, j, k;
7. public:
8. int f();
9. void g();
10. };
11. int A :: f()
12. {
13. return i + j + k;
14. }
15. void A :: g()
16. {
17. i = j = k = 0;
18. }
19. class B
20. {
21. int i, j, k;
22. public:
23. int f();
24. void g();
25. };
26. int B :: f()
27. {
28. return i + j + k;
29. }
30. void B :: g()
31. {
32. i = j = k = 0;
33. }
34. int main()
35. {
36. A a;
37. B b;
38. a.f();
39. a.g();
40. b.f();
41. b.g();
42. cout << "Identical results would be produced";
43. }

a) 50  
b) Identical results would be produced  
c) Error  
d) Runtime error

Answer: b  
Explanation: In this program, We apply the access specifiers to both the class and the structure.  
Output:

$ g++ acc1.cpp

$ a.out

7. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Cat
4. {
5. public:
6. int age;
7. int weight;
8. };
9. int main()
10. {
11. Cat f;
12. f.age = 56;
13. cout << "Gates is " ;
14. cout << f.age << " years old.**\n**";
15. }

a) Gates is  
b) Gates is 56 years old  
c) Error  
d) Gates is 53 years old

Answer: b  
Explanation: In this program, We passed the value from main function to class and returning it to the main and then printing it.  
Output:

$ g++ acc2.cpp

$ a.out

Gates is 56 years old

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. struct X;
4. struct Y
5. {
6. void f(X\*);
7. };
8. struct X
9. {
10. private:
11. int i;
12. public:
13. void initialize();
14. friend void g(X\* , int);
15. friend void Y :: f(X\*);
16. friend struct Z;
17. friend void h();
18. };
19. void X :: initialize()
20. {
21. i = 0;
22. }
23. void g(X\* x, int i)
24. {
25. x -> i = i;
26. }
27. void Y :: f(X \* x)
28. {
29. x -> i = 47;
30. cout << x->i;
31. }
32. struct Z
33. {
34. private:
35. int j;
36. public:
37. void initialize();
38. void g(X\* x);
39. };
40. void Z::initialize()
41. {
42. j = 99;
43. }
44. void Z::g(X\* x)
45. {
46. x -> i += j;
47. }
48. void h()
49. {
50. X x;
51. x.i = 100;
52. cout << x.i;
53. }
54. int main()
55. {
56. X x;
57. Z z;
58. z.g(&x);
59. cout << "Data accessed";
60. }

a) 99  
b) 47  
c) Data accessed  
d) 67

Answer: c  
Explanation: In this program, We are using the access specifiers to friend function to manipulate the values.  
Output:

$ g++ acc3.cpp

$ a.out

Data accessed

9. Members of which access specifiers are not inherited?  
a) Public  
b) Protected  
c) Private  
d) None of the mentioned

Answer: c  
Explanation: All the data members and member functions of a class are private by default.

10. What is the importance of mutable keyword?  
a) It allows the data member to change within a const member function  
b) It will not allow the data member to change within a const member function  
c) It will copy the values of the variable  
d) It allows the data member to change outside a const member function

Answer: a  
Explanation: Mutable keyword allows assigning values to a data member belonging to a class defined as “Const” or constant.

# C++ Programming Questions and Answers – Run Time Type Information

1. What is the Run-Time Type Information?  
a) Information about an object’s data type at runtime  
b) Information about the variables  
c) Information about the given block  
d) Information about the functions

Answer: a  
Explanation: With the help of RTTI, We can get the information about the data type at the runtime.

2. Which operators are part of RTTI?  
a) dynamic\_cast()  
b) typeid  
c) both dynamic\_cast<> & typeid  
d) dynamic\_cast[]

Answer: c  
Explanation: The dynamic\_cast<> operation and typeid operator in C++ are part of RTTI.

3. To which type of class, We can apply RTTI?  
a) Encapsulation  
b) Polymorphic  
c) Derived  
d) Static

Answer: b  
Explanation: RTTI is available only for classes which are polymorphic, which means they have at least one virtual method.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. class base { virtual void dummy() {} };
5. class derived: public base { int a; };
6. int main ()
7. {
8. try
9. {
10. base \* pba = new derived;
11. base \* pbb = new base;
12. derived \* pd;
13. pd = dynamic\_cast<derived\*>(pba);
14. if (pd == 0)
15. cout << "Null pointer on first type-cast" << endl;
16. pd = dynamic\_cast<derived\*>(pbb);
17. if (pd == 0)
18. cout << "Null pointer on second type-cast" << endl;
19. }
20. catch (exception& e)
21. {
22. cout << "Exception: " << e.what();
23. }
24. return 0;
25. }

a) Null pointer on first type-cast  
b) Null pointer on second type-cast  
c) Exception  
d) Null pointer on third type-cast

Answer: b  
Explanation: In this program, We apply the dynamic cast to pd. Based on the value in the pd, it produces the output.  
Output:

$ g++ rtti.cpp

$ a.out

Null pointer on second type-cast

5. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <typeinfo>
3. using namespace std;
4. int main ()
5. {
6. int \* a;
7. int b;
8. a = 0; b = 0;
9. if (typeid(a) != typeid(b))
10. {
11. cout << typeid(a).name();
12. cout << typeid(b).name();
13. }
14. return 0;
15. }

a) Pi  
b) i  
c) Both pi & i  
d) f

Answer: c  
Explanation: In this program, We are finding the typeid of the given variables.  
Output:

$ g++ rtti1.cpp

$ a.out

Pii

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <typeinfo>
3. #include <exception>
4. using namespace std;
5. class base
6. {
7. virtual void f(){}
8. };
9. class derived : public base {};
10. int main ()
11. {
12. try
13. {
14. base\* a = new base;
15. base\* b = new derived;
16. cout << typeid(\*a).name() << '**\t**';
17. cout << typeid(\*b).name();
18. }
19. catch (exception& e)
20. {
21. cout << "Exception: " << e.what() << endl;
22. }
23. return 0;
24. }

a) base\*  
b) derived\*  
c) 4base and 7derived  
d) Exception:derived

Answer: c  
Explanation: In this program, We apply the typeid to the polymorphic class.  
Output:

$ g++ rtti2.cpp

$ a.out

4base 7derived

7. What will be the output of the following C++ code?

1. #include <typeinfo>
2. #include <iostream>
3. using namespace std;
4. class A
5. {
6. public:
7. virtual ~A();
8. };
9. int main()
10. {
11. A\* a = NULL;
12. try
13. {
14. cout << typeid(\*a).name() << endl;
15. }
16. catch (bad\_typeid)
17. {
18. cout << "Object is NULL" << endl;
19. }
20. }

a) int  
b) float  
c) double  
d) object is NULL

Answer: d  
Explanation: In this program, We are using the bad typeid() for a. So it is arising an exception.  
Output:

$ g++ rtti3.cpp

$ a.out

object is NULL

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. struct A
4. {
5. virtual void f()
6. {
7. cout << "Class A" << endl;
8. }
9. };
10. struct B : A
11. {
12. virtual void f()
13. {
14. cout << "Class B" << endl;
15. }
16. };
17. struct C : A
18. {
19. virtual void f()
20. {
21. cout << "Class C" << endl;
22. }
23. };
24. void f(A\* arg)
25. {
26. B\* bp = dynamic\_cast<B\*>(arg);
27. C\* cp = dynamic\_cast<C\*>(arg);
28. if (bp)
29. bp -> f();
30. else if (cp)
31. cp -> f();
32. else
33. arg -> f();
34. };
35. int main()
36. {
37. A aobj;
38. C cobj;
39. A\* ap = &cobj;
40. A\* ap2 = &aobj;
41. f(ap);
42. f(ap2);
43. }

a) Class C  
b) Class A  
c) Both Class C & A  
d) Class D

Answer: c  
Explanation: In this program, We applied the dynamic casting to structure and produced the output.  
Output:

$ g++ rtti4.cpp

$ a.out

Class C

Class A

9. What is meant by type\_info?  
a) Used to hold the type information returned by the typeid operator  
b) Used to hold the type information returned by the dynamic\_cast  
c) Used to hold the type information returned by the static\_cast  
d) Used to hold the type information returned by the static\_id

Answer: a  
Explanation: type\_info is used to hold the type information returned by the typeid operator.

10. At which time does the static\_cast can be applied?  
a) Compile-time construct  
b) Runtime construct  
c) Both Compile-time & Runtime construct  
d) Runtime deconstruct

Answer: a  
Explanation: Static\_cast can be applied to only compile-time construct and not during run time construct.

# C++ Programming Questions and Answers – Pointers to Members

1. Which is referred by pointers to member?  
a) Static members of class objects  
b) Non-static members of class objects  
c) Referring to whole class  
d) Dynamic members of class objects

Answer: b  
Explanation: We cannot use a pointer to member to point to a static class member because the address of a static member is not associated with any particular object.

2. What should be used to point to a static class member?  
a) Smart pointer  
b) Dynamic pointer  
c) Normal pointer  
d) Static pointer

Answer: c  
Explanation: Normal pointer is sed to point to a static class member.

3. Which operator is used in pointer to member function?  
a) .\*  
b) ->\*  
c) Both .\* & ->\*  
d) $\*

Answer: c  
Explanation: The pointer to member operators .\* and ->\* are used to bind a pointer to a member of a specific class object.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class X
4. {
5. public:
6. int a;
7. void f(int b)
8. {
9. cout<< b << endl;
10. }
11. };
12. int main()
13. {
14. int X :: \*ptiptr = &X :: a;
15. void (X :: \* ptfptr) (int) = &X :: f;
16. X xobject;
17. xobject.\*ptiptr = 10;
18. cout << xobject.\*ptiptr << endl;
19. (xobject.\*ptfptr) (20);
20. }

a) 10

20

b)20

10

c) 20  
d) 10

Answer: a  
Explanation: In this program, We are assigning 10 and printing it in the  
main function and then for value 20, We are passing the value to class and  
printing it.  
Output:

$ g++ ptm.cpp

$ a.out

10

20

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Testpm
4. {
5. public:
6. void m\_func1()
7. {
8. cout << "func1**\n**";
9. }
10. int m\_num;
11. };
12. void (Testpm :: \*pmfn)() = &Testpm :: m\_func1;
13. int Testpm :: \*pmd = &Testpm :: m\_num;
14. int main()
15. {
16. Testpm ATestpm;
17. Testpm \*pTestpm = new Testpm;
18. (ATestpm.\*pmfn)();
19. (pTestpm ->\* pmfn)();
20. ATestpm.\*pmd = 1;
21. pTestpm ->\* pmd = 2;
22. cout << ATestpm.\*pmd << endl
23. << pTestpm ->\* pmd << endl;
24. }

a) func1  
b) func1

func1

c)1

2

d)func1

func1

1

2

Answer: d  
Explanation: In this program, As we are passing the value twice to the method  
in the class, It is printing the func1 twice and then it is printing the given  
value.  
Output:

$ g++ ptm1.cpp

$ a.out

func1

func1

1

2

 6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Car
4. {
5. public:
6. int speed;
7. };
8. int main()
9. {
10. int Car :: \*pSpeed = &Car :: speed;
11. Car c1;
12. c1.speed = 1;
13. cout << c1.speed << endl;
14. c1.\*pSpeed = 2;
15. cout << c1.speed << endl;
16. return 0;
17. }

a) 1  
b) 2  
c) Both 1 & 2  
d) 4

Answer: c  
Explanation: In this program, We are printing the value by direct access and another one by using pointer to member.  
Output:

$ g++ ptm2.cpp

$ a.out

1

2

7. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class bowl
4. {
5. public:
6. int apples;
7. int oranges;
8. };
9. int count\_fruit(bowl \* begin, bowl \* end, int bowl :: \*fruit)
10. {
11. int count = 0;
12. for (bowl \* iterator = begin; iterator != end; ++ iterator)
13. count += iterator ->\* fruit;
14. return count;
15. }
16. int main()
17. {
18. bowl bowls[2] = {{ 1, 2 },{ 3, 5 }};
19. cout << "I have " << count\_fruit(bowls, bowls + 2, & bowl :: apples) << " apples**\n**";
20. cout << "I have " << count\_fruit(bowls, bowls + 2, & bowl :: oranges) << " oranges**\n**";
21. return 0;
22. }

a)I have 4 apples

I have 7 oranges

b) I have 3 apples

I have 5 oranges

c) I have 1 apples

I have 5 oranges

d) I have 1 apples

I have 7 oranges

Answer: a  
Explanation: In this program, We are passing the value to the class and adding the values and printing it in the main.  
Output:

$ g++ ptm3.cpp

$ a.out

I have 4 apples

I have 7 oranges

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Foo
4. {
5. public:
6. Foo(int i = 0){ \_i = i;}
7. void f()
8. {
9. cout << "Executed"<<endl;
10. }
11. private:
12. int \_i;
13. };
14. int main()
15. {
16. Foo \*p = 0;
17. p -> f();
18. }

a) Executed  
b) Error  
c) Runtime error  
d) 10

Answer: a  
Explanation: In this program, We passes the value to the class and printing it.  
Output:

$ g++ ptm4.cpp

$ a.out

Executed

9. Which is the best design choice for using pointer to member function?  
a) Interface  
b) Class  
c) Structure  
d) Block

Answer: a  
Explanation: Interface is the best design choice for using pointer to member function.

10. What is the operation for .\*?  
a) It combines the first operand and the second operand  
b) It separates the first operand and the second operand  
c) It reduces the data size  
d) It combines the first operand and the second operand and terminates third operand

Answer: a  
Explanation: The binary operator .\* combines its first operand, which must be an object of class type, with its second operand, which must be a pointer-to-member type.

# C++ Programming Questions and Answers – Free Store

1. Which is used to allocate and deallocate storage for objects during the execution?  
a) Stack  
b) Heap  
c) Freestore  
d) Queue

Answer: c  
Explanation: Free store is a pool of memory available for you to allocate and deallocate storage for objects during the execution of your program.

2. Which operators are used in the free store?  
a) new  
b) delete  
c) both new & delete  
d) terminate

Answer: c  
Explanation: new and delete operators are used to allocate and deallocate the memory for the program.

3. What type of class member is operator new?  
a) static  
b) dynamic  
c) const  
d) smart

Answer: a  
Explanation: static is a type of class member is operator new.

4. What will be the output of the following C++ code?

1. #include <new>
2. #include <iostream>
3. using namespace std;
4. struct A
5. {
6. virtual ~A() { };
7. void operator delete(void\* p)
8. {
9. cout << "A :: operator delete" << endl;
10. }
11. };
12. struct B : A
13. {
14. void operator delete(void\* p)
15. {
16. cout << "B :: operator delete" << endl;
17. }
18. };
19. int main()
20. {
21. A\* ap = new B;
22. delete ap;
23. }

a) A::operator delete  
b) B::operator delete  
c) Both A::operator delete & B::operator delete  
d) A:operator new

Answer: b  
Explanation: In this program, We are passing the value to the B, So we are printing B::operator delete.  
Output:

$ g++ free.cpp

$ a.out

B::operator delete

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. struct A
4. {
5. virtual ~A()
6. {
7. cout << "~A()" << endl;
8. }
9. void operator delete[](void\* p, size\_t)
10. {
11. cout << "A :: operator delete[]" << endl;
12. delete [] p;
13. }
14. };
15. struct B : A
16. {
17. void operator delete[](void\* p, size\_t)
18. {
19. cout << "B :: operator delete[]" << endl;
20. delete [] p;
21. }
22. };
23. int main()
24. {
25. A\* bp = new B[3];
26. delete[] bp;
27. };

a) ~A()  
b) A :: operator delete[]  
c) B :: operator delete[]  
d) Warning

Answer: d  
Explanation: In this program, the behavior of the statement delete[] bp is undefined.

$ g++ a.cpp

a.cpp: In static member function ‘static void A::operator delete [](void\*, size\_t)’:

a.cpp:12: warning: deleting ‘void\*’ is undefined

a.cpp: In static member function ‘static void B::operator delete [](void\*, size\_t)’:

a.cpp:20: warning: deleting ‘void\*’ is undefined

$ a.out

~A()

~A()

~A()

A :: operator delete[].

6. What will be the output of the following C++ code?

1. #include <cstdlib>
2. #include <iostream>
3. using namespace std;
4. class X
5. {
6. public:
7. void\* operator new(size\_t sz) throw (const char\*)
8. {
9. void\* p = malloc(sz);
10. if (p == 0)
11. throw "malloc() failed";
12. return p;
13. }
14. void operator delete(void\* p)
15. {
16. cout << "X :: operator delete(void\*)" << endl;
17. free(p);
18. }
19. };
20. class Y
21. {
22. int filler[100];
23. public:
24. void operator delete(void\* p, size\_t sz) throw (const char\*)
25. {
26. cout << "Freeing " << sz << " bytes" << endl;
27. free(p);
28. };
29. };
30. int main()
31. {
32. X\* ptr = new X;
33. delete ptr;
34. Y\* yptr = new Y;
35. delete yptr;
36. }

a) X::operator delete(void\*)  
b) Freeing 400 bytes  
c) Depends on the compiler  
d) Both X::operator delete(void\*) & Depends on the compiler

Answer: d  
Explanation: The memory value allocated for the program depends on compiler only.

$ g++ free2.cpp

$ a.out

X :: operator delete(void\*)

Freeing 400 bytes

7. What will be the output of the following C++ code?

1. #include <new>
2. #include<cstdlib>
3. #include <iostream>
4. using namespace std;
5. class X;
6. struct Node
7. {
8. X\* data;
9. bool filled;
10. Node() : filled(false) { }
11. };
12. class X
13. {
14. static Node buffer[];
15. public:
16. int number;
17. enum { size = 3};
18. void\* operator new(size\_t sz) throw (const char\*)
19. {
20. void\* p = malloc(sz);
21. if (sz == 0)
22. throw "Error: malloc() failed";
23. cout << "X :: operator new(size\_t)" << endl;
24. return p;
25. }
26. void \*operator new(size\_t sz, int location) throw (const char\*)
27. {
28. cout << "X :: operator new(size\_t, " << location << ")" << endl;
29. void\* p = 0;
30. if (location < 0 || location >= size || buffer[location].filled == true)
31. {
32. throw "Error: buffer location occupied";
33. }
34. else
35. {
36. p = malloc(sizeof(X));
37. if (p == 0)
38. throw "Error: Creating X object failed";
39. buffer[location].filled = true;
40. buffer[location].data = (X\*) p;
41. }
42. return p;
43. }
44. static void printbuffer()
45. {
46. for (int i = 0; i < size; i++)
47. {
48. cout << buffer[i].data->number << endl;
49. }
50. }
51. };
52. Node X::buffer[size];
53. int main()
54. {
55. try
56. {
57. X\* ptr1 = new X;
58. X\* ptr2 = new(0) X;
59. X\* ptr3 = new(1) X;
60. X\* ptr4 = new(2) X;
61. ptr2->number = 10000;
62. ptr3->number = 10001;
63. ptr4->number = 10002;
64. X :: printbuffer();
65. X\* ptr5 = new(0) X;
66. }
67. catch (const char\* message)
68. {
69. cout << message << endl;
70. }
71. }

a) X::operator new(size\_t)  
b) Error  
c) Runtime error  
d) operator new(size\_d)

Answer: c  
Explanation: In this program, We are giving a location to two variables in the program, So it is arising an exception.  
Output:

$ g++ free3.cpp

$ a.out

X::operator new(size\_t)

X::operator new(size\_t, 0)

X::operator new(size\_t, 1)

X::operator new(size\_t, 2)

10000

10001

10002

X::operator new(size\_t, 0)

Error: buffer location occupied

8. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <new>
3. #include <cstdlib>
4. using namespace std;
5. const int bsize = 512;
6. int \*pa;
7. bool allocate = true;
8. void get\_memory()
9. {
10. cerr << "free store exhausted" << endl;
11. delete [] pa;
12. allocate = false;
13. }
14. void eat\_memory(int size)
15. {
16. int \*p = new int[size];
17. if (allocate)
18. eat\_memory(size);
19. else
20. cerr << "free store addr = " << p << endl;
21. }
22. int main()
23. {
24. set\_new\_handler(get\_memory);
25. pa = new int[bsize];
26. cerr << "free store addr = " << pa << endl;
27. eat\_memory(bsize);
28. return 0;
29. }

a) free store addr  
b) Error  
c) Segmentation fault  
d) free store exhausted

Answer: c  
Explanation: In this program, The memory will go beyond the limit, So there will be exhaustion in memory.  
Output:

$ g++ free4.cpp

$ a.out

free store addr = 0x80a8008

Segmentation fault

9. What must be an operand of operator delete?  
a) Pointer  
b) Array  
c) Stack  
d) Queue

Answer: a  
Explanation: The operand of delete must be a pointer returned by new.

10. How can object be allocated outside the object lifetime?  
a) int  
b) float  
c) void\*  
d) char$

Answer: c  
Explanation: void\* object can be allocated outside the object lifetime.

# C++ Programming Questions and Answers – Standard Library Design

1. Pick out the wrong header file about strings.  
a) <string>  
b) <regex>  
c) <ios>  
d) <reg>

Answer: c  
Explanation: The standard header files for string is string and regex. So the wrong one presented here is ios.

2. Which is best for coding the standard library for c++?  
a) no trailing underscores on names  
b) complex objects are returned by value  
c) have a member-swap()  
d) all of the mentioned

Answer: d  
Explanation: Best coding for the standard library for c++ is:  
-> No trailing underscores on names  
-> Complex objects are returned by value  
-> It should have a member-swap().

3. What is meant by vector in the container library contains?  
a) It is a sequence container that encapsulates dynamic size arrays  
b) It is a sequence container that encapsulates static size arrays  
c) It manages the memory  
d) It manages the length and size

Answer: a  
Explanation: Vector in the container library contains sequence container that manipulates and encapsulates dynamic size arrays.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. int main()
5. {
6. vector<int> v;
7. v.assign( 10, 42 );
8. for (int i = 0; i < v.size(); i++)
9. {
10. cout << v[i] << " ";
11. }
12. }

a) 42  
b) 42 42  
c) 424  
d) 42 for 10 times

Answer: d  
Explanation: In this program, We used the vector to print the 42 for 10 times.  
Output:

$ g++ std.cpp

$ a.out

42 42 42 42 42 42 42 42 42 42

5. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <list>
3. #include <queue>
4. using namespace std;
5. int main()
6. {
7. queue<char> q;
8. q.push('a');
9. q.push('b');
10. q.push('c');
11. cout << q.front();
12. q.pop();
13. cout << q.front();
14. q.pop();
15. cout << q.front();
16. q.pop();
17. }

a) ab  
b) abc  
c) a  
d) error

Answer: b  
Explanation: We are using queue in this program and queue follows FIFO strategy to handle data hence the following output pattern is observed.  
Output:

$ g++ std1.cpp

$ a.out

abc

6. What will be the output of the following C++ code?

1. #include <list>
2. #include <string>
3. #include <iostream>
4. using namespace std ;
5. typedef list<string> LISTSTR;
6. int main()
7. {
8. LISTSTR :: iterator i;
9. LISTSTR test;
10. test.insert(test.end(), "one");
11. test.insert(test.end(), "two");
12. LISTSTR test2(test);
13. LISTSTR test3(3, "three");
14. LISTSTR test4(++test3.begin(),
15. test3.end());
16. cout << "test:";
17. for (i = test.begin(); i != test.end(); ++i)
18. cout << " " << \*i << endl;
19. cout << "test:";
20. for (i = test2.begin(); i != test2.end(); ++i)
21. cout << " " << \*i << endl;
22. cout << "test:";
23. for (i = test3.begin(); i != test3.end(); ++i)
24. cout << " " << \*i << endl;
25. cout << "test:";
26. for (i = test4.begin(); i != test4.end(); ++i)
27. cout << " " << \*i << endl;
28. }

a) test  
b) test one  
c) test two  
d)test: one

two

test: one

two

test: three

three

three

test: three

three

Answer: d  
Explanation: In this program, We used the list to manipulate the given value.  
Output:

$ g++ std3.cpp

$ a.out

test: one

two

test: one

two

test: three

three

three

test: three

three

7. Pick out the wrong header file.  
a) <algorithm>  
b) <containers>  
c) <iterators>  
d) <process>

Answer: d  
Explanation: There is no header file named <process> in C++.

8. What is meant by standard c++ library?  
a) It is the collection of class definitions for standard data structures and a collection of algorithms  
b) It is a header file  
c) Collection of algorithms  
d) Step by step process

Answer: a  
Explanation: It is the collection of class definitions for standard data structures. This part of the library was derived from the Standard Template Library.

9. Pick out parameter for rehash method in unordered\_set in c++?  
a) count  
b) size  
c) hash  
d) type

Answer: a  
Explanation: count is used to return the new number of buckets.

10. What is the use of <exception> header  
a) Contains the standard exception files  
b) Contains the standard library files  
c) It is used to arise an exception in the program  
d) Reduce the memory size

Answer: a  
Explanation: <exception> header file contains standard exception files used for exception handling in a C++ program.

# C++ Programming Questions and Answers – Container Design

1. How many sets of requirements are need in designing a container?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are three sets of requirements. They are container interface requirements, Allocator interface requirements and iterator requirements.

2. Which interface in the container is required for storage management?  
a) Memory management  
b) Allocator interface  
c) Memory interface  
d) Storage interface

Answer: b  
Explanation: Allocator interface in the container is required for storage management.

3. Which is present in the basic interface of the allocator interface?  
a) Set of typedefs  
b) A pair of allocation functions  
c) allocate()  
d) All of the mentioned

Answer: d  
Explanation: The basic interface of an allocator class consists of a set of typedefs, a pair of allocation functions, allocate() and deallocate() and a pair of construction/destruction members, construct() and destroy().

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. class Component
5. {
6. public:
7. virtual void traverse() = 0;
8. };
9. class Leaf: public Component
10. {
11. int value;
12. public:
13. Leaf(int val)
14. {
15. value = val;
16. }
17. void traverse()
18. {
19. cout << value << ' ';
20. }
21. };
22. class Composite: public Component
23. {
24. vector < Component \* > children;
25. public:
26. void add(Component \*ele)
27. {
28. children.push\_back(ele);
29. }
30. void traverse()
31. {
32. for (int i = 0; i < children.size(); i++)
33. children[i]->traverse();
34. }
35. };
36. int main()
37. {
38. Composite containers[4];
39. for (int i = 0; i < 4; i++)
40. for (int j = 0; j < 3; j++)
41. containers[i].add(new Leaf(i \*3+j));
42. for (int k = 1; k < 4; k++)
43. containers[0].add(&(containers[k]));
44. for (int p = 0; p < 4; p++)
45. {
46. containers[p].traverse();
47. }
48. }

a) 345  
b) 678  
c) 901  
d)0 1 2 3 4 5 6 7 8 9 10 11

3 4 5

6 7 8

9 10 11

Answer: d  
Explanation: In this program, We are choosing and printing the numbers based on the certain limit and this is a composite design pattern.  
Output:

$ g++ cont.cpp

$ a.out

0 1 2 3 4 5 6 7 8 9 10 11

3 4 5

6 7 8

9 10 11

5. Which container provides random access iterators?  
a) vector  
b) deque  
c) sort  
d) both vector & deque

Answer: d  
Explanation: Vector & deque container provides random access iterators.

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <vector>
3. #include <iterator>
4. #include <stddef.h>
5. using namespace std;
6. template<class myType>
7. class SimpleContainer
8. {
9. public:
10. SimpleContainer(size\_t xDim, size\_t yDim, myType const& defaultValue)
11. : objectData(xDim \* yDim, defaultValue)
12. , xSize(xDim)
13. , ySize(yDim)
14. {
15. }
16. myType& operator()(size\_t x, size\_t y)
17. {
18. return objectData[y \* xSize + x];
19. }
20. myType const& operator()(size\_t x, size\_t y) const
21. {
22. return objectData[y \* xSize + x];
23. }
24. int getSize()
25. {
26. return objectData.size();
27. }
28. void inputEntireVector(vector<myType> inputVector)
29. {
30. objectData.swap(inputVector);
31. }
32. void printContainer(ostream& stream)
33. {
34. copy(objectData.begin(), objectData.end(),
35. ostream\_iterator<myType>(stream, ""*/\*No Space\*/*));
36. }
37. private:
38. vector<myType> objectData;
39. size\_t xSize;
40. size\_t ySize;
41. };
42. template<class myType>
43. inline ostream& operator<<(ostream& stream, SimpleContainer<myType>& object)
44. {
45. object.printContainer(stream);
46. return stream;
47. }
48. void sampleContainerInterfacing();
49. int main()
50. {
51. sampleContainerInterfacing();
52. return 0;
53. }
54. void sampleContainerInterfacing()
55. {
56. static int const ConsoleWidth = 80;
57. static int const ConsoleHeight = 25;
58. size\_t width = ConsoleWidth;
59. size\_t height = ConsoleHeight;
60. SimpleContainer<int> mySimpleContainer(width, height, 0);
61. cout << mySimpleContainer.getSize() << endl;
62. mySimpleContainer(0, 0) = 5;
63. }

a) 2000  
b) No Space  
c) Error  
d) Depends on the compiler

Answer: d  
Explanation: In this program, We formed a simple container and got the size of it and printing it.  
Output:

$ g++ cont1.cpp

$ a.out

200

7. Which is used for manually writing lookup table?  
a) std:map  
b) std:lookup  
c) std:find  
d) std:lock

Answer: a  
Explanation: Lookup table means storing values in a table with unique keys for each value so that can be checked in future easily. So for such lookup tables maps are used in C++.

8. How can the member functions in the container be accessed?  
a) Iterator  
b) Indirect  
c) Both Iterator & Indirect  
d) Vector

Answer: a  
Explanation: The container manages the storage space for its elements and provides member functions to access them, either directly or through iterators which reference objects with similar properties to pointers.

9. Which of the following type does the container should define?  
a) Iterator type  
b) Vector type  
c) Storage type  
d) Memory type

Answer: a  
Explanation: Every container must define an iterator type. Iterators allow algorithms to iterate over the container’s contents.

10. Which are the parameters for the content of the buffer?  
a) start  
b) finish  
c) both start & finish  
d) pause

Answer: c  
Explanation: The contents of the buffer are initialized using the values from the iterator range supplied to the constructor by the start and finish parameters.

# C++ Programming MCQ – Vector

1. What do vectors represent?  
a) Static arrays  
b) Dynamic arrays  
c) Stack  
d) Queue

Answer: b  
Explanation: Vectors are sequence containers representing arrays that can change in size.

2. In which type of storage location are the vector members stored?  
a) Contiguous storage locations  
b) Non-contiguous storage locations  
c) Contiguous & Non-contiguous storage locations  
d) Memory storage locations

Answer: a  
Explanation: Vectors use contiguous storage locations for their elements, which means that their elements can also be accessed using offsets on regular pointers to its elements

3. How many vector container properties are there in c++?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are three container properties in c++. They are sequence, Dynamic array and allocator-aware.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. int main ()
5. {
6. unsigned int i;
7. vector<int> first;
8. vector<int> second (4, 100);
9. vector<int> third (second.begin(), second.end());
10. vector<int> fourth (third);
11. int myints[] = {16, 2, 77, 29};
12. vector<int> fifth (myints, myints + sizeof(myints) / sizeof(int) );
13. for (vector<int> :: iterator it = fifth.begin(); it != fifth.end(); ++it)
14. cout << ' ' << \*it;
15. return 0;
16. }

a) 16  
b) 16 2  
c) 16 2 77  
d) 16 2 77 29

Answer: d  
Explanation: In this program, We got the values and printing it by using the vector and we are contructing vectors.  
Output:

$ g++ vect.cpp

$ a.out

16 2 77 29

5. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. int main ()
5. {
6. vector<int> myvector;
7. int sum (0);
8. myvector.push\_back (100);
9. myvector.push\_back (200);
10. myvector.push\_back (300);
11. while (!myvector.empty())
12. {
13. sum += myvector.back();
14. myvector.pop\_back();
15. }
16. cout << sum << '**\n**';
17. return 0;
18. }

a) 500  
b) 600  
c) 700  
d) Error

Answer: b  
Explanation: In this program, We are forming a stack and adding the elements and We are finding the total number of elements that are in stack.  
Output:

$ g++ vect1.cpp

$ a.out

600

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. int main ()
5. {
6. vector<int> a (3, 0);
7. vector<int> b (5, 0);
8. b = a;
9. a = vector<int>();
10. cout << "Size of a " << int(a.size()) << '**\n**';
11. cout << "Size of b " << int(b.size()) << '**\n**';
12. return 0;
13. }

a)Size of a 0

Size of b 3

b)Size of a 3

Size of b 5

c) Error  
d)Size of a 3

Size of b 9

Answer: a  
Explanation: In this program, We are finding the size of the vector elements.  
Output:

$ g++ vect2.cpp

$ a.out

Size of a 0

Size of b 3

7. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. int main ()
5. {
6. vector<int> first;
7. first.assign (7,100);
8. vector<int>::iterator it;
9. it=first.begin()+1;
10. int myints[] = {1776,7,4};
11. cout << int (first.size()) << '**\n**';
12. return 0;
13. }

a) 10  
b) 9  
c) 8  
d) 7

Answer: d  
Explanation: In this program, We are finding the size of the vector elements and resizing it.  
Output:

$ g++ vect3.cpp

$ a.out

7

8. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. int main ()
5. {
6. vector<int> myvector (5);
7. int\* p = myvector.data();
8. \*p = 10;
9. ++p;
10. \*p = 20;
11. p[2] = 100;
12. for (unsigned i = 0; i < myvector.size(); ++i)
13. cout << ' ' << myvector[i];
14. return 0;
15. }

a) 10 20 0 100 0  
b) 10 20 0 100  
c) 10 20 0  
d) 10 20

Answer: a  
Explanation: In this program, We are allocating the values to the vector and unallocated values are left as zero.  
Output:

$ g++ vect4.cpp

$ a.out

10 20 0 100 0

9. Pick out the correct statement about vector.  
a) vector<int> values (5)  
b) vector values (5)  
c) vector<int> (5)  
d) vector<5>

Answer: a  
Explanation: The syntax for declaring the vector element is vector<type> variable\_name (number\_of\_elements);

10. Which is optional in the declaration of vector?  
a) Type  
b) Name  
c) Vector  
d) Number\_of\_elements

Answer: d  
Explanation: The number of elements is optional. An empty vector means, A vector that contains zero elements.

# C++ Programming MCQ – Sequences

1. How many items are there in sequence container?  
a) 2  
b) 3  
c) 4  
d) 5

Answer: d  
Explanation: There are five items in sequence container. They are array, vector, list, forward\_list and dequeue.

2. Which of the following class template are based on arrays?  
a) vector  
b) list  
c) dequeue  
d) both vector & dequeue

Answer: d  
Explanation: Class template vector and class template dequeue both are based on arrays.

3. Which of the following will return the new element at the end of container?  
a) front  
b) back  
c) push\_back  
d) pop\_back

Answer: b  
Explanation: Q3: back() in containers are used to access the last element of the sequence.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <deque>
3. using namespace std;
4. int main ()
5. {
6. deque<int> mydeque (5);
7. deque<int>::reverse\_iterator rit = mydeque.rbegin();
8. int i = 0;
9. for (rit = mydeque.rbegin(); rit!= mydeque.rend(); ++rit)
10. \*rit = ++i;
11. for (deque<int> :: iterator it = mydeque.begin();
12. it != mydeque.end(); ++it)
13. cout << ' ' << \*it;
14. return 0;
15. }

a) 12345  
b) 1234  
c) 54321  
d) 43210

Answer: c  
Explanation: In this program, We used the operation of rbegin and rend on dequeue and produced the result.  
Output:

$ g++ seq.cpp

$ a.out

5 4 3 2 1

5. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <deque>
3. using namespace std;
4. int main ()
5. {
6. unsigned int i;
7. deque<int> a (3,100);
8. deque<int> b (5,200);
9. a.swap(b);
10. cout << "a contains:";
11. for (deque<int>::iterator it = a.begin(); it != a.end(); ++it)
12. cout << ' ' << \*it;
13. cout << "b contains:";
14. for (deque<int>::iterator it = b.begin(); it != b.end(); ++it)
15. cout << ' ' << \*it;
16. return 0;
17. }

a) a contains: 200 200 200 200 200b contains: 100 100 100  
b) a contains: 100 100 100 100 100b contains: 200 200 200  
c) a contains: 200 200 200 200 200b contains: 200 200 200  
d) a contains: 200 200 200 200 200b contains: 100 200 150

Answer: a  
Explanation: In this program, We swapped the values of both dequeues and printing the dequeues.  
Output:

$ g++ seq1.cpp

$ a.out

a contains: 200 200 200 200 200b contains: 100 100 100

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <deque>
3. using namespace std;
4. int main ()
5. {
6. unsigned int i;
7. deque<int> mydeque;
8. mydeque.push\_back (100);
9. mydeque.push\_back (200);
10. mydeque.push\_back (300);
11. for(deque<int> :: iterator it = mydeque.begin(); it != mydeque.end(); ++it)
12. {
13. }
14. mydeque.clear();
15. mydeque.push\_back (110);
16. mydeque.push\_back (220);
17. for(deque<int> :: iterator it = mydeque.begin(); it != mydeque.end(); ++it)
18. cout << ' ' << \*it;
19. cout << '**\n**';
20. return 0;
21. }

a) 110  
b) 220  
c) Both 110 & 220  
d) 330

Answer: c  
Explanation: In this program, We cleared the old values presented in the dequeue with the new values.  
Output:

$ g++ seq2.cpp

$ a.out

110 220

7. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <vector>
3. using namespace std;
4. int main ()
5. {
6. vector<int> myvector;
7. int \* p;
8. unsigned int i;
9. p = myvector.get\_allocator().allocate(5);
10. for (i = 0; i < 5; i++)
11. myvector.get\_allocator().construct(&p[i], i);
12. for (i = 0; i < 5; i++)
13. cout << ' ' << p[i];
14. for (i = 0; i < 5; i++)
15. myvector.get\_allocator().destroy(&p[i]);
16. myvector.get\_allocator().deallocate(p, 5);
17. return 0;
18. }

a) 1 2 3 4 5  
b) 0 1 2 3 4  
c) 1 2 3 4  
d) 5 4 3 2 1

Answer: b  
Explanation: In this program, We allocated the values to the vector by using get allocater and then we are destroying it.  
Output:

$ g++ seq3.cpp

$ a.out

0 1 2 3 4

8. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <cmath>
3. #include <list>
4. using namespace std;
5. bool same\_integral\_part (double first, double second)
6. {
7. return ( int(first) == int(second) );
8. }
9. struct is\_near
10. {
11. bool operator() (double first, double second)
12. {
13. return (fabs(first - second) < 5.0);
14. }
15. };
16. int main ()
17. {
18. double mydoubles[] = { 12.15, 2.72, 73.0, 12.77, 3.14, 12.77, 73.35, 72.25, 15.3, 72.25 };
19. list<double> mylist (mydoubles, mydoubles + 10);
20. mylist.sort();
21. mylist.unique();
22. mylist.unique (same\_integral\_part);
23. mylist.unique (is\_near());
24. for (list<double> :: iterator it = mylist.begin(); it != mylist.end(); ++it)
25. cout << ' ' << \*it;
26. cout << '**\n**';
27. return 0;
28. }

a) 2.72 12.15 72.25  
b) 12.15 73.0 12.77  
c) 73.35  
d) 74.45

Answer: a  
Explanation: In this program, We are eliminating the values by using the unique operation in the list.  
Output:

$ g++ seq4.cpp

$ a.out

2.72 12.15 72.25

9. How the list containers are implemented?  
a) Using Double linked list  
b) Using Single linked list  
c) Using Single & Double linked list  
d) Using linear linked list

Answer: a  
Explanation: List containers are implemented as doubly-linked lists. Doubly linked lists can store each of the elements they contain in different and unrelated storage locations.

10. Which of the following does not support any insertion or deletion?  
a) Array  
b) Vector  
c) Dequeue  
d) List

Answer: a  
Explanation: Because array is not dynamic in nature, So they can’t be manipulated.