1. What are the difference between C and C++ ?

C language	C++ language
1. C is a structure programming language	1. C++ is a Object Oriented programming language
2. C language is founded by Dennis Ritchie	2. C++ language is founded by Bjarne Stroustrup
3. C is a subset language.	3. C++ is a superset language.
4. C language is top down approach	4. C++ language is bottom up approach
5. C language is easily manipulated by other	5. C++ language is not easily manipulated by other.
6. C language is consist of 32 keywords	6. C++ language is consist of 52 keyword.
7. C language use printf, scanf	7. C++ language use Cin, Cup.

2. What are the class and object ?

⇒ Class :-

Class is a blueprint for objects. A class is a user-defined type that describes what a certain type of object will look like. A class description consists of a declaration and a defination. Usually these pieces are split into separate file.

An object is a single instance of a class. You can create many objects from the same type.

Object :-

Object is an encapsulation of data along with function that act upon that data.

An object consist of :-

- Name - the variable name we gave it.
- Member data - the data that described the object.
- Member functions - behavior aspects of the object (function related to the object itself).

3. What is Operator Overloading ?

In C++, we can make Operators work for user-defined classes. This means C++ has the ability to provide the Operators with a special meaning for a data type. This ability is known as Operator Overloading. For example, we can overload an Operator '+' in a class like String so that we can concentenate two Strings by just using +. Other example classes where arithemetic Operator may be overloaded are Complex Numbers, Fractional Numbers, Big Integers, etc.

Operator Overloading is a Compile time polymorphism. It is an idea of giving special meaning to an existing Operator in C++ without changing their Original meaning.

Example :-

```
int a;
float b, sum;
sum = a+b;
```

1. What is polymorphism in C++ ?

The word "polymorphism" means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. A real-life example of polymorphism is a person who at the same time can have different characteristics. Like a man at the same time is a father, a husband and an employee. So the same person exhibits different behaviour in different situation. This is called polymorphism. Polymorphism is considered as one of the important features of Object-Oriented Programming. In C++, polymorphism is mainly divided into two types:

- Compile-time Polymorphism.
- Runtime Polymorphism.

5. Explain Constructor in C++ ?

Constructor is a member function of class, whose name is same as the class.

A Constructor is a special type of member function of a class which initializes object of a class. In C++, Constructor is automatically called when object (instance of class) is created.

Constructor is invoked at the time of object creation. It constructs the values i.e provides data for the object that is why it is known as Constructors

Constructor does not have a return value, hence they do not a return type. Constructors can be defined inside or outside the class declaration.

1. Syntax for defining the constructor within the class:-

```
<class-name> ( list-of-parameters)
{
  // Constructor definition.
}
```

2. Syntax for defining the Constructor outside the class:-

```
<class-name>::<class-name> (list-of-parameters)
{
    // constructor definition
}
```

6. What are the C++ access specifiers?

The public keyword is an access specifier. Access specifiers define how the members (attributes and methods) of class can be accessed. In the example above, the members are public - which means that they can be accessed and modified from outside the code.

In C++, there are three access specifiers :-

- public - members are accessible from outside the class.
- private - members cannot be accessed (or viewed) from outside the class.
- protected - members cannot be accessed from outside the class, however, they can be accessed in inherited classes. You will learn more about Inheritance later.

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

Call by value mean calling a methods with a parameter as value. Through this, the argument value is passed to the parameter.

Call by Reference mean calling a method with a parameter as a reference. Through this, the argument reference is passed to the parameter.

In call by value, the modification done to the parameter passed does not reflect in the Caller's Scope while in the call by reference, the modification done to the parameter passed are permitant and changes are reflected in the caller's scope.

Example of call by Value :-

```
void main()
{
  int a = 10,
  void increment (int);
  Cout << " before function
calling " << a;
  increment (a);
  Cout << " after function
  calling " << a;
getch ();
}
  void increment (int x)
{
  int n = x + 1;
Cout << "Value is " << n;
}
```

Example of call by Reference :-

```
Public static void main (String
    args [])
  {  int a = 10;
   System.Out.println ("Before
call Value of a = ", a);
    void increment ();
  System.Out.println ("After
  Call Value of a = ", a);
}
  void increment (int n)
   {  int x = x + 1;
   }
```

8. Explain Inheritance ?

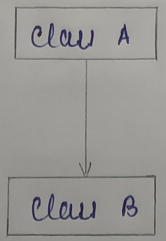
Inheritance is a mechanism in which one class acquires the property of another class. For example, a child inherits the traits of his / her parents. With inheritance we can reuse the fields and methods of the fields and methods of existing class. Hence, inheritance facilitates Reusability and is an important concepts of OOPs.

There are 5 types of inheritance in C++

1. Single Inheritance
2. Multiple Inheritance
3. Multilevel Inheritance
4. Hierarchical Inheritance
5. Hybrid Inheritance.

- Single Inheritance :-

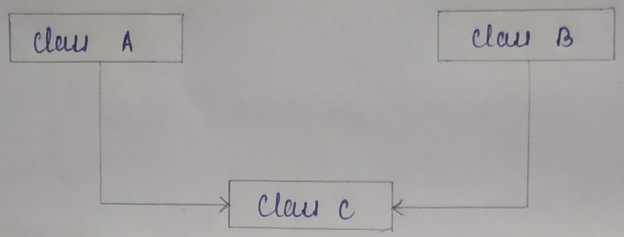
In single inheritance one class extends another class (one class only).



In above diagram, class B extends only class A. class A is a super class and class B is a sub-class.

- Multiple Inheritance :-

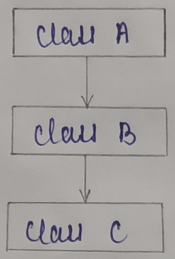
Multiple inheritance is one of the inheritance in C++ type where one class extending more than one class.



As per above diagram, class C extends class A and class B both.

- Multilevel Inheritance :-

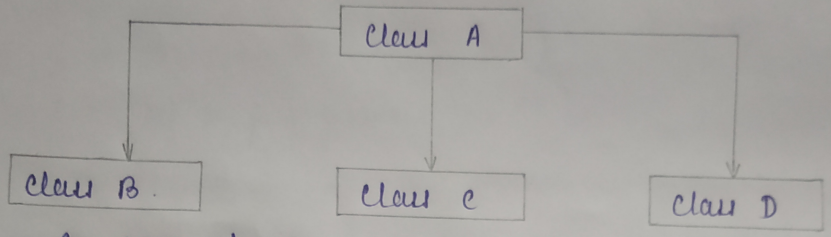
In Multilevel inheritance, one class can inherit from a derived class. Hence, the derived class becomes the base class for the new class.



As per shown in diagram class C is subclass of B and B is a of subclass class A.

Hierarchical Inheritance :-

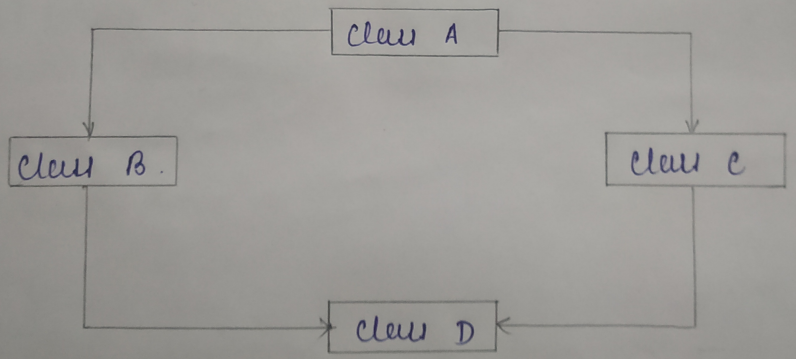
In hierarchical inheritance, one class is inherited by many sub classes.



As per above example, class B, C, and D inherit the same class A.

- ## Hybrid Inheritance :-

Hybrid inheritance is one of the inheritance type in C++ which is a combination of single and Multiple inheritance.



As per the above example, all the public and protected members of class A are inherited into class D, first via class B and secondly via class C.

9. What do you mean by exception handling?

Exception handling is the process of responding to unwanted or unexpected events when a computer program runs. Exception handling deals with these events to avoid the program or system crashing, and without this process, exceptions would disrupt the normal operation of a program.

Exception occurs for numerous reasons, including invalid user input, code errors, device failure, the loss of a network connection, insufficient memory to run an application, a memory conflict with another program, a program attempting to divide by zero or a user attempting to open files that are unavailable.

When an exception occurs, specialized programming language constructs, interrupt hardware mechanisms or operating system interprocess communication facilities handle the exception.

Example :-

- Null Pointer Exception :-

Null Pointer Exception is an unchecked exception that occurs when a user tries to access an object using a reference variable that is null or empty.

What is virtual function and pure virtual function?

Virtual Function :-

A virtual function is a member function in the base class that we expect to redefine in derived classes. Basically, a virtual function is used in the base class in order to ensure that the function is overridden. This especially applies to cases where a pointer of base class points to an object of a derived class. A virtual function is a member function of base class which can be redefined by derived class.

Pure Virtual Function :-

A pure virtual function or pure virtual method is a virtual function that is required to be implemented by a derived class if the derived class is not abstract. classes containing pure virtual methods are termed "abstract" and they cannot be instantiated directly. A pure virtual function is a member function of base class where only declaration is provided in base class and should be defined in derived class otherwise derived class also becomes abstract.