**C++ Programming Multiple Choice Questions on Basic Concepts**

**This set of C++ Programming MCQs focuses on “Basics”**

1. Which of the following is the correct syntax of including a user defined header files in C++?  
a) #include <userdefined.h>  
b) #include <userdefined>  
c) #include “userdefined”  
d) #include [userdefined]  
View Answer

Answer: c  
Explanation: C++ uses double quotes to include a user-defined header file. The correct syntax of including user-defined is #include “userdefinedname”.

2. Which of the following is a correct identifier in C++?  
a) 7var\_name  
b) 7VARNAME  
c) VAR\_1234  
d) $var\_name  
View Answer

Answer: c  
Explanation: The rules for writing an identifier is as follows:  
i) may contain lowercase/uppercase letters, digits or underscore(\_) only  
ii) should start with a non-digit character  
iii) should not contain any special characters like @, $, etc.

3. Which of the following is called address operator?  
a) \*  
b) &  
c) \_  
d) %  
View Answer

Answer: b  
Explanation: & operator is called address operator and is used to access the address of a variable.

4. Which of the following is used for comments in C++?  
a) // comment  
b) /\* comment \*/  
c) both // comment or /\* comment \*/  
d) // comment \*/  
View Answer

Answer: c  
Explanation: Both the ways are used for commenting in C++ programming. // is used for single line comments and /\* … \*/ is used for multiple line comments.

5. What are the actual parameters in C++?  
a) Parameters with which functions are called  
b) Parameters which are used in the definition of a function  
c) Variables other than passed parameters in a function  
d) Variables that are never used in the function  
View Answer

Answer: a  
Explanation: Actual parameters are those using which a function call is made i.e. which are actually passed in a function when that function is called.

6. What are the formal parameters in C++?  
a) Parameters with which functions are called  
b) Parameters which are used in the definition of the function  
c) Variables other than passed parameters in a function  
d) Variables that are never used in the function  
View Answer

Answer: b  
Explanation: Formal parameters are those which are used in the definition of a function. They are the parameters that represent the actual parameters passed and they are the one which is used inside the function.

7. Which function is used to read a single character from the console in C++?  
a) cin.get(ch)  
b) getline(ch)  
c) read(ch)  
d) scanf(ch)  
View Answer

Answer: a  
Explanation: C++ provides cin.get() function to read a single character from console whereas others are used to read either a single or multiple characters.

8. Which function is used to write a single character to console in C++?  
a) cout.put(ch)  
b) cout.putline(ch)  
c) write(ch)  
d) printf(ch)  
View Answer

Answer: a  
Explanation: C++ provides cout.put() function to write a single character to console whereas others are used to write either a single or multiple characters.

9. What are the escape sequences?  
a) Set of characters that convey special meaning in a program  
b) Set of characters that whose use are avoided in C++ programs  
c) Set of characters that are used in the name of the main function of the program  
d) Set of characters that are avoided in cout statements  
View Answer

Answer: a  
Explanation: Escape sequence is a set of characters that convey a special meaning to the program. They are used to convey a meaning which cannot be conveyed directly.

10. Which of the following escape sequence represents carriage return?  
a) \r  
b) \n  
c) \n\r  
d) \c  
View Answer

Answer: a  
Explanation: \r is used to represent carriage return which means move the cursor to the beginning of the next line.

11. Which of the following escape sequence represents tab?  
a) \t  
b) \t\r  
c) \b  
d) \a  
View Answer

Answer: a  
Explanation: \t is used to represent tab which means a set of blank spaces in a line.

12. Who created C++?  
a) Bjarne Stroustrup  
b) Dennis Ritchie  
c) Ken Thompson  
d) Brian Kernighan  
View Answer

Answer: a  
Explanation: Bjarne Stroustrup is the original creator of C++ during 1979 at AT&T Bell Labs.

13. Which of the following is called insertion/put to operator?  
a) <<  
b) >>  
c) >  
d) <  
View Answer

Answer: a  
Explanation: << operator is called insertion or put to operator i.e. insert/put things to console/files.

14. Which of the following is called extraction/get from operator?  
a) <<  
b) >>  
c) >  
d) <  
View Answer

Answer: b  
Explanation: >> operator is called extraction or get from operator i.e. extract/get things from console/files.

15. A language which has the capability to generate new data types are called \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) Extensible  
b) Overloaded  
c) Encapsulated  
d) Reprehensible  
View Answer

Answer: a  
Explanation: Languages that can produce/generate new data types are called extensible languages as they have the ability to handle new data types.

**This set of C++ Programming Objective Questions & Answers focuses on “OOPs – 1”**

1. Wrapping data and its related functionality into a single entity is known as \_\_\_\_\_\_\_\_\_\_\_\_\_  
a) Abstraction  
b) Encapsulation  
c) Polymorphism  
d) Modularity  
View Answer

Answer: b  
Explanation: In OOPs, the property of enclosing data and its related functions into a single entity(in C++ we call them classes) is called encapsulation.

2. How structures and classes in C++ differ?  
a) In Structures, members are public by default whereas, in Classes, they are private by default  
b) In Structures, members are private by default whereas, in Classes, they are public by default  
c) Structures by default hide every member whereas classes do not  
d) Structures cannot have private members whereas classes can have  
View Answer

Answer: a  
Explanation: Structure members are public by default whereas, class members are private by default. Both of them can have private and public members.

3. What does polymorphism in OOPs mean?  
a) Concept of allowing overiding of functions  
b) Concept of hiding data  
c) Concept of keeping things in differnt modules/files  
d) Concept of wrapping things into a single unit  
View Answer

Answer: a  
Explanation: In OOPs, Polymorphism is the concept of allowing a user to override functions either by changing the types or number of parameters passed.

4. Which concept allows you to reuse the written code?  
a) Encapsulation  
b) Abstraction  
c) Inheritance  
d) Polymorphism  
View Answer

Answer: c  
Explanation: Inheritance allows you to reuse your already written code by inheriting the properties of written code into other parts of the code, hence allowing you to reuse the already written code.

5. Which of the following explains Polymorphism?  
a)

int func(int, int);

float func1(float, float);

b)

int func(int);

int func(int);

c)

int func(float);

float func(int, int, char);

d)

int func();

int new\_func();

View Answer

Answer: c  
Explanation: Polymorphism means overriding the same function by changing types or number of arguments. So we have only two options which has the same function names, but as one can observe that in one option types, name and number of parameters all are same which will lead to an error. Hence that is wrong so the option having same name and different types or number of parameters is correct.

6. Which of the following shows multiple inheritances?  
a) A->B->C  
b) A->B; A->C  
c) A,B->C  
d) B->A  
View Answer

Answer: c  
Explanation: In multiple inheritance, a single class is inherited from two classes. So in A,B->C, Class C is inherited from A and B, whereas in A->B->C, C from B and B from A called simple inheritance, in A->B; A->C, B and C are inherited from A which is called hierarchical inheritance.

7. How access specifiers in Class helps in Abstraction?  
a) They does not helps in any way  
b) They allows us to show only required things to outer world  
c) They help in keeping things together  
d) Abstraction concept is not used in classes  
View Answer

Answer: b  
Explanation: Abstraction is the concept of hiding things from the outer world and showing only the required things to the world, which is where access specifiers private, protected and public helps in keeping our knowledge hidden from the world.

8. C++ is \_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) procedural programming language  
b) object oriented programming language  
c) functional programming language  
d) both procedural and object oriented programming language  
View Answer

Answer: d  
Explanation: C++ supports both procedural(step by step instruction) and object oriented programming(using concept of classes and objects).

9. What does modularity mean?  
a) Hiding part of program  
b) Subdividing program into small independent parts  
c) Overriding parts of program  
d) Wrapping things into single unit  
View Answer

Answer: b  
Explanation: Modularity means dividing a program into independent sub programs so that it can be invoked from other parts of the same program or any other program.

10. Which of the following feature of OOPs is not used in the following C++ code?

class A

{

int i;

public:

void print(){cout<<"hello"<<i;}

}

class B: public A

{

int j;

public:

void assign(int a){j = a;}

}

a) Abstraction  
b) Encapsulation  
c) Inheritance  
d) Polymorphism  
View Answer

Answer: d  
Explanation: As i and j members are private i.e. they are hidden from outer world therefore we have used the concept of abstraction. Next data members and there related functions are put together into single class therefore encapsulation is used. Also as class B is derived from A therefore Inheritance concept is used. But as no function is overloaded in any of the classes therefore, the concept of polymorphism is missing here.

**This set of C++ Programming Multiple Choice Questions & Answers (MCQs) focuses on “OOPs – 2”**

1. Which of the following class allows to declare only one object of it?  
a) Abstract class  
b) Virtual class  
c) Singleton class  
d) Friend class  
View Answer

Answer: c  
Explanation: Singleton class allows the programmer to declare only one object of it, If one tries to declare more than one object the program results into error.

2. Which of the following is not a type of Constructor?  
a) Friend constructor  
b) Copy constructor  
c) Default constructor  
d) Parameterized constructor  
View Answer

Answer: a  
Explanation: Friend function is not a constructor whereas others are a type of constructor used for object initialization.

3. Which of the following is correct?  
a) Base class pointer object cannot point to a derived class object  
b) Derived class pointer object cannot point to a base class object  
c) A derived class cannot have pointer objects  
d) A base class cannot have pointer objects  
View Answer

Answer: b  
Explanation: C++ does not allow a derived class pointer to point a base class pointer whereas Base class can point to a derived class object. Both base class and derived class can have pointer objects.

4. Out of the following, which is not a member of the class?  
a) Static function  
b) Friend function  
c) Constant function  
d) Virtual function  
View Answer

Answer: b  
Explanation: Friend function is not a member of the class. They are given the same access rights as the class member function have but they are not actual members of the class.

5. What is the other name used for functions inside a class?  
a) Member variables  
b) Member functions  
c) Class functions  
d) Class variables  
View Answer

Answer: b  
Explanation: Functions of a class are also known as member functions of a class.

6. Which of the following cannot be a friend?  
a) Function  
b) Class  
c) Object  
d) Operator function  
View Answer

Answer: c  
Explanation: Objects of any class cannot be made a friend of any other or same class whereas functions, classes and operator functions can be made a friend.

7. Why references are different from pointers?  
a) A reference cannot be made null  
b) A reference cannot be changed once initialized  
c) No extra operator is needed for dereferencing of a reference  
d) All of the mentioned  
View Answer

Answer: d  
Explanation: References cannot be made null whereas a pointer can be. References cannot be changed whereas pointers can be modified.  
Pointers need \* operator to dereference the value present inside it whereas reference does not need an operator for dereferencing.

8. Which of the following provides a programmer with the facility of using object of a class inside other classes?  
a) Inheritance  
b) Composition  
c) Abstraction  
d) Encapsulation  
View Answer

Answer: b  
Explanation: The concept of using objects of one class into another class is known as Composition.

9. How many types of polymorphism are there in C++?  
a) 1  
b) 2  
c) 3  
d) 4  
View Answer

Answer: b  
Explanation: There are two types of polymorphism in C++ namely run-time and compile-time polymorphisms.

10. How run-time polymorphisms are implemented in C++?  
a) Using Inheritance  
b) Using Virtual functions  
c) Using Templates  
d) Using Inheritance and Virtual functions  
View Answer

Answer: d  
Explanation: Run-time polymorphism is implemented using Inheritance and virtual in which object decides which function to call.

11. How compile-time polymorphisms are implemented in C++?  
a) Using Inheritance  
b) Using Virtual functions  
c) Using Templates  
d) Using Inheritance and Virtual functions  
View Answer

Answer: c  
Explanation: Compile-time polymorphism is implemented using templates in which the types(which can be checked during compile-time) are used decides which function to be called.

12. Which of the following is an abstract data type?  
a) int  
b) float  
c) class  
d) string  
View Answer

Answer: c  
Explanation: Class is used as an abstract data type as it can be used to give implementation independent view whereas no other data type can be used to provide this.

13. Which concept means the addition of new components to a program as it runs?  
a) Data hiding  
b) Dynamic binding  
c) Dynamic loading  
d) Dynamic typing  
View Answer

Answer: c  
Explanation: Dynamic loading is the concept of adding new components to a program as it runs.

14. Which of the following explains the overloading of functions?  
a) Virtual polymorphism  
b) Transient polymorphism  
c) Ad-hoc polymorphism  
d) Pseudo polymorphism  
View Answer

Answer: c  
Explanation: Ad-hoc polymorphism is a type of polymorphism in which a function denotes heterogeneous implementation depending upon the types of argument.

15. Which of the following approach is used by C++?  
a) Top-down  
b) Bottom-up  
c) Left-right  
d) Right-left  
View Answer

Answer: b  
Explanation: C++ is an object-oriented language and OOL uses a bottom-up approach to solve/view a problem.

**This set of C++ Programming Objective Questions & Answers focuses on “OOPs – 3”**

1. Which operator is overloaded for a cout object?  
a) >>  
b) <<  
c) <  
d) >  
View Answer

Answer: b  
Explanation: cout in C++ uses << operator to print anything so << operator is overloaded for a cout object.

2. Which of the following cannot be used with the virtual keyword?  
a) Class  
b) Member functions  
c) Constructors  
d) Destructors  
View Answer

Answer: c  
Explanation: Virtual keyword cannot be used with constructors as constructors are defined to initialized an object of particular class hence no other class needs constructor of other class.

3. Which concept is used to implement late binding?  
a) Virtual functions  
b) Operator functions  
c) Constant functions  
d) Static functions  
View Answer

Answer: a  
Explanation: Virtual functions are used to implement the concept of late binding i.e. binding actual functions to their calls.

4. Which of the following is correct?  
a) C++ allows static type checking  
b) C++ allows dynamic type checking.  
c) C++ allows static member function to be of type const.  
d) C++ allows both static and dynamic type checking  
View Answer

Answer: d  
Explanation: C++ allows both static and dynamic type checking i.e. types are checked by the compiler.

5. Which of the following supports the concept that reusability is a desirable feature of a language?  
a) It reduces the testing time  
b) It reduces maintenance cost  
c) It decreases the compilation time  
d) It reduced both testing and maintenance time  
View Answer

Answer: d  
Explanation: As we will be using the existing code therefore we don’t need to check the code again and again so testing and maintenance time decreases but the compiler time may increase or remains same because though we are reusing the code but every part needs to be compiled and extra include statement needs to be executed therefore compilation time may remain same or increases.

6. Which of the following is a static polymorphism mechanism?  
a) Function overloading  
b) Operator overloading  
c) Templates  
d) All of the mentioned  
View Answer

Answer: d  
Explanation: All the options mentioned above uses static polymorphism mechanism. As the conflicts in all these types of functions are resolved during compile-time.

7. Which of the following is true?  
I) All operators in C++ can be overloaded.  
II) The basic meaning of an operator can be changed.  
a) I only  
b) II only  
c) Both I and II  
d) Neither I nor II  
View Answer

Answer: d  
Explanation: Both statements are false because all the operators of C++ cannot be overloaded and the basic meaning of an operator cannot be changed, we can only give new meaning to an operator

8. Which of the following is not a type of inheritance?  
a) Multiple  
b) Multilevel  
c) Distributive  
d) Hierarchical  
View Answer

Answer: c  
Explanation: Distributive is not a type of inheritance whereas others are a type of inheritance having their own meaning.

9. What happens if a class does not have a name?  
a) It will not have a constructor  
b) It will not have a destructor  
c) It is not allowed  
d) It will neither have a constructor or destructor  
View Answer

Answer: b  
Explanation: A class without a name will not have a destructor. The object is made so constructor is required but the destructor is not. Check the code below:

#include <iostream>

using namespace std;

class

{

public:

void func()

{

cout<<"Hello world";

}

}a;

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

{

a.func();

return 0;

}

10. Which of the following statement is true?  
I) In Procedural programming languages, all function calls are resolved at compile-time  
II) In Object Oriented programming languages, all function calls are resolved at compile-time  
a) I only  
b) II only  
c) Both I and II  
d) Neither I nor II  
View Answer

Answer: a  
Explanation: In Procedural programming like C we don’t have the concept of polymorphism, therefore, all the function calls are resolved at the compile-time but in case of OOP languages sue to polymorphism concept all function calls are not resolved at compile-time.

11. Which members are inherited but are not accessible in any case?  
a) Private  
b) Public  
c) Protected  
d) Both private and protected  
View Answer

Answer: a  
Explanation: Private members of a class are inherited to the child class but are not accessible from the child class.

12. Which of the following is correct?  
a) Friend functions can access public members of a class  
b) Friend functions can access protected members of a class  
c) Friend functions can access private members of a class  
d) All of the mentioned  
View Answer

Answer: d  
Explanation: Friend functions can access any member of a class without caring about the type of member i.e. without caring whether it is private, protected or public.

13. Which of the following is correct in C++?  
a) Classes cannot have protected data members  
b) Structures can have member functions  
c) Class members are public by default  
d) Structure members are private by default  
View Answer

Answer: b  
Explanation: Though C does not allows member functions in structures but C++ allows structures to have member functions. Members of structures are public by default and those of classes are private by default. Classes can have protected data members.

14. Which of the following is used to make an abstract class?  
a) By using virtual keyword in front of a class declaration  
b) By using an abstract keyword in front of a class declaration  
c) By declaring a virtual function in a class  
d) By declaring a pure virtual function in a class  
View Answer

Answer: d  
Explanation: Abstract class should have at least one pure virtual function. Therefore to declare an abstract class one should declare a pure virtual function in a class.

15. Which of the following is correct?  
a) A class is an instance of its objects  
b) An object is an instance of its class  
c) A class is an instance of the data type that the class have  
d) An object is an instance of the data type of the class  
View Answer

Answer: b  
Explanation: An object is an instance of a class i.e. an object represents a class i.e. what class has(data members) and what it can do(member functions).

**This set of C++ Programming Multiple Choice Questions & Answers (MCQs) focuses on “OOPs – 4”**

1. Which of the following is correct about new and malloc?  
a) Both are available in C  
b) Pointer object initialization of a class with both new and malloc calls the constructor of that class  
c) Pointer object initialization of a class using new involves constructor call whereas using malloc does not involve constructor call  
d) Pointer object initialization of a class using malloc involves constructor call whereas using new does not involve constructor call  
View Answer

Answer: c  
Explanation: Object initialization using new keyword involves constructor call whereas malloc does not involve constructor call. That’s why new is explicitly added in C++. Also, malloc is used to assign memory to any pointer hence it assigns memory equals to the size of the class however new keyword involves initialization also hence calls the constructor of that class.

2. What is virtual inheritance?  
a) C++ technique to avoid multiple copies of the base class into children/derived class  
b) C++ technique to avoid multiple inheritances of classes  
c) C++ technique to enhance multiple inheritance  
d) C++ technique to ensure that a private member of the base class can be accessed somehow  
View Answer

Answer: a

Explanation: Virtual inheritance is a C++ technique with which it ensures that a derived class contains only one copy of the base class’s variables. Refer Wikipedia for more info.

3. What is the difference between delete and delete[] in C++?  
a) delete is used to delete normal objects whereas delete[] is used to pointer objects  
b) delete is a keyword whereas delete[] is an identifier  
c) delete is used to delete single object whereas delete[] is used to multiple(array/pointer of) objects  
d) delete is syntactically correct but delete[] is wrong and hence will give an error if used in any case  
View Answer

Answer: c  
Explanation: delete is used to delete a single object initiated using new keyword whereas delete[] is used to delete a group of objects initiated with the new operator.

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

#include <iostream>

using namespace std;

class A{

public:

A(){

cout<<"Constructor called**\n**";

}

~A(){

cout<<"Destructor called**\n**";

}

};

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

{

A \*a = new A[5];

delete a;

return 0;

}

a) “Constructor called” five times and then “Destructor called” five times  
b) “Constructor called” five times and then “Destructor called” once  
c) Error  
d) Segmentation fault  
View Answer

Answer: d  
Explanation: The program will result in segmentation fault as we are trying to delete only one pointer variable and leaving other variables as it is which will result in segmentation fault i.e. improper handling of memory.

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

#include <iostream>

using namespace std;

class A{

public:

A(){

cout<<"Constructor called**\n**";

}

~A(){

cout<<"Destructor called**\n**";

}

};

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

{

A \*a = new A[5];

delete[] a;

return 0;

}

a) “Constructor called” five times and then “Destructor called” five times  
b) “Constructor called” five times and then “Destructor called” once  
c) Error  
d) Segmentation fault

View Answer

Answer: a  
Explanation: In the above program we have first initiated five-pointer variables using new keyword hence fives time constructor will be called after that as we using delete[](used for deleting multiple objects) to delete variables hence all the five objects created will be destroyed and hence five times destructor will be called.

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

#include<iostream>

using namespace std;

class Base {

public:

Base()

{ cout<<"Constructing Base **\n**"; }

~Base()

{ cout<<"Destructing Base **\n**"; }

};

class Derived: public Base {

public:

Derived()

{ cout<<"Constructing Derived **\n**"; }

~Derived()

{ cout<<"Destructing Derived **\n**"; }

};

int main(void)

{

Derived \*d = new Derived();

Base \*b = d;

delete b;

return 0;

}

a)

Constructing Base

Constructing Derived

Destructing Base

b)

Constructing Base

Constructing Derived

Destructing Derived

Destructing Base

c)

Constructing Base

Constructing Derived

Destructing Base

Destructing Derived

d)

Constructing Derived

Constructing Base

Destructing Base

Destructing Derived

View Answer

Answer: a  
Explanation: As we are storing a derived class object into base class pointer therefore when the object is destroyed the program has not called the Derived class destructor which shows that the object is not destroyed therefore the program may give unusual behaviour.

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

#include<iostream>

using namespace std;

class Base {

public:

Base()

{ cout<<"Constructing Base **\n**"; }

virtual~Base()

{ cout<<"Destructing Base **\n**"; }

};

class Derived: public Base {

public:

Derived()

{ cout<<"Constructing Derived **\n**"; }

~Derived()

{ cout<<"Destructing Derived **\n**"; }

};

int main(void)

{

Derived \*d = new Derived();

Base \*b = d;

delete b;

return 0;

}

a)

Constructing Base

Constructing Derived

Destructing Base

b)

Constructing Base

Constructing Derived

Destructing Derived

Destructing Base

c)

Constructing Base

Constructing Derived

Destructing Base

Destructing Derived

d)

Constructing Derived

Constructing Base

Destructing Base

Destructing Derived

View Answer

Answer: b  
Explanation: In this case, we have made the destructor of base class virtual which will ensure that any derived class object which is pointed by a base class pointer object on deletion should call both base and derived class destructor.

8. What is the correct syntax of declaring array of pointers of integers of size 10 in C++?  
a) int arr = new int[10];  
b) int \*\*arr = new int\*[10];  
c) int \*arr = new int[10];  
d) int \*arr = new int\*[10];  
View Answer

Answer: b  
Explanation: As we have to declare an array of pointers of integers therefore we need double pointer array in which each element is collection pointers to integers. Therefore the correct syntax is int \*\*arr = new int\*[10];

9. Which of the following is correct about new and malloc?  
i) new is an operator whereas malloc is a function  
ii) new calls constructor malloc does not  
iii) new returns required pointer whereas malloc returns void pointer and needs to be typecast  
a) i and ii  
b) ii and iii  
c) i and iii  
d) i, ii and iii  
View Answer

Answer: d  
Explanation: All the statements about the new and malloc are correct. new is an operator whereas malloc() is a function. The constructor is called when new is used and new returns required type memory pointer.

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

#include <iostream>

using namespace std;

class A

{

int a;

A() { a = 5;}

};

int main()

{

A \*obj = new A;

cout << obj->a;

}

a) 5  
b) Garbage value  
c) Compile-time error  
d) Run-time error  
View Answer

Answer: c  
Explanation: As Test() constructor is private member of the class therefore cannot be accessed from the outside world therefore the program gives error.

11. What happens if the following C++ statement is compiled and executed?

int \*ptr = NULL;

delete ptr;

a) The program compiled successfully but throws an error during run-time  
b) The program gives a compile-time error  
c) The program is not semantically correct  
d) The program is compiled and executed successfully  
View Answer

Answer: d  
Explanation: The above statement is syntactically and semantically correct as C++ allows the programmer to delete a NULL pointer, therefore, the program is compiled and executed successfully.

12. What happens if a pointer is deleted twice in a program as shown in the following C++ statements?

int \*ptr = new int;

delete ptr;

delete ptr;

a) Undefined behaviour  
b) Syntactically incorrect  
c) Semantically incorrect  
d) The program runs perfectly  
View Answer

Answer: a  
Explanation: Deleting a pointer twice in a program may lead to run-time error or may run perfectly. It depends on the compiler how it handles the situation so the program may compile and run successfully but actually the program should give a run-time error(segmentation fault) as you are trying to access the unauthorized memory of the system.

**This set of C++ Programming Multiple Choice Questions & Answers (MCQs) focuses on “C++ vs C”**

1. What happens if the following program is executed in C and C++?

#include<stdio.h>

int main()

{

foo();

}

int foo()

{

printf("Hello");

return 0;

}

a) Error in both C and C++  
b) Warning in both C and C++  
c) Error in C++ but Warning in C  
d) Error in C but Warning in C++  
View Answer

Answer: c  
Explanation: In C++ all the functions should be declared before it is called otherwise the C++ compiler will give an error but in case of C the compiler just gives a warning and the program can be executed.

2. What happens if the following program is executed in C and C++?

#include <stdio.h>

int main(void)

{

const int j = 20;

int \*ptr = &j;

printf("\*ptr: %d**\n**", \*ptr);

return 0;

}

a) Error in both C and C++  
b) Warning in both C and C++  
c) Error in C but Warning in C++  
d) Error in C++ but Warning in C  
View Answer

Answer: d  
Explanation: C++ is strict on the use of types of variables hence when the programmer tries to assign const int to a normal pointer the program gives error whereas C is not strict on types therefore it gives warning only.

3. What happens if the following line is executed in C and C++?

int \*p = malloc(10);

a) Error in both C and C++  
b) Warning in both C and C++  
c) Error in C++ and successful execution in C  
d) Error in C and successful execution in C++  
View Answer

Answer: c  
Explanation: C++ is strict in type check but C is not and as malloc returns a void\* which we are trying to assign to an int\*, therefore, the C++ compiler gives error whereas C compiler executes the program successfully.

4. What happens if the following line is executed in C and C++?

const int a;

a) Error in both C and C++  
b) Warning in both C and C++  
c) Error in C and successful execution in C++  
d) Error in C++ and successful execution in C  
View Answer

Answer: d  
Explanation: C++ compiler does not allow the programmer to declare a constant variable without initializing it hence the C++ compiler gives an error whereas C allows such declaration, therefore, the program compiles and runs successfully.

5. What happens if the following program is executed in C and C++?

#include <stdio.h>

int main(void)

{

int new = 5;

printf("%d", new);

}

a) Error in both C and C++  
b) A successful run in both C and C++  
c) Error in C and successful execution in C++  
d) Error in C++ and successful execution in C  
View Answer

Answer: d  
Explanation: new is a keyword in C++, therefore, we cannot declare a variable with name new but as there is no such keyword new in C, therefore, the program is compiled and executed successfully in C.

6. What happens if the following program is executed in C and C++?

#include <stdio.h>

void main()

{

printf("Hello World");

}

a) Error in both C and C++  
b) Successful run in both C and C++  
c) Error in C and successful execution in C++  
d) Error in C++ and successful execution in C  
View Answer

Answer: d  
Explanation: main() function in C++ must return int otherwise the C++ compiler gives the error whereas C does not forces such things on main() function. Thereas when we aremaking void main(){} function in this program the C++ compiler gives error whereas C compiler runs successfully.

7. What happens if the following program is executed in C and C++?

#include <stdio.h>

void func(void)

{

printf("Hello");

}

void main()

{

func();

func(2);

}

a) Error in both C and C++  
b) Outputs Hello twice in both C and C++  
c) Error in C and successful execution in C++  
d) Error in C++ and successful execution in C  
View Answer

Answer: a  
Explanation: As the func(void) needs no argument during its call, hence when we are calling func(2) with 2 as passed as a parameter then this statement gives the error in both C++ and C compiler.

8. What happens if the following program is executed in C and C++?

#include <stdio.h>

void func()

{

printf("Hello");

}

void main()

{

func();

func(2);

}

a) Error in both C and C++  
b) Outputs Hello twice in both C and C++  
c) Error in C and Outputs Hello twice in C++  
d) Error in C++ and Outputs Hello twice in C  
View Answer

Answer: d  
Explanation: In C++ whenever a function without argument is declared it is equivalent to function with void arguments i.e. func() == func(void) whereas in C a function without argument is equivalent to func(…) i.e. it can take any number of arguments so func(2) call is also valid in C but not valid in C++. Hence it gives error in C++ whereas no error in C.

9. Which of the following type is provided by C++ but not C?  
a) int  
b) bool  
c) float  
d) double  
View Answer

Answer: b  
Explanation: C++ provides the boolean type to handle true and false values whereas no such type is provided in C.

10. Which of the following feature is not provided by C?  
a) Pointers  
b) Structures  
c) References  
d) Functions  
View Answer

Answer: c  
Explanation: References are introduced in C++. They are not present in C.

**This set of C++ Programming Multiple Choice Questions & Answers (MCQs) focuses on “C++ Concepts – 1”**

1. Which of the following is not a fundamental type is not present in C but present in C++?  
a) int  
b) float  
c) bool  
d) void  
View Answer

Answer: c  
Explanation: Boolean type is not present as a fundamental type in C. int type is used as boolean in C whereas in C++ bool is defined as a fundamental type for handling boolean outputs.

2. What is the size of a boolean variable in C++?  
a) 1 bit  
b) 1 byte  
c) 4 bytes  
d) 2 bytes  
View Answer

Answer: b

Explanation: Even though boolean represents only the truth values which can be true(1) or false(0), it typically occupies 1 byte of memory due to various practical considerations.

3. Which of the following is C++ equivalent for scanf()?  
a) cin  
b) cout  
c) print  
d) input  
View Answer

Answer: a  
Explanation: C++ uses cin to read input form uses. However C++ also uses scanf().

4. Which of the following is C++ equivalent for printf()?  
a) cin  
b) cout  
c) print  
d) input  
View Answer

Answer: b  
Explanation: C++ uses cout to print output to console. However C++ also uses printf().

5. Which of the following is the correct difference between cin and scanf()?  
a) both are the same  
b) cin is a stream object whereas scanf() is a function  
c) scanf() is a stream object whereas cin is a function  
d) cin is used for printing whereas scanf() is used for reading input  
View Answer

Answer: b  
Explanation: cin is a stream object available in C++ whereas scanf() is a function available in both C and C++. both are used for reading input from users.

6. Which of the following is an exit-controlled loop?  
a) for  
b) while  
c) do-while  
d) all of the mentioned  
View Answer

Answer: c  
Explanation: do-while is called exit controlled loop because in do-while termination condition is checked when we have executed the body of the loop i.e. we are exiting the body and then checking the condition, therefore, it is called exit controlled loop.

7. Which of the following is an entry-controlled loop?  
a) for  
b) while  
c) do-while  
d) both while and for  
View Answer

Answer: d  
Explanation: Both while and for loops are called entry controlled loop because in both of them the termination condition is checked before we enter the body of the loop hence they are called entry controlled loop.

8. In which part of the for loop termination condition is checked?  
for(I;II;III)  
{IV}  
a) I  
b) II  
c) III  
d) IV  
View Answer

Answer: b  
Explanation: In II part the termination condition of the for loop is checked.

9. What is dynamic binding?  
a) The process of linking the actual code with a procedural call during run-time  
b) The process of linking the actual code with a procedural call during compile-time  
c) The process of linking the actual code with a procedural call at any-time  
d) All of the mentioned  
View Answer

Answer: a  
Explanation: Binding of calls and variables with actual code at run-time is called dynamic binding. For example in the concept of polymorphism types are decided are defined during the execution of code which leads to the different function calls depending upon the types used this is called dynamic binding. As the function call is decided during the run-time therefore dynamic binding happens at run-time.

10. What is static binding?  
a) The process of linking the actual code with a procedural call during run-time  
b) The process of linking the actual code with a procedural call during compile-time  
c) The process of linking the actual code with a procedural call at any-time  
d) All of the mentioned  
View Answer

Answer: b  
Explanation: Binding of calls and variables with actual code at compile-time is called static binding. For example normally whenever we declare a variable we define its type hence compiler knows what type should be binded to that variable i.e. compiler can decide about that variable this is called static binding.

11. What is name mangling in C++?  
a) The process of adding more information to a function name so that it can be distinguished from other functions by the compiler  
b) The process of making common names for all the function of C++ program for better use  
c) The process of changing the names of variable  
d) The process of declaring variables of different types  
View Answer

Answer: a  
Explanation: Name mangling is the process of adding some more information to a function name so that it can be distinguished from other functions by the compiler. This is used when a programmer uses the concept of function overloading in his/her program.

12. What will be the output of the following program in both C and C++?

#include<stdio.h>

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

{

printf("%d**\n**", (int)sizeof('a'));

return 0;

}

a) Output in C is 1 and in C++ is 4  
b) Output in C is 4 and in C++ is 1  
c) Output in C is 1 and in C++ is 1  
d) Output in C is 4 and in C++ is 4  
View Answer

Answer: b  
Explanation: In C a character is stored as int therefore the size of ‘a’ is printed as 4 whereas in C++ it is stored as char only therefore in C++ it prints 1.

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

#include<stdio.h>

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

{

char a = 'a';

printf("%d**\n**", (int)sizeof(a));

return 0;

}

a) Output in C is 1 and in C++ is 4  
b) Output in C is 4 and in C++ is 1  
c) Output in C is 1 and in C++ is 1  
d) Output in C is 4 and in C++ is 4  
View Answer

Answer: c  
Explanation: Both in C and C++ the type char has same size which is 1. But a character enclosed inside single quotes has difference sizes i.e. in case of char a; the size of a will be 1 in both C and C++ but in case of ‘a’ size will be 4 in case of C but 1 in case of C++.

14. Which of the following syntax for declaring a variable of struct STRUCT can be used in both C and C++?  
a) struct STRUCT S;  
b) STRUCT S;  
c) Both struct STRUCT S; and STRUCT S;  
d) Both C and C++ have different syntax  
View Answer

Answer: a  
Explanation: C program requires struct keyword while defining a variable of any structure, therefore, we cannot use the second STRUCT S; definition to declare a variable.

15. What if we define the below structure in C and C++?

struct mymem {

int x = 0;

int y;

};

a) Error in C but not in C++  
b) Error in C++ but not in C  
c) No error in both C and C++  
d) Error in both C and C++  
View Answer

Answer: a  
Explanation: The above definition will give an error in C but not in C++ as C does not allows the programmer to give any default values to any member of structure but C++ does allow.

**This set of C++ Programming Multiple Choice Questions & Answers (MCQs) focuses on “C++ Concepts – 2”**

1. Which of the following is the scope resolution operator?  
a) .  
b) \*  
c) ::  
d) ~  
View Answer

Answer: c  
Explanation: :: operator is called scope resolution operator used for accessing a global variable from a function which is having the same name as the variable declared in the function.

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

#include<iostream>

using namespace std;

int x = 1;

int main()

{

int x = 2;

{

int x = 3;

cout << ::x << endl;

}

return 0;

}

a) 1  
b) 2  
c) 3  
d) 123  
View Answer

Answer: a  
Explanation: While printing x we are using :: operator hence the refernce is given to global variable hence the global variable x = 1 is printed.

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

#include<iostream>

using namespace std;

class A

{

~A(){

cout<<"Destructor called**\n**";

}

};

int main()

{

A a;

return 0;

}

a) Destructor called  
b) Nothing will be printed  
c) Error  
d) Segmentation fault  
View Answer

Answer: c  
Explanation: Whenever a destructor is private then one should not define any normal object as it will be destroyed at the end of the program which will call destructor and as destructor is private the program gives error during compile while in case of pointer object the compiler at compile does not know about the object, therefore, does not gives compile error. Hence when the destructor is private then the programmer can declare pointer object but cannot declare a normal object.

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

#include<iostream>

using namespace std;

class A

{

~A(){

cout<<"Destructor called**\n**";

}

};

int main()

{

A \*a1 = new A();

A \*a2 = new A();

return 0;

}

a) Destructor called  
b)Destructor called

c) Error  
d) Nothing is printed  
View Answer

Answer: d  
Explanation: The pointer object is created is not deleted hence the destructor for these objects is not called hence nothing is printed on the screen.

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

#include<iostream>

using namespace std;

int x[100];

int main()

{

cout << x[99] << endl;

}

a) Garbage value  
b) 0  
c) 99  
d) Error  
View Answer

Answer: b  
Explanation: In C++, all the uninitialized global variables are set to 0. Therefore, the value of all elements of the array is set to 0.

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

#include<iostream>

using namespace std;

int main ()

{

int cin;

cin >> cin;

cout << "cin: " << cin;

return 0;

}

a) cin: garbage value  
b) Error  
c) Segmentation fault  
d) Nothing is printed  
View Answer

Answer: a  
Explanation: cin is a variable hence overrides the cin object. cin >> cin has no meaning so no error.

7. Which of the following operator has left to right associativity?  
a) Unary operator  
b) Logical not  
c) Array element access  
d) addressof  
View Answer

Answer: c  
Explanation: Array element has left to right associativity i.e. expressions are evaluated from left to right in case of array element access.

8. Which of the following is accessed by a member function of a class?  
a) The object of that class  
b) All members of a class  
c) The public part of a class  
d) The private part of a class  
View Answer

Answer: b  
Explanation: A member function of a class can access all the members of its class whether they are private, protected or public.

9. What is the size of a character literal in C and C++?

a) 4 and 1  
b) 1 and 4  
c) 1 and 1  
d) 4 and 4  
View Answer

Answer: a  
Explanation: The size of a character literal is 4 in case of C but it is one in case of C++. You can do printf(“%d”, (int)sizeof(‘a’)); in both C and C++ to check this.

10. What is the size of a character type in C and C++?  
a) 4 and 1  
b) 1 and 4  
c) 1 and 1  
d) 4 and 4  
View Answer

Answer: c  
Explanation: The size of a character type in both C and C++ is 1. You can do printf(“%d”, (int)sizeof(char)); in both C and C++ to check this.

11. Which of the following is correct?  
a) struct tag is required in both C and C++ while declaring an object of the structure  
b) struct is not required in C but required in C++ while declaring an object of the structure  
c) struct is not required in C++ but required in C while declaring an object of the structure  
d) struct tag is not required in both C and C++ while declaring an object of the structure  
View Answer

Answer: c  
Explanation: C++ does not require struct keyword while declaring an object of the structure whereas in C we require struct tag for declaring an object.

12. Which of the following is correct?  
a) struct cannot have member function in C but it can in C++  
b) struct cannot have member function in C++ but it can in C  
c) struct cannot have member function in both C and C++  
d) struct can have member function in both C and C++  
View Answer

Answer: a  
Explanation: struct can have member function in C++ whereas member functions are not allowed in case of C.

13. What happens if we run the following code in both C and C++?

#include<stdio.h>

struct STRUCT

{

int a;

int func()

{

printf("HELLO THIS IS STRUCTURE**\n**");

}

};

int main()

{

struct STRUCT s;

s.func();

return 0;

}

a) The program runs fine and both prints output “HELLO THIS IS STRUCTURE”  
b) The program gives an error in case of C but runs perfectly in case of C++  
c) The program gives an error in case of C++ but runs perfectly in case of C  
d) The program gives an error in case of both C and C++  
View Answer

Answer: b  
Explanation: As C does not allows the structure to have member functions, therefore, it gives an error in case of C but as C++ does allow structures to have member functions, therefore, the C++ does not give an error.

14. What happens if we run the following code in both C and C++?

#include<stdio.h>

struct STRUCT

{

int a = 5;

int func()

{

printf("%d**\n**", a);

}

};

int main()

{

struct STRUCT s;

s.func();

return 0;

}

a) The program runs fine and both prints output “HELLO THIS IS STRUCTURE”  
b) The program gives an error in case of C but runs perfectly in case of C++  
c) The program gives an error in case of C++ but runs perfectly in case of C  
d) The program gives an error in case of both C and C++

View Answer

Answer: b  
Explanation: As C does not allows to initialize any member inside the structure, therefore, the program gives error whereas in case of C++ this is allowed therefore the program does not give any error.

15. What happens if the following program is compiled in both C and C++?

#include<stdio.h>

struct STRUCT

{

int static a;

};

int main()

{

struct STRUCT s;

return 0;

}

a) The program runs fine and both prints output “HELLO THIS IS STRUCTURE”  
b) The program gives an error in case of C but runs perfectly in case of C++  
c) The program gives an error in case of C++ but runs perfectly in case of C  
d) The program gives an error in case of both C and C++  
View Answer

Answer: b

Explanation: C does not allow the programmer to declare any static members inside a class whether in C++ it is allowed to declare static variables.

**This set of C++ Programming Question Paper focuses on “C++ Concepts – 3”**

1. Which of the following statement is correct?  
a) Structure in C allows Constructor definition  
b) Structure in C++ allows Constructor definition  
c) Both allow Constructor definition  
d) C allows constructor definition while C++ does not  
View Answer

Answer: b  
Explanation: As C does not allow the programmer to define a function inside a structure and constructor itself is a function, therefore, the constructor definition is not allowed in C whereas such definitions are allowed in C++.

2. What happens if the following code is compiled on both C and C++?

#include<stdio.h>

struct STRUCT

{

private:

int a;

};

int main()

{

printf("%d**\n**", (int)sizeof(struct STRUCT));

return 0;

}

a) The program runs fine and both prints output “HELLO THIS IS STRUCTURE”  
b) The program gives an error in case of C but runs perfectly in case of C++  
c) The program gives an error in case of C++ but runs perfectly in case of C  
d) The program gives an error in case of both C and C++  
View Answer

Answer: b  
Explanation: Access specifiers like private, protected and the public are used because the OOPs concept and as C is not an Object Oriented language, therefore, access specifiers are not defined in C and hence C gives error whereas C++ does not.

3. Which of the following is correct about this pointer in C++?  
a) this pointer is passed as a hidden argument in all the functions of a class  
b) this pointer is passed as a hidden argument in all non-static functions of a class  
c) this pointer is passed as a hidden argument in all static functions of a class  
d) this pointer is passed as a hidden argument in all static variables of a class  
View Answer

Answer: b  
Explanation: As static functions are a type of global function for a class so all the object shares the common instance of that static function whereas all the objects have there own instance for non-static functions and hence they are passed as a hidden argument in all the non-static members but not in static members.

4. Which of the following operator is used with this pointer to access members of a class?  
a) .  
b) !  
c) ->  
d) ~  
View Answer

Answer: c  
Explanation: this pointer is a type of pointer and as we know pointer object uses the arrow(->) operator to access the members of the class, therefore, this pointer uses -> operator.

5. Why this pointer is used?  
a) To access the members of a class which have the same name as local variables in that scope  
b) To access all the data stored under that class  
c) To access objects of other class  
d) To access objects of other variables  
View Answer

Answer: a  
Explanation: this pointer is used to access the members of a class which have the same name as local variables in that part of the code.

6. How many types of polymorphism are there?  
a) 1  
b) 2  
c) 3  
d) 4  
View Answer

Answer: b  
Explanation: There are two types of polymorphism in C++ namely compile-time polymorphism and run-time polymorphism.

7. What is the other name of compile-time polymorphism?  
a) Static polymorphism  
b) Dynamic polymorphism  
c) Executing polymorphism  
d) Non-executing polymorphism

View Answer

Answer: a  
Explanation: Compile-time polymorphism is also known as static polymorphism as it is implemented during the compile-time.

8. What is the other name of run-time polymorphism?  
a) Static polymorphism  
b) Dynamic polymorphism  
c) Executing polymorphism  
d) Non-executing polymorphism  
View Answer

Answer: b  
Explanation: Run-time polymorphism is also known as dynamic polymorphism as it is implemented during the run-time of the program.

9. Which of the following is correct about static polymorphism?  
a) In static polymorphism, the conflict between the function call is resolved during the compile time  
b) In static polymorphism, the conflict between the function call is resolved during the run time  
c) In static polymorphism, the conflict between the function call is never resolved during the execution of a program  
d) In static polymorphism, the conflict between the function call is resolved only if it required  
View Answer

Answer: a  
Explanation: The conflict between which function to call is resolved during the compile time in static polymorphism i.e. before the execution of the program starts.

10. Which of the following is correct about dynamic polymorphism?  
a) In dynamic polymorphism, the conflict between the function call is resolved during the compile time  
b) In dynamic polymorphism, the conflict between the function call is resolved during the run time  
c) In dynamic polymorphism, the conflict between the function call is never resolved during the execution of the program  
d) In dynamic polymorphism, the conflict between the function call is resolved at the beginning of the program  
View Answer

Answer: b  
Explanation: The conflict between which function to call is resolved during the run time in dynamic polymorphism i.e. the conflict is resolved when the execution reaches the function call statement.

11. Which of the following operator(s) can be used with pointers?  
i) – only  
ii) +, \*  
iii) +, –  
iv) +, -, \*  
v) /  
vi) + only  
a) i only  
b) vi only  
c) ii and v  
d) iv  
View Answer

Answer: a  
Explanation: The only arithmetic operator that can be used with a pointer is – subtraction operator. No arithmetic operator can be used with pointers.

12. What is std in C++?  
a) std is a standard class in C++  
b) std is a standard namespace in C++  
c) std is a standard header file in C++  
d) std is a standard file reading header in C++  
View Answer

Answer: b  
Explanation: std is a standard namespace present in C++ which contains different stream classes and objects like cin, cout, etc. and other standard functions.

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

#include <iostream>

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

{

cout<<"Hello World";

return 0;

}

a) Hello World  
b) Compile-time error  
c) Run-time error  
d) Segmentation fault  
View Answer

Answer: b  
Explanation: cout is defined under the namespace std and without including std namespace we cannot cout, therefore, the program gives an error.

14. Which of the following syntax can be used to use a member of a namespace without including that namespace?  
a) namespace::member  
b) namespace->member  
c) namespace.member  
d) namespace~member  
View Answer

Answer: a  
Explanation: To use a member of a namespace without including the namespace is done by this syntax namespace::member.

15. Which of the following C++ code will give error on compilation?

================code 1=================

#include <iostream>

using namespace std;

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

{

cout<<"Hello World";

return 0;

}

========================================

================code 2=================

#include <iostream>

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

{

std::cout<<"Hello World";

return 0;

}

========================================

a) Both code 1 and code 2  
b) Code 1 only  
c) Code 2 only  
d) Neither code 1 nor code 2  
View Answer

Answer: d  
Explanation: Neither code 1 nor code2 will give error as both are syntactically correct as in first code we have included namespace std and in second one we have used scope resolution operator to resolve the conflict.

**This set of C++ Programming MCQs focuses on “Static Constant Keyword”**

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

#include <iostream>

using namespace std;

class Test

{

static int x;

public:

Test() { x++; }

static int getX() {return x;}

};

int Test::x = 0;

int main()

{

cout << Test::getX() << " ";

Test t[5];

cout << Test::getX();

}

a) 0 0  
b) 5 0  
c) 0 5  
d) 5 5  
View Answer

Answer: c  
Explanation: Static function can be called without using objects therefore the first call is fine. Next when we are creating 5 objects of the class then value of x is updated each time and as static variables are global to class therefore each updation of x is reflected back to each object hence value of x is 5.

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

#include <iostream>

using namespace std;

class Player

{

private:

int id;

static int next\_id;

public:

int getID() { return id; }

Player() { id = next\_id++; }

};

int Player::next\_id = 1;

int main()

{

Player p1;

Player p2;

Player p3;

cout << p1.getID() << " ";

cout << p2.getID() << " ";

cout << p3.getID();

return 0;

}

a) 1 2 3  
b) 2 2 2  
c) 1 3 1  
d) 1 1 1  
View Answer

Answer: a  
Explanation: In this as next\_id is a static variable so and initialized with 1 therefore the id value for 1st objects is 1 and next\_id is updated to 2. In this way next\_id is assigned to id in each object creation and updated by 1 so in this way value of each Id is updated.

3. Which of the following is correct about static variables?  
a) Static functions do not support polymorphism  
b) Static data members cannot be accessed by non-static member functions  
c) Static data members functions can access only static data members  
d) Static data members functions can access both static and non-static data members  
View Answer

Answer: c  
Explanation: Static member functions can access static data members only. Static member functions can be overloaded. Static data members can be accessed by non-static member functions.

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

#include <iostream>

using namespace std;

class A

{

private:

int x;

public:

A(int \_x) { x = \_x; }

int get() { return x; }

};

class B

{

static A a;

public:

static int get()

{ return a.get(); }

};

int main(void)

{

B b;

cout << b.get();

return 0;

}

a) Garbage value  
b) Compile-time Error  
c) Run-time Error  
d) Nothing is printed  
View Answer

Answer: b  
Explanation: Every static member function of a class must be initialized explicitly before use and a data member, a of class A declared inside class B is used without initializing ‘a’ therefore the program gives an error.

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

#include<iostream>

using namespace std;

class Test

{

private:

static int count;

public:

Test& fun();

};

int Test::count = 0;

Test& Test::fun()

{

Test::count++;

cout << Test::count << " ";

return \*this;

}

int main()

{

Test t;

t.fun().fun().fun().fun();

return 0;

}

a) 4 4 4 4  
b) 1 2 3 4  
c) 1 1 1 1  
d) 0 1 2 3  
View Answer

Answer: b  
Explanation: Here we are returning the reference of object by the function call fun() therefore this type of call is allowed. Also as count is static member of the class therefore updation is reflected to the whole class and to every object of the class. Therefore the four function calls to fun() function updates the value of count and prints.

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

#include <iostream>

class Test

{

public:

void fun();

};

static void Test::fun()

{

std::cout<<"fun() is static";

}

int main()

{

Test::fun();

return 0;

}

a) fun() is static  
b) Compile-time Error  
c) Run-time Error  
d) Nothing is printed  
View Answer

Answer: b  
Explanation: The prototype of the functions are not matched. The function declared inside a class does not have static linkage however the class defined outside the class has the static linkage, therefore, the program gives an error.

7. Const qualifier can be applied to which of the following?  
i) Functions inside a class  
ii) Arguments of a function  
iii) Static data members  
iv) Reference variables  
a) i, ii and iii  
b) i, ii, iii, and iv  
c) ii, iii and iv  
d) i only  
View Answer

Answer: b  
Explanation: const keyword can be applied to all of the following mentioned above.

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

#include <iostream>

using namespace std;

class Point

{

int x, y;

public:

Point(int i = 0, int j =0)

{ x = i; y = j; }

int getX() const { return x; }

int getY() {return y;}

};

int main()

{

const Point t;

cout << t.getX() << " ";

cout << t.gety();

return 0;

}

a) 0 0  
b) Garbage values  
c) Compile error  
d) Segmentation fault  
View Answer

Answer: c  
Explanation: C++ does not allows a constant object to access any non constant member functions and as getY() is a non constant function and t is a constant object therefore the program gives the error.

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

#include <stdio.h>

int main()

{

const int x;

x = 10;

printf("%d", x);

return 0;

}

a) 10  
b) Garbage value  
c) Error  
d) Segmentation fault  
View Answer

Answer: c  
Explanation: In C++, all the constant variables must be initialized while declaration and they cannot be modified later in the program. Now in this program as we have declared the constant variable x in first line and initializing it in the next line therefore the program gives the error.

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

#include <iostream>

int const s=9;

int main()

{

std::cout << s;

return 0;

}

a) 9  
b) Garbage value  
c) Error  
d) Segmentation fault  
View Answer

Answer: a  
Explanation: The program is syntactically and semantically correct hence the program is compiled and executed successfully.