What is C++?

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

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

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

What are the advantages of C++?

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

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

What are the different data types present in C++?

- Primitive Datatype(basic datatype). Example- char, short, int, float, long, double, bool, etc.
- Derived datatype. Example- array, pointer, etc.
- Enumeration. Example- enum
- User-defined data types. Example- structure, class, etc.

What is the difference between C and C++?

C	C++
C is a procedure-oriented programming language.	C++ is an object-oriented programming language.
C does not support data hiding.	Data is hidden by encapsulation to ensure that data structures and operators are used as intended.
C is a subset of C++	C++ is a superset of C.
Function and operator overloading are not supported in C	Function and operator overloading is supported in C++
Namespace features are not present in C	Namespace is used by C++, which avoids name collisions.
Functions can not be defined inside structures.	Functions can be defined inside structures.
calloc() and malloc() functions are used for memory allocation and free() function is used for memory deallocation.	new operator is used for memory allocation and deletes operator is used for memory deallocation.

What is a class?

The class is a user-defined data type. The class is declared with the keyword class. The class contains the data members, and member functions whose access is defined by the three modifiers are private, public and protected. The class defines the type definition of the category of things. It defines a datatype, but it does not define the data it just specifies the structure of data.

What is the difference between reference and pointer?

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

Object:

An object is a run-time entity. An object is the instance of the class. An object can represent a person, place or any other item. An object can operate on both data members and member functions. The class does not occupy any memory space. When an object is created using a new keyword, then space is allocated for the variable in a heap, and the starting address is stored in the stack memory. When an object is created without a new keyword, then space is not allocated in the heap memory, and the object contains the null value in the stack.

```
class Student
{
//data members;
//Member functions
}
```

The syntax for declaring the object:

```
Student s = new Student();

```

Inheritance:

Inheritance provides reusability. Reusability means that one can use the functionalities of the existing class. It eliminates the redundancy of code. Inheritance is a technique of deriving a new class from the old class. The old class is known as the base class, and the new class is known as derived class.

Syntax

```
class derived_class :: visibility-mode base_class;
```

Note: The visibility-mode can be public, private, protected.

Encapsulation:

Encapsulation is a technique of wrapping the data members and member functions in a single unit. It binds the data within a class, and no outside method can access the data. If the data member is private, then the member function can only access the data.

Abstraction:

Abstraction is a technique of showing only essential details without representing the implementation details. If the members are defined with a public keyword, then the

members are accessible outside also. If the members are defined with a private keyword, then the members are not accessible by the outside methods.

Data binding:

Data binding is a process of binding the application UI and business logic. Any change made in the business logic will reflect directly to the application UI.

Polymorphism:

Polymorphism means multiple forms. Polymorphism means having more than one function with the same name but with different functionalities. Polymorphism is of two types:

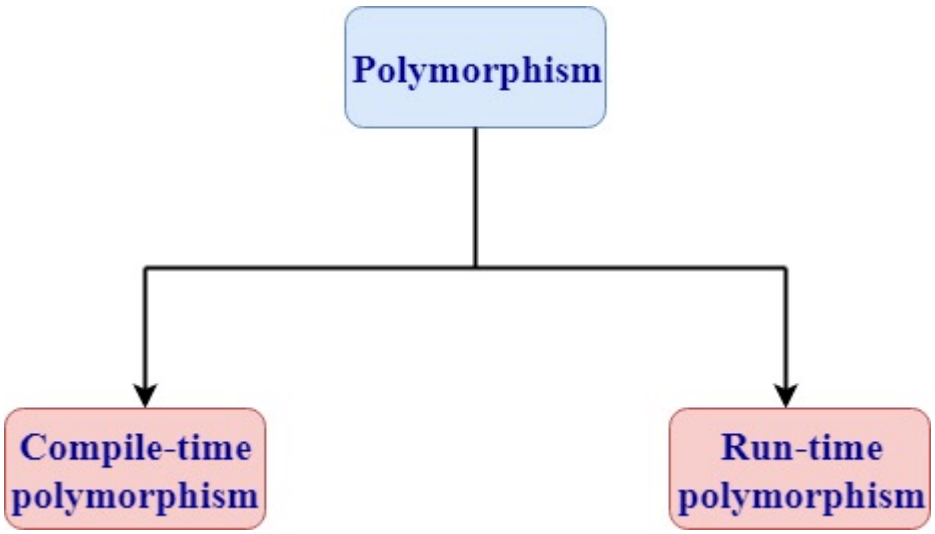
1. Static polymorphism is also known as early binding.
2. Dynamic polymorphism is also known as late binding.

---

7) What are the different types of polymorphism in C++?

Polymorphism: Polymorphism means multiple forms. It means having more than one function with the same function name but with different functionalities.

Polymorphism is of two types:



-
-

Runtime polymorphism

Runtime polymorphism is also known as dynamic polymorphism. Function overriding is an example of runtime polymorphism. Function overriding means when the child class contains the method which is already present in the parent class. Hence, the child class overrides the method of the parent class. In case of function overriding, parent and child class both contains the same function with the different definition. The call to the function is determined at runtime is known as runtime polymorphism.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. class Base
4. {
5.     public:
6.     virtual void show()
7.     {
8.         cout<<"javaTpoint";
9.      }
10. };
11. class Derived:public Base
12. {
13.     public:
14.     void show()
15.     {
16.         cout<<"javaTpoint tutorial";
17.     }
18. };
19.
20. int main()
21. {
22.     Base* b;
23.     Derived d;
24.     b=&d;
25.     b->show();
26.                 return 0;
27. }
```

Output:

```
javaTpoint tutorial
```

Compile time polymorphism

Compile-time polymorphism is also known as static polymorphism. The polymorphism which is implemented at the compile time is known as compile-time polymorphism. Method overloading is an example of compile-time polymorphism.

Method overloading: Method overloading is a technique which allows you to have more than one function with the same function name but with different functionality.

Method overloading can be possible on the following basis:

- The return type of the overloaded function.
- The type of the parameters passed to the function.
- The number of parameters passed to the function.

Define namespace in C++.

- The namespace is a logical division of the code which is designed to stop the naming conflict.
- The namespace defines the scope where the identifiers such as variables, class, functions are declared.
- The main purpose of using namespace in C++ is to remove the ambiguity. Ambiquity occurs when the different task occurs with the same name.

Define token in C++.

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

Who was the creator of C++?

Bjarne Stroustrup

Define 'std'.

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

delete [] is different from delete?

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

What are the C++ access specifiers?

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

There are three types of access specifiers:

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

What is the difference between an array and a list?

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

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

- new() is a preprocessor while malloc() is a function.
- There is no need to allocate the memory while using "new" but in malloc() you have to use sizeof().

Define friend function.

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

of the class. Sometimes, it is necessary for the non-member function to access the data. The friend function is a non-member function and has the ability to access the private data of the class.

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

```
class sample
{
   // data members;
 public:
friend void abc(void);
};
```

Following are the characteristics of a friend function:

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

Let's understand this through an example:

```
#include <iostream>
using namespace std;
class Addition
{
```

```
5.  int a=5;
6.  int b=6;
7.  public:
8.  friend int add(Addition a1)
9.  {
10.     return(a1.a+a1.b);
11. }
12. };
13. int main()
14. {
15. int result;
16. Addition a1;
17.  result=add(a1);
18.  cout<<result;
19. return 0;
20. }
```

Output:

```
11
```

What is a virtual function?

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

Rules of a virtual function:

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

---

24) When should we use multiple inheritance?

You can answer this question in three manners:

1. Never
2. Rarely
3. If you find that the problem domain cannot be accurately modeled any other way.

---

25) What is a destructor?

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

Rules of destructor:

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

---

26) What is an overflow error?

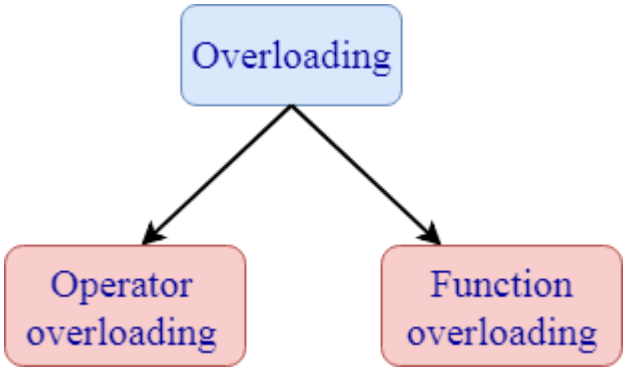
It is a type of arithmetical error. It happens when the result of an arithmetical operation been greater than the actual space provided by the system.

---

27) What is overloading?

- When a single object behaves in many ways is known as overloading. A single object has the same name, but it provides different versions of the same function.

- C++ facilitates you to specify more than one definition for a function name or an operator in the same scope. It is called function overloading and operator overloading respectively.
- **Overloading is of two types:**



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

Operator overloading can be implemented in the following functions:

- Member function
- Non-member function
- Friend function

Syntax of Operator overloading:

```
Return_type classname :: Operator Operator_symbol(argument_list)
{
    // body_statements;
}
```

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

function to be invoked depends on the number of arguments and the type of the arguments in the argument list.

---

28) What is function overriding?

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

---

29) What is virtual inheritance?

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

What is a constructor?

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

What is the purpose of the "delete" operator?

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

Explain this pointer?

This pointer holds the address of the current object.

What does Scope Resolution operator do?

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

What is the difference between delete and delete[]?

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

What is a pure virtual function?

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

Syntax of a pure virtual function:

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

Let's understand this through an example:

```
#include<iostream>
using namespace std;
class Base
{
    public:
```

```
6.      virtual void show()=0;
7.  };
8.
9.  class Derived:public Base
10. {
11.     public:
12.     void show()
13.     {
14.         cout<<"javaTpoint";
15.     }
16. };
17. int main()
18. {
19.     Base* b;
20.     Derived d;
21.     b=&d;
22.     b->show();
23.     return 0;
24. }
```

Output:

```
javaTpoint
```

What is the difference between struct and class?

Structures	class
A structure is a user-defined data type which contains variables of dissimilar data types.	The class is a user-defined data type wh variables and member functions.
The variables of a structure are stored in the stack memory.	The variables of a class are stored in the
We cannot initialize the variables directly.	We can initialize the member variables d
If access specifier is not specified, then by default the access specifier of the variable is "public".	If access specifier is not specified, then specifier of a variable is "private".

The instance of a structure is a "structure variable".	

Declaration of a structure:
`struct structure_name`
`{`
`  // body of structure;`
`} ;`
	Declaration of class:
	`class class_name`
	`{`
	`  // body of class;`
	`}`

A structure is declared by using a struct keyword.	The class is declared by using a class ke
The structure does not support the inheritance.	The class supports the concept of inheri
The type of a structure is a value type.	The type of a class is a reference type.

What is a class template?

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

The syntax of a class template:

```
template<class T>
class classname
{
  // body of class;
};
```

Syntax of a object of a template class:

1. classname<type> objectname(arglist);

What is the difference between function overloading and operator overloading?

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

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

What is a virtual destructor?

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

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

Let's understand this through an example

- Example without using virtual destructor

```
#include <iostream>
using namespace std;
class Base
{
   public:
   Base()
   {
      cout<<"Base constructor is called"<<"\n";
   }
```

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

Output:

```
Base constructor is called
Derived class constructor is called
Base class object is destroyed
```

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

- Example with a virtual destructor

```
1.  #include <iostream>
2.  using namespace std;
3.  class Base
4.  {
5.      public:
6.      Base()
7.      {
8.          cout<<"Base constructor is called"<<"\n";
9.      }
10.     virtual ~Base()
11.     {
12.         cout<<"Base class object is destroyed"<<"\n";
13.     }
14. };
15. class Derived:public Base
16. {
17.     public:
18.     Derived()
19.     {
20.         cout<<"Derived class constructor is called"<<"\n";
21.     }
22.     ~Derived()
23.     {
24.         cout<<"Derived class object is destroyed"<<"\n";
25.     }
26. };
27. int main()
28. {
29.   Base* b= new Derived;
30.   delete b;
31.   return 0;
32.
33. }
```

Output:

```
Base constructor is called
Derived class constructor is called
Derived class object is destroyed
```

```
Base class object is destroyed
```

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

What is a reference in C++?

A reference is like a pointer. It is another name of an already existing variable. Once a reference name is initialized with a variable, that variable can be accessed by the variable name or reference name both.

For example-

```
int x=10;
int &ref=x;            //reference variable
```

If we change the value of ref it will be reflected in x. Once a reference variable is initialized it cannot refer to any other variable. We can declare an array of pointers but an array of references is not possible.

What do you mean by call by value and call by reference?

In call by value method, we pass a copy of the parameter is passed to the functions. For these copied values a new memory is assigned and changes made to these values do not reflect the variable in the main function.

In call by reference method, we pass the address of the variable and the address is used to access the actual argument used in the function call. So changes made in the parameter alter the passing argument.

What is an abstract class and when do you use it?

A class is called an abstract class whose objects can never be created. Such a class exists as a parent for the derived classes. We can make a class abstract by placing a pure virtual function in the class.

What are the static members and static member functions?

When a variable in a class is declared static, space for it is allocated for the lifetime of the program. No matter how many objects of that class have been created, there is only one copy of the static member. So same static member can be accessed by all the objects of that class.

A static member function can be called even if no objects of the class exist and the static function are accessed using only the class name and the scope resolution operator :

What is a copy constructor?

A copy constructor is a member function that initializes an object using another object of the same class.

Example-

```
class A{
int x,y;
A(int x, int y){
 this->x=x;
 this->y=y;
}

};
int main(){
A a1(2,3);
A a2=a1;     //default copy constructor is called
return 0;
}
```

We can define our copy constructor. If we don't define a copy constructor then the default copy constructor is called.

What is the difference between shallow copy and deep copy?

The difference between shallow copy and a deep copy is given below:

Shallow Copy	Deep Copy
Shallow copy stores the references of objects to the original memory address.	Deep copy makes a new and separate copy of an entire object with its unique memory address.

Shallow Copy	Deep Copy
Shallow copy is faster.	Deep copy is comparatively slower.
Shallow copy reflects changes made to the new/copied object in the original object.	Deep copy doesn't reflect changes made to the new/copied object in the original object

What is the difference between virtual functions and pure virtual functions?

A virtual function is a member function in the base class that you redefine in a derived class. It is declared using the virtual keyword.

Example-

```
class base{
public:
 virtual void fun(){

 }
};
```

A pure virtual function is a function that has no implementation and is declared by assigning 0. It has no body.

Example-

```
class base{
public:
 virtual void fun()=0;
};
```

Here, = sign has got nothing to do with the assignment, and value 0 is not assigned to anything. It is used to simply tell the compiler that a function will be pure and it will not have anybody.

Can we call a virtual function from a constructor?

Yes, we can call a virtual function from a constructor. But the behavior is a little different in this case. When a virtual function is called, the virtual call is resolved at runtime. It is always the member function of the current class that gets called. That is the virtual machine doesn't work within the constructor.

For example-

```
class base{
 private:
  int value;
 public:
  base(int x){
   value=x;
  }
  virtual void fun(){

  }
}

class derived{
 private:
  int a;
 public:
  derived(int x, int y):base(x){
   base *b;
   b=this;
   b->fun();       //calls derived::fun()
  }
  void fun(){
   cout<<"fun inside derived class"<<endl;
  }
}
```

What are void pointers?

A void pointer is a pointer which is having no datatype associated with it. It can hold addresses of any type.

For example-

```
void *ptr;
char *str;
p=str;                  // no error
str=p;                  // error because of type mismatch
```

We can assign a pointer of any type to a void pointer but the reverse is not true unless you typecast it as

```
str=(char*) ptr;
```

What is this pointer in C++?

The member functions of every object have a pointer named this, which points to the object itself. The value of this is set to the address of the object for which it is called. It can be used to access the data in the object it points to.

Example

```
class A{
 private:
  int value;
 public:
  void setvalue(int x){
   this->value=x;
  }
};

int main(){
 A a;
 a.setvalue(5);
 return 0;
}
```

How do you allocate and deallocate memory in C++?

The new operator is used for memory allocation and deletes operator is used for memory deallocation in C++.

For example-

```
int value=new int;              //allocates memory for storing 1 integer
delete value;                           // deallocates memory taken by value

int *arr=new int[10];           //allocates memory for storing 10 int
delete []arr;                   // deallocates memory occupied by arr
```

What is an inline function?

An inline function when called expands in line. When you call this function, the whole code of the inline function gets inserted or substituted at the inline function call.

Syntax:

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

{

}
```

What is the block scope variable in C++?

A variable whose scope is applicable only within a block is said so. Also a variable in C++ can be declared anywhere within the block.

What is the scope resolution operator?

The scope resolution operator is used to

- Resolve the scope of global variables.
- To associate function definition to a class if the function is defined outside the class.
-

Where an automatic variable is stored?

Every local variable by default being an auto variable is stored in stack memory

What is a container class?

A class containing at least one member variable of another class type in it is called so.