C++ Inheritance Job Interview Questions And Answers

Interview Questions Answers

https://interviewquestionsanswers.org/

About Interview Questions Answers

Interview Questions Answers . ORG is an interview preparation guide of thousands of Job Interview Questions And Answers, Job Interviews are always stressful even for job seekers who have gone on countless interviews. The best way to reduce the stress is to be prepared for your job interview. Take the time to review the standard interview questions you will most likely be asked. These interview questions and answers on C++ Inheritance will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts.

If you find any **question or answer** is incorrect or incomplete then you can **submit your question or answer** directly with out any registration or login at our website. You just need to visit C++ Inheritance Interview Questions And Answers to add your answer click on the *Submit Your Answer* links on the website; with each question to post your answer, if you want to ask any question then you will have a link *Submit Your Question*; that's will add your question in C++ Inheritance category. To ensure quality, each submission is checked by our team, before it becomes live. This C++ Inheritance Interview preparation PDF was generated at **Wednesday 29th November, 2023**

You can follow us on FaceBook for latest Jobs, Updates and other interviews material.
www.facebook.com/InterviewQuestionsAnswers.Org

Follow us on Twitter for latest Jobs and interview preparation guides.
https://twitter.com/InterviewQA

If you need any further assistance or have queries regarding this document or its material or any of other inquiry, please do not hesitate to contact us.

Best Of Luck.

Interview Questions Answers.ORG Team
https://InterviewQuestionsAnswers.ORG/
Support@InterviewQuestionsAnswers.ORG

C++ Inheritance Interview Questions And Answers Guide.

Question - 1:

Explain about virtual destructor?

Ans:

If the destructor in the base class is not made virtual, then an object that might have been declared of type base class and instance of child class would simply call the base class destructor without calling the derived class destructor.
Hence, by making the destructor in the base class virtual, we ensure that the derived class destructor gets called before the base class destructor.

```
class a
{
        public:
        a(){printf("nBase Constructorn");}
        ~a(){printf("nBase Destructorn");}
};
class b : public a
{
       public:
       b(){printf("nDerived Constructorn");}
       ~b(){printf("nDerived Destructorn");}
};
int main()
{
       a* obj=new b;
       delete obj;
       return 0;
}
```

Output:

```
Base Constructor
Derived Constructor
Base Destructor
```

By Changing

```
~a(){printf("nBase Destructorn");}
```

to

```
virtual ~a(){printf("nBase Destructorn");}
```

Output:

```
Base Constructor
Derived Constructor
Derived Destructor
Base Destructor
```

View All Answers

Question - 2:

Explain what is Private Inheritance?

Ans:

The Public and protected members of Base class become private members of the derived class.

View All Answers

Question - 3:

Explain what is Public Inheritance?

Ans:

All the public members and protected members are inherited as public and protected respectively.

View All Answers

Question - 4:

List the advantages of inheritance?

Ans:

Allows the code to be reused as many times as needed. The base class once defined and once it is compiled, it need not be reworked.
Saves time and effort as the main code need not be written again.

View All Answers

Question - 5:

Explain about Protected Inheritance?

Ans:

Public and Protected members are derived as protected members.

View All Answers

Question - 6:

Explain what is protected inheritance?

Ans:

When a class is being derived from another class, we can make use of access specifiers. This is essentially useful to control the access the derived class members have to the base class. When inheritance is protected:
Private members of base class are not accessible to derived class.
Protected members of base class remain protected in derived class.
Public members of base class become protected in derived class.

```
#include <iostream>
using namespace std;
class base
{
    protected:
    int i, j;
    public:
    void setij(int a, int b)
    {
        i = a;
        j = b;
    }
    void showij()
    {
        cout <<"nI:"<<i<<"n J:<<j;
    }
};
class derived : protected base
{
     int k;
     public:
     void setk()
     {
         setij();
         k = i + j;
     }
     void showall()
     {
          cout <<"nK:"<<k<<show();
     }
};
int main()
{
     derived ob;
     //ob.setij(); // not allowed. Setij() is protected member of derived
     ob.setk(); //ok setk() is public member of derived
     //ob.showij(); // not allowed. Showij() is protected member of derived
    ob.showall(); // ok showall() is public member of derived
    return 0;
}
```

View All Answers

Question - 7:

Explain about the private inheritance?

Ans:

When a class is being derived from another class, we can make use of access specifiers. This is essentially useful to control the access the derived class members have to the base class. When inheritance is private:
i. Private members of base class are not accessible to derived class.
ii. Protected members of base class become private members of derived class.
iii. Public members of base class become private members of derived class.

```
#include <iostream>
using namespace std;
class base
{
    int i, j;
    public:
```

```
    void setij(int a, int b)
    {
        i = a;
        j = b;
    }
    void showij()
    {
        cout <<"nI:"<<i<<"n J:"<<j;
    }
};
class derived : private base
{
    int k;
    public:
    void setk()
    {
        //setij();
        k = i + j;
    }
    void showall()
    {
        cout <<"nK:"<<k<<show();
    }
};
int main()
{
        derived ob;
        //ob.setij(); // not allowed. Setij() is private member of derived
        ob.setk(); //ok setk() is public member of derived
        //ob.showij(); // not allowed. Showij() is private member of derived
        ob.showall(); // ok showall() is public member of derived
        return 0;
}
```

View All Answers

Question - 8:

Give example of pure virtual functions?

Ans:

A pure virtual function is a function which has no definition in the base class. Its definition lies only in the derived class ie it is compulsory for the derived class to provide definition of a pure virtual function. Since there is no definition in the base class, these functions can be equated to zero.
The general form of pure virtual function is:
virtual type func-name(parameter-list) = 0;
Consider following example of base class Shape and classes derived from it viz Circle, Rectangle, Triangle etc.

```
class Shape
{
    int x, y;
    public:
        virtual void draw() = 0;
};
class Circle: public Shape
{
        public:
        draw()
        {
            //Code for drawing a circle
        }
};
class Rectangle: public Shape
{
      Public:
      void draw()
      {
          //Code for drawing a rectangle
      }
};
class Triangle: public Shape
{
       Public:
       void draw()
       {
           //Code for drawing a triangle
       }
};
```

Thus, base class Shape has pure virtual function draw(); which is overridden by all the derived classes.

View All Answers

Question - 9:

Explain the difference between Overriding vs. overloading?

Ans:

Overloading helps to create different behaviors of methods with the same name and scope. For instance we can overload a method to return float values and integer values.
Overriding on the other hand changes the behavior of a class to make it behave different than its parent class.

View All Answers

Question - 10:

Explain about overriding?

Ans:

Defining a function in the derived class with same name as in the parent class is called overriding. In C++, the base class member can be overridden by the derived class function with the same signature as the base class function. Method overriding is used to provide different implementations of a function so that a more specific behavior can be realized.

View All Answers

Question - 11:

Explain pure virtual functions?

Ans:

Pure virtual functions are also called 'do nothing functions'.
e.g. virtual void abc() = 0;
When a pure virtual fnction is declared in the base class, the compiler necessitates the derived classes to define those functions or redeclare them are pure virtual functions. The classes containing pure virtual functions cannot be used to declare objects of their own. Such classes are called as abstract base classes.

View All Answers

Question - 12:

What is base class?

Ans:

Inheritance is one of the important features of OOP which allows us to make hierarchical classifications of classes. In this, we can create a general class which defines the most common features. Other more specific classes can inherit this class to define those features that are unique to them. In this case, the class from which other classes are inherited is referred as base class.
For example, a general class vehicle can be inherited by more specific classes car and bike. The class vehicle is base class in this case.

```
class Base
{
    int a;
    public:
       Base()
       {
              a = 1;
              cout <<"inside Base class";
       }
};
class Derived:: public Base //class Derived is inheriting class Base publically
{
    int b;
    public:
         Derived()
         {
              b = 1;
              cout <<"inside Derived class";
         }
};
```

View All Answers

Question - 13:

Do you know private inheritance?

Ans:

When a class is being derived from another class, we can make use of access specifiers. This is essentially useful to control the access the derived class members have to the base class. When inheritance is private:
i. Private members of base class are not accessible to derived class.
ii. Protected members of base class become private members of derived class.
iii. Public members of base class become private members of derived class.

```
#include <iostream>
using namespace std;
class base
{
     int i, j;
     public:
      void setij(int a, int b)
      {
          i = a;
          j = b;
      }
      void showij()
       {
```

```
        cout <<â€•
I:â€•<<i<<â€•
 J:â€•<<j;
    }
};
class derived : private base
{
    int k;
    public:
    void setk()
    {
        //setij();
        k = i + j;
    }
    void showall()
    {
          cout <<â€•
K:â€•<<k<<show();
    }
};
int main()
{
        derived ob;
        //ob.setij(); // not allowed. Setij() is private member of derived
        ob.setk(); //ok setk() is public member of derived
        //ob.showij(); // not allowed. Showij() is private member of derived
        ob.showall(); // ok showall() is public member of derived
        return 0;
}
```

View All Answers

Question - 14:

How to implement inheritance in C++?

Ans:

```
class Square: public Shape
{
    ...
};
```

In the above example all public members of Shape can be reused by the class Square. If public key word is left, private inheritance takes place by default. If protected is specified the inheritance is applied for any descendant or friend class.

View All Answers

Question - 15:

Explain the advantages of inheritance?

Ans:

Allows the code to be reused as many times as needed. The base class once defined and once it is compiled, it need not be reworked.
Saves time and effort as the main code need not be written again.

View All Answers

Question - 16:

What is a base class?

Ans:

Inheritance is one of the important features of OOP which allows us to make hierarchical classifications of classes. In this, we can create a general class which defines the most common features. Other more specific classes can inherit this class to define those features that are unique to them. In this case, the class from which other classes are inherited is referred as base class.
For example, a general class vehicle can be inherited by more specific classes car and bike. The class vehicle is base class in this case.

```
class Base
{
    int a;
    public:
      Base()
      {
            a = 1;
            cout <<â€•inside Base classâ€•;
      }
};
class Derived:: public Base //class Derived is inheriting class Base publically
{
    int b;
    public:
        Derived()
        {
            b = 1;
            cout <<â€•inside Derived classâ€•;
        }
};
```

View All Answers

Question - 17:

Explain protected inheritance?

Ans:

When a class is being derived from another class, we can make use of access specifiers. This is essentially useful to control the access the derived class members have to the base class. When inheritance is protected:

Private members of base class are not accessible to derived class.
Protected members of base class remain protected in derived class.
Public members of base class become protected in derived class.

```
#include <iostream>
using namespace std;
class base
{
    protected:
    int i, j;
    public:
    void setij(int a, int b)
    {
        i = a;
        j = b;
    }
    void showij()
    {
        cout <<â€•
I:â€•<<i<<â€•
 J:<<j;
    }
};
class derived : protected base
{
     int k;
     public:
     void setk()
     {
         setij();
         k = i + j;
     }
     void showall()
     {
          cout <<â€•
K:â€•<<k<<show();
     }
};
int main()
{
      derived ob;
      //ob.setij(); // not allowed. Setij() is protected member of derived
      ob.setk(); //ok setk() is public member of derived
      //ob.showij(); // not allowed. Showij() is protected member of derived
     ob.showall(); // ok showall() is public member of derived
     return 0;
}
```

View All Answers

Question - 18:

Do you know what is Protected Inheritance?

Ans:

Public and Protected members are derived as protected members.

View All Answers

Question - 19:

Explain Public Inheritance?

Ans:

All the public members and protected members are inherited as public and protected respectively.

View All Answers

Question - 20:

Explain Private Inheritance?

Ans:

The Public and protected members of Base class become private members of the derived class.

View All Answers

C++ Most Popular & Related Interview Guides

1 : **C++ Pointers & Functions Interview Questions and Answers.**

2 : C++ Operator Overloading Interview Questions and Answers.

3 : C++ Exception Handling Interview Questions and Answers.

4 : C++ Template Interview Questions and Answers.

5 : C++ Friend Interview Questions and Answers.

6 : C++ Virtual Functions Interview Questions and Answers.

7 : C++ Constructors Interview Questions and Answers.

8 : C++ Type Checking Interview Questions and Answers.

9 : C++ Access Control Interview Questions and Answers.

10 : C++ Inline Function Interview Questions and Answers.

Follow us on FaceBook
www.facebook.com/InterviewQuestionsAnswers.Org

Follow us on Twitter
https://twitter.com/InterviewQA

For any inquiry please do not hesitate to contact us.

Interview Questions Answers.ORG Team
https://InterviewQuestionsAnswers.ORG/
support@InterviewQuestionsAnswers.ORG