By OnlineInterviewQuestions.com

C++ Interview Questions

C++ is an object-oriented programming language and is meant for general purposes. It has imperative and generic programming features and it especially provides low-level memory manipulation facilities. The International Organization for Standardization (ISO) standardizes C++ and the latest version is C++ 2.0. Similar to C, the C++ language supports four types of memory management, namely static storage duration objects, automatic storage duration objects, thread storage duration objects, and dynamic storage duration objects.

It is widely used for partial or complete scripting and application development such as with Adobe Systems, Google apps, MySQL servers, Mozilla Firefox and Thunderbird, various media players, and more. C++ also finds applications in the development of Operating Systems such as Apple OS X, Microsoft OS, Symbian OS, and more. The development and evolution of C++ are guided by the principle that each program written in the programming language must be driven by actual problems and that its features should be used immediately in real-world programs. C++ is exceptionally compatible with many programming languages in order to foster a compatible and useful programming environment. Finally, practice here the top 35+ **C++ Interview Questions & Answers**, that are mostly asked during C Plus Plus Job Interviews.

Q1. Explain what is OOP?

Object-oriented programming (OOP) is a type of computer programming, also more commonly known as software design, in which the programmers define the data type of a data structure and the types of operations or functions which can be applied to the data structure.

The data structure becomes an object that includes both data and functions. Additionally, programmers can create relationships between one object and another. Object-oriented programming basically aims to implement real-world entities in programming. These include data abstraction, data encapsulation, inheritance, polymorphism and more.

Q2. What is the difference between #import and #include in C++?

The statement #include is used in C++ to include the source file or import the header files containing definitions of function declarations, macro definitions to be shared between several source files and variables using the preprocessor. On the other hand, #import is a Microsoft-specific statement used for binary library like DLL or .Lib files. It is very similar to include that it load all the header function definitions from the DLL file so that the user can use the header file just like the case with #include.

The statement #include allows the user to include the same file many times while #import ensures that the preprocessor only includes a file once.

Q3. What is ‘this’ pointer?

The ‘this’ pointer is passed as a hidden argument to all the non-static member method calls and is available as a local variable within the body of all non-static methods. The pointer ‘this’ is a constant pointer which holds the memory address of the current object. The pointer ‘this’ is not available in static member functions as static member functions can be called without any object (using class name).

Q4. What are the vectors in C++?

Vectors are a kind of data structure sequence containers, which represent arrays but can change in size. Just like arrays, vectors make use of contiguous storage locations for their elements, which implies that the elements of vectors can also be accessed using offsets on regular pointers to its elements, and just as efficiently as in arrays. However, unlike arrays, their size can change dynamically, with their storage being handled automatically by the container.

Q5. Differentiate between structure and class in C++.

In C++, a class is simply an extension of a structure used in the C programming language. A class is a user-defined data type, which actually binds the data and its related functions in one unit. A structure and a class in C++ differs a lot as a structure has limited functionality and features when compared to a class.The structure is also a user-defined data type with a certain template. It is used to represent a record. In C++, a structure can have both data members as well as methods as classes. They also differ in the purpose for which they are used- class is used for data abstraction and further inheritance whereas a structure is generally meant for grouping of data.

Q6. What is the use of dot in C++?

The dot basically means to reference the method or property of the object before it in object-oriented programming. The connection between the attributes or the methods with the object is indicated by a “dot” (“.”) written between them. Both, the “.” (dot) Operator as well as the -> (arrow) operator, are used to reference individual members of classes, structures, and unions. The dot operator is applied to the actual object defined in class.

Q7. Differentiate between a pointer and a reference with respect to C++.

A pointer can be reassigned n number of times while a reference cannot be re-assigned after binding. Pointers can point nowhere or to a NULL value, whereas a reference always refers to an object. The users cannot take the address of a reference like they can with pointers.

There’s no “reference arithmetic” but the user can take the address of an object pointed by a reference and perform pointer arithmetic on it.

Q8. What is the Identity function in C++? How is it useful?

In general, an ID is an abbreviation for identification. Identification is the act of being identified so a system can verify whom you say you really are. For example, if a service or system is protected with a username and password and the correct username and password is entered you are identified and can log in to the service or system. Implementing an identity function is usually with the goal of substituting every occurrence of an expression in the code by an identity function without changing the semantics of the program at all. However, the identity function is supposedly nothing more than a cast and it doesn't affect the performance of the program. Lifetime extension applies to temporaries whose construction is visible. Interposing the call to the identity function disables lifetime extension.

Q9. Differentiate between C and C++?

C	**C++**
The C language follows the procedural style programming.	C++ is a multi-paradigm programming language. It supports both procedural and object-oriented styles of programming.
Data is less secure in C as compared to C++.	In case of C++, you can use modifiers for class members to make it inaccessible for outside users. Thus, the data is more secure in C++.
The C language follows the top-down approach.	The C++ language follows the bottom-up approach.
There is no scope for function overloading in C.	C++ supports function overloading.
The user cannot use functions in structure in the case of C.	In case of C++, the user can use functions in structure.
C also does not support reference variables.	C++ supports use of reference variables.
In C, the methods scanf() and printf() are mainly used for input or output.	The C++ language mainly uses streams cin and cout to perform the input and output operations.

Q10. Differentiate between an array and a list?

An array is a collection of homogeneous elements whereas a list is a collection of heterogeneous elements. Array memory allocation is always static and continuous while the memory allocation of a list is dynamic and quite random.

In case of arrays, users need not keep a track of next memory allocation, whereas in case of lists, a user has to keep a track of next location where memory is allocated due to its dynamism.

Q11. Explain what are accessor methods?

An accessor method is a method that fetches private data that is stored within an object. An accessor provides the means by which to obtain the state of an object from other program parts. This is a preferred method in object-oriented paradigms as it provides an abstraction layer that hides the implementation details of functionality sets. An accessor doesn't need arguments and it has the same type as the retrieved variable. The

name of the accessor begins with the Get prefix and a naming convention is necessary while defining accessor methods.
Though a new dependent code is contained within accessor methods, yet they directly access state data. Furthermore, within a database fetch, the dependent code need not be changed. This is also an advantage of this type of object-oriented programming.

When comparing two data items, two access method calls are necessary in order to make the comparison. Accessors seek underlying data such as data creation, data retrieval, initialization, as well as modification. The accessor method is basically a type of instance method that contains a sequence of programming statements for the purpose of performing an action, customizing those actions with a parameter and producing a return value of some sort. An accessor function or method must be called to access a private object member.

Q12. Explain what is class definition in C++ ?

A class describes the behavior and properties common to any particular type of object. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. Defining a class means defining a blueprint for a data type or an object. A class definition begins with the keyword class followed by the class name and the class body, which is enclosed by a pair of curly braces. A class definition must always be followed either by a semicolon or a list of declarations for it to be valid.

Q13. Explain what are mutator methods in C++?

While an accessor function makes a protected data member or private object accessible to an outside user, it does not give permission for editing it or modifying it in any way. Modification of a protected data member always requires calling a mutator function. Since the mutator methods provide direct access to protected data, both mutator and accessor functions must be written and used carefully by the user.

Q14. Explain what are single and multiple inheritances in C++?

The concept of inheritance brings something of a real-world view to programming. It allows a class to be defined that has a certain set of characteristics (such as methods and instance variables) and then other classes to be created, which are derived from that class. The derived class inherits all of the features of the parent class and typically then adds some features of its own. Multiple Inheritances is a feature of C++ wherein a single class can inherit objects and methods from more than one classes. The constructors of the inherited classes are called in the same order by the base class in which they are inherited. Multiple inheritances is a feature which is not common to many programming languages. Even though it is an extremely useful feature which allows a user to apply multiple interfaces to a class, a user might encounter problems such as ambiguity and the Diamond Problem if it is not used or executed correctly.

Q15. Explain what is polymorphism in C++?

Polymorphism is a kind of runtime generalization that is enabled by the type system in object-oriented as well as functional programming languages like C++ and Haskell. Typically, we talk about two types of polymorphism in programming languages:

- Subtype polymorphism – which allows a function that is defined over a formal parameter having one type

also to be defined over subtypes of same.
- Parametric polymorphism – which allows a template, already defined class over a template parameter having one type also to be defined over subtypes of same.

Q16. Explain what data encapsulation is in C++?

Data encapsulation is a mechanism of bundling the data and the functions that use them where the implementation details of a class are kept hidden from the user. With this, the user can only perform a restricted set of operations on the hidden members of the class by executing special functions, which are also commonly known as methods. This vital concept of object-oriented programming is also often used to hide the internal representation, or state, of an object from the outside world.

Q17. What is data abstraction? How is it different from data encapsulation?

Data abstraction is a mechanism of exposing only the interfaces and hiding the implementation details from the user. Encapsulation can merely be understood as wrapping up of data and methods in a capsule, which means just hiding properties and methods. Encapsulation is used for hiding the code and data in a single unit to protect the data from the outside the world. The class is the best example of encapsulation in C++. On the other hand, abstraction means showing only the necessary details to the intended user.

Q18. How is new() different from malloc()?

New () is a preprocessor whereas malloc() is a function or method. There is no need for the user to allocate the memory while using “new” but in malloc() you have to use the function sizeof() to allocate the memory. “new” initializes the new memory to 0 while malloc() stores a random value in the newly allotted memory location.

Q19. What are inline functions? What is the syntax for defining an inline function?

The C++ language provides inline functions to reduce the function call overhead. An inline function is a method or function that is expanded in line when it is called. Whenever the inline function is called, the complete code of the inline function gets inserted or substituted at the point of the inline function call. The C++ compiler does this substitution at compile time. The inline function may increase the efficiency of the code if it is small.

The syntax for defining the function inline is:

inline return-type function-name(parameters)

```
{
// function code
}
```

Q20. Is it possible to have a recursive inline function in C++?

Although a user can call an inline function from within itself, the compiler may not generate the inline code as the compiler is unable to determine the depth of recursion at compile time. A compiler with a good optimizer can inline recursive calls until some depth fixed at compile-time (say three or five recursive calls), and insert non-recursive calls at compile time for cases when the actual depth gets exceeded at run time.

Q21. What is a responder chain?

The responder chain is a series of linked responder objects. The responder chain begins with the first responder object and ends with the app object. In case the first responder is unable to handle an event, the chain forwards the event to the next responder in line.

Q22. What do you mean by delegate? Can a user retain delegates?

A delegate is an object, which acts on behalf of, or in coordination or sync with, another object when the other/second object encounters an event during program execution.
Delegates can be retained if the user wants to. If the user declares it to be retained (which means the delegate will be influential in ARC) it'll be retained.
Generally, the norm is not to retain the delegates because they are already retained elsewhere and more importantly, the user can avoid retaining cycles by opting not to retain them.

Q23. How is Objective C different from C++?

There are a considerable amount of difference between c and c++ :

Objective C	C++
The classes in Objective C allow a method or function and a variable with the exact same name.	In case of C++, they must be different.
Objective C does not support a constructor or destructor. Instead, it has init and dealloc methods, which must be called explicitly when needed.	C++ supports constructors and destructors.
Objective C uses the signs + and – to differentiate between class methods (which are also known as factory methods in Java) and instance methods.	On the other hand, C++ uses static to specify a factory method.
Multiple inheritances is not allowed in Objective C, however, a user can use a protocol to some extent.	Multiple inheritances are allowed in C++.
Objective C does not directly allow for method overloading and has a workaround, but that is not the case for operator overloading.	C++ directly allows for method overloading.
Objective C does not allow stack-based objects either. Each object must be a pointer to a block of memory for it to exist.	C++ can use stack based objects which is automatically allocated, instantiated and deallocated on the stack.
In Objective C the message overloading is somehow faked by naming the parameters. The user is required to mangle the names manually.	C++ actually does the same but the compiler does the name mangling for the user instead of having to do it himself.
Objective C doesn't allow automatic type coercion.	C++ allows automatic type coercion.
Objective C has references. But as pointers can be used wherever a reference is employed, there isn't any specific need for references in general.	C++ has references as the language requires.

Objective C doesn’t have templates.	The users of C++ need templates because the programming language has a strong typing and static binding that prevents generic classes, such as Lists and Arrays.

Q24. Explain the volatile and mutable keywords.

The keyword volatile informs the compiler that a variable may change without the compiler knowing it. Variables that are declared as volatile will not be cached by the compiler, and will therefore always be read from memory.

The keyword mutable can be used for class member variables. Mutable variables are allowed to change from within constant member functions of the class.

Q25. What is Dynamic and Static Typing?

Static typed languages concerning computer applications are those in which type checking is done only at compile-time, while the dynamic typed languages are those in which type checking is done during run-time. Since C++ is a statically typed language, the user needs to tell the compiler what type of object they’re working with at compile time.

Q26. What is a static member?

Static is a keyword in C++ programming used to give unique characteristics to an element. The static elements are allocated storage only once in an entire program lifetime in the static storage area. Also, they have a scope for the span of the program lifetime. Static member functions do not have this pointer, and they cannot be of a virtual type. Member method declarations with the same name and the name parameter-type-list cannot be overloaded if any of them is a static member function declaration. Furthermore, a static member function cannot be declared as const, volatile, or const volatile.

Q27. What is a virtual function?

A virtual function is a method, which is used to replace the implementation provided by the base class. The said replacement is always called whenever the object in question is actually of the derived class, even if a base pointer instead of a derived pointer accesses the object. Virtual functions are always used with inheritance, and they are called according to the type of the object pointed or referred, not according to the type of pointer or reference. In other words, virtual functions are resolved later than usual, at runtime. The keyword virtual is used to make a method virtual.

Q28. What do you mean by function and operator overloading in C++?

C++ facilitates the user to specify more than one definition for a function name or an operator in the same scope. This is called function overloading and operator overloading respectively. Function overloading is primarily a feature in C++ where two or more functions can have not only the same name but also different types and numbers of parameters. Operating overloading, on the other hand, allows the user to make operators work for user-defined classes. For example, we can overload an operator such as ‘+’ in a class like String so that we can concatenate two strings by just using + instead of other descriptive functions. Other example classes where the user may overload arithmetic operators are Complex Number, Fractional Number, Big Integer, and more.

Q29. What is function overriding?

If a user inherits a class into a derived class and provides a definition for one of the base class’s function again inside the derived class, then this function is called an overridden function, and this mechanism is known as function overriding. When the member function exists in both classes (base and derived) is called in the case of overridden functions, the member function of the derived class is invoked, and the function of the base class is ignored.

Q30. Differentiate between a constructor and a destructor in C++.

Constructor	**Destructor**
A constructor is used to initialize the instance of a class or object.	A destructor simply destroys the objects when they are no longer needed.
A constructor is called when a new instance of a class or object is created.	A destructor is called when an instance of a class or object is deleted or released.
A constructor allocates the memory.	A destructor releases or frees the memory.
Constructors can have arguments.	A destructor cannot have any arguments.
Overloading of the constructor is possible in C++.	Overloading of destructors is not possible.
A constructor has the same name as class name.	A destructor also has the same name as class name but with (~) tiled operator.

Q31. What are the two main components of C++?

The C++ language itself has two main components, namely a direct mapping of hardware features, which are provided primarily by the C subset, and zero-overhead abstractions, which are based on those mappings. This is also why C++ is considered to be a highly portable language as compared to others. It is often the language of choice for multi-device, multi-platform app development and large-scale commercial applications for end users.

Q32. Why is C++ a mid-level programming language?

C++ is also considered as (but never proven to be) the superset of C. Since it is like an extension of C language is a mid-level programming language, and it acquires the features of both, low level as well as high-level programming languages.

Q33. Explain the properties and principles of OOP.

Abstraction is the process of providing only essential information regarding the standard features of objects and procedures while hiding the details from the user.

The class is a category of objects, which defines all the common properties of the various objects that belong to it.

Encapsulation is the process of combining or wrapping up of data or elements to create a single, new entity.

Information hiding is the process of hiding details of an object or function for reducing complexity.

Inheritance is a feature that represents the “is-a” relationship between the various classes.

The interface is the set of languages and codes that the applications or programs use to communicate with each other as well as with the hardware.

Messaging or message passing is a kind of communication used in parallel programming and OOP.

The object is a self-contained entity, which consists of both data and procedures employed to manipulate the data.

Polymorphism is a programming language’s ability to process objects differently depending on their data type or class.

The procedure is merely a section of a program that performs a specific task.

Q34. How many characters are recognized by ANSI C++?

128 characters are recognized by ANSI C++.

Q35. What is copy constructor?

Just like the name, a copy constructor is one the other constructors used for creating a copy of an object of class type that has been in existence already. The translator known as compiler makes available a default copy constructor for every class. The named constructor comes from creating while copy simply means new object from an existing one. In a simpler term, a copy constructor creates a copy of an already existing or available object.

Q36. What is function overloading in C++?

Function overloading is one of the features in C++ programming language where more than one function can one particular name but distinct parameters.