name: Anita

3.10 OBJECTS AS FUNCTION ARGUMENT

- In C++ an object may be used as a function argument.
- The following two approaches used in objects as a function argument.

1. **Passing Object to function:**
   - Objects may be passed to a function just like any other variable. Arguments can be passed to a function in two ways:
     - Call by value (A copy of entire object is passed to the function)
     - Call by reference (Only the address of the object is transferred to the function).
   - Objects are passed through the call by value mechanism by default. In this method value of the variable (object) will be passed to function as an argument or parameter.
2. **Returning objects from the function:**
   - A function may return an object to the calling function.
   - It works similar to the normal integer or float variables. For such functions return type will be name of a class.

Program 3.6: Program for object as function argument concept.

```
/* Program which adds 2 complex numbers */
#include<iostream.h>
#include<conio.h>
class COMPLEX
{
   int real, imag;
   public:
      void get_no(int a, int b)
      {
         real = a;
         imag = b;
      }
```

```
    void display_no( )
    {
        cout<<"\n Real ="<<real;
        cout<<"\n Imag = "<<imag;
    }
    COMPLEX add_no (COMPLEX C2)
    {
        COMPLEX C3;
        C3.real = real + C2.real;
        C3.imag = imag + C2.imag;
        return(C3);
    }
}; // end of class
int main( )
{
    COMPLEX C1, C2, C3;
    C1.get_no(10, 20);
    C2.get_no(30, 40);
    C3=C1.add_no(C2);
    cout<<"\n Addition";
    C3.display_no( );
}
```

Output:

```
Addition
Real = 40
Imag = 60
Press any key to continue . . .
```

3.11 FRIEND FUNCTION [S - 19]

- The functions that are declared with the keyword friend are known as friend functions.
- To make an outside function friendly to a class, we have to simply declare this function as a friend of the class.
- A friend function is a function that is not a member of a class but has access to the class's private and protected members. Friend functions are not considered class members; they are normal external functions that are given special access privileges.
- Friends are not in the class's scope, and they are not called using the member selection operators (. and ->) unless they are members of another class.

- **For example:**

```
class xyz
{
    ......
    public:
    ......
    ......
    friend void abc(void);
};
```

- The function declaration in above example should be preceded by the keyword friend.

Program 3.7: C++ Program to Add Two Numbers using Friend Function. [S - 19]

```
#include<iostream>
using namespace std;
class temp
{
  int x, y, z;
  public:
    void input()
    {
        cout << "Enter the value of x and y:";
        cin >> x>>y;
    }
    friend void add(temp &t);
    void display()
    {
        cout << "The sum is :" << z;
    }
};
void add(temp &t)
{
   t.z = t.x + t.y;
}
int main()
{
   temp t1;
   t1.input();
   add(t1);
   t1.display();
   return 0;
}
```

Output:

```
Enter the value of x and y:10 20
The sum is :30
```

3.11.1 Friend Class

- A friend class is a class all of whose member functions are friend functions of a class, that is, whose member functions have access to the other class's private and protected members.
- A friend class in C++, can access the "private" and "protected" members of the class in which it is declared as a friend. On declaration of friend class all member function of the friend class become friend of the class in which the friend class was declared. Friend status is not inherited; every friendship has to be explicitly declared.
- Classes are declared as friends within the definition of the class to whom access is to be given; this prevents a class from giving itself access to another's protected members, which enforces encapsulation.
- The friend class has the same level of access irrespective of whether the friend declaration appears in either the public, protected or private sections of the class definition. Friend status is granted by using the friend keyword.

```
friend class ClassName;
```

Program 3.8: Program illustrate friend class concept.

```
#include <iostream.h>
#include <conio.h>
class B
{
   // B declares A as a friend...
   friend class A;
   private:
      void privatePrint()
      {
         std::cout << "hello, world" << std::endl;
      }
};
class A
{
   public:
       A()
      {
         B b;
         // ... and A now has access to B's private members
         b.privatePrint();
      }
};
int main()
{
 A a;
 return 0;
}
```

Output: hello, world

Advantages of using friend class:

1. It provides additional functionality which is kept outside the class.
2. It provides functions with data which is not normally used by the class.
3. It allows sharing private class information by a non member function.

3.12 FUNCTION RETURNING OBJECTS

- A function may return an object to the calling function.
- It works similar to the normal integer or float variables. For such functions return type will be name of a class.

Program 3.9: Program for returning objects.

```
/* Program which adds 2 complex numbers */
   #include<iostream.h>
   #include<conio.h>
   class COMPLEX
   {
      int real, imag;
      public:
      void get_no(int a, int b)
      {
         real = a;
         imag = b;
      }
      void display_no( )
      {
         cout<<"\n Real ="<<real;
         cout<<"\n Imag = "<<imag;
      }
      COMPLEX add_no (COMPLEX C2)
      {
         COMPLEX C3;
         C3.real = C1.real + C2.real;
         C3.imag = C1.imag + C2.imag;
         return(C3);
      }
   }; // end of class
void main( )
{
   COMPLEX C1, C2, C3;
   C1.get_no(10, 20);
   C2.get_no(30, 40);
   C3=C1.add_no(C2);
   cout<<"\n Addition =";
   C3.display_no( );
}
```

Output:

```
Addition
real = 40
imag = 60
```