# C++ Short question and Answer:

## 1.Copy constructor

The **copy constructor** is a **constructor** which creates an object by initializing it with an object of the same class, which has been created previously.

The **copy constructor** is used to: Initialize one object from another of the same type.

**Copy** an object to pass it as an argument to a function.

## 2.Static data member and function

Classes can contain static member data and member functions.

When a data member is declared as **static**, only one copy of the data is maintained for all objects of the class.

## 3.Manipulators

Manipulators are the operators that are used to format the data display.

Endl()

Setw()

Setprecision()

Setfill()



## 4.Protected access specifier

Protected - The members declared as Protected are accessible from outside the class BUT only in a class derived from it.

*Protected* are inaccessible outside the class but they can be accessed by any subclass(derived class) of that class.

## 5.this poiner

Every object in **C++** has access to its own address through an important **pointer** called this **pointer**.

The this **pointer** is an implicit parameter to all member functions.

Therefore, inside a member function, this may be **used** to refer to the invoking object.

## 6.Containership

When a class contains objects of another class or its members, this kind of relationship is called **containership** or nesting and the class which contains objects of another class as its members is called as container class.

Syntax for the declaration of another class is: Class class_name1. { ———

## 7.Scope resolution operator

*Scope resolution operator*(::) is used to define a function outside a class or when we want to use a global variable but also has a local variable with same name.

The :: (**scope resolution**) **operator** is used to qualify hidden names so that you can still use them.

You can use the unary **scope operator** if a namespace **scope** or global **scope** name is hidden by an explicit declaration of the same name in a block or class.

## 8.Friend function

A **friend function** of a class is defined outside that class' scope but it has the right to access all private and protected members of the class.

Even though the prototypes for **friend functions** appear in the class definition, **friends** are not member **functions**.



## 9.Inline function

**C++ inline function** is powerful concept that is commonly used with classes.

If a **function** is **inline**, the compiler places a copy of the code of that **function** at each point where the **function** is called at compile time.

*Inline function* is a function that is expanded in line when it is called.

At the time of declaration or definition, function name is preceded by word inline.

When inline functions are used, the overhead of function call is eliminated. Instead, the executable statements of the function are copied at the place of each function call.

This is done by the compiler.

Consider the following example :

```
#include <iostream>
using namespace std;

inline int sqr(int x)
{
  int y;
  y = x * x;
  return y;
}
int main()
{
  int a =3, b;
  b = sqr(a);
  cout <<b;
  return 0;
}
```

## 10.Function overloading

**Function overloading** (also **method overloading**) is a **programming** concept that allows programmers to define two or more **functions** with the same name and in the same scope.

Each **function** has a unique signature (or header), which is derived from: **function**/procedure name. number of arguments. arguments' type.



## 11.Exceptin handling

A **C++ exception** is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

**Exceptions** provide a way to transfer control from one part of a program to another.

**C++ exception handling** is built upon three keywords: try, **catch**, and throw.

### 1 .Try block

Try block is basically a code block which surrounds the code where there is a chance of occurrence of error or exceptions. We enclose that piece of code within the try block. Syntax of catch block is very simple
Code:

```
try
{
        result = numerator/denominator;
        cout<<"\nThe result of division is:" <<result;
}
```

### 2. Catch Block

Catch block executes only if an exception occurs inside the try block. When an exception occurs, a message is passed to the corresponding catch block along with some parameters. After, exception occurs, the statement in the corresponding catch block.

Code:

```
catch(int num)
{
        cout<<"You cannot enter "<<num<<" in denominator";
}
```

**3. Throw Statement**

We have discussed that we pass a message from the catch block to try block that an exception has occurred and you will have to catch this exception.

But how we send this message? We send this message from the try block to a catch block via special statement called 'throw'.

Throw statement basically throws the exception to catch block and passes the parameter to catch block.

```
try {
        if(denominator == 0) {
                throw denominator;
        }
        result = numerator/denominator;
        cout<<"\nThe result of division is:" <<result;
}
```



## 12.Operator overloading

**Operator overloading** (less commonly known as ad-hoc polymorphism) is a specific case of polymorphism (part of the OO nature of the language) in which some or all **operators** like +, = or == are treated as polymorphic functions and as such have different behaviors depending on the types of its arguments.

## 13.Overloading and 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.

## 13.Inheritance

**Inheritance** in Object Oriented Programming can be described as a process of creating new classes from existing classes.

New classes **inherit** some of the properties and behavior of the existing classes.
An existing class that is "parent" of a new class is called a base class. ... **Inheritance** is a technique of code.

## 14.Constructor

A constructor is a member function of a class which initializes objects of a class.

In C++,Constructor is automatically called when object(instance of class) create.It is special member function of the class.

**How constructors are different from a normal member function?**

A constructor is different from normal functions in following ways:

- Constructor has same name as the class itself
- Constructors don’t have return type
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).