**Make Up Test solution:**

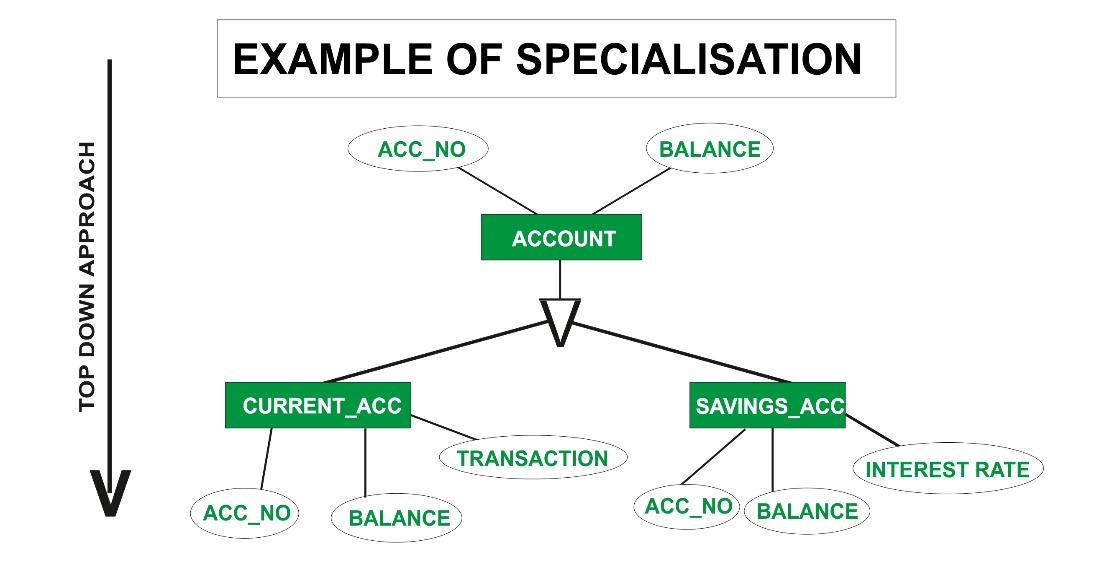
**1.a** Contrast on generalization and specialization with the help of suitable example and discover the role of link and association in object modelling.

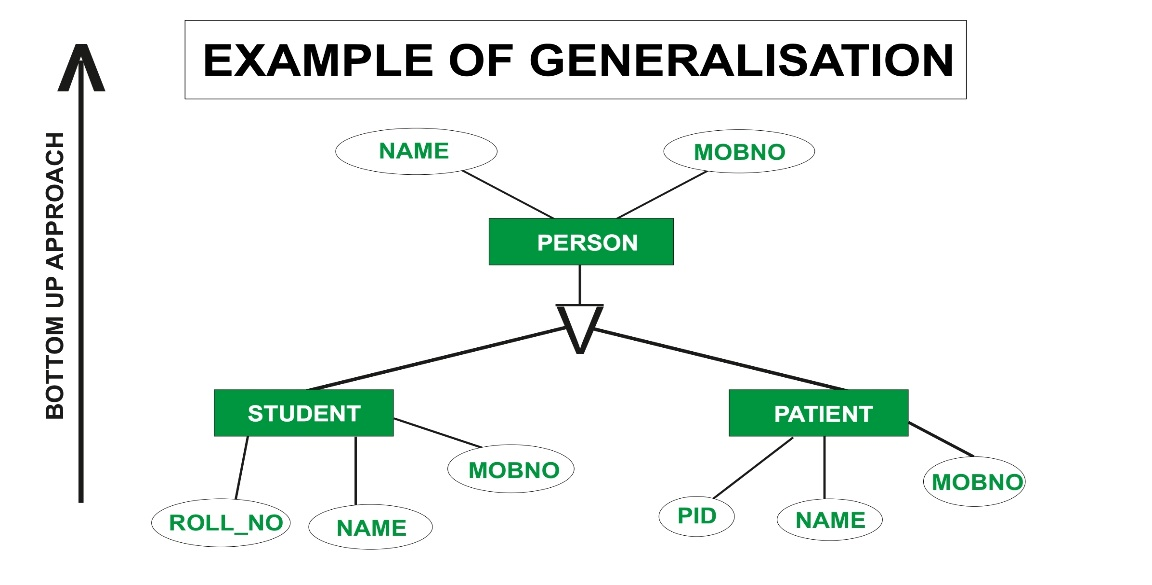
**Ans.**

Generalization and Specialization both the terms are more common in **Object Oriented Technology**,

**Generalization** occurs when we ignore the differences and acknowledge the similarities between lower entities or child classes or relations (tables in DBMS) to form a higher entity.

We can say that Specialization is opposite of Generalization. In Specialization things are broken down into smaller things to simplify it further

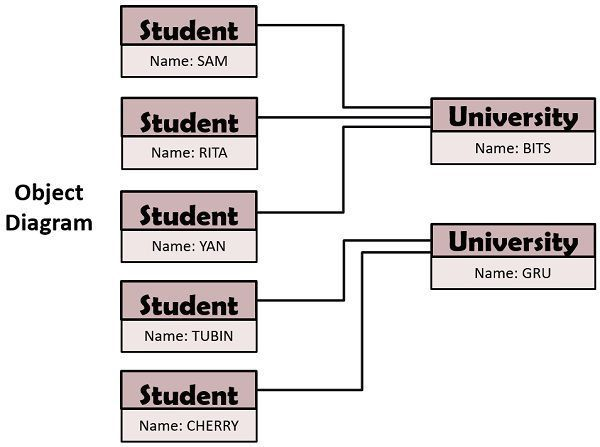




**Link and association** in UML represent the **relation** between **objects** and **classes**. Link is used to establish the relation between the objects in the object diagram.

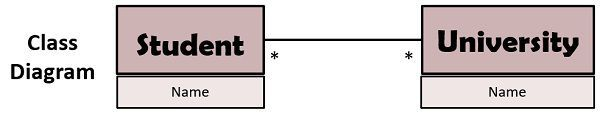
**Link** defines the relationship between two or more objects and a link is considered as an **instance** of an association

The links in the below example represent the relationship between the different objects. The links can be of three types – One-to-one, one-to-many, many-to-many.



A collection of links is specified by an **association** which have common structure and semantics. Association is essentially bidirectional.

Let’s take an example, a student X studies at Z university. In the below example “studies in” is a link between student and university class. The various kinds of Association are unary, binary, ternary, quaternary, and higher order association.



**1.b “**Inheritance can increase the reusability of code." Justify. Explain the different types of inheritance.

**Ans.** Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.

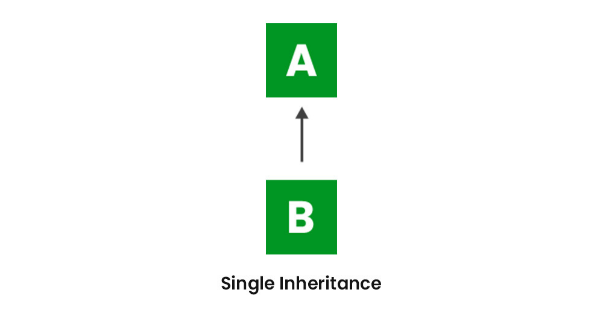
**Code Reusability:**

Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

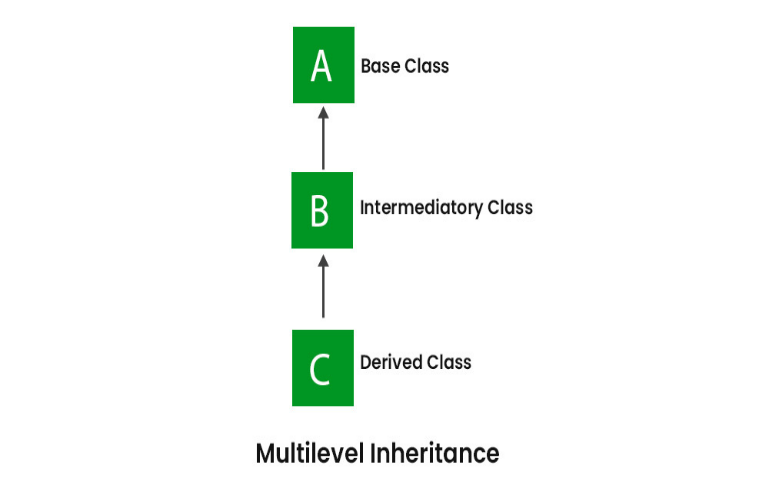
“Code reusability” has a straightforward meaning independent of how it’s achieved—code is reusable when you reuse it! In particular, it’s reusable when you can use the same piece of code in multiple contexts without copying or changing it.

**Type of Inheritance:**

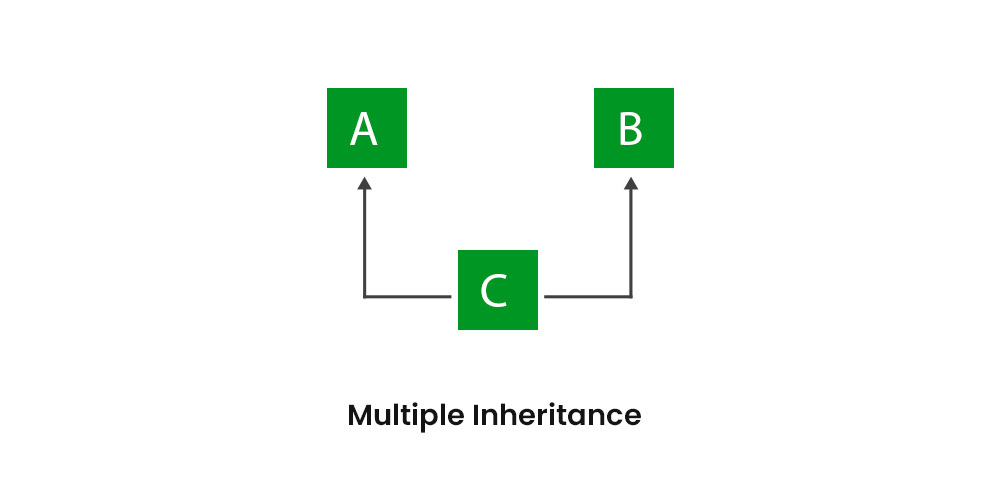
1. **Single Inheritance:**In single inheritance, subclasses inherit the features of one superclass. In the image below, class A serves as a base class for the derived class B.



1. **Multilevel Inheritance:**In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class.

****

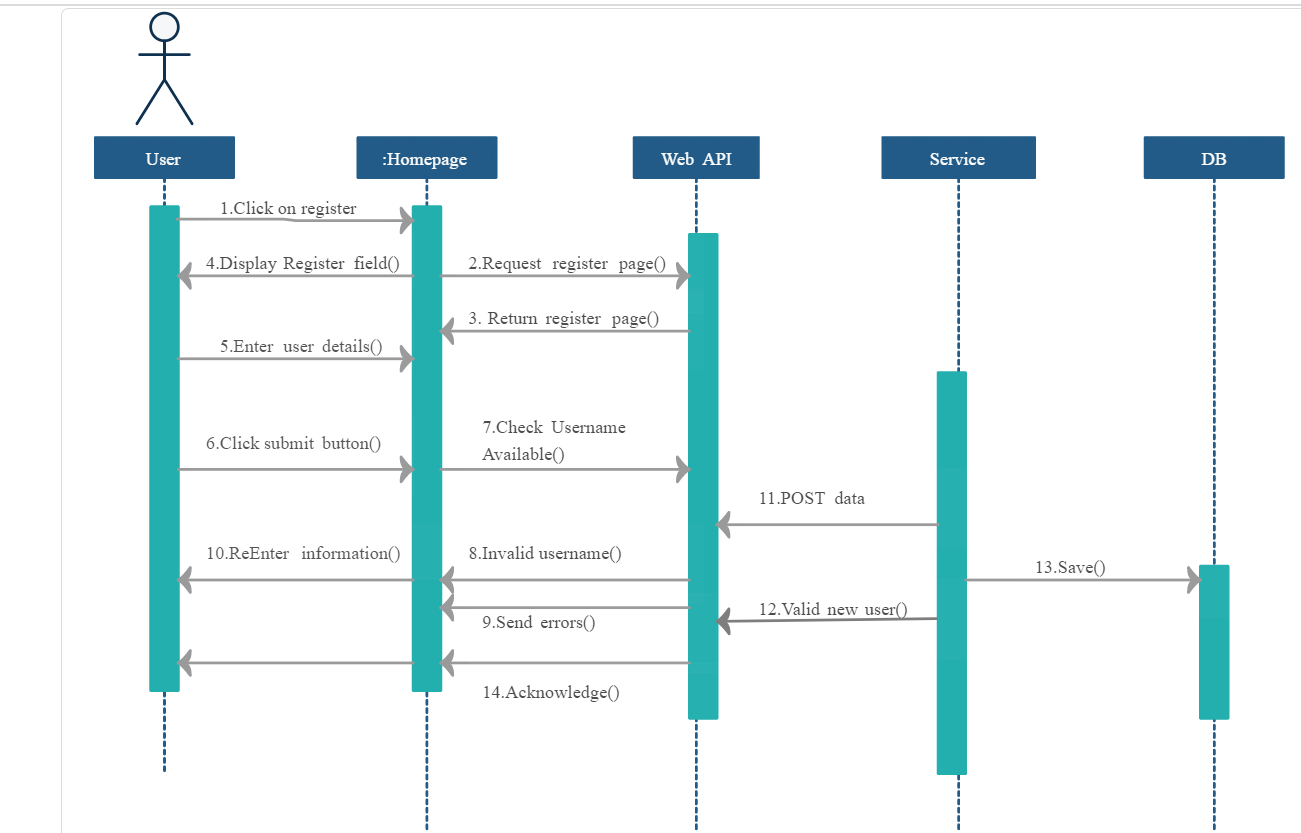
1. **Hierarchical Inheritance:** In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass. In the below image, class A serves as a base class for the derived class B, C and D.

****

**2.a** Extract the significant of sequence diagram, also define various terms and symbols used in a sequence diagram. Draw a sequence diagram for creating new email registration system.

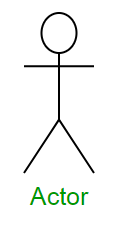
**Ans.**

Sequence diagrams are a popular dynamic modelling solution in UML because they specifically focus on *lifelines*, or the processes and objects that live simultaneously, and the messages exchanged A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios. Between them to perform a function before the lifeline ends.

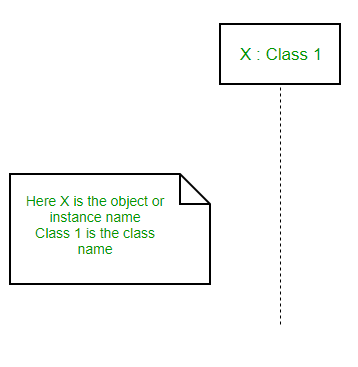
****

**Sequence diagrams are made up of the following icons and elements:**

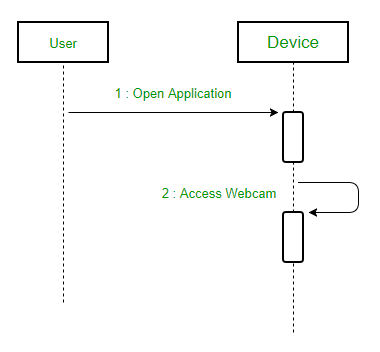
**Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects.

****

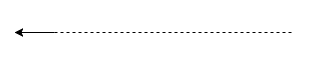
**Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline.

****

**Messages –** Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.



**Reply Message –** Reply messages are used to show the message being sent from the receiver to the sender. We represent a return/reply message using an open arrowhead with a dotted line

****

**2.b** What do you mean by a collaboration diagram? Explain various terms and symbols used in a collaboration diagram. Draw collaboration diagram for ATM transaction system.

**Ans.**

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming.

**The four major components of a collaboration diagram are:**

1. **Objects :**

Objects are shown as rectangles with naming labels inside. The naming label follows the convention of object name: class name. If an object has a property or state that specifically influences the collaboration, this should also be noted.

1. **Actors :**

Actors are instances that invoke the interaction in the diagram. Each actor has a name and a role, with one actor initiating the entire use case.

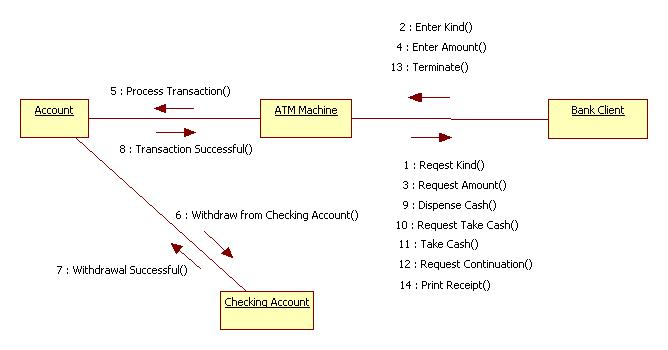
1. **Links :**

Links connect objects with actors and are depicted using a solid line between two elements. Each link is an instance where messages can be sent.

1. **Messages :**

Messages between objects are shown as a labelled arrow placed near a link. These messages are communications between objects that convey information about the activity and can include the sequence number.

**ATM collaboration Diagram:**

****

3.A Describe by applying the mechanism of accessing data members and member functions in the following cases : 1.Inside the main function. 2. Inside a member function of the same class. 3. Inside a member function of another class.

Ans. The data members of a class must be declared within the body of the class, whereas the member functions of the class can be defined in any one of the following ways.

• Inside the class specification

• Outside the class specification The syntax of a member function definition changes depending on whether it is defined inside or outside the class specification, but it performs the same operation

**All the member functions defined within the body of a class.**

**Eg:**

**Class classname**

**{**

**private:**

**int age;**

**int setage(int agein); //member function**

**{**

**age = agein; //body of the function**

**}**

**………..**

**public:**

**int b;**

**void rect( )**

**{**

**. . . . // body of a function**

**}**

**}**

**To declare function prototype within the body of a class and then define it outside**

**the body of a class. This is done by using the ‘scope resolution operator’ (::). It acts as**

**an identity-label to inform the compiler, the class to which the function belongs.**

**Syntax:**

**class classname**

**{**

**. . . .**

**Returntype memberfunction (arguments); //function declaration**

**. . . .**

**};**

**returntype classname :: memberfunction (arguments) //function definition**

**{**

**//body of the function**

**}**

**Accessing member functions within the class**

**A member of a class is accessed by the objects of that class using the dot**

**operator.**

**Ex:**

**#include <iostream.h>**

**class number**

**{**

**int num1, num2; //private by default**

**public:**

**void read( )**

**{**

**cout<<”Enter first number: “);**

**cin>>num1;**

**cout<<”Enter second number: “);**

**cin>>num2;**

**}**

**int max( )**

**{**

**if(num1>num2)**

**return num1;**

**else**

**return num2;**

**}**

**//Nesting of member function**

**void showmax( )**

**{**

**cout<<”maximum = “<<max( ) ;**

**}**

**};**

**void main( )**

**{**

**number n1;**

**n1.read( );**

**n1.showmax( );**

**}**

**3.b** In a program class encapsulation have the variable x is made private. This variable can be accessed and manipulated only using the functions get() and set() which are present inside the class. Thus we can say that here, the variable x and the functions get() and set() are binded together which is nothing but encapsulation. Implement above case in your c++ program by applying concept of encapsulation. Differentiate between encapsulation and data abstraction.

**Ans.**

In normal terms Encapsulation is defined as wrapping up of data and information under a single unit. In Object Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulates them. Consider a real life example of encapsulation, in a company there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial transactions and keep records of all the data related to finance.

We can not access any function from class directly. We need an object to access that function which is using the member the variable of that class.

The function which we are making inside the class ,it must use the all member variable then only it is called encapsulation.

If we don’t make function inside the class which is using the member variable of the class then we don’t call it encapsulation.

#include<iostream>

using namespace std;

class Encapsulation

{

    private:

        // data hidden from outside world

        int x;

    public:

        // function to set value of

        // variable x

        void set(int a)

        {

            x =a;

        }

        // function to return value of

        // variable x

        int get()

        {

            return x;

        }

};

// main function

int main()

{

    Encapsulation obj;

    obj.set(5);

    cout<<obj.get();

    return 0;

}

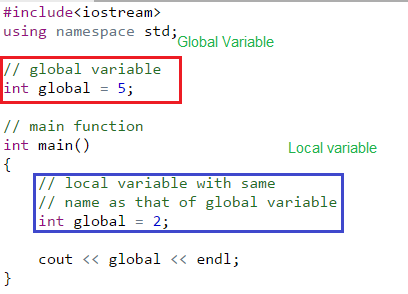
|  |  |
| --- | --- |
| ABSTRACTION | ENCAPSULATION |
| 1. | Abstraction is the process or method of gaining the information. | While encapsulation is the process or method to contain the information. |
| 2. | In abstraction, problems are solved at the design or interface level. | While in encapsulation, problems are solved at the implementation level. |
| 3. | Abstraction is the method of hiding the unwanted information. | Whereas encapsulation is a method to hide the data in a single entity or unit along with a method to protect information from outside. |
| 4. | We can implement abstraction using abstract class and interfaces. | Whereas encapsulation can be implemented using by access modifier i.e. private, protected and public. |
| 5. | In abstraction, implementation complexities are hidden using abstract classes and interfaces. | While in encapsulation, the data is hidden using methods of getters and setters. |
| 6. | The objects that help to perform abstraction are encapsulated. | Whereas the objects that result in encapsulation need not be abstracted. |

**4.a** In C++, "A variable can be declared anywhere in the scope." Justify. What is the significance of this features? Write a function using reference variable as arguments to swap the value of a pair of integers.

**Ans.**

In general, the scope is defined as the extent up to which something can be worked with. In programming also, the scope of a variable is defined as the extent of the program code within which the variable can be accessed or declared or worked with. There are mainly two types of variable scopes:

1. Local Variables
2. Global Variables

****

**#include<iostream.h>**

**#include<conio.h>**

**class ABES**

**{**

**private:**

**int t;**

**public:**

**void swap(int &a, int &b)**

**{**

**t= a;**

**a=b;**

**b=t;**

**}**

**};**

**void main()**

**{**

**int a,b;**

**clrscr();**

**ABES A;**

**cout<<"Enter the values of A: ";**

**cin>>a;**

**cout<<"Enter the values of B: ";**

**cin>>b;**

**A.swap(a,b);**

**cout<<"\nValues of A is: "<<a;**

**cout<<"\nValues of B is: "<<b;**

**}**

**4.b** Explicate role of function in C++. Differentiate between call by value and call by reference with Appropriate example.

**Ans.**

A function is a code module that performs a single task. Some examples such as sorting, search for a given item, and invert a square matrix. Once a function is created it is tested extensively. After this, it becomes a part of the library of functions. A user can use such a library function as many times as needed. This idea improves software robustness and also reduced code development time. Functions are classified into two categories: system defined, and user defined. Examples for a system defined math library functions are: sqrt, exp, and log.

**Advantage of functions in C**

There are many importance of functions like.

**1) Code Reusability**

By creating functions in C++, you can call it many times. So we don't need to write the same code again and again.

**2) Code optimization**

It makes the code optimized, we don't need to write much code. Suppose, you have to check 3 numbers (531, 883 and 781) whether it is prime number or not. Without using function, you need to write the prime number logic 3 times. So, there is repetition of code. But if you use functions, you need to write the logic only once and you can reuse it several times.

**Types of Functions**

There are two types of functions in C programming:

**1. Library Functions:** are the functions which are declared in the C++ header files such as ceil(x), cos(x), exp(x), etc.

**2. User-defined functions:** are the functions which are created by the C++ programmer, so that he/she can use it many times. It reduces complexity of a big program and optimizes the code.

Functions can be invoked in two ways: Call by Value or Call by Reference. These two ways are generally differentiated by the type of values passed to them as parameters.

The parameters passed to function are called *actual parameters* whereas the parameters received by function are called *formal parameters*.

Call By Value: In this parameter passing method, values of actual parameters are copied to function’s formal parameters and the two types of parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.

Call by Reference: Both the actual and formal parameters refer to the same locations, so any changes made inside the function are actually reflected in actual parameters of the caller.

|  |  |
| --- | --- |
| **Call By Value** | **Call By Reference** |
| While calling a function, we pass values of variables to it. Such functions are known as “Call By Values”. | While calling a function, instead of passing the values of variables, we pass address of variables (location of variables) to the function known as “Call by References. |
| In this method, the value of each variable in calling function is copied into corresponding dummy variables of the called function. | In this method, the address of actual variables in the calling function are copied into the dummy variables of the called function. |
| With this method, the changes made to the dummy variables in the called function have no effect on the values of actual variables in the calling function. | With this method, using addresses we would have an access to the actual variables and hence we would be able to manipulate them. |

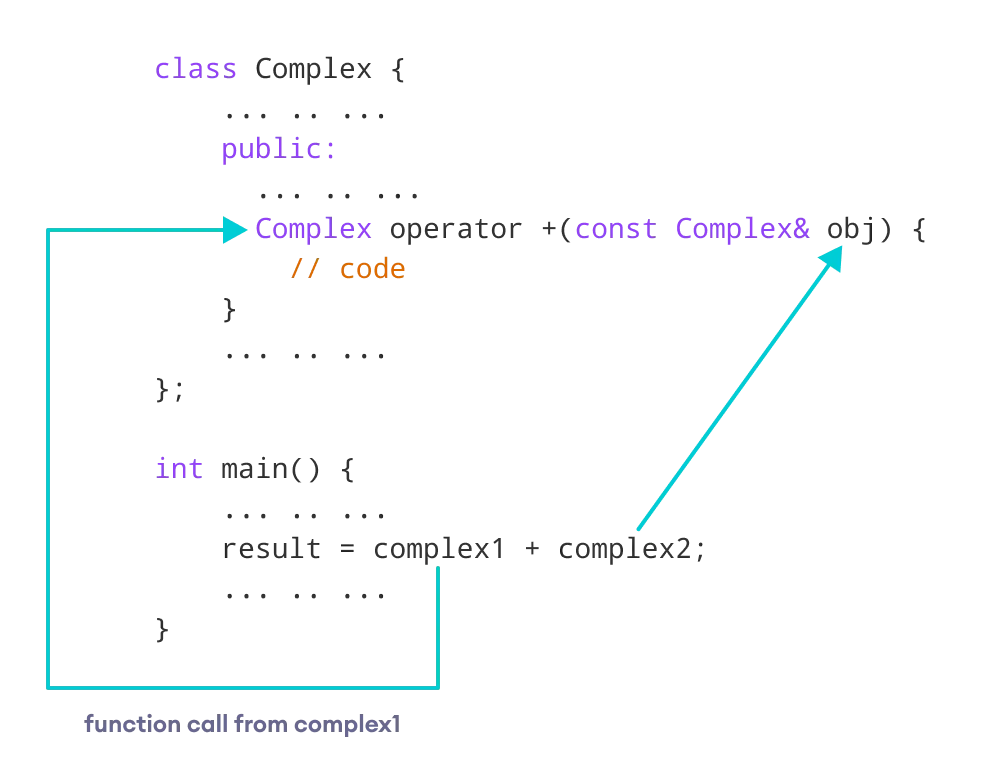
**5.a** In case of non-static member functions how many maximum object arguments a binary operator overloaded function can take? Applying concept of overloading write a program in C++ for (+) operator overloading.

Ans.

*C++ provides a special function to change the current functionality of some operators within its class which is often called as operator overloading. Operator Overloading is the method by which we can change the function of some specific operators to do some different tasks.*

**Below are some criteria/rules to define the operator function:**

* In case of a non-static function, the binary operator should have only one argument and unary should not have an argument.
* In the case of a friend function, the binary operator should have only two argument and unary should have only one argument.
* All the class member object should be public if operator overloading is implemented.
* Operators that cannot be overloaded are **.** **.\*** **::** **?:**
* Operator cannot be used to overload when declaring that function as friend function **=** **()** **[]** **->**.

****

**5.b** Can we have more than one constructor in a class? If yes, write a C++ program that shows the concept of multiple constructors in a class. Write output of below code

**Ans. Constructor in C++** is a special method that is invoked automatically at the time of object creation. It is used to initialize the data members of new objects generally. The constructor in C++ has the same name as the class or structure. Constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object which is why it is known as constructors.

Constructor does not have a return value, hence they do not have a return type.

The prototype of Constructors is as follows:

<class-name> (list-of-parameters);

Yes, we have more than one constructor is declare in a program this is the concept of constructor overloading in C++. Through this constructor overloading we can achieve polymorphism.

EXAMPLE:

C++ program to demonstrate more than one constructor in a class.

#include <iostream>

using namespace std;

class Person {

private:

int age;

public:

// 1. Constructor with no arguments

Person() {

age = 20;

}

// 2. Constructor with an argument

Person(int a) {

age = a;

}

int getAge() {

return age;

}

};

int main() {

Person person1, person2(45);

cout << "Person1 Age = " << person1.getAge() << endl;

cout << "Person2 Age = " << person2.getAge() << endl;

return 0;

}

**Output of C++ program is : Ronav 101**