C++ Virtual Functions 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++ Virtual Functions 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++ Virtual Functions 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++ Virtual Functions category. To ensure quality, each submission is checked by our team, before it becomes live. This C++ Virtual Functions 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++ Virtual Functions Interview Questions And Answers Guide.

Question - 1:

What is the use of Vtable?

Ans:

Vtables are used for virtual functions. Its a shortform for Virtual Function Table.
It's a static table created by the compiler. Compiler creates a static table per class and the data consists on pointers to the virtual function definitions. They are automatically initialised by the compiler's constructor code.

View All Answers

Question - 2:

Explain 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.

View All Answers

Question - 3:

Explain data encapsulation?

Ans:

The wrapping up of data and functions into a single unit (Class) is known as Encapsulation. By encapsulating the data inside the class; data is not accessible to the outside world.

View All Answers

Question - 4:

Explain polymorphism?

Ans:

Polymorphism means the ability to take more than one form. An operation may exhibit different behavior in different instances. The behavior depends on the types of data used in the operation.

View All Answers

Question - 5:

Tell me can a pure virtual function have an implementation?

Ans:

The quick answer to that question is yes! A pure virtual function can have an implementation in C++ - which is something that even many veteran C++ developers do not know. So, using the SomeClass class from our example above, we can have the following code:

```
class SomeClass {
public:
  virtual void pure_virtual() = 0;  // a pure virtual function
  // note that there is no function body
};
/*This is an implementation of the pure_virtual function
   which is declared as a pure virtual function.
   This is perfectly legal:
*/
void SomeClass::pure_virtual() {
   cout <<"This is a test"<< endl;
}< /endl;
```

View All Answers

Question - 6:

Give example of a pure virtual function in C++?

Ans:

```
class SomeClass {
public:
  virtual void pure_virtual() = 0;  // a pure virtual function
  // note that there is no function body
};
```

View All Answers

Question - 7:

Can you please explain the difference between using macro and inline functions?

Ans:

A textual substitution is provided by a macro as a constant, where as an inline function is procedure which is called at each time. Although the macros have few advantages over inline functions, the disadvantages are numerous.

View All Answers

Question - 8:

Explain object slicing in C++?

Ans:

When a Derived Class object is assigned to Base class, the base class' contents in the derived object are copied to the base class leaving behind the derived class specific contents. This is referred as Object Slicing. That is, the base class object can access only the base class members. This also implies the separation of base class members from derived class members has happened.

```
class base
{
    public:
        int i, j;
};
class derived : public base
{
     public:
         int k;
};
int main()
{
     base b;
     derived d;
     b=d;
     return 0;
}
```

here b contains i and j where as d contains i, j& k. On assignment only i and j of the d get copied into i and j of b. k does not get copied. on the effect object d got sliced.

View All Answers

Question - 9:

What is problem with overriding functions?

Ans:

Overriding of functions occurs in Inheritance. A derived class may override a base class member function. In overriding, the function names and parameter list are same in both the functions.

View All Answers

Question - 10:

Do you know what is 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.

View All Answers

Question - 11:

Do you know what are static and dynamic type checking?

Ans:

Static type checking performs the type checking operation before the execution of the program. To perform this operation, the arguments, expressions, variables must be given a data type.

View All Answers

Question - 12:

Tell me what are static member functions?

Ans:

Static member functions are used to maintain a single copy of a class member function across various objects of the class. Static member functions can be called either by itself, independent of any object, by using class name and :: (scope resolution operator) or in connection with an object.

View All Answers

Question - 13:

Can you please explain the difference between static and dynamic binding of functions?

Ans:

Static Binding:
By default, matching of function call with the correct function definition happens at compile time. This is called static binding or early binding or compile-time binding. Static binding is achieved using function overloading and operator overloading. Even though there are two or more functions with same name, compiler uniquely identifies each function depending on the parameters passed to those functions.
Dynamic Binding:
C++ provides facility to specify that the compiler should match function calls with the correct definition at the run time; this is called dynamic binding or late binding or run-time binding. Dynamic binding is achieved using virtual functions. Base class pointer points to derived class object. And a function is declared virtual in base class, then the matching function is identified at run-time using virtual table entry.

View All Answers

Question - 14:

What is general form of pure virtual function? Explain?

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 - 15:

Do you know what are 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 - 16:

Explain the problem with overriding functions?

Ans:

Overriding of functions occurs in Inheritance. A derived class may override a base class member function. In overriding, the function names and parameter list are same in both the functions.

View All Answers

Question - 17:

What is Virtual base class uses?

Ans:

When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class' name with the word virtual.
Consider following example:

```
class A
{
    public:
       int i;
};
class B : virtual public A
{
     public:
          int j;
};
class C: virtual public A
{
      public:
         int k;
};
class D: public B, public C
{
public:
    int sum;
};
int main()
{
       D ob;
       ob.i = 10; //unambiguous since only one copy of i is inherited.
       ob.j = 20;
       ob.k = 30;
       ob.sum = ob.i + ob.j + ob.k;
       cout << "Value of i is : "<< ob.i<<"n";
       cout << "Value of j is : "<< ob.j<<"n"; cout << "Value of k is :"<< ob.k<<"n";
       cout << "Sum is : "<< ob.sum <<"n";
return 0;
}.
```

View All Answers

Question - 18:

What is a virtual base class?

Ans:

An ambiguity can arise when several paths exist to a class from the same base class. This means that a child class could have duplicate sets of members inherited from a single base class.

View All Answers

Question - 19:

What is virtual methods?

Ans:

virtual methods are used to use the polymorphism feature in C++. Say class A is inherited from class B. If we declare say function f() as virtual in class B and override the same function in class A then at run time appropriate method of the class will be called depending upon the type of the object.

View All Answers

Question - 20:

Explain the 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 - 21:

What is virtual base class?

Ans:

When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class' name with the word virtual.
Consider following example:

```
class A
{
   public:
      int i;
};
class B : virtual public A
{
    public:
         int j;
};
class C: virtual public A
{
     public:
        int k;
};
class D: public B, public C
{
public:
    int sum;
};
int main()
{
      D ob;
      ob.i = 10; //unambiguous since only one copy of i is inherited.
      ob.j = 20;
      ob.k = 30;
      ob.sum = ob.i + ob.j + ob.k;
      cout << â€œValue of i is : â€•<< ob.i<<â€•
â€•;
      cout << â€œValue of j is : â€•<< ob.j<<â€•
â€•; cout << â€œValue of k is :â€•<< ob.k<<â€•
â€•;
      cout << â€œSum is : â€•<< ob.sum <<â€•
â€•;
return 0;
}.
```

View All Answers

Question - 22:

Do you know the use of Vtable?

Ans:

Vtables are used for virtual functions. Its a shortform for Virtual Function Table.
It's a static table created by the compiler. Compiler creates a static table per class and the data consists on pointers to the virtual function definitions. They are automatically initialised by the compiler's constructor code.

Since virtual function pointers are stored in each instance, the compiler is enabled to call the correct vrtual function at runtime.

View All Answers

Question - 23:

What is Dynamic Binding?

Ans:

C++ provides facility to specify that the compiler should match function calls with the correct definition at the run time; this is called dynamic binding or late binding or run-time binding. Dynamic binding is achieved using virtual functions. Base class pointer points to derived class object. And a function is declared virtual in base class, then the matching function is identified at run-time using virtual table entry.

View All Answers

Question - 24:

What is Static Binding?

Ans:

By default, matching of function call with the correct function definition happens at compile time. This is called static binding or early binding or compile-time binding. Static binding is achieved using function overloading and operator overloading. Even though there are two or more functions with same name, compiler uniquely identifies each function depending on the parameters passed to those functions.

View All Answers

Question - 25:

Do you know the problem with overriding functions?

Ans:

Overriding of functions occurs in Inheritance. A derived class may override a base class member function. In overriding, the function names and parameter list are same in both the functions. Depending upon the caller object, proper function is invoked.
Consider following sample code:

```
class A
{
      int a;
      public:
        A()
        {
              a = 10;
        }
        void show()
        {
             cout << a;
        }
};
class B: public A
{
      int b;
      public:
      B()
      {
          b = 20;
      }
      void show()
      {
            cout << b;
      }
};
int main()
{
      A ob1;
      B ob2;
      ob2.show(); // calls derived class show() function. o/p is 20
      return 0;
}
```

As seen above, the derived class functions override base class functions. The problem with this is, the derived class objects can not access base class member functions which are overridden in derived class.
Base class pointer can point to derived class objects; but it has access only to base members of that derived class.
Therefore, pA = &ob2; is allowed. But pa->show() will call Base class show() function and o/p would be 10.

View All Answers

Question - 26:

Can you please explain the difference between Overloading and Overriding?

Ans:

Overriding of functions occurs when one class is inherited from another class. Overloading can occur without inheritance.
Overloaded functions must differ in function signature ie either number of parameters or type of parameters should differ. In overriding, function signatures must be same.
Overridden functions are in different scopes; whereas overloaded functions are in same scope.
Overriding is needed when derived class function has to do some added or different job than the base class function.

Overloading is used to have same name functions which behave differently depending upon parameters passed to them.

View All Answers

Question - 27:

What is Virtual Table?

Ans:

A virtual table is a mechanism to perform dynamic polymorphism i.e., run time binging. Virtual table is used to resolve the function calls at runtime. Every class that uses virtual functions is provided with its own virtual functions.
Every entry in the virtual table is a pointer that points to the derived function that is accessible by that class. A hidden pointer is added by a compiler to the base class which in turn calls *_vptr which is automatically set when an instance of the class is created and it points to the virtual table for that class..

View All Answers

Question - 28:

What is virtual function?

Ans:

A virtual function is a member function that is declared within a base class and redefined by a derived class. To create virtual function, precede the function's declaration in the base class with the keyword virtual. When a class containing virtual function is inherited, the derived class redefines the virtual function to suit its own needs.
Base class pointer can point to derived class object. In this case, using base class pointer if we call some function which is in both classes, then base class function is invoked. But if we want to invoke derived class function using base class pointer, it can be achieved by defining the function as virtual in base class, this is how virtual functions support runtime polymorphism.
Consider following program code:

```
Class A
{
      int a;
      public:
      A()
      {
            a = 1;
      }
      virtual void show()
      {
             cout <<a;
      }
};
Class B: public A
{
       int b;
       public:
       B()
       {
            b = 2;
       }
       virtual void show()
       {
             cout <<b;
       }
};
int main()
{
        A *pA;
        B oB;
        pA = &oB;
        pA->show();
        return 0;
}
```

Output is 2 since pA points to object of B and show() is virtual in base class A.

View All Answers

Question - 29:

What is Object slicing?

Ans:

When a Derived Class object is assigned to Base class, the base class' contents in the derived object are copied to the base class leaving behind the derived class specific contents. This is referred as Object Slicing. That is, the base class object can access only the base class members. This also implies the separation of base class members from derived class members has happened.

```
class base
{
    public:
       int i, j;
};
class derived : public base
{
     public:
        int k;
};
int main()
```

```
{
    base b;
    derived d;
    b=d;
    return 0;
}
```

here b contains i and j where as d contains i, j& k. On assignment only i and j of the d get copied into i and j of b. k does not get copied. on the effect object d got sliced.

View All Answers

Question - 30:

What is Virtual destructor ans explain its use?

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("
Base Constructor
");}
        ~a(){printf("
Base Destructor
");}
};
class b : public a
{
       public:
       b(){printf("
Derived Constructor
");}
       ~b(){printf("
Derived Destructor
");}
};
int main()
{
       a* obj=new b;
       delete obj;
       return 0;
}
```

Output:

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

By Changing

```
~a(){printf("
Base Destructor
");}
```

to

```
virtual ~a(){printf("
Base Destructor
");}
```

Output:

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

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++ Constructors Interview Questions and Answers.

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

8 : C++ Inheritance 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