Durvesh Naresh Patil 05/10/2021

2019BTEEN00035 Batch: EN-4 (T.Y. B.Tech)

Object Oriented Programming.

Exp. 6. : Operator overloading (binary)

⊙ Activity :

1) Write a program to overload '+' operator to add two complex numbers.

⇒

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

class Complex
{
        int real, imag;
   public:
        Complex()
        {
            real = imag = 0;
        }
        Complex(int r, int i)
        {
            real = r; imag = i;
        }
        void setData()
        {
            cout << "\nEnter real part: ";
            cin >> real;
            cout << "\nEnter imaginary part: ";
            cin >> imag;
        }
```

```
    void getData()
    {
        cout <<"\n" <<real <<" + " << imag << "i";
    }

    Complex operator + (Complex & c1)
    {
        Complex res;
        res.real = real + c1.real;
        res.imag = imag + c1.imag;
        return res;
    }
};

int main()
{
    Complex c1(20,10);
    Complex c2(4,6);

    Complex c3 = c1 + c2;
    c3.getData();

    return 0;
}
```

<u>output</u> : 24 + 16i

2) Write a program to overload '<' operator to compare two distances entered by user.

⇒

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

class Distance
{
        int feet;
        float inches;
    public:
        Distance()
        {
            feet = 0; inches = 0.0;
        }
        Distance(int f, float i)
        {
            feet = f;
            inches = i;
        }
        void setData()
        {
            cout << "\nEnter feet: ";
            cin >> feet;
            cout << "\n Enter inches: ";
            cin >> inches;
        }

        void getData()
        {
            cout << "feet : " << feet << "\t Inches : "
            << inches << "\n";
        }
        bool operator < (Distance & d2)
        {
            float dist1 = feet + inches/12 ;
            float dist2 = d2.feet + d2.inches/12 ;

            return (dist1 < dist2) ? true : false ;
        }
};

int main()
{
    Distance d1 (5, 8.2) ;
    Distance d2 ( 6, 2.5) ;
    if (d1 < d2)
        cout <<"\n d1 is less than d2 " ;
    else
        cout << "\n d1 is greater than or equal to d2 " ;

    return 0;
}
```

Output:

```
d1 is less than d2
```

⊙ Questions :

1) Which is the correct statement about operators overloading.
⇒ Associativity & precedence of operators does not change.

2) How many maximum object arguments a binary operator overloaded function can take ?
⇒ 1

3) When using binary operators, overloaded through a member function, the left-hand operand must be an object of the relevant class.

⊙ **Conclusion :** Using operator overloading, we can give special meanings to operators. In binary operator overloading, there should be one argument to be passed. It is overloading of an operator operating on two operands. We can achieve polymorphism using operator overloading.