C++ Inheritance Interview Questions And Answers Guide.

Global Guideline.

https://globalguideline.com/

C++ Inheritance Job Interview Preparation Guide.

Question # 1

What is base class?

Answer:-

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";
         }
};
```

Read More Answers.

Question # 2

Explain pure virtual functions?

Answer:-

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.

Read More Answers.

Question # 3

Explain about overriding?

Answer:-

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.

Read More Answers.

Question # 4

Explain the difference between Overriding vs. overloading?

Answer:-

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.

Read More Answers.

Question # 5

Give example of pure virtual functions?

Answer:-

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.

Read More Answers.

Question # 6

Explain about the private inheritance?

Answer:-

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;
}
```

Read More Answers.

Question # 7

Explain what is protected inheritance?

Answer:-

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;
}
```

Read More Answers.

Question # 8

Explain about Protected Inheritance?

Answer:-

Public and Protected members are derived as protected members.

Read More Answers.

Question # 9

List the advantages of inheritance?

Answer:-

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.

Read More Answers.

Question # 10

Explain what is Public Inheritance?

Answer:-

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

Read More Answers.

Question # 11

Explain what is Private Inheritance?

Answer:-

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

Read More Answers.

Question # 12

Explain about virtual destructor?

Answer:-

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
```

Read More Answers.

Question # 13

What is a base class?

Answer:-

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€•;
        }
};
```

Read More Answers.

Question # 14

Explain Private Inheritance?

Answer:-

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

Read More Answers.

Question # 15

How to implement inheritance in C++?

Answer:-

```
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.

Read More Answers.

Question # 16

Explain Public Inheritance?

Answer:-

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

Read More Answers.

Question # 17

Do you know what is Protected Inheritance?

Answer:-

Public and Protected members are derived as protected members.

Read More Answers.

Question # 18

Explain protected inheritance?

Answer:-

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;
}
```

Read More Answers.

Question # 19

Explain the advantages of inheritance?

Answer:-

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.

Read More Answers.

Question # 20

Do you know private inheritance?

Answer:-

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;
}
```

Read More Answers.

C++ Programming Most Popular Interview Topics.

1 : C++ Operator Overloading Frequently Asked Interview Questions and Answers Guide.

2 : C++ Virtual Functions Frequently Asked Interview Questions and Answers Guide.

3 : C++ Exception Handling Frequently Asked Interview Questions and Answers Guide.

4 : C++ Constructors Frequently Asked Interview Questions and Answers Guide.

5 : C++ Template Frequently Asked Interview Questions and Answers Guide.

6 : Basic C++ Syntax Frequently Asked Interview Questions and Answers Guide.

7 : C++ Friend Frequently Asked Interview Questions and Answers Guide.

8 : C++ Inline Function Frequently Asked Interview Questions and Answers Guide.

9 : C++ New And Delete Frequently Asked Interview Questions and Answers Guide.

10 : C++ Containers Frequently Asked Interview Questions and Answers Guide.

About Global Guideline.

Global Guideline is a platform to develop your own skills with thousands of job interview questions and web tutorials for fresher's and experienced candidates. These interview questions and web tutorials will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts. Global Guideline invite you to unlock your potentials with thousands of **Interview Questions with Answers** and much more. Learn the most common technologies at Global Guideline. We will help you to explore the resources of the World Wide Web and develop your own skills from the basics to the advanced. Here you will learn anything quite easily and you will really enjoy while learning. Global Guideline will help you to become a professional and Expert, well prepared for the future.

* This PDF was generated from https://GlobalGuideline.com at **November 29th, 2023**

* If any answer or question is incorrect or inappropriate or you have correct answer or you found any problem in this document then don't hesitate feel free and e-mail us we will fix it.

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

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

Best Of Luck.

Global Guideline Team
https://GlobalGuideline.com
Info@globalguideline.com