|  |
| --- |
|  |

**What is C++?**Released in 1985, C++ is an object-oriented programming language created by Bjarne Stroustrup. C++ maintains almost all aspects of the C language, while simplifying memory management and adding several features - including a new datatype known as a class (you will learn more about these later) - to allow object-oriented programming. C++ maintains the features of C which allowed for low-level memory access but also gives the programmer new tools to simplify memory management.   
  
C++ used for:  
  
C++ is a powerful general-purpose programming language. It can be used to create small programs or large applications. It can be used to make CGI scripts or console-only DOS programs. C++ allows you to create programs to do almost anything you need to do. The creator of C++, Bjarne Stroustrup, has put together a partial list of applications written in C++.

**How do you find out if a linked-list has an end? (i.e. the list is not a cycle)**   
  
You can find out by using 2 pointers. One of them goes 2 nodes each time. The second one goes at 1 nodes each time. If there is a cycle, the one that goes 2 nodes each time will eventually meet the one that goes slower. If that is the case, then you will know the linked-list is a cycle.

**What is the difference between realloc() and free()?**  
The free subroutine frees a block of memory previously allocated by the malloc subroutine. Undefined results occur if the Pointer parameter is not a valid pointer. If the Pointer parameter is a null value, no action will occur. The realloc subroutine changes the size of the block of memory pointed to by the Pointer parameter to the number of bytes specified by the Size parameter and returns a new pointer to the block. The pointer specified by the Pointer parameter must have been created with the malloc, calloc, or realloc subroutines and not been deallocated with the free or realloc subroutines. Undefined results occur if the Pointer parameter is not a valid pointer.

**What is function overloading and operator overloading?**

Function overloading: C++ enables several functions of the same name to be defined, as long as these functions have different sets of parameters (at least as far as their types are concerned). This capability is called function overloading. When an overloaded function is called, the C++ compiler selects the proper function by examining the number, types and order of the arguments in the call. Function overloading is commonly used to create several functions of the same name that perform similar tasks but on different data types.   
Operator overloading allows existing C++ operators to be redefined so that they work on objects of user-defined classes. Overloaded operators are syntactic sugar for equivalent function calls. They form a pleasant facade that doesn't add anything fundamental to the language (but they can improve understandability and reduce maintenance costs).

**What is the difference between declaration and definition?**  
The declaration tells the compiler that at some later point we plan to present the definition of this declaration.  
E.g.: void stars () //function declaration   
The definition contains the actual implementation.  
E.g.: void stars () // declarator  
{  
for(int j=10; j > =0; j--) //function body  
cout << \*;  
cout << endl; }

**What are the advantages of inheritance?**  
It permits code reusability. Reusability saves time in program development. It encourages the reuse of proven and debugged high-quality software, thus reducing problem after a system becomes functional.

**How do you write a function that can reverse a linked-list?**  
void reverselist(void)  
{  
if(head==0)  
return;  
if(head->next==0)  
return;  
if(head->next==tail)  
{  
head->next = 0;  
tail->next = head;  
}  
else  
{  
node\* pre = head;  
node\* cur = head->next;  
node\* curnext = cur->next;  
head->next = 0;  
cur-> next = head;  
  
for(; curnext!=0; )  
{  
cur->next = pre;  
pre = cur;  
cur = curnext;  
curnext = curnext->next;  
}  
  
curnext->next = cur;  
}  
}

|  |
| --- |
|  |

**What do you mean by inline function?**The idea behind inline functions is to insert the code of a called function at the point where the function is called. If done carefully, this can improve the application's performance in exchange for increased compile time and possibly (but not always) an increase in the size of the generated binary executables.

Write a program that ask for user input from 5 to 9 then calculate the average   
#include "iostream.h"  
int main() {  
int MAX = 4;  
int total = 0;  
int average;  
int numb;  
for (int i=0; i<MAX; i++) {  
cout << "Please enter your input between 5 and 9: ";  
cin >> numb;  
while ( numb<5 || numb>9) {  
cout << "Invalid input, please re-enter: ";  
cin >> numb;  
}  
total = total + numb;  
}  
average = total/MAX;  
cout << "The average number is: " << average << "\n";  
return 0;  
}

**Write a short code using C++ to print out all odd number from 1 to 100 using a for loop**   
for( unsigned int i = 1; i < = 100; i++ )  
if( i & 0x00000001 )  
cout << i << \",\";

**What is public, protected, private?**Public, protected and private are three access specifier in C++.   
Public data members and member functions are accessible outside the class.   
Protected data members and member functions are only available to derived classes.   
Private data members and member functions can’t be accessed outside the class. However there is an exception can be using friend classes.  
Write a function that swaps the values of two integers, using int\* as the argument type.   
void swap(int\* a, int\*b) {  
int t;  
t = \*a;  
\*a = \*b;  
\*b = t;  
}

**Tell how to check whether a linked list is circular.**   
Create two pointers, each set to the start of the list. Update each as follows:

while (pointer1) {  
pointer1 = pointer1->next;  
pointer2 = pointer2->next; if (pointer2) pointer2=pointer2->next;  
if (pointer1 == pointer2) {  
print (\"circular\n\");  
}  
}

**OK, why does this work?**  
If a list is circular, at some point pointer2 will wrap around and be either at the item just before pointer1, or the item before that. Either way, it’s either 1 or 2 jumps until they meet.

**What is virtual constructors/destructors?**   
Answer1  
Virtual destructors:  
If an object (with a non-virtual destructor) is destroyed explicitly by applying the delete operator to a base-class pointer to the object, the base-class destructor function (matching the pointer type) is called on the object.  
There is a simple solution to this problem declare a virtual base-class destructor.  
This makes all derived-class destructors virtual even though they don’t have the same name as the base-class destructor. Now, if the object in the hierarchy is destroyed explicitly by applying the delete operator to a base-class pointer to a derived-class object, the destructor for the appropriate class is called. Virtual constructor: Constructors cannot be virtual. Declaring a constructor as a virtual function is a syntax error.   
  
Answer2  
Virtual destructors: If an object (with a non-virtual destructor) is destroyed explicitly by applying the delete operator to a base-class pointer to the object, the base-class destructor function (matching the pointer type) is called on the object.  
There is a simple solution to this problem – declare a virtual base-class destructor. This makes all derived-class destructors virtual even though they don’t have the same name as the base-class destructor. Now, if the object in the hierarchy is destroyed explicitly by applying the delete operator to a base-class pointer to a derived-class object, the destructor for the appropriate class is called.

**Virtual constructor: Constructors cannot be virtual. Declaring a constructor as a virtual function is a syntax error. Does c++ support multilevel and multiple inheritance?**  
Yes.

**What are the advantages of inheritance?**  
• It permits code reusability.  
• Reusability saves time in program development.  
• It encourages the reuse of proven and debugged high-quality software, thus reducing problem after a system becomes functional.

**What is the difference between declaration and definition?**  
The declaration tells the compiler that at some later point we plan to present the definition of this declaration.  
E.g.: void stars () //function declaration  
The definition contains the actual implementation.  
E.g.: void stars () // declarator  
{  
for(int j=10; j>=0; j--) //function body  
cout<<”\*”;  
cout<<endl; }

Top of Form

|  |
| --- |
|  |

Bottom of Form

**C++ Interview Questions and Answers**

|  |
| --- |
|  |

**What is the difference between an ARRAY and a LIST?**Answer1  
Array is collection of homogeneous elements.  
List is collection of heterogeneous elements.   
  
For Array memory allocated is static and continuous.  
For List memory allocated is dynamic and Random.   
  
Array: User need not have to keep in track of next memory allocation.  
List: User has to keep in Track of next location where memory is allocated.   
  
Answer2  
Array uses direct access of stored members, list uses sequencial access for members.  
  
//With Array you have direct access to memory position 5  
Object x = a[5]; // x takes directly a reference to 5th element of array  
  
//With the list you have to cross all previous nodes in order to get the 5th node:  
list mylist;  
list::iterator it;  
  
for( it = list.begin() ; it != list.end() ; it++ )  
{  
if( i==5)  
{  
x = \*it;  
break;  
}  
i++;  
}

**Does c++ support multilevel and multiple inheritance?**   
Yes.

**What is a template?**   
Templates allow to create generic functions that admit any data type as parameters and return value without having to overload the function with all the possible data types. Until certain point they fulfill the functionality of a macro. Its prototype is any of the two following ones:  
  
  
template <class indetifier> function\_declaration; template <typename indetifier> function\_declaration;  
The only difference between both prototypes is the use of keyword class or typename, its use is indistinct since both expressions have exactly the same meaning and behave exactly the same way.

**Define a constructor - What it is and how it might be called** (2 methods).   
Answer1  
constructor is a member function of the class, with the name of the function being the same as the class name. It also specifies how the object should be initialized.   
  
Ways of calling constructor:  
1) Implicitly: automatically by complier when an object is created.  
2) Calling the constructors explicitly is possible, but it makes the code unverifiable.  
  
Answer2  
class Point2D{  
int x; int y;  
public Point2D() : x(0) , y(0) {} //default (no argument) constructor  
};  
  
main(){  
  
Point2D MyPoint; // Implicit Constructor call. In order to allocate memory on stack, the default constructor is implicitly called.  
  
Point2D \* pPoint = new Point2D(); // Explicit Constructor call. In order to allocate memory on HEAP we call the default constructor.

**You have two pairs: new() and delete() and another pair : alloc() and free().   
Explain differences between eg. new() and malloc()**   
Answer1  
1.) “new and delete” are preprocessors while “malloc() and free()” are functions. [we dont use brackets will calling new or delete].   
2.) no need of allocate the memory while using “new” but in “malloc()” we have to use “sizeof()”.   
3.) “new” will initlize the new memory to 0 but “malloc()” gives random value in the new alloted memory location [better to use calloc()]   
  
Answer2  
new() allocates continous space for the object instace  
malloc() allocates distributed space.  
new() is castless, meaning that allocates memory for this specific type,  
malloc(), calloc() allocate space for void \* that is cated to the specific class type pointer.

**What is the difference between class and structure?**   
Structure: Initially (in C) a structure was used to bundle different type of data types together to perform a particular functionality. But C++ extended the structure to contain functions also. The major difference is that all declarations inside a structure are by default public.   
Class: Class is a successor of Structure. By default all the members inside the class are private.

|  |
| --- |
|  |

**What is RTTI?**Runtime type identification (RTTI) lets you find the dynamic type of an object when you have only a pointer or a reference to the base type. RTTI is the official way in standard C++ to discover the type of an object and to convert the type of a pointer or reference (that is, dynamic typing). The need came from practical experience with C++. RTTI replaces many Interview Questions - Homegrown versions with a solid, consistent approach.

**What is encapsulation?**   
Packaging an object’s variables within its methods is called encapsulation.

**Explain term POLIMORPHISM and give an example using eg. SHAPE object: If I have a base class SHAPE, how would I define DRAW methods for two objects CIRCLE and SQUARE**   
  
Answer1  
POLYMORPHISM : A phenomenon which enables an object to react differently to the same function call.  
in C++ it is attained by using a keyword virtual  
  
Example  
public class SHAPE  
{  
public virtual void SHAPE::DRAW()=0;  
}  
Note here the function DRAW() is pure virtual which means the sub classes must implement the DRAW() method and SHAPE cannot be instatiated   
  
public class CIRCLE::public SHAPE  
{  
public void CIRCLE::DRAW()  
{  
// TODO drawing circle  
}  
}  
public class SQUARE::public SHAPE  
{  
public void SQUARE::DRAW()  
{  
// TODO drawing square  
}  
}  
now from the user class the calls would be like  
globally  
SHAPE \*newShape;  
  
When user action is to draw  
public void MENU::OnClickDrawCircle(){  
newShape = new CIRCLE();  
}  
  
public void MENU::OnClickDrawCircle(){  
newShape = new SQUARE();  
  
}  
  
the when user actually draws  
public void CANVAS::OnMouseOperations(){  
newShape->DRAW();  
}  
  
  
Answer2  
class SHAPE{  
public virtual Draw() = 0; //abstract class with a pure virtual method  
};  
  
class CIRCLE{  
public int r;  
public virtual Draw() { this->drawCircle(0,0,r); }  
};  
  
class SQURE  
public int a;  
public virtual Draw() { this->drawRectangular(0,0,a,a); }  
};  
  
Each object is driven down from SHAPE implementing Draw() function in its own way.

**What is an object?**   
Object is a software bundle of variables and related methods. Objects have state and behavior.

**How can you tell what shell you are running on UNIX system?**   
You can do the Echo $RANDOM. It will return a undefined variable if you are from the C-Shell, just a return prompt if you are from the Bourne shell, and a 5 digit random numbers if you are from the Korn shell. You could also do a ps -l and look for the shell with the highest PID.

**What do you mean by inheritance?**   
Inheritance is the process of creating new classes, called derived classes, from existing classes or base classes. The derived class inherits all the capabilities of the base class, but can add embellishments and refinements of its own.

|  |
| --- |
|  |

**Describe PRIVATE, PROTECTED and PUBLIC – the differences and give examples.**class Point2D{  
int x; int y;  
  
public int color;  
protected bool pinned;  
public Point2D() : x(0) , y(0) {} //default (no argument) constructor  
};  
  
Point2D MyPoint;  
  
You cannot directly access private data members when they are declared (implicitly) private:   
  
MyPoint.x = 5; // Compiler will issue a compile ERROR  
//Nor yoy can see them:  
int x\_dim = MyPoint.x; // Compiler will issue a compile ERROR  
  
On the other hand, you can assign and read the public data members:   
  
MyPoint.color = 255; // no problem  
int col = MyPoint.color; // no problem  
  
With protected data members you can read them but not write them: MyPoint.pinned = true; // Compiler will issue a compile ERROR   
  
bool isPinned = MyPoint.pinned; // no problem

**What is namespace?**   
Namespaces allow us to group a set of global classes, objects and/or functions under a name. To say it somehow, they serve to split the global scope in sub-scopes known as namespaces.  
The form to use namespaces is:  
namespace identifier { namespace-body }  
Where identifier is any valid identifier and namespace-body is the set of classes, objects and functions that are included within the namespace. For example:  
namespace general { int a, b; } In this case, a and b are normal variables integrated within the general namespace. In order to access to these variables from outside the namespace we have to use the scope operator ::. For example, to access the previous variables we would have to put:  
general::a general::b   
The functionality of namespaces is specially useful in case that there is a possibility that a global object or function can have the same name than another one, causing a redefinition error.

**What is a COPY CONSTRUCTOR and when is it called?**   
A copy constructor is a method that accepts an o bject of the same class and copies it’s data members to the object on the left part of assignement:   
  
class Point2D{  
int x; int y;  
  
public int color;  
protected bool pinned;  
public Point2D() : x(0) , y(0) {} //default (no argument) constructor  
public Point2D( const Point2D & ) ;  
};  
  
 {  
this->x = p.x;  
this->y = p.y;  
this->color = p.color;  
this->pinned = p.pinned;  
}  
  
main(){  
Point2D MyPoint;  
MyPoint.color = 345;  
Point2D AnotherPoint = Point2D( MyPoint ); // now AnotherPoint has color = 345

**What is Boyce Codd Normal form?**   
A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies in F+ of the form a-> , where a and b is a subset of R, at least one of the following holds:  
\* a- > b is a trivial functional dependency (b is a subset of a)  
\* a is a superkey for schema R

**What is virtual class and friend class?**   
Friend classes are used when two or more classes are designed to work together and need access to each other's implementation in ways that the rest of the world shouldn't be allowed to have. In other words, they help keep private things private. For instance, it may be desirable for class DatabaseCursor to have more privilege to the internals of class Database than main() has.

**What is the word you will use when defining a function in base class to allow this function to be a polimorphic function?**   
virtual

**What do you mean by binding of data and functions?**   
Encapsulation.

|  |
| --- |
|  |

**What are 2 ways of exporting a function from a DLL?**1.Taking a reference to the function from the DLL instance.  
2. Using the DLL ’s Type Library

**What is the difference between an object and a class?**   
Classes and objects are separate but related concepts. Every object belongs to a class and every class contains one or more related objects.  
- A Class is static. All of the attributes of a class are fixed before, during, and after the execution of a program. The attributes of a class don't change.  
- The class to which an object belongs is also (usually) static. If a particular object belongs to a certain class at the time that it is created then it almost certainly will still belong to that class right up until the time that it is destroyed.  
- An Object on the other hand has a limited lifespan. Objects are created and eventually destroyed. Also during that lifetime, the attributes of the object may undergo significant change.

**Suppose that data is an array of 1000 integers. Write a single function call that will sort the 100 elements data [222] through data [321].**   
quicksort ((data + 222), 100);

**What is a class?**   
Class is a user-defined data type in C++. It can be created to solve a particular kind of problem. After creation the user need not know the specifics of the working of a class.

**What is friend function?**   
As the name suggests, the function acts as a friend to a class. As a friend of a class, it can access its private and protected members. A friend function is not a member of the class. But it must be listed in the class definition.

**Which recursive sorting technique always makes recursive calls to sort subarrays that are about half size of the original array?**   
Mergesort always makes recursive calls to sort subarrays that are about half size of the original array, resulting in O(n log n) time.

**What is abstraction?**   
Abstraction is of the process of hiding unwanted details from the user.

**What are virtual functions?**   
A virtual function allows derived classes to replace the implementation provided by the base class. The compiler makes sure 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. This allows algorithms in the base class to be replaced in the derived class, even if users don't know about the derived class.

**What is the difference between an external iterator and an internal iterator? Describe an advantage of an external iterator.**   
An internal iterator is implemented with member functions of the class that has items to step through. .An external iterator is implemented as a separate class that can be "attach" to the object that has items to step through. .An external iterator has the advantage that many difference iterators can be active simultaneously on the same object.

**What is a scope resolution operator?**   
A scope resolution operator (::), can be used to define the member functions of a class outside the class.

**What do you mean by pure virtual functions?**   
A pure virtual member function is a member function that the base class forces derived classes to provide. Normally these member functions have no implementation. Pure virtual functions are equated to zero.  
class Shape { public: virtual void draw() = 0; };

**What is polymorphism? Explain with an example?**   
"Poly" means "many" and "morph" means "form". Polymorphism is the ability of an object (or reference) to assume (be replaced by) or become many different forms of object.  
Example: function overloading, function overriding, virtual functions. Another example can be a plus ‘+’ sign, used for adding two integers or for using it to concatenate two strings.

|  |
| --- |
|  |

**What’s the output of the following program? Why?**#include <stdio.h>  
main()  
{  
typedef union  
{  
int a;  
char b[10];  
float c;  
}  
Union;  
  
Union x,y = {100};  
x.a = 50;  
strcpy(x.b,\"hello\");  
x.c = 21.50;  
  
printf(\"Union x : %d %s %f \n\",x.a,x.b,x.c );  
printf(\"Union y :%d %s%f \n\",y.a,y.b,y.c);  
}  
  
Given inputs X, Y, Z and operations | and & (meaning bitwise OR and AND, respectively)  
What is output equal to in   
output = (X & Y) | (X & Z) | (Y & Z)

**Why are arrays usually processed with for loop?**   
The real power of arrays comes from their facility of using an index variable to traverse the array, accessing each element with the same expression a[i]. All the is needed to make this work is a iterated statement in which the variable i serves as a counter, incrementing from 0 to a.length -1. That is exactly what a loop does.

**What is an HTML tag?**   
Answer: An HTML tag is a syntactical construct in the HTML language that abbreviates specific instructions to be executed when the HTML script is loaded into a Web browser. It is like a method in Java, a function in C++, a procedure in Pascal, or a subroutine in FORTRAN.

**Explain which of the following declarations will compile and what will be constant - a pointer or the value pointed at: \* const char \*   
\* char const \*   
\* char \* const**   
  
Note: Ask the candidate whether the first declaration is pointing to a string or a single character. Both explanations are correct, but if he says that it’s a single character pointer, ask why a whole string is initialized as char\* in C++. If he says this is a string declaration, ask him to declare a pointer to a single character. Competent candidates should not have problems pointing out why const char\* can be both a character and a string declaration, incompetent ones will come up with invalid reasons.

**You’re given a simple code for the class Bank Customer. Write the following functions:   
\* Copy constructor   
\* = operator overload  
\* == operator overload  
\* + operator overload (customers’ balances should be added up, as an example of joint account between husband and wife)**   
  
Note:Anyone confusing assignment and equality operators should be dismissed from the interview. The applicant might make a mistake of passing by value, not by reference. The candidate might also want to return a pointer, not a new object, from the addition operator. Slightly hint that you’d like the value to be changed outside the function, too, in the first case. Ask him whether the statement customer3 = customer1 + customer2 would work in the second case.

**What problems might the following macro bring to the application?**   
#define sq(x) x\*x

**Anything wrong with this code?**  
T \*p = new T[10];   
delete p;   
  
Everything is correct, Only the first element of the array will be deleted”, The entire array will be deleted, but only the first element destructor will be called.

**Anything wrong with this code?  
T \*p = 0;  
delete p;**   
  
Yes, the program will crash in an attempt to delete a null pointer.

**How do you decide which integer type to use?**   
It depends on our requirement. When we are required an integer to be stored in 1 byte (means less than or equal to 255) we use short int, for 2 bytes we use int, for 8 bytes we use long int.   
  
A char is for 1-byte integers, a short is for 2-byte integers, an int is generally a 2-byte or 4-byte integer (though not necessarily), a long is a 4-byte integer, and a long long is a 8-byte integer.

|  |
| --- |
|  |

**What does extern mean in a function declaration?**Using extern in a function declaration we can make a function such that it can used outside the file in which it is defined.   
  
An extern variable, function definition, or declaration also makes the described variable or function usable by the succeeding part of the current source file. This declaration does not replace the definition. The declaration is used to describe the variable that is externally defined.   
  
If a declaration for an identifier already exists at file scope, any extern declaration of the same identifier found within a block refers to that same object. If no other declaration for the identifier exists at file scope, the identifier has external linkage.

**What can I safely assume about the initial values of variables which are not explicitly initialized?**   
It depends on complier which may assign any garbage value to a variable if it is not initialized.

**What is the difference between char a[] = “string”; and char \*p = “string”;?**   
In the first case 6 bytes are allocated to the variable a which is fixed, where as in the second case if \*p is assigned to some other value the allocate memory can change.

**What’s the auto keyword good for?**   
Answer1  
Not much. It declares an object with automatic storage duration. Which means the object will be destroyed at the end of the objects scope. All variables in functions that are not declared as static and not dynamically allocated have automatic storage duration by default.   
  
For example  
int main()  
{  
int a; //this is the same as writing “auto int a;”  
}   
  
Answer2  
Local variables occur within a scope; they are “local” to a function. They are often called automatic variables because they automatically come into being when the scope is entered and automatically go away when the scope closes. The keyword auto makes this explicit, but local variables default to auto auto auto auto so it is never necessary to declare something as an auto auto auto auto.

**What is the difference between char a[] = “string”; and char \*p = “string”; ?**   
Answer1  
a[] = “string”;  
char \*p = “string”;  
  
The difference is this:  
p is pointing to a constant string, you can never safely say  
p[3]=’x';  
however you can always say a[3]=’x';  
  
char a[]=”string”; - character array initialization.  
char \*p=”string” ; - non-const pointer to a const-string.( this is permitted only in the case of char pointer in C++ to preserve backward compatibility with C.)   
  
Answer2  
a[] = “string”;  
char \*p = “string”;  
  
a[] will have 7 bytes. However, p is only 4 bytes. P is pointing to an adress is either BSS or the data section (depending on which compiler — GNU for the former and CC for the latter).   
  
Answer3  
char a[] = “string”;  
char \*p = “string”;  
  
for char a[]…….using the array notation 7 bytes of storage in the static memory block are taken up, one for each character and one for the terminating nul character.   
  
But, in the pointer notation char \*p………….the same 7 bytes required, plus N bytes to store the pointer variable “p” (where N depends on the system but is usually a minimum of 2 bytes and can be 4 or more)……

How do I declare an array of N pointers to functions returning pointers to functions returning pointers to characters?   
Answer1  
If you want the code to be even slightly readable, you will use typedefs.   
typedef char\* (\*functiontype\_one)(void);  
typedef functiontype\_one (\*functiontype\_two)(void);  
functiontype\_two myarray[N]; //assuming N is a const integral   
  
Answer2  
char\* (\* (\*a[N])())()  
Here a is that array. And according to question no function will not take any parameter value.

**What does extern mean in a function declaration?**   
It tells the compiler that a variable or a function exists, even if the compiler hasn’t yet seen it in the file currently being compiled. This variable or function may be defined in another file or further down in the current file.

|  |
| --- |
|  |

**How do I initialize a pointer to a function?**This is the way to initialize a pointer to a function  
void fun(int a)  
{  
  
}  
  
void main()  
{  
void (\*fp)(int);  
fp=fun;  
fp(1);  
  
}

**How do you link a C++ program to C functions?**   
By using the extern "C" linkage specification around the C function declarations.

**Explain the scope resolution operator.**   
It permits a program to reference an identifier in the global scope that has been hidden by another identifier with the same name in the local scope.

**What are the differences between a C++ struct and C++ class?**   
The default member and base-class access specifier are different.

**How many ways are there to initialize an int with a constant?**   
Two.   
There are two formats for initializers in C++ as shown in the example that follows. The first format uses the traditional C notation. The second format uses constructor notation.   
int foo = 123;  
int bar (123);

**How does throwing and catching exceptions differ from using setjmp and longjmp?**   
The throw operation calls the destructors for automatic objects instantiated since entry to the try block.

**What is a default constructor?**   
Default constructor WITH arguments class B { public: B (int m = 0) : n (m) {} int n; }; int main(int argc, char \*argv[]) { B b; return 0; }

**What is a conversion constructor?**   
A constructor that accepts one argument of a different type.

**What is the difference between a copy constructor and an overloaded assignment operator?**   
A copy constructor constructs a new object by using the content of the argument object. An overloaded assignment operator assigns the contents of an existing object to another existing object of the same class.

**When should you use multiple inheritance?**   
There are three acceptable answers: "Never," "Rarely," and "When the problem domain cannot be accurately modeled any other way."

**Explain the ISA and HASA class relationships. How would you implement each in a class design?**   
A specialized class "is" a specialization of another class and, therefore, has the ISA relationship with the other class. An Employee ISA Person. This relationship is best implemented with inheritance. Employee is derived from Person. A class may have an instance of another class. For example, an employee "has" a salary, therefore the Employee class has the HASA relationship with the Salary class. This relationship is best implemented by embedding an object of the Salary class in the Employee class.

**When is a template a better solution than a base class?**   
When you are designing a generic class to contain or otherwise manage objects of other types, when the format and behavior of those other types are unimportant to their containment or management, and particularly when those other types are unknown (thus, the generosity) to the designer of the container or manager class.

**What is a mutable member?**   
One that can be modified by the class even when the object of the class or the member function doing the modification is const.

**What is an explicit constructor?**   
A conversion constructor declared with the explicit keyword. The compiler does not use an explicit constructor to implement an implied conversion of types. It’s purpose is reserved explicitly for construction.

|  |
| --- |
|  |

**What is the Standard Template Library (STL)?**A library of container templates approved by the ANSI committee for inclusion in the standard C++ specification.   
A programmer who then launches into a discussion of the generic programming model, iterators, allocators, algorithms, and such, has a higher than average understanding of the new technology that STL brings to C++ programming.

**Describe run-time type identification.**  
The ability to determine at run time the type of an object by using the typeid operator or the dynamic\_cast operator.

**What problem does the namespace feature solve?**   
Multiple providers of libraries might use common global identifiers causing a name collision when an application tries to link with two or more such libraries. The namespace feature surrounds a library’s external declarations with a unique namespace that eliminates the potential for those collisions.   
This solution assumes that two library vendors don’t use the same namespace identifier, of course.

**Are there any new intrinsic (built-in) data types?**   
Yes. The ANSI committee added the bool intrinsic type and its true and false value keywords.

Will the following program execute?  
void main()  
{  
void \*vptr = (void \*) malloc(sizeof(void));  
vptr++;  
}  
  
Answer1  
It will throw an error, as arithmetic operations cannot be performed on void pointers.   
  
Answer2  
It will not build as sizeof cannot be applied to void\* ( error “Unknown size” )   
  
Answer3  
How can it execute if it won’t even compile? It needs to be int main, not void main. Also, cannot increment a void \*.   
  
Answer4  
According to gcc compiler it won’t show any error, simply it executes. but in general we can’t do arthematic operation on void, and gives size of void as 1   
  
Answer5  
The program compiles in GNU C while giving a warning for “void main”. The program runs without a crash. sizeof(void) is “1? hence when vptr++, the address is incremented by 1.   
  
Answer6  
Regarding arguments about GCC, be aware that this is a C++ question, not C. So gcc will compile and execute, g++ cannot. g++ complains that the return type cannot be void and the argument of sizeof() cannot be void. It also reports that ISO C++ forbids incrementing a pointer of type ‘void\*’.   
  
Answer7  
in C++  
voidp.c: In function `int main()’:  
voidp.c:4: error: invalid application of `sizeof’ to a void type  
voidp.c:4: error: `malloc’ undeclared (first use this function)  
voidp.c:4: error: (Each undeclared identifier is reported only once for each function it appears in.)  
voidp.c:6: error: ISO C++ forbids incrementing a pointer of type `void\*’  
  
But in c, it work without problems

**void main()  
{  
char \*cptr = 0?2000;  
long \*lptr = 0?2000;  
cptr++;  
lptr++;  
printf(” %x %x”, cptr, lptr);  
}  
Will it execute or not?**  
Answer1  
For Q2: As above, won’t compile because main must return int. Also, 0×2000 cannot be implicitly converted to a pointer (I assume you meant 0×2000 and not 0?2000.)   
  
Answer2  
Not Excute.  
Compile with VC7 results following errors:  
error C2440: ‘initializing’ : cannot convert from ‘int’ to ‘char \*’  
error C2440: ‘initializing’ : cannot convert from ‘int’ to ‘long \*’  
  
  
Not Excute if it is C++, but Excute in C.  
The printout:  
2001 2004   
  
Answer3  
In C++  
[$]> g++ point.c  
point.c: In function `int main()’:  
point.c:4: error: invalid conversion from `int’ to `char\*’  
point.c:5: error: invalid conversion from `int’ to `long int\*’   
  
in C  
———————————–  
[$] etc > gcc point.c  
point.c: In function `main’:  
point.c:4: warning: initialization makes pointer from integer without a cast  
point.c:5: warning: initialization makes pointer from integer without a cast  
[$] etc > ./a.exe  
2001 2004

|  |
| --- |
|  |

**What is the difference between Mutex and Binary semaphore?**semaphore is used to synchronize processes. where as mutex is used to provide synchronization between threads running in the same process.

**In C++, what is the difference between method overloading and method overriding?**   
Overloading a method (or function) in C++ is the ability for functions of the same name to be defined as long as these methods have different signatures (different set of parameters). Method overriding is the ability of the inherited class rewriting the virtual method of the base class.

**What methods can be overridden in Java?**   
In C++ terminology, all public methods in Java are virtual. Therefore, all Java methods can be overwritten in subclasses except those that are declared final, static, and private.

**What are the defining traits of an object-oriented language?**   
The defining traits of an object-oriented langauge are:  
\* encapsulation  
\* inheritance  
\* polymorphism

**Write a program that ask for user input from 5 to 9 then calculate the average**   
int main()  
{  
int MAX=4;  
int total =0;  
int average=0;  
int numb;  
cout<<"Please enter your input from 5 to 9";  
cin>>numb;  
if((numb <5)&&(numb>9))  
cout<<"please re type your input";  
else  
for(i=0;i<=MAX; i++)  
{  
total = total + numb;  
average= total /MAX;  
}  
cout<<"The average number is"<<average<<endl;  
  
return 0;  
}

**Assignment Operator - What is the diffrence between a "assignment operator" and a "copy constructor"?**   
Answer1.   
In assignment operator, you are assigning a value to an existing object. But in copy constructor, you are creating a new object and then assigning a value to that object. For example:   
complex c1,c2;  
c1=c2; //this is assignment  
complex c3=c2; //copy constructor  
  
Answer2.   
A copy constructor is used to initialize a newly declared variable from an existing variable. This makes a deep copy like assignment, but it is somewhat simpler:   
  
There is no need to test to see if it is being initialized from itself.   
There is no need to clean up (eg, delete) an existing value (there is none).   
A reference to itself is not returned.

**RTTI - What is RTTI?**   
Answer1.   
RTTI stands for "Run Time Type Identification". In an inheritance hierarchy, we can find out the exact type of the objet of which it is member. It can be done by using:   
  
1) dynamic id operator   
2) typecast operator   
  
Answer2.   
RTTI is defined as follows: Run Time Type Information, a facility that allows an object to be queried at runtime to determine its type. One of the fundamental principles of object technology is polymorphism, which is the ability of an object to dynamically change at runtime.

**STL Containers - What are the types of STL containers?**   
There are 3 types of STL containers:   
  
1. Adaptive containers like queue, stack   
2. Associative containers like set, map   
3. Sequence containers like vector, deque

**What is the need for a Virtual Destructor ?**   
Destructors are declared as virtual because if do not declare it as virtual the base class destructor will be called before the derived class destructor and that will lead to memory leak because derived classâ€™s objects will not get freed.Destructors are declared virtual so as to bind objects to the methods at runtime so that appropriate destructor is called.

|  |
| --- |
|  |

**What is "mutable"?**Answer1.   
"mutable" is a C++ keyword. When we declare const, none of its data members can change. When we want one of its members to change, we declare it as mutable.   
  
Answer2.   
A "mutable" keyword is useful when we want to force a "logical const" data member to have its value modified. A logical const can happen when we declare a data member as non-const, but we have a const member function attempting to modify that data member. For example:   
class Dummy {  
public:  
bool isValid() const;  
private:  
mutable int size\_ = 0;  
mutable bool validStatus\_ = FALSE;   
// logical const issue resolved  
};  
  
bool Dummy::isValid() const   
// data members become bitwise const  
{  
if (size > 10) {  
validStatus\_ = TRUE; // fine to assign  
size = 0; // fine to assign  
}  
}  
  
  
Answer2.   
"mutable" keyword in C++ is used to specify that the member may be updated or modified even if it is member of constant object. Example:   
class Animal {  
private:  
string name;  
string food;  
mutable int age;  
public:  
void set\_age(int a);  
};  
  
void main() {  
const Animal Tiger(â€™Fulffyâ€™,'antelopeâ€™,1);  
Tiger.set\_age(2);   
// the age can be changed since its mutable  
}

**Differences of C and C++  
Could you write a small program that will compile in C but not in C++ ?**   
In C, if you can a const variable e.g.   
const int i = 2;   
you can use this variable in other module as follows   
extern const int i;   
C compiler will not complain.   
  
But for C++ compiler u must write   
extern const int i = 2;   
else error would be generated.

**Bitwise Operations - Given inputs X, Y, Z and operations | and & (meaning bitwise OR and AND, respectively), what is output equal to in?**   
output = (X & Y) | (X & Z) | (Y & Z);

## C++ Object-Oriented Interview Questions And Answers

**What is a modifier?**   
A modifier, also called a modifying function is a member function that changes the value of  at least one data member. In other words, an operation that modifies the state of an object. Modifiers are also known as ‘mutators’. Example: The function mod is a modifier in the following code snippet:  
  
class test  
{  
int x,y;  
public:  
test()  
{  
x=0; y=0;  
}  
void mod()  
{  
x=10;  
y=15;  
}  
};

**What is an accessor?**   
An accessor is a class operation that does not modify the state of an object. The accessor functions need to be declared as const operations

|  |
| --- |
|  |

**Differentiate between a template class and class template.**Template class: A generic definition or a parameterized class not instantiated until the client provides the needed information. It’s jargon for plain templates. Class template: A class template specifies how individual classes can be constructed much like the way a class specifies how individual objects can be constructed. It’s jargon for plain classes.

**When does a name clash occur?**   
A name clash occurs when a name is defined in more than one place. For example., two different class libraries could give two different classes the same name. If you try to use many class libraries at the same time, there is a fair chance that you will be unable to compile or link the program because of name clashes.

**Define namespace.**   
It is a feature in C++ to minimize name collisions in the global name space. This namespace keyword assigns a distinct name to a library that allows other libraries to use the same identifier names without creating any name collisions. Furthermore, the compiler uses the namespace signature for differentiating the definitions.

**What is the use of ‘using’ declaration. ?**  
A using declaration makes it possible to use a name from a namespace without the scope operator.

**What is an Iterator class ?**   
A class that is used to traverse through the objects maintained by a container class. There are five categories of iterators: input iterators, output iterators, forward iterators, bidirectional iterators, random access. An iterator is an entity that gives access to the contents of a container object without violating encapsulation constraints. Access to the contents is granted on a one-at-a-time basis in order. The order can be storage order (as in lists and queues) or some arbitrary order (as in array indices) or according to some ordering relation (as in an ordered binary tree). The iterator is a construct, which provides an interface that, when called, yields either the next element in the container, or some value denoting the fact that there are no more elements to examine. Iterators hide the details of access to and update of the elements of a container class.  
The simplest and safest iterators are those that permit read-only access to the contents of a container class.

**What is an incomplete type?**   
Incomplete types refers to pointers in which there is non availability of the implementation of the referenced location or it points to some location whose value is not available for modification.   
  
int \*i=0x400 // i points to address 400  
\*i=0; //set the value of memory location pointed by i.   
  
Incomplete types are otherwise called uninitialized pointers.

**What is a dangling pointer?**   
A dangling pointer arises when you use the address of an object after  
its lifetime is over. This may occur in situations like returning  
addresses of the automatic variables from a function or using the  
address of the memory block after it is freed. The following  
code snippet shows this:  
  
class Sample  
{  
public:  
int \*ptr;  
Sample(int i)  
{  
ptr = new int(i);  
}  
  
~Sample()  
{  
delete ptr;  
}  
void PrintVal()  
{  
cout << "The value is " << \*ptr;  
}  
};  
  
void SomeFunc(Sample x)  
{  
cout << "Say i am in someFunc " << endl;  
}  
  
int main()  
{  
Sample s1 = 10;  
SomeFunc(s1);  
s1.PrintVal();  
}  
  
In the above example when PrintVal() function is  
called it is called by the pointer that has been freed by the  
destructor in SomeFunc.

|  |
| --- |
|  |

**Differentiate between the message and method.**Message:  
\* Objects communicate by sending messages to each other.  
\* A message is sent to invoke a method.  
  
Method  
\* Provides response to a message.  
\* It is an implementation of an operation.

**What is an adaptor class or Wrapper class?**   
A class that has no functionality of its own. Its member functions hide the use of a third party software component or an object with the non-compatible interface or a non-object-oriented implementation.

**What is a Null object?**   
It is an object of some class whose purpose is to indicate that a real object of that class does not exist. One common use for a null object is a return value from a member function that is supposed to return an object with some specified properties but cannot find such an object.

**What is class invariant?**   
A class invariant is a condition that defines all valid states for an object. It is a logical condition to ensure the correct working of a class. Class invariants must hold when an object is created, and they must be preserved under all operations of the class. In particular all class invariants are both preconditions and post-conditions for all operations or member functions of the class.

**What do you mean by Stack unwinding?**   
It is a process during exception handling when the destructor is called for all local objects between the place where the exception was thrown and where it is caught.

**Define precondition and post-condition to a member function.**   
Precondition: A precondition is a condition that must be true on entry to a member function. A class is used correctly if preconditions are never false. An operation is not responsible for doing anything sensible if its precondition fails to hold. For example, the interface invariants of stack class say nothing about pushing yet another element on a stack that is already full. We say that isful() is a precondition of the push operation. Post-condition: A post-condition is a condition that must be true on exit from a member function if the precondition was valid on entry to that function. A class is implemented correctly if post-conditions are never false. For example, after pushing an element on the stack, we know that isempty() must necessarily hold. This is a post-condition of the push operation.

**What are the conditions that have to be met for a condition to be an invariant of the class?**   
\* The condition should hold at the end of every constructor.  
\* The condition should hold at the end of every mutator (non-const) operation.

**What are proxy objects?**   
Objects that stand for other objects are called proxy objects or surrogates.   
template <class t="">  
class Array2D  
{  
public:  
class Array1D  
{  
public:  
T& operator[] (int index);  
const T& operator[] (int index)const;  
};  
  
Array1D operator[] (int index);  
const Array1D operator[] (int index) const;  
};  
  
The following then becomes legal:  
  
Array2D<float>data(10,20);  
cout<<data[3][6]; // fine  
  
Here data[3] yields an Array1D object and the operator [] invocation on that object yields the float in position(3,6) of the original two dimensional array. Clients of the Array2D class need not be aware of the presence of the Array1D class. Objects of this latter class stand for one-dimensional array objects that, conceptually, do not exist for clients of Array2D. Such clients program as if they were using real, live, two-dimensional arrays. Each Array1D object stands for a one-dimensional array that is absent from a conceptual model used by the clients of Array2D. In the above example, Array1D is a proxy class. Its instances stand for one-dimensional arrays that, conceptually, do not exist.

**Name some pure object oriented languages.**   
Smalltalk, Java, Eiffel, Sather.

**What is an orthogonal base class?**   
If two base classes have no overlapping methods or data they are said to be independent of, or orthogonal to each other. Orthogonal in the sense means that two classes operate in different dimensions and do not interfere with each other in any way. The same derived class may inherit such classes with no difficulty.

|  |
| --- |
|  |

**What is a node class?**A node class is a class that,  
\* relies on the base class for services and implementation,  
\* provides a wider interface to the users than its base class,  
\* relies primarily on virtual functions in its public interface  
\* depends on all its direct and indirect base class  
\* can be understood only in the context of the base class  
\* can be used as base for further derivation  
\* can be used to create objects.  
A node class is a class that has added new services or functionality beyond the services inherited from its base class.

**What is a container class? What are the types of container classes?**   
A container class is a class that is used to hold objects in memory or external storage. A container class acts as a generic holder. A container class has a predefined behavior and a well-known interface. A container class is a supporting class whose purpose is to hide the topology used for maintaining the list of objects in memory. When a container class contains a group of mixed objects, the container is called a heterogeneous container; when the container is holding a group of objects that are all the same, the container is called a homogeneous container.

**How do you write a function that can reverse a linked-list?**   
Answer1:  
  
void reverselist(void)  
{  
if(head==0)  
return;  
if(head-<next==0)  
return;  
if(head-<next==tail)  
{  
head-<next = 0;  
tail-<next = head;  
}  
else  
{  
node\* pre = head;  
node\* cur = head-<next;  
node\* curnext = cur-<next;  
head-<next = 0;  
cur-<next = head;  
  
for(; curnext!=0; )  
{  
cur-<next = pre;  
pre = cur;  
cur = curnext;  
curnext = curnext-<next;  
}  
  
curnext-<next = cur;  
}  
}  
  
Answer2:  
  
node\* reverselist(node\* head)  
{  
if(0==head || 0==head->next)   
//if head->next ==0 should return head instead of 0;  
return 0;  
  
{  
node\* prev = head;  
node\* curr = head->next;  
node\* next = curr->next;  
  
for(; next!=0; )  
{  
curr->next = prev;  
prev = curr;  
curr = next;  
next = next->next;  
}  
curr->next = prev;  
  
head->next = 0;  
head = curr;  
}  
  
return head;  
}

**What is polymorphism?**   
Polymorphism is the idea that a base class can be inherited by several classes. A base class pointer can point to its child class and a base class array can store different child class objects.

**How do you find out if a linked-list has an end? (i.e. the list is not a cycle)**   
You can find out by using 2 pointers. One of them goes 2 nodes each time. The second one goes at 1 nodes each time. If there is a cycle, the one that goes 2 nodes each time will eventually meet the one that goes slower. If that is the case, then you will know the linked-list is a cycle.

|  |
| --- |
|  |

**How can you tell what shell you are running on UNIX system?**You can do the Echo $RANDOM. It will return a undefined variable if you are from the C-Shell, just a return prompt if you are from the Bourne shell, and a 5 digit random numbers if you are from the Korn shell. You could also do a ps -l and look for the shell with the highest PID.

**What is Boyce Codd Normal form?**   
A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies in F+ of the form a->b, where a and b is a subset of R, at least one of the following holds:   
  
\* a->b is a trivial functional dependency (b is a subset of a)  
\* a is a superkey for schema R

**What is pure virtual function?**   
A class is made abstract by declaring one or more of its virtual functions to be pure. A pure virtual function is one with an initializer of = 0 in its declaration

**Write a Struct Time where integer m, h, s are its members**   
struct Time  
{  
int m;  
int h;  
int s;  
};

**How do you traverse a Btree in Backward in-order?**   
Process the node in the right subtree  
Process the root  
Process the node in the left subtree

**What is the two main roles of Operating System?**   
As a resource manager  
As a virtual machine

**In the derived class, which data member of the base class are visible?**   
In the public and protected sections.

## C++ programming on UNIX

**Could you tell something about the Unix System Kernel?**   
The kernel is the heart of the UNIX openrating system, it’s reponsible for controlling the computer’s resouces and scheduling user jobs so that each one gets its fair share of resources.

**What are each of the standard files and what are they normally associated with?**   
They are the standard input file, the standard output file and the standard error file. The first is usually associated with the keyboard, the second and third are usually associated with the terminal screen.

**Detemine the code below, tell me exectly how many times is the operation sum++ performed ?**   
for ( i = 0; i < 100; i++ )  
for ( j = 100; j > 100 - i; j–)  
sum++;   
  
(99 \* 100)/2 = 4950  
The sum++ is performed 4950 times.

**Give 4 examples which belongs application layer in TCP/IP architecture?**   
FTP, TELNET, HTTP and TFTP

**What’s the meaning of ARP in TCP/IP?**   
The "ARP" stands for Address Resolution Protocol. The ARP standard defines two basic message types: a request and a response. a request message contains an IP address and requests the corresponding hardware address; a replay contains both the IP address, sent in the request, and the hardware address.

**What is a Makefile?**   
Makefile is a utility in Unix to help compile large programs. It helps by only compiling the portion of the program that has been changed.   
A Makefile is the file and make uses to determine what rules to apply. make is useful for far more than compiling programs.

**What is deadlock?**   
Deadlock is a situation when two or more processes prevent each other from running. Example: if T1 is holding x and waiting for y to be free and T2 holding y and waiting for x to be free deadlock happens.

|  |
| --- |
|  |

**What is semaphore?**Semaphore is a special variable, it has two methods: up and down. Semaphore performs atomic operations, which means ones a semaphore is called it can not be inturrupted.   
  
The internal counter (= #ups - #downs) can never be negative. If you execute the “down” method when the internal counter is zero, it will block until some other thread calls the “up” method. Semaphores are use for thread synchronization.

**Is C an object-oriented language?**   
C is not an object-oriented language, but limited object-oriented programming can be done in C.

**Name some major differences between C++ and Java.**   
C++ has pointers; Java does not. Java is platform-independent; C++ is not. Java has garbage collection; C++ does not. Java does have pointers. In fact all variables in Java are pointers. The difference is that Java does not allow you to manipulate the addresses of the pointer

## C++ Networking Interview Questions and Answers

**What is the difference between Stack and Queue?**   
Stack is a Last In First Out (LIFO) data structure.   
Queue is a First In First Out (FIFO) data structure

**Write a fucntion that will reverse a string.**   
char \*strrev(char \*s)  
{  
int i = 0, len = strlen(s);  
char \*str;  
if ((str = (char \*)malloc(len+1)) == NULL)  
/\*cannot allocate memory \*/  
err\_num = 2;  
return (str);  
}  
while(len)  
str[i++]=s[–len];  
str[i] = NULL;  
return (str);  
}

**What is the software Life-Cycle?**   
The software Life-Cycle are  
1) Analysis and specification of the task  
2) Design of the algorithms and data structures  
3) Implementation (coding)  
4) Testing  
5) Maintenance and evolution of the system  
6) Obsolescence

**What is the difference between a Java application and a Java applet?**   
The difference between a Java application and a Java applet is that a Java application is a program that can be executed using the Java interpeter, and a JAVA applet can be transfered to different networks and executed by using a web browser (transferable to the WWW).

**Name 7 layers of the OSI Reference Model?**   
-Application layer  
-Presentation layer  
-Session layer  
-Transport layer  
-Network layer  
-Data Link layer  
-Physical layer

## C++ Algorithm Interview Questions and Answers

**What are the advantages and disadvantages of B-star trees over Binary trees?**  
Answer1  
B-star trees have better data structure and are faster in search than Binary trees, but it’s harder to write codes for B-start trees.   
  
Answer2  
The major difference between B-tree and binary tres is that B-tree is a external data structure and binary tree is a main memory data structure. The computational complexity of binary tree is counted by the number of comparison operations at each node, while the computational complexity of B-tree is determined by the disk I/O, that is, the number of node that will be loaded from disk to main memory. The comparision of the different values in one node is not counted.   
  
  
**Write the psuedo code for the Depth first Search.**  
dfs(G, v) //OUTLINE  
Mark v as "discovered"  
For each vertex w such that edge vw is in G:  
If w is undiscovered:  
dfs(G, w); that is, explore vw, visit w, explore from there as much as possible, and backtrack from w to v. Otherwise:  
"Check" vw without visiting w. Mark v as "finished".   
  
  
**Describe one simple rehashing policy.**  
The simplest rehashing policy is linear probing. Suppose a key K hashes to location i. Suppose other key occupies H[i]. The following function is used to generate alternative locations:   
rehash(j) = (j + 1) mod h   
where j is the location most recently probed. Initially j = i, the hash code for K. Notice that this version of rehash does not depend on K.   
  
  
**Describe Stacks and name a couple of places where stacks are useful.**  
A Stack is a linear structure in which insertions and deletions are always made at one end, called the top. This updating policy is called last in, first out (LIFO). It is useful when we need to check some syntex errors, such as missing parentheses.   
  
  
**Suppose a 3-bit sequence number is used in the selective-reject ARQ, what is the maximum number of frames that could be transmitted at a time?**  
If a 3-bit sequence number is used, then it could distinguish 8 different frames. Since the number of frames that could be transmitted at a time is no greater half the numner of frames that could be distinguished by the sequence number, so at most 4 frames can be transmitted at a time.

http://www.pdftutorials.com/tutorials/cpp-c-interview-questions-answers+219.html

Browse: [Home](http://www.pdftutorials.com/tutorials/default.html) / [Software/ProgrammingTutorials](http://www.pdftutorials.com/tutorials/topiclist.php?cat=Software/ProgrammingTutorials) / [CPP](http://www.pdftutorials.com/tutorials/C/C++-books-shelf1.html) / CPP ( C++ ) Interview Questions & ANswers

[**CPP ( C++ ) Interview Questions & ANswers**](http://www.pdftutorials.com/tutorials/cpp-c-interview-questions-answers+219.html)

What is encapsulation , What is inheritance , What is Polymorphism, What is constructor or ctor, What is destructor, When are copy constructors called, What is conversion constructor, What is Memory alignment, What is a dangling pointer, What is overloading, What is a Make file

What is encapsulation??  
Containing and hiding information about an object, such as internal [data structures](http://www.pdftutorials.com/tutorials/cpp-c-interview-questions-answers+219.html" \t "undefined) and   
code. Encapsulation isolates the internal complexity of an object's operation from the rest   
of the application. For example, a client component asking for net revenue from a business   
object need not know the data's origin.

What is inheritance?  
Inheritance allows one class to reuse the state and behavior of another class. The derived   
class inherits the properties and method implementations of the base class and extends it by   
overriding methods and adding additional properties and methods.  
What is Polymorphism??

Polymorphism allows a client to treat different objects in the same way even if they were   
created from different classes and exhibit different behaviors. You can use implementation   
inheritance to achieve polymorphism in languages such as C++ and Java. Base class object's   
pointer can invoke methods in derived class objects. You can also achieve polymorphism in   
C++ by function overloading and operator overloading.

What is constructor or ctor?

Constructor creates an object and initializes it. It also creates vtable for virtual   
functions. It is different from other methods in a class.  
�

What is destructor?  
Destructor usually deletes any extra resources allocated by the object.   
What is default constructor?  
Constructor with no arguments or all the arguments has default values.  
�

What is copy constructor?  
Constructor which initializes the it's object member variables ( by shallow copying) with   
another object of the same class. If you don't implement one in your class then compiler   
implements one for you.   
for example:  
Boo Obj1(10); // calling Boo constructor  
Boo Obj2(Obj1); // calling boo copy constructor  
Boo Obj2 = Obj1;// calling boo copy constructor  
�

When are copy constructors called?   
Copy constructors are called in following cases:   
a) when a function returns an object of that class by value  
b) when the object of that class is passed by value as an argument to a function  
c) when you construct an object based on another object of the same class  
d) When compiler generates a temporary object  
�

What is assignment operator?   
Default assignment operator handles assigning one object to another of the same class.   
Member to member copy (shallow copy)  
�

What are all the implicit member functions of the class? Or what are all the functions which   
compiler implements for us if we don't define one.??  
default ctor  
copy ctor  
assignment operator  
default destructor  
address operator  
�

What is conversion constructor?  
constructor with a single argument makes that constructor as conversion ctor and it can be   
used for type conversion.  
for example:  
class Boo  
{  
public:  
Boo( int i );  
};  
Boo BooObject = 10 ; // assigning int 10 Boo object  
�

What is conversion operator??  
class can have a public method for specific data type conversions.  
for example:  
class Boo  
{  
double value;  
public:  
Boo(int i )  
operator double()   
{   
return value;  
}  
};  
Boo BooObject;  
double i = BooObject; // assigning object to variable i of type double. now conversion   
operator gets called to assign the value.  
�

What is diff between malloc()/free() and new/delete?  
malloc allocates memory for object in heap but doesn't invoke object's constructor to   
initiallize the object.  
new allocates memory and also invokes constructor to initialize the object.  
malloc() and free() do not support object semantics   
Does not construct and destruct objects   
string \* ptr = (string \*)(malloc (sizeof(string)))  
Are not safe   
Does not calculate the size of the objects that it construct   
Returns a pointer to void   
int \*p = (int \*) (malloc(sizeof(int)));  
int \*p = new int;  
Are not extensible   
new and delete can be overloaded in a class   
"delete" first calls the object's termination routine (i.e. its destructor) and then   
releases the space the object occupied on the heap memory. If an array of objects was   
created using new, then delete must be told that it is dealing with an array by preceding   
the name with an empty []:-   
Int\_t \*my\_ints = new Int\_t[10];  
...  
delete []my\_ints;  
What is the diff between "new" and "operator new" ?   
  
"operator new" works like malloc.  
What is difference between template and macro??  
There is no way for the compiler to verify that the macro parameters are of compatible   
types. The macro is expanded without any special type checking.  
If macro parameter has a postincremented variable ( like c++ ), the increment is performed   
two times.  
Because macros are expanded by the preprocessor, compiler error messages will refer to the   
expanded macro, rather than the macro definition itself. Also, the macro will show up in   
expanded form during debugging.   
for example:  
Macro:  
#define min(i, j) (i < j ? i : j)  
template:  
template   
T min (T i, T j)   
{   
return i < j ? i : j;  
}  
What are C++ storage classes?  
auto  
register  
static  
extern  
auto: the default. Variables are automatically created and initialized when they are defined   
and are destroyed at the end of the block containing their definition. They are not visible   
outside that block  
register: a type of auto variable. a suggestion to the compiler to use a CPU register for   
performance  
static: a variable that is known only in the function that contains its definition but is   
never destroyed and retains its value between calls to that function. It exists from the   
time the program begins execution  
extern: a static variable whose definition and placement is determined when all object and   
library modules are combined (linked) to form the executable code file. It can be visible   
outside the file where it is defined.  
What are storage qualifiers in C++ ?  
They are..  
const  
volatile  
mutable  
Const keyword indicates that memory once initialized, should not be altered by a program.  
volatile keyword indicates that the value in the memory location can be altered even though   
nothing in the program  
code modifies the contents. for example if you have a pointer to hardware location that   
contains the time, where hardware changes the value of this pointer variable and not the   
program. The intent of this keyword to improve the optimization ability of the compiler.   
mutable keyword indicates that particular member of a structure or class can be altered even   
if a particular structure variable, class, or class member function is constant.  
struct data  
{  
char name[80];  
mutable double salary;  
}  
const data MyStruct = { "Satish Shetty", 1000 }; //initlized by complier  
strcpy ( MyStruct.name, "Shilpa Shetty"); // compiler error  
MyStruct.salaray = 2000 ; // complier is happy allowed  
What is reference ??  
reference is a name that acts as an alias, or alternative name, for a previously defined   
variable or an object. prepending variable with "&" symbol makes it as reference. for   
example:   
int a;  
int &b = a;   
What is passing by reference?  
Method of passing arguments to a function which takes parameter of type reference. for   
example:  
void swap( int & x, int & y )  
{  
int temp = x;  
x = y;  
y = x;   
}  
int a=2, b=3;  
swap( a, b );  
Basically, inside the function there won't be any copy of the arguments "x" and "y" instead   
they refer to original variables a and b. so no extra memory needed to pass arguments and it   
is more efficient.   
When do use "const" reference arguments in function?  
a) Using const protects you against programming errors that inadvertently alter data.  
b) Using const allows function to process both const and non-const actual arguments, while a   
function without const in the prototype can only accept non constant arguments.  
c) Using a const reference allows the function to generate and use a temporary variable   
appropriately.  
When are temporary variables created by C++ compiler?  
Provided that function parameter is a "const reference", compiler generates temporary   
variable in following 2 ways.   
a) The actual argument is the correct type, but it isn't Lvalue  
double Cuberoot ( const double & num )  
{  
num = num \* num \* num;  
return num;  
}  
double temp = 2.0;  
double value = cuberoot ( 3.0 + temp ); // argument is a expression and not a Lvalue;  
b) The actual argument is of the wrong type, but of a type that can be converted to the   
correct type  
long temp = 3L;  
double value = cuberoot ( temp); // long to double conversion   
What is virtual function?  
When derived class overrides the base class method by redefining the same function, then if   
client wants to access redefined the method from derived class through a pointer from base   
class object, then you must define this function in base class as virtual function.  
class parent  
{  
void Show()   
{   
cout << "i'm parent" << endl;  
}  
};  
class child: public parent  
{  
void Show()   
{   
cout << "i'm child" << endl;  
}  
};  
parent \* parent\_object\_ptr = new child;  
parent\_object\_ptr->show() // calls parent->show() i   
now we goto virtual world...  
class parent  
{  
virtual void Show()   
{   
cout << "i'm parent" << endl;  
}  
};  
class child: public parent  
{  
void Show()   
{   
cout << "i'm child" << endl;  
}  
};  
parent \* parent\_object\_ptr = new child;  
parent\_object\_ptr->show() // calls child->show()   
What is pure virtual function? or what is abstract class?  
When you define only function prototype in a base class without and do the complete   
implementation in derived class. This base class is called abstract class and client won't   
able to instantiate an object using this base class.  
You can make a pure virtual function or abstract class this way..   
class Boo  
{  
void foo() = 0;  
}  
Boo MyBoo; // compilation error  
What is Memory alignment??  
The term alignment primarily means the tendency of an address pointer value to be a multiple   
of some power of two. So a pointer with two byte alignment has a zero in the least   
significant bit. And a pointer with four byte alignment has a zero in both the two least   
significant bits. And so on. More alignment means a longer sequence of zero bits in the   
lowest bits of a pointer.  
What problem does the namespace feature solve?   
Multiple providers of libraries might use common global identifiers causing a name collision   
when an application tries to link with two or more such libraries. The namespace feature   
surrounds a library's external declarations with a unique namespace that eliminates the   
potential for those collisions.   
namespace [identifier] { namespace-body }  
A namespace declaration identifies and assigns a name to a declarative region.  
The identifier in a namespace declaration must be unique in the declarative region in which   
it is used. The identifier is the name of the namespace and is used to reference its   
members.  
What is the use of 'using' declaration?  
A using declaration makes it possible to use a name from a namespace without the scope   
operator.   
What is an Iterator class?   
A class that is used to traverse through the objects maintained by a container class. There   
are five categories of iterators: input iterators, output iterators, forward iterators,   
bidirectional iterators, random access. An iterator is an entity that gives access to the   
contents of a container object without violating encapsulation constraints. Access to the   
contents is granted on a one-at-a-time basis in order. The order can be storage order (as in   
lists and queues) or some arbitrary order (as in array indices) or according to some   
ordering relation (as in an ordered binary tree). The iterator is a construct, which   
provides an interface that, when called, yields either the next element in the container, or   
some value denoting the fact that there are no more elements to examine. Iterators hide the   
details of access to and update of the elements of a container class. Something like a   
pointer.   
What is a dangling pointer?   
A dangling pointer arises when you use the address of an object after its lifetime is over.   
This may occur in situations like returning addresses of the automatic variables from a   
function or using the address of the memory block after it is freed.  
What do you mean by Stack unwinding?   
It is a process during exception handling when the destructor is called for all local   
objects in the stack between the place where the exception was thrown and where it is   
caught.  
Name the operators that cannot be overloaded??  
sizeof, ., .\*, .->, ::, ?:   
What is a container class? What are the types of container classes?   
A container class is a class that is used to hold objects in memory or external storage. A   
container class acts as a generic holder. A container class has a predefined behavior and a   
well-known interface. A container class is a supporting class whose purpose is to hide the   
topology used for maintaining the list of objects in memory. When a container class contains   
a group of mixed objects, the container is called a heterogeneous container; when the   
container is holding a group of objects that are all the same, the container is called a   
homogeneous container.   
What is inline function??  
The \_\_inline keyword tells the compiler to substitute the code within the function   
definition for every instance of a function call. However, substitution occurs only at the   
compiler's discretion. For example, the compiler does not inline a function if its address   
is taken or if it is too large to inline.  
What is overloading??  
With the C++ language, you can overload functions and operators. Overloading is the practice   
of supplying more than one definition for a given function name in the same scope.  
- Any two functions in a set of overloaded functions must have different argument lists.  
- Overloading functions with argument lists of the same types, based on return type alone,   
is an error.   
What is Overriding?  
To override a method, a subclass of the class that originally declared the method must   
declare a method with the same name, return type (or a subclass of that return type), and   
same parameter list.  
The definition of the method overriding is:   
• Must have same method name.   
• Must have same data type.   
• Must have same argument list.   
Overriding a method means that replacing a method functionality in child class. To imply   
overriding functionality we need parent and child classes. In the child class you define the   
same method signature as one defined in the parent class.  
What is "this" pointer?  
The this pointer is a pointer accessible only within the member functions of a class,   
struct, or union type. It points to the object for which the member function is called.   
Static member functions do not have a this pointer.  
When a nonstatic member function is called for an object, the address of the object is   
passed as a hidden argument to the function. For example, the following function call  
myDate.setMonth( 3 );  
can be interpreted this way:  
setMonth( &myDate, 3 );  
The object's address is available from within the member function as the this pointer. It is   
legal, though unnecessary, to use the this pointer when referring to members of the class.  
What happens when you make call "delete this;" ??  
The code has two built-in pitfalls. First, if it executes in a member function for an   
extern, static, or automatic object, the program will probably crash as soon as the delete   
statement executes. There is no portable way for an object to tell that it was instantiated   
on the heap, so the class cannot assert that its object is properly instantiated. Second,   
when an object commits suicide this way, the using program might not know about its demise.   
As far as the instantiating program is concerned, the object remains in scope and continues   
to exist even though the object did itself in. Subsequent dereferencing of the pointer can   
and usually does lead to disaster.  
You should never do this. Since compiler does not know whether the object was allocated on   
the stack or on the heap, "delete this" could cause a disaster.  
How virtual functions are implemented C++?  
Virtual functions are implemented using a table of function pointers, called the vtable.   
There is one entry in the table per virtual function in the class. This table is created by   
the constructor of the class. When a derived class is constructed, its base class is   
constructed first which creates the vtable. If the derived class overrides any of the base   
classes virtual functions, those entries in the vtable are overwritten by the derived class   
constructor. This is why you should never call virtual functions from a constructor: because   
the vtable entries for the object may not have been set up by the derived class constructor   
yet, so you might end up calling base class implementations of those virtual functions  
What is name mangling in C++??  
The process of encoding the parameter types with the function/method name into a unique name   
is called name mangling. The inverse process is called demangling.  
For example Foo::bar(int, long) const is mangled as `bar\_\_C3Fooil'.   
For a constructor, the method name is left out. That is Foo::Foo(int, long) const is mangled   
as `\_\_C3Fooil'.  
What is the difference between a pointer and a reference?   
A reference must always refer to some object and, therefore, must always be initialized;   
pointers do not have such restrictions. A pointer can be reassigned to point to different   
objects while a reference always refers to an object with which it was initialized.  
How are prefix and postfix versions of operator++() differentiated?   
The postfix version of operator++() has a dummy parameter of type int. The prefix version   
does not have dummy parameter.  
What is the difference between const char \*myPointer and char \*const myPointer?   
Const char \*myPointer is a non constant pointer to constant data; while char \*const   
myPointer is a constant pointer to non constant data.   
How can I handle a constructor that fails?  
throw an exception. Constructors don't have a return type, so it's not possible to use   
return codes. The best way to signal constructor failure is therefore to throw an exception.   
How can I handle a destructor that fails?  
Write a message to a log-file. But do not throw an exception.   
The C++ rule is that you must never throw an exception from a destructor that is being   
called during the "stack unwinding" process of another exception. For example, if someone   
says throw Foo(), the stack will be unwound so all the stack frames between the throw Foo()   
and the } catch (Foo e) { will get popped. This is called stack unwinding.   
During stack unwinding, all the local objects in all those stack frames are destructed. If   
one of those destructors throws an exception (say it throws a Bar object), the C++ runtime   
system is in a no-win situation: should it ignore the Bar and end up in the } catch (Foo e)   
{ where it was originally headed? Should it ignore the Foo and look for a } catch (Bar e) {   
handler? There is no good answer -- either choice loses information.   
So the C++ language guarantees that it will call terminate() at this point, and terminate()   
kills the process. Bang you're dead.   
What is Virtual Destructor?  
Using virtual destructors, you can destroy objects without knowing their type - the correct   
destructor for the object is invoked using the virtual function mechanism. Note that   
destructors can also be declared as pure virtual functions for abstract classes.   
if someone will derive from your class, and if someone will say "new Derived", where   
"Derived" is derived from your class, and if someone will say delete p, where the actual   
object's type is "Derived" but the pointer p's type is your class.  
Can you think of a situation where your program would crash without reaching the breakpoint   
which you set at the beginning of main()?  
C++ allows for dynamic initialization of global variables before main() is invoked. It is   
possible that initialization of global will invoke some function. If this function crashes   
the crash will occur before main() is entered.   
Name two cases where you MUST use initialization list as opposed to assignment in   
constructors.  
Both non-static const data members and reference data members cannot be assigned values;   
instead, you should use initialization list to initialize them.   
Can you overload a function based only on whether a parameter is a value or a reference?  
No. Passing by value and by reference looks identical to the caller.   
What are the differences between a C++ struct and C++ class?  
The default member and base class access specifiers are different.   
The C++ struct has all the features of the class. The only differences are that a struct   
defaults to public member access and public base class inheritance, and a class defaults to   
the private access specifier and private base class inheritance.   
What does extern "C" int func(int \*, Foo) accomplish?  
It will turn off "name mangling" for func so that one can link to code compiled by a C   
compiler.   
How do you access the static member of a class?  
::  
What is multiple inheritance(virtual inheritance)? What are its advantages and   
disadvantages?  
Multiple Inheritance is the process whereby a child can be derived from more than one parent   
class. The advantage of multiple inheritance is that it allows a class to inherit the   
functionality of more than one base class thus allowing for modeling of complex   
relationships. The disadvantage of multiple inheritance is that it can lead to a lot of   
confusion(ambiguity) when two base classes implement a method with the same name.   
What are the access privileges in C++? What is the default access level?  
The access privileges in C++ are private, public and protected. The default access level   
assigned to members of a class is private. Private members of a class are accessible only   
within the class and by friends of the class. Protected members are accessible by the class   
itself and it's sub-classes. Public members of a class can be accessed by anyone.  
What is a nested class? Why can it be useful?  
A nested class is a class enclosed within the scope of another class. For example:  
// Example 1: Nested class  
//  
class OuterClass  
{  
class NestedClass  
{  
// ...  
};  
// ...  
};  
Nested classes are useful for organizing code and controlling access and dependencies.   
Nested classes obey access rules just like other parts of a class do; so, in Example 1, if   
NestedClass is public then any code can name it as OuterClass::NestedClass. Often nested   
classes contain private implementation details, and are therefore made private; in Example   
1, if NestedClass is private, then only OuterClass's members and friends can use   
NestedClass.  
When you instantiate as outer class, it won't instantiate inside class.  
What is a local class? Why can it be useful?  
local class is a class defined within the scope of a function -- any function, whether a   
member function or a free function. For example:  
// Example 2: Local class  
//  
int f()  
{  
class LocalClass  
{  
// ...  
};  
// ...  
};  
Like nested classes, local classes can be a useful tool for managing code dependencies.   
Can a copy constructor accept an object of the same class as parameter, instead of reference   
of the object?  
No. It is specified in the definition of the copy constructor itself. It should generate an   
error if a programmer specifies a copy constructor with a first argument that is an object   
and not a reference.  
(From Microsoft) Assume I have a linked list contains all of the alphabets from ‘A’ to ‘Z’.   
I want to find the letter ‘Q’ in the list, how does you perform the search to find the ‘Q’?  
How do you write a function that can reverse a linked-list? (Cisco System)  
void reverselist(void)  
{  
if(head==0)  
return;  
if(head->next==0)  
return;  
if(head->next==tail)  
{  
head->next = 0;  
tail->next = head;  
}  
else  
{  
node\* pre = head;  
node\* cur = head->next;  
node\* curnext = cur->next;  
head->next = 0;  
cur->next = head;  
for(; curnext!=0; )  
{  
cur->next = pre;  
pre = cur;  
cur = curnext;  
curnext = curnext->next;  
}  
curnext->next = cur;  
}  
}  
How do you find out if a linked-list has an end? (i.e. the list is not a cycle)  
You can find out by using 2 pointers. One of them goes 2 nodes each time. The second one   
goes at 1 nodes each time. If there is a cycle, the one that goes 2 nodes each time will   
eventually meet the one that goes slower. If that is the case, then you will know the   
linked-list is a cycle.  
How can you tell what shell you are running on [UNIX system](http://www.pdftutorials.com/tutorials/cpp-c-interview-questions-answers+219.html" \t "undefined)?  
You can do the Echo $RANDOM. It will return a undefined variable if you are from the   
C-Shell, just a return prompt if you are from the Bourne shell, and a 5 digit random numbers   
if you are from the Korn shell. You could also do a ps -l and look for the shell with the   
highest PID.  
What is Boyce Codd Normal form?   
A relation schema R is in BCNF with respect to a set F of functional dependencies if for all   
functional dependencies in F+ of the form a->b, where a and b is a subset of R, at least one   
of the following holds:   
• a->b is a trivial functional dependency (b is a subset of a)   
• a is a superkey for schema R   
Could you tell something about the Unix System Kernel?  
The kernel is the heart of the UNIX openrating system, it’s reponsible for controlling the   
computer’s resouces and scheduling user jobs so that each one gets its fair share of   
resources.   
What is a Make file?  
Make file is a utility in Unix to help compile large programs. It helps by only compiling   
the portion of the program that has been changed  
How do you link a C++ program to C functions?   
By using the extern "C" linkage specification around the C function declarations.   
Explain the scope resolution operator.   
Design and implement a String class that satisfies the following:  
Supports embedded nulls   
Provide the following methods (at least)   
Constructor   
Destructor   
Copy constructor   
Assignment operator   
Addition operator (concatenation)   
Return character at location   
Return substring at location   
Find substring   
Provide versions of methods for String and for char\* arguments   
Suppose that data is an array of 1000 integers. Write a single function call that will sort   
the 100 elements data [222] through data [321].  
Answer: quicksort ((data + 222), 100);  
What is a modifier?  
What is an accessor?  
  
Differentiate between a template class and class template.  
  
When does a name clash occur?  
  
Define namespace.  
  
What is the use of ‘using’ declaration.  
  
What is an Iterator class?  
  
List out some of the OODBMS available.  
  
List out some of the object-oriented methodologies.  
  
What is an incomplete type?  
  
What is a dangling pointer?  
  
Differentiate between the message and method.  
  
What is an adaptor class or Wrapper class?  
  
What is a Null object?  
  
What is class invariant?  
What do you mean by Stack unwinding?  
Define precondition and post-condition to a member function.  
  
What are the conditions that have to be met for a condition to be an invariant of the class?  
  
What are proxy objects?  
  
Name some pure object oriented languages.  
  
Name the operators that cannot be overloaded.   
  
What is a node class?  
  
What is an orthogonal base class?  
  
What is a container class? What are the types of container classes?  
  
What is a protocol class?  
  
What is a mixin class?  
  
What is a concrete class?  
  
What is the handle class?  
  
What is an action class?  
  
When can you tell that a memory leak will occur?  
What is a parameterized type?  
Differentiate between a deep copy and a shallow copy?  
What is an opaque pointer?  
What is a smart pointer?  
What is reflexive association?  
What is slicing?  
What is name mangling?  
What are proxy objects?  
What is cloning?  
Describe the main characteristics of static functions.  
Will the inline function be compiled as the inline function always? Justify.  
Define a way other than using the keyword inline to make a function inline.  
How can a '::' operator be used as unary operator?  
What is placement new?  
What do you mean by analysis and design?  
What are the steps involved in designing?  
  
What are the main underlying concepts of object orientation?  
  
  
What do u meant by "SBI" of an object?  
  
  
Differentiate persistent & non-persistent objects?  
  
  
What do you meant by active and passive objects?  
  
  
What is meant by software development method?  
  
  
What do you meant by static and dynamic modeling?  
  
  
How to represent the interaction between the modeling elements?  
  
  
Why generalization is very strong?  
  
  
Differentiate Aggregation and containment?  
Can link and Association applied interchangeably?  
  
  
What is meant by "method-wars"?  
  
  
Whether unified method and unified modeling language are same or different?  
  
  
Who were the three famous amigos and what was their contribution to the object community?  
  
  
Differentiate the class representation of Booch, Rumbaugh and UML?  
  
  
What is an USECASE? Why it is needed?  
  
  
Who is an Actor?  
  
  
What is guard condition?  
  
  
Differentiate the following notations?  
  
  
USECASE is an implementation independent notation. How will the designer give the   
implementation details of a particular USECASE to the programmer?  
  
  
Suppose a class acts an Actor in the problem [domain](http://www.pdftutorials.com/tutorials/cpp-c-interview-questions-answers+219.html" \t "undefined), how to represent it in the static   
model?  
  
  
Why does the function arguments are called as "signatures"?

**What is encapsulation??**

Containing and hiding information about an object, such as internal data structures and code. Encapsulation isolates the internal complexity of an object's operation from the rest of the application. For example, a client component asking for net revenue from a business object need not know the data's origin.

**What is inheritance?**

Inheritance allows one class to reuse the state and behavior of another class. The derived class inherits the properties and method implementations of the base class and extends it by overriding methods and adding additional properties and methods.

**What is Polymorphism??**

Polymorphism allows a client to treat different objects in the same way even if they were created from different classes and exhibit different behaviors.

You can use implementation inheritance to achieve polymorphism in languages such as C++ and Java.

Base class object's pointer can invoke methods in derived class objects.

You can also achieve polymorphism in C++ by function overloading and operator overloading.

**What is constructor or ctor?**

Constructor creates an object and initializes it. It also creates vtable for virtual functions. It is different from other methods in a class.

**What is destructor?**

Destructor usually deletes any extra resources allocated by the object.

**What is default constructor?**

Constructor with no arguments or all the arguments has default values.

**What is copy constructor?**

Constructor which initializes the it's object member variables ( by shallow copying) with another object of the same class. If you don't implement one in your class then compiler implements one for you.

for example:  
Boo Obj1(10); // calling Boo constructor

Boo Obj2(Obj1); // calling boo copy constructor  
Boo Obj2 = Obj1;// calling boo copy constructor

**When are copy constructors called?**

Copy constructors are called in following cases:   
a) when a function returns an object of that class by value  
b) when the object of that class is passed by value as an argument to a function  
c) when you construct an object based on another object of the same class  
d) When compiler generates a temporary object

**What is assignment operator?**

Default assignment operator handles assigning one object to another of the same class. Member to member copy (shallow copy)

**What are all the implicit member functions of the class? Or what are all the functions which compiler implements for us if we don't define one.??**

default ctor  
copy ctor  
assignment operator  
default destructor  
address operator

**What is conversion constructor?**

constructor with a single argument makes that constructor as conversion ctor and it can be used for type conversion.

for example:

class Boo  
{  
  public:  
    Boo( int i );  
};

Boo BooObject = 10 ; // assigning int 10 Boo object

**What is conversion operator??**

class can have a public method for specific data type conversions.

for example:  
class Boo  
{  
  double value;  
  public:  
    Boo(int i )  
    operator double()   
    {   
  return value;  
    }  
};

Boo BooObject;

double i  = BooObject; // assigning object to variable i of type double. now conversion  operator gets called to assign the value.

**What is diff between malloc()/free() and new/delete?**

malloc allocates memory for object in heap but doesn't invoke object's constructor to initiallize the object.

new allocates memory and also invokes constructor to initialize the object.

malloc() and free() do not support object semantics   
Does not construct and destruct objects   
string \* ptr = (string \*)(malloc (sizeof(string)))  
Are not safe   
Does not calculate the size of the objects that it construct   
Returns a pointer to void   
int \*p = (int \*) (malloc(sizeof(int)));  
int \*p = new int;  
Are not extensible   
new and delete can be overloaded in a class

"delete" first calls the object's termination routine (i.e. its destructor) and then releases the space the object occupied on the heap memory. If an array of objects was created using new, then delete must be told that it is dealing with an array by preceding the name with an empty []:-

Int\_t \*my\_ints = new Int\_t[10];

...

delete []my\_ints;

**what is the diff between "new" and "operator new" ?**

"operator new" works like malloc.

**What is difference between template and macro??**

There is no way for the compiler to verify that the macro parameters are of compatible types. The macro is expanded without any special type checking.

If macro parameter has a postincremented variable ( like c++ ), the increment is performed two times.

Because macros are expanded by the preprocessor, compiler error messages will refer to the expanded macro, rather than the macro definition itself. Also, the macro will show up in expanded form during debugging.

for example:

Macro:

#define min(i, j) (i < j ? i : j)

template:  
template<class T>   
T min (T i, T j)   
{   
return i < j ? i : j;  
}

**What are C++ storage classes?**

auto  
register  
static  
extern

**auto:** the default. Variables are automatically created and initialized when they are defined and are destroyed at the end of the block containing their definition. They are not visible outside that block

**register:** a type of auto variable. a suggestion to the compiler to use a CPU register for performance

**static:** a variable that is known only in the function that contains its definition but is never destroyed and retains its value between calls to that function. It exists from the time the program begins execution

**extern:** a static variable whose definition and placement is determined when all object and library modules are combined (linked) to form the executable code file. It can be visible outside the file where it is defined.

**What are storage qualifiers in C++ ?**

They are..

const  
volatile  
mutable

**Const** keyword indicates that memory once initialized, should not be altered by a program.

**volatile** keyword indicates that the value in the memory location can be altered even though nothing in the program  
code modifies the contents. for example if you have a pointer to hardware location that contains the time, where hardware changes the value of this pointer variable and not the program. The intent of this keyword to improve the optimization ability of the compiler.

**mutable** keyword indicates that particular member of a structure or class can be altered even if a particular structure variable, class, or class member function is constant.

struct data  
{  
char name[80];  
mutable double salary;  
}

const data MyStruct = { "Satish Shetty", 1000 }; //initlized by complier

strcpy ( MyStruct.name, "Shilpa Shetty"); // compiler error  
MyStruct.salaray = 2000 ; // complier is happy allowed

**What is reference ??**

reference is a name that acts as an alias, or alternative name, for a previously defined variable or an object.

prepending variable with "&" symbol makes it as reference.

for example:

int a;  
int &b = a; 

**What is passing by reference?**

Method of passing arguments to a function which takes parameter of type reference.

for example:

void swap( int & x, int & y )  
{  
 int temp = x;  
 x = y;  
 y = temp;   
}

int a=2, b=3;

swap( a, b );

Basically, inside the function there won't be any copy of the arguments "x" and "y" instead they refer to original variables a and b. so no extra memory needed to pass arguments and it is more efficient. 

**When do use "const" reference arguments in function?**

a) Using const protects you against programming errors that inadvertently alter data.  
b) Using const allows function to process both const and non-const actual arguments, while a function without const in the prototype can only accept non constant arguments.  
c) Using a const reference allows the function to generate and use a temporary variable appropriately.

**When are temporary variables created by C++ compiler?**

Provided that function parameter is a "const reference", compiler generates temporary variable in following 2 ways.

a) The actual argument is the correct type, but it isn't Lvalue

double Cube(const double & num)  
{  
  num = num \* num \* num;  
  return num;

}

double temp = 2.0;  
double value = cube(3.0 + temp); // argument is a expression and not a Lvalue;

b) The actual argument is of the wrong type, but of a type that can be converted to the correct type

long temp = 3L;  
double value = cuberoot ( temp); // long to double conversion 

**What is virtual function?**

When derived class overrides the base class method by redefining the same function, then if client wants to access redefined the method from derived class through a pointer from base class object, then you must define this function in base class as virtual function.

class parent  
{  
   void Show()   
{   
cout << "i'm parent" << endl;  
}  
};

class child: public parent  
{  
   void Show()   
{   
cout << "i'm child" << endl;  
}

};

parent \* parent\_object\_ptr = new child;

parent\_object\_ptr->show() // calls parent->show() i

now we goto virtual world...

class parent  
{  
   virtual void Show()   
{   
cout << "i'm parent" << endl;  
}  
};

class child: public parent  
{  
   void Show()   
{   
cout << "i'm child" << endl;  
}

};

parent \* parent\_object\_ptr = new child;

parent\_object\_ptr->show() // calls child->show()  

**What is pure virtual function? or what is abstract class?**

When you define only function prototype in a base class without implementation and do the complete implementation in derived class. This base class is called abstract class and client won't able to instantiate an object using this base class.

You can make a pure virtual function or abstract class this way..

class Boo  
{  
void foo() = 0;  
}

Boo MyBoo; // compilation error

**What is Memory alignment??**

The term alignment primarily means the tendency of an address pointer value to be a multiple of some power of two. So a pointer with two byte alignment has a zero in the least significant bit. And a pointer with four byte alignment has a zero in both the two least significant bits. And so on. More alignment means a longer sequence of zero bits in the lowest bits of a pointer.

**What problem does the namespace feature solve?**

Multiple providers of libraries might use common global identifiers causing a name collision when an application tries to link with two or more such libraries. The namespace feature surrounds a library's external declarations with a unique namespace that eliminates the potential for those collisions.

namespace [identifier] { namespace-body }

A namespace declaration identifies and assigns a name to a declarative region.  
The identifier in a namespace declaration must be unique in the declarative region in which it is used. The identifier is the name of the namespace and is used to reference its members.

**What is the use of 'using' declaration?**

A using declaration makes it possible to use a name from a namespace without the scope operator.

**What is an Iterator class?**

A class that is used to traverse through the objects maintained by a container class. There are five categories of iterators: input iterators, output iterators, forward iterators, bidirectional iterators, random access. An iterator is an entity that gives access to the contents of a container object without violating encapsulation constraints. Access to the contents is granted on a one-at-a-time basis in order. The order can be storage order (as in lists and queues) or some arbitrary order (as in array indices) or according to some ordering relation (as in an ordered binary tree). The iterator is a construct, which provides an interface that, when called, yields either the next element in the container, or some value denoting the fact that there are no more elements to examine. Iterators hide the details of access to and update of the elements of a container class. Something like a pointer.

**What is a dangling pointer?**

A dangling pointer arises when you use the address of an object after its lifetime is over. This may occur in situations like returning addresses of the automatic variables from a function or using the address of the memory block after it is freed.

**What do you mean by Stack unwinding?**

It is a process during exception handling when the destructor is called for all local objects in the stack between the place where the exception was thrown and where it is caught.

**Name the operators that cannot be overloaded??**

sizeof, ., .\*, .->, ::, ?:

**What is a container class? What are the types of container classes?**

A container class is a class that is used to hold objects in memory or external storage. A container class acts as a generic holder. A container class has a predefined behavior and a well-known interface. A container class is a supporting class whose purpose is to hide the topology used for maintaining the list of objects in memory. When a container class contains a group of mixed objects, the container is called a heterogeneous container; when the container is holding a group of objects that are all the same, the container is called a homogeneous container.

**What is inline function??**

The \_\_inline keyword tells the compiler to substitute the code within the function definition for every instance of a function call. However, substitution occurs only at the compiler's discretion. For example, the compiler does not inline a function if its address is taken or if it is too large to inline.

**What is overloading??**

With the C++ language, you can overload functions and operators. Overloading is the practice of supplying more than one definition for a given function name in the same scope.

- Any two functions in a set of overloaded functions must have different argument lists.  
- Overloading functions with argument lists of the same types, based on return type alone, is an error.

**What is Overriding?**

To override a method, a subclass of the class that originally declared the method must declare a method with the same name, return type (or a subclass of that return type), and same parameter list.  
The definition of the method overriding is:   
· Must have same method name.    
· Must have same data type.    
· Must have same argument list.    
Overriding a method means that replacing a method functionality in child class. To imply overriding functionality we need parent and child classes. In the child class you define the same method signature as one defined in the parent class.

**What is "this" pointer?**

The this pointer is a pointer accessible only within the member functions of a class, struct, or union type. It points to the object for which the member function is called. Static member functions do not have a this pointer.

When a nonstatic member function is called for an object, the address of the object is passed as a hidden argument to the function. For example, the following function call

myDate.setMonth( 3 );

can be interpreted this way:

setMonth( &myDate, 3 );

The object's address is available from within the member function as the this pointer. It is legal, though unnecessary, to use the this pointer when referring to members of the class.

**What happens when you make call "delete this;" ??**

The code has two built-in pitfalls. First, if it executes in a member function for an extern, static, or automatic object, the program will probably crash as soon as the delete statement executes. There is no portable way for an object to tell that it was instantiated on the heap, so the class cannot assert that its object is properly instantiated. Second, when an object commits suicide this way, the using program might not know about its demise. As far as the instantiating program is concerned, the object remains in scope and continues to exist even though the object did itself in. Subsequent dereferencing of the pointer can and usually does lead to disaster.

You should never do this. Since compiler does not know whether the object was allocated on the stack or on the heap, "delete this" could cause a disaster.

**How virtual functions are implemented C++?**

Virtual functions are implemented using a table of function pointers, called the vtable.  There is one entry in the table per virtual function in the class.  This table is created by the constructor of the class.  When a derived class is constructed, its base class is constructed first which creates the vtable.  If the derived class overrides any of the base classes virtual functions, those entries in the vtable are overwritten by the derived class constructor.  This is why you should never call virtual functions from a constructor: because the vtable entries for the object may not have been set up by the derived class constructor yet, so you might end up calling base class implementations of those virtual functions

**What is name mangling in C++??**

The process of encoding the parameter types with the function/method name into a unique name is called name mangling. The inverse process is called demangling.

For example Foo::bar(int, long) const is mangled as `bar\_\_C3Fooil'.   
For a constructor, the method name is left out. That is Foo::Foo(int, long) const is mangled as `\_\_C3Fooil'.

**What is the difference between a pointer and a reference?**

A reference must always refer to some object and, therefore, must always be initialized; pointers do not have such restrictions. A pointer can be reassigned to point to different objects while a reference always refers to an object with which it was initialized.

**How are prefix and postfix versions of operator++() differentiated?**

The postfix version of operator++() has a dummy parameter of type int. The prefix version does not have dummy parameter.

**What is the difference between const char \*myPointer and char \*const myPointer?**

Const char \*myPointer is a non constant pointer to constant data; while char \*const myPointer is a constant pointer to non constant data.

**How can I handle a constructor that fails?**

throw an exception. Constructors don't have a return type, so it's not possible to use return codes. The best way to signal constructor failure is therefore to throw an exception.

**How can I handle a destructor that fails?**

Write a message to a log-file. But do not throw an exception.   
The C++ rule is that you must never throw an exception from a destructor that is being called during the "stack unwinding" process of another exception. For example, if someone says throw Foo(), the stack will be unwound so all the stack frames between the throw Foo() and the } catch (Foo e) { will get popped. This is called stack unwinding.   
During stack unwinding, all the local objects in all those stack frames are destructed. If one of those destructors throws an exception (say it throws a Bar object), the C++ runtime system is in a no-win situation: should it ignore the Bar and end up in the } catch (Foo e) { where it was originally headed? Should it ignore the Foo and look for a } catch (Bar e) { handler? There is no good answer -- either choice loses information.   
So the C++ language guarantees that it will call terminate() at this point, and terminate() kills the process. Bang you're dead.

**What is Virtual Destructor?**

Using virtual destructors, you can destroy objects without knowing their type - the correct destructor for the object is invoked using the virtual function mechanism. Note that destructors can also be declared as pure virtual functions for abstract classes.

if someone will derive from your class, and if someone will say "new Derived", where "Derived" is derived from your class, and if someone will say delete p, where the actual object's type is "Derived" but the pointer p's type is your class.

**Can you think of a situation where your program would crash without reaching the breakpoint which you set at the beginning of main()?**

C++ allows for dynamic initialization of global variables before main() is invoked. It is possible that initialization of global will invoke some function. If this function crashes the crash will occur before main() is entered.

**Name two cases where you MUST use initialization list as opposed to assignment in constructors.**

Both non-static const data members and reference data members cannot be assigned values; instead, you should use initialization list to initialize them.

**Can you overload a function based only on whether a parameter is a value or a reference?**

No. Passing by value and by reference looks identical to the caller.

**What are the differences between a C++ struct and C++ class?**

The default member and base class access specifiers are different.

The C++ struct has all the features of the class. The only differences are that a struct defaults to public member access and public base class inheritance, and a class defaults to the private access specifier and private base class inheritance.

**What does extern "C" int func(int \*, Foo) accomplish?**

It will turn off "name mangling" for func so that one can link to code compiled by a C compiler.

**How do you access the static member of a class?**

<ClassName>::<StaticMemberName>

**What is multiple inheritance(virtual inheritance)? What are its advantages and disadvantages?**

Multiple Inheritance is the process whereby a child can be derived from more than one parent class. The advantage of multiple inheritance is that it allows a class to inherit the functionality of more than one base class thus allowing for modeling of complex relationships. The disadvantage of multiple inheritance is that it can lead to a lot of confusion(ambiguity) when two base classes implement a method with the same name.

**What are the access privileges in C++? What is the default access level?**

The access privileges in C++ are private, public and protected. The default access level assigned to members of a class is private. Private members of a class are accessible only within the class and by friends of the class. Protected members are accessible by the class itself and it's sub-classes. Public members of a class can be accessed by anyone.

**What is a nested class? Why can it be useful?**

A nested class is a class enclosed within the scope of another class. For example:

  //  Example 1: Nested class  
  //  
  class OuterClass  
  {  
    class NestedClass  
    {  
      // ...  
    };  
    // ...  
  };  
Nested classes are useful for organizing code and controlling access and dependencies. Nested classes obey access rules just like other parts of a class do; so, in Example 1, if NestedClass is public then any code can name it as OuterClass::NestedClass. Often nested classes contain private implementation details, and are therefore made private; in Example 1, if NestedClass is private, then only OuterClass's members and friends can use NestedClass.

When you instantiate as outer class, it won't instantiate inside class.

**What is a local class? Why can it be useful?**

local class is a class defined within the scope of a function -- any function, whether a member function or a free function. For example:

  //  Example 2: Local class  
  //  
  int f()  
  {  
    class LocalClass  
    {  
      // ...  
    };  
    // ...  
  };  
Like nested classes, local classes can be a useful tool for managing code dependencies. 

**Can a copy constructor accept an object of the same class as parameter, instead of reference of the object?**

No. It is specified in the definition of the copy constructor itself. It should generate an error if a programmer specifies a copy constructor with a first argument that is an object and not a reference.