Object Oriented Programming Using Cpp

Topic:

Object As Function Argument

6.6 Object as function argument

Now, if we need to pass class data members to function as arguments, it can be done by passing the data members as arguments. However, this approach is efficient only when class contains less number of data members. But practically, a class may contain large number of data members and in this scenario it is not feasible to pass data members as function arguments. In this situation, instead of passing data members as arguments, we can pass object as function argument because an object has all the data members defined inside a class.

Program 31: Write a program in C++ to find out useful life of a product using class and object. Also illustrate the concept of object as an argument.

Code:

```
#include<iostream.h>

#include<conio.h>

class  PRODUCT

{

        int     mfg_yr, exp_yr;

        public:

                void input();

                void lifetime(PRODUCT);

};

void PRODUCT: :input()

{

        cout<<"Enter the year of manufacturing and expiry:";

        cin>>mfg_yr>>exp_yr;
}
void PRODUCT : : lifetime(PRODUCT    p1)
{
        int      LOP=0;
        LOP=p1.exp_yr -  p1.mfg_yr;
        cout<<"Useful life of product: "<<LOP<<"yrs.";
}
void    main()
{
        PRODUCT    p2;
        p2.input();
        p2.lifetime(p2);          // Object p2 is passed as argument
        getch();
}
```

O/P:

```
Enter year of manufacturing and expiry
2002            2015
Useful life of product:13yrs.
```

6.7 Friend Function

The main idea behind the data hiding is that, only member function of the class has access the private data of the class. A non-member function can't access the class data members. However, C++ provides a mechanism where a non-member function can access the data members of a class. This mechanism is known as friend function. A function can be made a friend to a class if it is declared using friend keyword inside the class. However, the function must be defined outside the class like a normal function. It is because friend function is not a member function of a class.

A friend function can be declared using following syntax:

Syntax for friend function

```
class <class_name>
{
        ........................;
        ........................;
        public:
                friend return_type function_name(class_name);
};
Return_type        function_name(class_name      obj)
{



}
```

6.7.1 Properties of friend function

- A friend function is not a member function of a class. Rather it is just a friend of the class.
- The friend keyword must be present in the prototype of the function which must be written inside the function. However, the function must be defined outside the class without friend keyword.
- Since it is not a member of the class, a friend function is directly called without using object.
- One function can be friend of multiple classes. Similarly, a class can have multiple friend function.

Program 32: Write a program in C++ to find out useful life of a product using class and object. Use friend function to calculate the lifetime of product.

Code:

```
#include<iostream.h>
#include<conio.h>
class  PRODUCT
```

```
{
        int     mfg_yr, exp_yr;
        public:
                void        input();
                friend      void        lifetime(PRODUCT);
};
void PRODUCT: :input()
{
        cout<<"Enter the year of manufacturing and expiry:";
        cin>>mfg_yr>>exp_yr;
}
void    lifetime(PRODUCT            p1)
{
        int     LOP=0;
        LOP=p1.exp_yr -  p1.mfg_yr;
        cout<<"Useful life of product: "<<LOP<<"yrs.";
}
void    main()
{
        PRODUCT     p2;
        p2.input();
        lifetime(p2);   // Friend function is called.
        getch();
}
```

O/P:

```
Enter year of manufacturing and expiry
2002        2015
Useful life of product:13yrs.
```

Program 33: Write a C++ program to illustrate the concept that a function can be friend of multiple classes.

Code:

```
#include<iostream.h>

#include<conio.h>

class       TWO;        // Forward declaration of a class

class       ONE

{
```

```
          int       n1;
          public:
                    void input();
                    **friend void add(ONE, TWO);      // Friend of class ONE**
};
void    ONE: : input()
{
          cout<<"Enter the value for n1:";
          cin>>n1;
}
class           TWO
{
          int       n2;
          public:
                    void input();
                     **friend void add(ONE, TWO);     // Friend of class TWO**
};
void    TWO: : input()
{
          cout<<"Enter the value for n2:";
          cin>>n2;
}
void    add(ONE      o1, TWO       t1)
{
          int       sum=0;
          sum = o1.n1 + t1.n2;
          cout<<"Sum is "<<sum;
```

```
}
void   main()
{
        ONE  o2;
        o2.input();
        TWO  t2;
        t2.input();
        add(o2,t2);              // Friend function is called
}
```

O/P:

```
Enter the value for n1:
5
Enter the value for n2:
4
Sum is 9
```

Program 34: Write a C++ program to illustrate the concept that a class can have multiple friend functions.

Code:

```
#include<iostream.h>
#include<conio.h>
class          ONE
{
        int      n1,n2;
        public:
                void input();
                friend void display(ONE);//1st Friend function of class ONE
                friend void add(ONE);     //2ndFriend function of class ONE
};
void   ONE: : input()
{
        cout<<"Enter the value for n1 and n2:";
        cin>>n1>>n2;
}
void   display(ONE  o1)
{
        cout<<"n1="<<n1<<"\t"<<"n2="<<n2<<endl;
}
void   add(ONE      o2)
{
        int      sum=0;
        sum = o2.n1 + o2.n2;
        cout<<"Sum is "<<sum;
}
void   main()
{
         ONE  o3;
        O3.input();
        display(o3);// Friend function is called
        add(o3);      // Friend function is called
}
```

O/P:

```
Enter the value for n1 and n2:
5      4
n1=5          n2=4
```

```
Sum is 9
```

Note: Friend function is not frequently used in practical programming due to the fact that it violates the data security.

6.8 Friend Class

Till now we have seen that a member function of a class can access the data members of the same class. However, C++ provides a mechanism where member function of a class can access the data members of another class. It can be done by establishing a friendship between the two classes using a member function. This concept is known as friend class. However, this friendship is not mutual i.e. If class A is made friend of class B then member function of class A can access the data members of class B but reverse is not true.

Program 35: Write a C++ program to illustrate the concept of friend class.

Code:

```
        #inlcude<iostream.h>

        #inlcude<conio.h>

        class   B;     // Forward declaration of class

        class   A

        {

int   a;

          public:

            void input()

            {

                a=5;

            }

            void   show(B);

        };

        class   B

        {
```

```
int     b;
            public:
                void input()
                {
                    b=10;
                }
        friend    void    A : : show(B); //A becomes friend of B using show
        };
        void     A : : show(B  ob)
        {
        cout<<"a="<<a<<"\t"<<"b="<<ob.b;
        }
        void    main()
        {
            A     obj1;
            obj1.input();
            B     obj2;
            obj2.input();
            obj1.show(obj2);
        }
```

O/P:

```
        a=5         b=10
```

Explanation: Here, there should not be any confusion that show is a friend function. You may check how show function is called? It is called using object of class A. SO, it is not a friend function. In fact, the friendship between class A and class B has been established using this member function show().

6.9 Local Class

When a class is defined inside a function, it is known as local class. Generally it is not used while practical programming as it restricts the accessibility of the class.

Consider the following example:

```
void main()
{
     class   A
     {
        int    a;
        public:
          void fun()
          {
               a=5;
               cout<<"a="<<a;
          }
        };
        A     ob1;
        ob1.input();
  }
```

Here, class A behaves as local class cause it is defined inside the main() function.

6.10 Nested class

If a class contains another class as its member then it is known as nested class. While using nesting of classes some special care should be taken while accessing inner class members. This is because the inner class is not independent. It is a member of outer class.

Consider the following example:

```
class  A
{
        int      a;
        public:
                void show()
                {
                        a=10;
                        cout<<"a="<<a<<endl;
                }
                class  B
                {
                        int   b;
                        public:
                                void show()
                                {
                                        b=20;
                                        cout<<"b="<<b<<endl;
                                }
                };                      //closing of inner class B
};              // closing of outer class A
void main()
{
        A       o1;
        o1.show();
        A : : B   o2;
        o2.show();

        getch();

}
```

Note:

- We know that a program execution starts from main(). So, a curiosity may arise that whether main can be used as a member function or not? The answer is, YES. We can use main as a member function inside a class without any problem. The reason is, the scope of this main is limited inside the class where it is defined.

  Example:

```
        class  sample
        {
                int      n1;
                public:
                        void main()            //main as a member function
                        {
                                n1=10;
                                cout<<"n1="<<n1;
                        }
        };
        void main()
        {
                sample  s1;
                s1.main();
        }
```

- It is possible to write multiple member function inside a class with the same name. The concept is known as **overloading of member functions.**It can be performed by using the same three rules which were applicable for function overloading as it is very important for the system to distinguish between the member functions so that the functions can be called successfully.

  Example:

```
        class  sample
        {
                int      n1;
                public:
                        void fun(int x)
                        {
                             n1=x;
                        }
                        void fun()
                        {
                            cout<<"n1="<<n1;
                    }
        };
        void main()
        {
                sample  s1;
                s1.fun(10);
                s1.fun();
        }
```

- Now if we combine the first two points we can conclude that it is possible to do the overloading the main function but that should be only when main is used as a member function inside a class. Failing to do so will create two main function in a program which will create error. So, overloading for the main function inside a class must be done carefully.

  Example:

```
        class  sample
        {
                int     n1;
                public:
                        void main(int x)
                        {
                                n1=x;
                        }
                        void main()
                        {
                                cout<<"n1="<<n1;
                        }
        };
        void main()
        {
                sample  s1;
                s1.main(10);
                s1.main();
        }
```