C++ Interview Questions

A list of top frequently asked C++ interview questions and answers are given below.

1) What is C++?

C++ is an object-oriented programming language created by Bjarne Stroustrup. It was released in 1985.

C++ is a superset of C with the major addition of classes in C language.

Initially, Stroustrup called the new language "C with classes". However, after sometime the name was changed to C++. The idea of C++ comes from the C increment operator ++.

---

2) What are the advantages of C++?

C++ doesn't only maintains all aspects from C language, it also simplifies memory management and adds several features like:

- C++ is a highly portable language means that the software developed using C++ language can run on any platform.
- C++ is an object-oriented programming language which includes the concepts such as classes, objects, inheritance, polymorphism, abstraction.
- C++ has the concept of inheritance. Through inheritance, one can eliminate the redundant code and can reuse the existing classes.
- Data hiding helps the programmer to build secure programs so that the program cannot be attacked by the invaders.
- Message passing is a technique used for communication between the objects.
- C++ contains a rich function library.

---

3) What is the difference between C and C++?

Following are the differences between C and C++:

C	C++
C language was developed by Dennis Ritchie.	C++ language was developed by Bjarne Stroustrup.
C is a structured programming language.	C++ supports both structural and object-oriented programming language.
C is a subset of C++.	C++ is a superset of C.

In C language, data and functions are the free entities.	In the C++ language, both data and functions are encapsulated together in the form of a project.
C does not support the data hiding. Therefore, the data can be used by the outside world.	C++ supports data hiding. Therefore, the data cannot b accessed by the outside world.
C supports neither function nor operator overloading.	C++ supports both function and operator overloading.
In C, the function cannot be implemented inside the structures.	In the C++, the function can be implemented inside th structures.
Reference variables are not supported in C language.	C++ supports the reference variables.
C language does not support the virtual and friend functions.	C++ supports both virtual and friend functions.
In C, scanf() and printf() are mainly used for input/output.	C++ mainly uses stream cin and cout to perform input output operations.

4) What is the difference between reference and pointer?

Following are the differences between reference and pointer:

Reference	Pointer
Reference behaves like an alias for an existing variable, i.e., it is a temporary variable.	The pointer is a variable which stores the address of a variable.
Reference variable does not require any indirection operator to access the value. A reference variable can be used directly to access the value.	Pointer variable requires an indirection operator to access the value of a variable.
Once the reference variable is assigned, then it cannot be reassigned with different address values.	The pointer variable is an independent variable means that it can be reassigned to point to different objects.
A null value cannot be assigned to the reference variable.	A null value can be assigned to the reference variable.
It is necessary to initialize the variable at the time of declaration.	It is not necessary to initialize the variable at the time of declaration.

---

9) Define token in C++.

A token in C++ can be a keyword, identifier, literal, constant and symbol.

---

10) Who was the creator of C++?

Bjarne Stroustrup.

---

11) Which operations are permitted on pointers?

Following are the operations that can be performed on pointers:

- **Incrementing or decrementing a pointer**: Incrementing a pointer means that we can increment the pointer by the size of a data type to which it points.

There are two types of increment pointers:

1. Pre-increment pointer: The pre-increment operator increments the operand by 1, and the value of the expression becomes the resulting value of the incremented. Suppose ptr is a pointer then pre-increment pointer is represented as ++ptr.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int a[5]={1,2,3,4,5};
6. int *ptr;
7. ptr=&a[0];
8. cout<<"Value of *ptr is : "<<*ptr<<"\n";
9. cout<<"Value of *++ptr : "<<*++ptr;
10. return 0;
11. }
```

Test it Now

Output:

*Value of *ptr is : 1*

*Value of *++ptr : 2*

2. Post-increment pointer: The post-increment operator increments the operand by 1, but the value of the expression will be the value of the operand prior to the incremented value of the operand. Suppose ptr is a pointer then post-increment pointer is represented as ptr++.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int a[5]={1,2,3,4,5};
6. int *ptr;
7. ptr=&a[0];
8. cout<<"Value of *ptr is : "<<*ptr<<"\n";
9. cout<<"Value of *ptr++ : "<<*ptr++;
10. return 0;
11. }
```

Test it Now

Output:

*Value of *ptr is : 1*

*Value of *ptr++ : 1*

- **Subtracting a pointer from another pointer:** When two pointers pointing to the members of an array are subtracted, then the number of elements present between the two members are returned.

---

12) Define 'std'.

Std is the default namespace standard used in C++.

---

13) Which programming language's unsatisfactory performance led to the discovery of C++?

C++was discovered in order to cope with the disadvantages of C.

---

14) How delete [] is different from delete?

Delete is used to release a unit of memory, delete[] is used to release an array.

---

15) What is the full form of STL in C++?

STL stands for Standard Template Library.

---

16) What is an object?

The Object is the instance of a class. A class provides a blueprint for objects. So you can create an object from a class. The objects of a class are declared with the same sort of declaration that we declare variables of basic types.

---

17) What are the C++ access specifiers?

The access specifiers are used to define how to functions and variables can be accessed outside the class.

There are three types of access specifiers:

- **Private**: Functions and variables declared as private can be accessed only within the same class, and they cannot be accessed outside the class they are declared.
- **Public**: Functions and variables declared under public can be accessed from anywhere.
- **Protected**: Functions and variables declared as protected cannot be accessed outside the class except a child class. This specifier is generally used in inheritance.

---

18)

---

19) What is the difference between an array and a list?

- An Array is a collection of homogeneous elements while a list is a collection of heterogeneous elements.
- Array memory allocation is static and continuous while List memory allocation is dynamic and random.
- In Array, users don't need to keep in track of next memory allocation while In the list, the user has to keep in track of next location where memory is allocated.

---

20) What is the difference between new() and malloc()?

- new() is a preprocessor while malloc() is a function.
- There is no need to allocate the memory while using "new" but in malloc() you have to use sizeof().
- "new" initializes the new memory to 0 while malloc() gives random value in the newly allotted memory location.
- The new() operator allocates the memory and calls the constructor for the object initialization and malloc() function allocates the memory but does not call the constructor for the object initialization.
- The new() operator is faster than the malloc() function as operator is faster than the function.

<<::Memory::>>

Memory is divided into two parts -

Stack - All the variables that are declared inside any function take memory from the stack.

Heap - It is unused memory in the program that is generally used for dynamic memory allocation.

Memory allocation::>>

Reserving or providing space to a variable is called memory allocation.

For storing the data, memory allocation can be done in two ways -

1.Static allocation or compile-time allocation - Static memory allocation means providing space for the variable. The size and data type of the variable is known, and it remains constant throughout the program.

2.Dynamic allocation or run-time allocation - Dynamic memory allocation means to perform memory allocation manually by a programmer. Dynamically allocated memory is allocated on Heap.

Pointers play a major role in dynamic memory allocation.

The most important use is the flexibility provided to programmers. We are free to allocate and deallocate memory whenever we need it and whenever we don't need it anymore.

There are many cases where this flexibility helps. Examples of such cases are Linked List.

Dynamic memory allocation using the new operator:

The new is a operator which is used to memory allocation at the runtime. The memory initialized by the new operator is allocated in a heap.

It returns the starting address of the memory, which gets assigned to the variable.

A malloc() is a function that allocates memory at the runtime. This function returns the void pointer, which means that it can be assigned to any pointer type.

This void pointer can be further typecast to get the pointer that points to the memory of a specified type.

malloc() takes a single argument, which is the number of bytes to allocate.

Unlike malloc(), calloc() takes two arguments:

1) Number of blocks to be allocated.

2) Size of each block in bytes.

Ex:

int *ptr; // pointer variable declaration

ptr=(int*) malloc(sizeof(int)*len); // allocating memory to the poiner variable

The sizeof() operator is required in the malloc() function that tell the malloc() function how much memory is required for the allocation.

How is memory allocated/deallocated in C++?

C uses the malloc() and calloc() function to allocate memory dynamically at run time and uses a free() function to free dynamically allocated memory.

C++ supports these functions and also has two operators new and delete, that perform the task of allocating and freeing the memory in a better and easier way.

The biggest difference between the malloc() and new.

The new operator invokes the constructor while malloc() function does not call the constructo, and the delete operator invokes the destructor to destroy the object.

The new is an operator, while malloc() is a predefined function in the stdlib header file.

The operator new can be overloaded while the malloc() function cannot be overloaded.

If the sufficient memory is not available in a heap, then the new operator will throw an exception while the malloc() function returns a NULL pointer.

In the new operator, we need to specify the number of objects to be allocated while in malloc() function, we need to specify the number of bytes to be allocated.

In the case of a new operator, we have to use the delete operator to deallocate the memory. But in the case of malloc() function, we have to use the free() function to deallocate the memory.

---

21) What are the methods of exporting a function from a DLL?

There are two ways:

- By using the DLL's type library.
- Taking a reference to the function from the DLL instance.

---

22) Define friend function.

Friend function acts as a friend of the class. It can access the private and protected members of the class. The friend function is not a member of the class, but it must be listed in the class definition. The non-member function cannot access the private data of the class. Sometimes, it is necessary for

the non-member function to access the data. The friend function is a non-member function and has the ability to access the private data of the class.

To make an outside function friendly to the class, we need to declare the function as a friend of the class as shown below:

```
1. class sample
2. {
3.    // data members;
4.  public:
5. friend void abc(void);
6. };
```

Test it Now

Following are the characteristics of a friend function:

- The friend function is not in the scope of the class in which it has been declared.
- Since it is not in the scope of the class, so it cannot be called by using the object of the class. Therefore, friend function can be invoked like a normal function.
- A friend function cannot access the private members directly, it has to use an object name and dot operator with each member name.
- Friend function uses objects as arguments.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. class Addition
4. {
5.  int a=5;
6.  int b=6;
7.  public:
8.  friend int add(Addition a1)
9.  {
10.    return(a1.a+a1.b);
11. }
12. };
13. int main()
```

```
14. {
15. int result;
16. Addition a1;
17.  result=add(a1);
18.  cout<<result;
19. return 0;
20. }
```

Test it Now

Output:

11

---

23) What is a virtual function?

- A virtual function is used to replace the implementation provided by the base class. The replacement is always called whenever the object in question is actually of the derived class, even if the object is accessed by a base pointer rather than a derived pointer.
- A virtual function is a member function which is present in the base class and redefined by the derived class.
- When we use the same function name in both base and derived class, the function in base class is declared with a keyword virtual.
- When the function is made virtual, then C++ determines at run-time which function is to be called based on the type of the object pointed by the base class pointer. Thus, by making the base class pointer to point different objects, we can execute different versions of the virtual functions.

Rules of a virtual function:

- The virtual functions should be a member of some class.
- The virtual function cannot be a static member.
- Virtual functions are called by using the object pointer.
- It can be a friend of another class.
- C++ does not contain virtual constructors but can have a virtual destructor.

---

24) When should we use multiple inheritance?

You can answer this question in three manners:

1. Never

2. Rarely
3. If you find that the problem domain cannot be accurately modeled any other way.

---

25) What is a destructor?

A Destructor is used to delete any extra resources allocated by the object. A destructor function is called automatically once the object goes out of the scope.

Rules of destructor:

- Destructors have the same name as class name and it is preceded by tilde.
- It does not contain any argument and no return type.

---

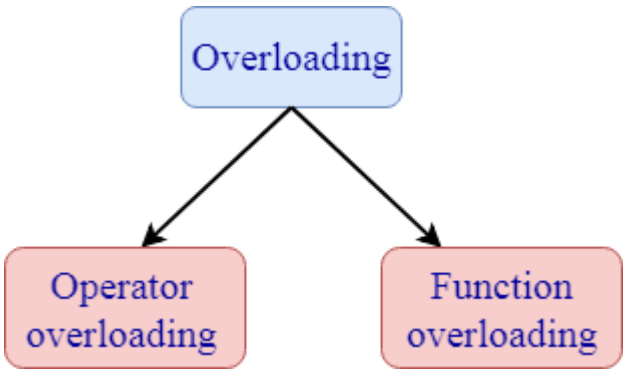
26) What is an overflow error?

It is a type of arithmetical error. It happens when the result of an arithmetical operation been greater than the actual space provided by the system.

---

27) What is overloading?

- When a single object behaves in many ways is known as overloading. A single object has the same name, but it provides different versions of the same function.
- C++ facilitates you to specify more than one definition for a function name or an operator in the same scope. It is called function overloading and operator overloading respectively.
- **Overloading is of two types:**



1. Operator overloading: Operator overloading is a compile-time polymorphism in which a standard operator is overloaded to provide a user-defined definition to it. For example, '+' operator is overloaded to perform the addition operation on data types such as int, float, etc.

Operator overloading can be implemented in the following functions:

- Member function
- Non-member function
- Friend function

Syntax of Operator overloading:

```
1. Return_type classname :: Operator Operator_symbol(argument_list)
2. {
3.       // body_statements;
4. }
```

Test it Now

2. Function overloading: Function overloading is also a type of compile-time polymorphism which can define a family of functions with the same name. The function would perform different operations based on the argument list in the function call. The function to be invoked depends on the number of arguments and the type of the arguments in the argument list.

---

28) What is function overriding?

If you inherit a class into a derived class and provide a definition for one of the base class's function again inside the derived class, then this function is called overridden function, and this mechanism is known as function overriding.

---

29) What is virtual inheritance?

Virtual inheritance facilitates you to create only one copy of each object even if the object appears more than one in the hierarchy.

---

30) What is a constructor?

A Constructor is a special method that initializes an object. Its name must be same as class name.

---

31) What is the purpose of the "delete" operator?

The "delete" operator is used to release the dynamic memory created by "new" operator.

---

32) Explain this pointer?

This pointer holds the address of the current object.

---

33) What does Scope Resolution operator do?

A scope resolution operator(::) is used to define the member function outside the class.

---

34) What is the difference between delete and delete[]?

Delete [] is used to release the array of allocated memory which was allocated using new[] whereas delete is used to release one chunk of memory which was allocated using new.

---

35) What is a pure virtual function?

The pure virtual function is a virtual function which does not contain any definition. The normal function is preceded with a keyword virtual. The pure virtual function ends with 0.

Syntax of a pure virtual function:

```
1. virtual void abc()=0;   //pure virtual function.
```

Test it Now

Let's understand this through an example:

```
1. #include<iostream>
2. using namespace std;
3. class Base
4. {
5.     public:
6.     virtual void show()=0;
7. };
8.
9. class Derived:public Base
10. {
11.     public:
12.     void show()
13.     {
14.         cout<<"javaTpoint";
15.     }
16. };
17. int main()
18. {
19.     Base* b;
20.     Derived d;
21.     b=&d;
22.     b->show();
```

```
23.     return 0;
24. }
```

Test it Now

Output:

javaTpoint

---

36) What is the difference between struct and class?

Structures	class
A structure is a user-defined data type which contains variables of dissimilar data types.	The class is a user-defined data type which contains member variables and member functions.
The variables of a structure are stored in the stack memory.	The variables of a class are stored in the heap memory.
We cannot initialize the variables directly.	We can initialize the member variables directly.
If access specifier is not specified, then by default the access specifier of the variable is "public".	If access specifier is not specified, then by default the access specifier of a variable is "private".
The instance of a structure is a "structure variable".	
Declaration of a structure: *struct structure_name* *{* *// body of structure;* *} ;*	**Declaration of class:** *class class_name* *{* *// body of class;* *}*
A structure is declared by using a struct keyword.	The class is declared by using a class keyword.
The structure does not support the inheritance.	The class supports the concept of inheritance.
The type of a structure is a value type.	The type of a class is a reference type.

---

37) What is a class template?

A class template is used to create a family of classes and functions. For example, we can create a template of an array class which will enable us to create an array of various types such as int, float, char, etc. Similarly, we can create a template for a function, suppose we have a function add(), then we can create multiple versions of add().

The syntax of a class template:

```
1. template<class T>
2. class classname
3. {
4.   // body of class;
5. };
```

Test it Now

Syntax of a object of a template class:

```
1. classname<type> objectname(arglist);
```

Test it Now

---

38) What is the difference between function overloading and operator overloading?

Function overloading: Function overloading is defined as we can have more than one version of the same function. The versions of a function will have different signature means that they have a different set of parameters.

Operator overloading: Operator overloading is defined as the standard operator can be redefined so that it has a different meaning when applied to the instances of a class.

---

39) What is a virtual destructor?

A virtual destructor in C++ is used in the base class so that the derived class object can also be destroyed. A virtual destructor is declared by using the ~ tilde operator and then virtual keyword before the constructor.

Note: Constructor cannot be virtual, but destructor can be virtual.

Let's understand this through an example

- Example without using virtual destructor

```
1. #include <iostream>
2. using namespace std;
3. class Base
4. {
5.     public:
```

```
6.      Base()
7.      {
8.         cout<<"Base constructor is called"<<"\n";
9.      }
10.     ~Base()
11.     {
12.        cout<<"Base class object is destroyed"<<"\n";
13.     }
14. };
15. class Derived:public Base
16. {
17.     public:
18.     Derived()
19.     {
20.        cout<<"Derived class constructor is called"<<"\n";
21.     }
22.     ~Derived()
23.     {
24.        cout<<"Derived class object is destroyed"<<"\n";
25.     }
26. };
27. int main()
28. {
29.    Base* b= new Derived;
30.    delete b;
31.    return 0;
32.
33. }
```

Test it Now

Output:

Base constructor is called

Derived class constructor is called

Base class object is destroyed

In the above example, delete b will only call the base class destructor due to which derived class destructor remains undestroyed. This leads to the memory leak.

- Example with a virtual destructor

```
1. #include <iostream>
2. using namespace std;
3. class Base
4. {
5.     public:
6.     Base()
7.     {
8.         cout<<"Base constructor is called"<<"\n";
9.     }
10.    virtual ~Base()
11.    {
12.        cout<<"Base class object is destroyed"<<"\n";
13.    }
14. };
15. class Derived:public Base
16. {
17.    public:
18.    Derived()
19.    {
20.        cout<<"Derived class constructor is called"<<"\n";
21.    }
22.    ~Derived()
23.    {
24.        cout<<"Derived class object is destroyed"<<"\n";
25.    }
26. };
```

```
27. int main()
28. {
29.   Base* b= new Derived;
30.   delete b;
31.   return 0;
32.
33. }
```

Test it Now

Output:

Base constructor is called

Derived class constructor is called

Derived class object is destroyed

Base class object is destroyed

When we use the virtual destructor, then the derived class destructor is called first, and then the base class destructor is called.

What is stack unwinding?

When we call some functions, it stores the address into call stack.

The stack unwinding is a process where the function call stack entries are removed at runtime.

To remove stack elements, we can use exceptions. If an exception is thrown from the inner function,

then all of the entries of the stack is removed, and return to the main invoker function

---

Tell me about Exceptional Handling. https://www.w3schools.com/cpp/cpp_exceptions.asp

One of the advantages of C++ over C is Exception Handling.

Exceptions can handle at runtime bcoz it occurs due to some unexpected conditions.

to handle runtime errors.

exception handling is performed using try/catch statement.

Try: Try is the block of a code that contains error causing program.

Throw: Whenever a error is occured in program then we are going to throw some value through throw block.

catch: Catch is a block of code that is executed when a particular exception is thrown.

What is difference between call by value and call by reference?

In the Call by Value method, there is no modification in the original value.

In the case of Call by Value, when we pass the value of the parameter during the calling of the function,

it copies them to the function's local argument.

In the Call by Reference method, there is a modification in the original value.

In the case of Call by Reference, when we pass the parameter's location reference/address, it copies and assigns them to the function's local argument.

Thus, both the actual argument and passed parameters refer to one similar location.

```
Def Display():
        print("Welcome")
Display()
```

Ans::> Welcome

```
class Employee_1:
   def __init__(self, name, age,salary):
      self.name = name
      self.age = age
      self.salary = 20000
E_1 = Employee_1("pqr", 20, 25000)   //E1 is the instance of class Employee.
#__init__ allocates memory for E1.
print(E_1.name)
print(E_1.age)
print(E_1.salary)
```

Ans::>

pqr

20

25000

What is self in Python?

Self is an instance or an object of a class.

The self-variable in the init method refers to the newly created object while in other methods, it refers to the object whose method was called.

What is PEP 8?

PEP 8 is a Python style guide. It is a document that provides the guidelines and best practices on how to write beautiful Python code.

It promotes a very readable and eye-pleasing coding style.

What is a generator in Python?

A generator is a normal function to generate a value that contains at least a yield statement rather than return. So, it becomes a generator.

The yield keyword pauses the current execution by saving its states and then resume from the same when required.

What are Decorators?

Decorators which means that functions in Python can be used or passed as arguments to another function.

You can store the function in a variable.

You can pass the function as a parameter to another function.

You can return the function from a function.

Ex:1

```
def shout(text):
    return text.upper()


print(shout('Hello'))


yell = shout

print(yell('Hello'))
```

Ex:2

```
def shout(text):
    return text.upper()

def whisper(text):
    return text.lower()

def greet(func):
    # storing the function in a variable
    greeting = func("""Hi, I am created by a function passed as an argument.""")
    print (greeting)

greet(shout)
greet(whisper)
```

What is Pickling and Unpickling?

Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using the dump function, this process is called pickling.

While the process of retrieving original Python objects from the stored string representation is called unpickling.

What is List Comprehension? Give an Example.

List comprehension is a square bracket containing a expression to creation of a list based on existing iterable.

```
my_list = [i for i in range(1, 10)]
```

What is a lambda function?

A lambda function is an anonymous function(the function is without a name.). This function can have any number of parameters but, can have just one statement.

For Example:

Add 10 to argument a, and return the result:

```
x = lambda a : a + 10

print(x(5))
```

What is Iterator

An Iterator is an object that allows the programmer to traverse through a sequence of data without having to store the entire data in the memory