**C++ Interview Questions (Frequently Asked)**

Table of Contents

C++ is still as relevant today as it was during its advent in the mid-80s. The imperative, object-oriented programming language is widely used as a general-purpose programming language. As such, several jobs require candidates to have a profound understanding of C++. If you wish to sharpen your C++ Language skills further, here are some of the best [C++ tutorials](https://hackr.io/tutorials/learn-c-plus-plus?ref=blog) that can help you out.

Best C++ Interview Questions and Answers

If you’re preparing for a job role with emphasis on C++, then here are the 20 most important C++ interview questions to self-assess your C++ interview preparation:

***Basic Level Interview Questions***

**Question: Define C++?**

**Answer:**C++ is a computer programming language that is a superset of C wherein additional features are made in the C language.

**Question: Can we call C++ OOPS? and Why?**

**Answer:**Yes, C++ can be called OOPS. The full form of OOPS is an Object-Oriented Programming System, which means a paradigm that provides an application of various concepts, including data binding, polymorphism, inheritance, and various others.

**Question: Define Class in C++?**

**Answer:**Class is referred to as the designing of the user-defined data type. It reflects the different entities, attributes, and actions.

**Question: Define Object in C++?**

**Answer:**Object is an instance of the class. An object can have fields, methods, constructors, and related. For example, a bike in real life is an object, but it has various features such as brakes, color, size, design, and others, which are instances of its class.

**Question: Define Encapsulation in C++?**

**Answer:**Encapsulation is the process of binding together the data and functions in a class. It is applied to prevent direct access to the data for security reasons. The functions of a class are applied for this purpose. For example, the customers' net banking facility allows only the authorized person with the required login id and password to get access. That is too only for his/her part of the information in the bank data source.

**Question: What is an abstraction in C++?**

**Answer:**An abstraction in C++ is hiding the internal implementations and displaying only the required details. For example, when you send an important message through email, at that time, only writing and clicking the send option is used. This outcome is just the success message that is displayed to confirm that your email has been sent. However, the process followed in transferring the data through email is not displayed because it is of no use to you.

**Question**: **Briefly explain the concept of Inheritance in C++.**

**Answer**: C++ allows classes to inherit some of the commonly used state and behavior from other classes. This process is known as inheritance.

**Question**: **Define access specifier and its various types in C++**

**Answer**: An access specifier offers how it is possible to define how the class members, i.e., functions and variables, will be accessed outside the class's scope. There are three types of access specifier in C++:

* Private – Such class members can’t be accessed outside the class in which they are declared and are only accessible within the same class. Even child classes are disabled to access private members of its parent class.
* Protected – In addition to the class in which they are declared, the child classes can access its parent class's protected members.
* Public – Class members declared as public can be accessed throughout the program (code)

**Question: Define a namespace?**

**Answer:** A namespace is used for resolving the name conflict of the identifier, which is accomplished by placing them under various namespaces. This way, it helps in the logical division of the different codes.

**Question: Define a class template?**

**Answer:**A class template is a name given to the generic class. The use of the keyword template is made for defining a class template.

**Question: What is the function of the keyword “Volatile”?**

**Answer:**"Volatile" is a function that helps in declaring that the particular variable is volatile and thereby directs the compiler to change the variable externally- this way, the compiler optimization on the variable reference can be avoided.

**Question: Define storage class in C++? Name some?**

**Answer:**Storage class in C++ specifically resemble life or even the scope of symbols, including the variables, functions, etc. Some of the storage class names in C++ include mutable, auto, static, extern, register, etc.

**Question**:**Can we have a recursive inline function in C++?**

**Answer**:Even though it is possible to call an inline function from within itself in C++, the compiler may not generate the inline code. This is so because the compiler won’t determine the depth of the recursion at the compile time.

Nonetheless, a compiler with a good optimizer is able to inline recursive calls until some depth is fixed at compile-time and insert non-recursive calls at compile time for the cases when the actual depth exceeds run time.

**Question**: **Define an Inline Function in C++? Write its syntax. Is it possible for the C++ compiler to ignore inlining?**

**Answer**: In order to reduce the function call overhead, C++ offers inline functions. As the name suggests, an inline function is expanded in line when it is called.

As soon as the inline function is called, the whole code of the same gets either inserted or substituted at the particular point of the inline function call. The substitution is complete by the C++ compiler at compile time. Small inline functions might increase program efficiency.

The syntax of a typical inline function is:

Inline **return**-type **function**-**name**(parameters)

{

// Function code goes here

}

As the inlining is a request, not a command, the compiler can ignore it.

**Question**: **Explain ‘this’ pointer?**

**Answer**: The ‘this’ pointer is a constant pointer, and it holds the memory address of the current object. It passes as a hidden argument to all the nonstatic member function calls. Also, it is available as a local variable within the body of all the nonstatic functions.

As static member functions can be called even without any object, i.e., with the class name, the ‘this’ pointer is not available for them.

**Question**: **What are the most important differences between C and C++?**

**Answer**:

* C++ supports references while C doesn’t
* Features like friend functions, function overloading, inheritance, templates, and virtual functions are inherent to C++. These are not available in the C programming language.
* In C, exception handling is taken care of in the traditional if-else style. On the other hand, C++ offers support for exception handling at the language level.
* Mainly used input and output in C are scanf() and printf(), respectively. In C++, cin is the standard input stream while cout serves as the standard output stream.
* While C is a [procedural programming language](https://hackr.io/blog/procedural-programming), C++ provides support for both procedural and object-oriented programming approaches.

**Question**: **Why do we need the Friend class and function?**

**Answer**: Sometimes, there is a need for allowing a particular class to access private or protected members of a class. The solution is a friend class, which can access the protected and private members of the class in which it is declared as a friend.

Similar to the friend class, a friend function is able to access private and protected class members. A friend function can either be a global function or a method of some class.

Some important points about friend class and friend function:

* Friendship is not inherited.
* Friendship isn’t mutual, i.e., if some class called Friend is a friend of some other class called NotAFriend, then it doesn’t automatically become a friend of the Friend class.
* The total number of friend classes and friend functions should be limited in a program as the overabundance of the same might lead to a depreciation of the concept of encapsulation of separate classes, which is an inherent and desirable quality of object-oriented programming.

***Intermediate Level Interview Questions***

**Question**: **Explain the significance of vTable and vptr in C++ and how the compiler deals with them**

**Answer**: vTable is a table containing function pointers. Every class has a vTable. vptr is a pointer to vTable. Each object has a vptr. In order to maintain and use vptr and vTable, the C++ compiler adds additional code at two places:

1. In every constructor – This code sets vptr:
   1. Of the object being created
   2. To point to vTable of the class
2. Code with the polymorphic functional call – At every location where a polymorphic call is made, the compiler inserts code in order to first look for vptr using the base class pointer or reference. The vTable of a derived class can be accessed once the vptr is successfully fetched. Address of derived class function show() is accessed and called using the vTable.

**Question**: **How is function overloading different from operator overloading?**

**Answer**: Function overloading allows two or more functions with different type and number of parameters to have the same name. On the other hand, operator overloading allows for redefining the way an operator works for user-defined types.

**Question**: **Is it possible for a C++ program to be compiled without the main() function?**

**Answer**: Yes, it is possible. However, as the main() function is essential for the execution of the program, the program will stop after compiling and will not execute.

**Question: What is a destructor?**

**Answer:**A destructor is the member function of the class. It has the same name as the class name and also prefixed with a tilde symbol. It can be executed automatically whenever an object loses its scope.

**Question: Can we overload a destructor?**

**Answer:**No, a destructor cannot be overloaded, and it has the only form without the parameters.

**Question: What is the default constructor?**

**Answer:**The compiler provides a constructor to every class in case the provider does not offer the same. This is when the programmer provides the constructor with no specific parameters than it is called a default constructor. The code for default constructor can be displayed in the following example.

// Cpp program to illustrate the  
// concept **of** Constructors  
#include <iostream>  
using namespace std;  
class construct {  
public:  
    int a, b;  
    // Default Constructor  
    construct()  
    {  
        a = 10;  
        b = 20;  
    }  
};  
int main()  
{  
    // Default constructor called automatically  
    // when the object is created  
    construct c;  
    cout << "a: " << c.a << endl  
         << "b: " << c.b;  
    return 1;  
}

**Question: Can we provide one default constructor for our class?**

**Answer:**No, we cannot provide one default constructor for our class. When a variable in the class type is set to null, it means that it was never initialized and the outcomes will be zero.

**Question: What is the main difference between the keyword struct and class?**

**Answer:**The keyword struct is used for resembling public members by default, while the keyword class is used for resembling private members by default.

**Question**: **Draw a comparison between C++ and Java**

**Answer**:

* C++ has destructors, which are invoked automatically when an object is destroyed. Java has something called automatic garbage collection
* C++ supports multiple inheritance, operator overloading, pointers, structures, templates, and unions. Java doesn’t have any of them
* Java has a Thread class that is inherited in order to create a new thread. C++ has no inbuilt support for threads
* In C++, a goto statement offers a way to jump from a location to some labeled statement in the same function. There is no goto statement in Java
* C++ run and compile using the compiler, which converts the source code into machine level language. Hence, it is platform-dependent. Java compiler, on the other hand, converts the source code into JVM bytecode, which is platform-independent.

**Question**: **Take a look at the following C++ program:**

#include

**using** **namespace** std;

**int** **main**()

{

**int** numbers[5], sum = 0;

cout << "Enter 5 numbers: ";

**for** (**int** i = 0; i < 5; ++i)

{

cin >> numbers[i];

sum += numbers[i];

}

cout << "Sum = " << sum << endl;

**return** 0;

}

**What will be the output?**

**Answer**: The program will ask the user to enter 5 numbers and then present with their sum. For instance,

Enter 5 numbers: 22  
25  
32  
46  
66

Sum = 191

**Question**: **Explain Virtual Functions and the concept of Runtime Polymorphism in C++ with a code example.**

**Answer**: Any function when accompanying the virtual keyword exhibits the behavior of a virtual function. Unlike normal functions that are called in accordance with the type of pointer or reference used, virtual functions are called as per the type of the object pointed or referred.

In simple terms, virtual functions resolve at runtime, not anytime sooner. Use of virtual functions could also be understood as writing a C++ program leveraging the concept of [runtime polymorphism](https://www.quora.com/What-exactly-is-the-runtime-polymorphism-in-C++). Things essential to writing a virtual function in C++ are:

* A base class
* A derived class
* A function with the same name in both the classes i.e. the base class and the derived class
* A pointer or reference of base class type that points or refers, respectively to an object of the derived class

An example demonstrating the use of virtual functions (or runtime polymorphism at play) is:

#include

**using** **namespace** std;

**class** **Base** {

**public**:

**virtual** **void** **show**() { cout<<" In Base \n"; }

};

**class** **Derived**: **public** Base {

**public**:

**void** **show**() { cout<<"In Derived \n"; }

};

**int** **main**(**void**) {

Base \*bp = **new** Derived;

bp->show(); // <- Runtime Polymorphism in Action

**return** 0;

}

In the aforementioned program bp is a pointer of type Base. A call to bp->show() calls show() function of the Derived class. This is because bp points to an object of the Derived class.

**Question**: **What differences separate structure from a class in C++?**

**Answer**: There are two important distinctions between a class and a structure in C++. These are:

1. When deriving a structure from a class or some other structure, the default access specifier for the base class or structure is public. On the contrary, default access specifier is private when deriving a class.
2. While the members of a structure are public by default, the members of a class are private by default

**Question**: **What does a Static member in C++ mean?**

**Answer**: Denoted by the static keyword, a static member is allocated storage, in the static storage area, only once during the program lifetime. Some important facts pertaining to the static members are:

* Any static member function can’t be virtual
* Static member functions don’t have ‘this’ pointer
* The const, const volatile, and volatile declaration aren’t available for static member functions

**Question: Define the Reference variable?**

**Answer:**The reference variable in C++ is the name given to the existing variables. The variable name and reference variable point share the same memory location in C++, which helps in updating the original variable using the reference variable. The code can be displayed in the following example.

#include<iostream>  
using namespace std;  
int **main**()  
{  
  int x = 10;  
  // ref is a reference to x.  
  int& ref = x;  
  // Value of x is now changed to 20  
  ref = 20;  
  cout << "x = " << x << endl ;  
  // Value of x is now changed to 30  
  x = 30;  
  cout << "ref = " << ref << endl ;  
  return 0;  
}

***Advanced Level Interview Questions***

**Question**: **Define the Copy Constructor used in C++ along with its general function prototype. Also, explain the various scenarios in which it is called.**

**Answer**: A member function that initializes an object using another object of the same class is known as a copy constructor in C++. Copy Constructor can also be made private. A call to the Copy Constructor can happen in any of the following 4 scenarios when:

1. The compiler generates a temporary object
2. An object is constructed or based on some another object of the same class
3. An object of the class is returned by value
4. An object of the class is passed (i.e., to a function) by value as an argument

The general function prototype for the Copy Constructor is:

ClassName (const ClassName &old\_obj);  
Point(int x1, int y1) { x=x1; y=y1;}  
Point(const Point &p2) { x=p2.x; y=p2.y; }

**Question**:**Observe the following code snippet:**

int i = 5;  
int j = i++;

**After execution, what will be the value of I and j? Explain your answer.**

**Answer**:Post the execution of the code above, i and j will be 6 and 5, respectively. For understanding the output, it’s important to understand how the unary ‘++’ operator and the decrement ‘--’ operator works in C++.

When any of these operators precede a variable, the value of the variable is first modified, and then this modified value is used. However, when any of the two operators follow a variable, the value is first used, and then it is modified.

Therefore, in the code above j is set to the unmodified value of 5, and then i is incremented to store 6.

**Question**:**Take a look at the following two code examples for printing a vector:**

**Sample Code 1:**

vector vec;

/\* ... .. ... \*/

**for** (**auto** itr = vec.begin(); itr != vec.end(); itr++) {

itr->print();

}

**Sample Code 2:**

vector vec;

/\* ... .. ... \*/

**for** (**auto** itr = vec.begin(); itr != vec.end(); ++itr) {

itr->print();

}

**Is there any advantage of using one over the other?**

**Answer**:Though both codes will generate the same output, sample code 2 is a more performant option. This is due to the fact that the post-increment ‘itr++’ operator is more expensive than the pre-increment ‘++itr’ operator.

The post-increment operator generates a copy of the element before proceeding with incrementing the element and returning the copy. Moreover, most compilers will automatically optimize sample code 1 by converting it implicitly into sample code 2.

**Question**: **Suppose you have the GPA (Grade Point Average) of n number of students, and you need to store and display it using C++. Can you write a program that accomplishes this?**

**Answer**:

#include

#include

**using** **namespace** std;

**int** **main**()

{

**int** num;

cout << "Enter the total number of students: ";

cin >> num;

**float**\* ptr;

ptr = **new** **float**[num];

cout << "Enter the GPA of students." << endl;

**for** (**int** i = 0; i < num; ++i)

{

cout << "Student" << i + 1 << ": ";

cin >> \*(ptr + i);

}

cout << "\nDisplaying GPA of students." << endl;

**for** (**int** i = 0; i < num; ++i) {

cout << "Student" << i + 1 << " :" << \*(ptr + i) << endl;

}

**delete** [] ptr;

**return** 0;

}

**Question: What is a mutable storage class specifier? How can they be used?**

**Answer:**A mutable storage class specifier is applied only on the class's non-static and non-constant member variable. It is used for altering the constant class object's member by declaring it. This can be done by using a storage class specifier.

**Question: What are the differences between a shallow copy and a deep copy?**

**Answer:**The differences between a shallow copy and a deep copy can be stated as under.

|  |  |
| --- | --- |
| Shallow Copy | Deep Copy |
| It allows memory dumping on a bit by bit basis from one object to another. | It allows the copy field, which is done by field from one object to another. |
| It is achieved by using a copy instructor and an overloading assignment operator. | It is used for shallow copy purposes. |

**Question: Define an Abstract class in C++?**

**Answer:**An abstract class in C++ is referred to as the base class, which has at least one pure virtual function. In such a function, a person cannot instantiate an abstract class. This way, an Abstract class a pure virtual function is defined by using a pure specifier which is equal to zero during the declaration of the virtual member function in the class declaration. The code sample can be displayed as follows in example.

// An abstract class  
class Test  
{   
    // Data members of class  
public:  
    // Pure Virtual Function  
    virtual void show() = 0;  
   /\* Other members \*/  
};

**Question; Can we have a String primitive data type in C++?**

**Answer:**No, we cannot have a String Primitive data type in C++. Instead, we can have a class from the Standard Template Library (STL).

**Question: Can we use access specifiers to achieve data hiding in C++?**

**Answer:**Yes, we can use access specifiers to achieve data hiding in C++. These include Private and Protected.

**Question: Define Block scope variable?**

**Answer:**A Block scope variable is the one that is specified as a block using the C++ that can be declared anywhere within the block.

**Question: What are the functions of the scope resolution operator?**

**Answer:**The functions of the scope resolution operator include the following.

1. It helps in resolving the scope of various global variables.
2. It helps in associating the function with the class when it is defined outside the class.

The code of the scope resolution operator can be displayed as follows.

#include <iostream>    
using namespace std;    
int my\_var = 0;  
int main(void) {  
   int my\_var = 0;  
   ::my\_var = 1;  // set global my\_var to 1  
   my\_var = 2;    // set local my\_var to 2  
   cout << ::my\_var << ", " << my\_var;  
   return 0;  
}

**Question: What is the function of the keyword "Auto"?**

**Answer:**The keyword “Auto” is used by default for various local variables to make function work automatically.

**Question: Define a token in C++? Give examples?**

**Answer:**A token is a name given to the various functions in C++ programs. Examples of tokens include a keyword, symbol, string literal, identifier, constant, etc. The code of token in C++ other than C, can be displayed in the following example.

**asm**      **bool**    **catch**      class  
const\_cast   **delete** **dynamic\_cast**   explicit  
export   **false**   **friend**         inline  
mutable  **namespace**   **new** operator  
private  **protected**   **public** reinterpret\_cast  
static\_cast  **template** **this**       throw  
true     **try**     **typeid**     typename  
using    **virtual** **wchar\_t**

**Question**:**What is the ‘diamond problem’ that occurs with multiple inheritance in C++? Explain using an example.**

**Answer**:The diamond problem in C++ represents the inability of the programming language to support hybrid inheritance using multiple and hierarchical inheritance.

Suppose we have a university with some faculty members and some graduate students. A simple inheritance scheme in this scenario might have different types of people in different roles. However, all of them inherit from the same Person class.

The Person class defines an abstract getRole() method that would then be overridden by its subclasses in order to return the correct role type. Things up till this point is simple, however, if we wish to model the role of a TA or Teaching Assistant then things get weird.

A Teaching Assistant is both a student and a faculty member. This will yield the diamond problem, as illustrated in the figure below:

The problem generates an inheritance diagram resembling a diamond, hence the name, diamond problem.

Which getRole() implementation should the Teaching Assistant inherit? Graduate Student or the Faculty Member? A potential answer might be to have the Teaching Assistant class override the getRole() method and return a newly-defined role, say TA.

However, such an answer would also be far from complete as it will hide the fact that a teaching assistant is both a faculty member and a graduate