Subject Name: Object Oriented Programming Using C++

Subject Code: BCA-301 N

Subject Topic: Operator Overloading using Non-member/ Friend function

Abhishek Dwivedi

Assistant Professor

Department of Computer Application

UIET, CSJM University, Kanpur

Operator Overloading using Non-member/ Friend function

- Overload binary plus (+) operator using non-member/Friend function

```
class Complex
    {
          int real, img;
       public:
          void getnumber()
          {
                cout<<"\n Enter Two Numbers : ";
                cin>>real>>img;
          }

          friend Complex operator+(Complex, Complex );

          void display()
          {
                cout<<real<<"+"<<img<<"i"<<"\n";
          }
    };
```

Friend complex operator+ function Definition

```
Complex operator+(Complex c1, Complex c2)
   {
          Complex c;
          c.real=c1.real+c2.real;
          c.img=c1.img+c2.img;
          return(c);
   }
```

Main Function Definition

```
void main()
     {
            Complex c1,c2, c3;

            c1.getnumber();
            c2.getnumber();

            c3 = c1+c2;                         //Addition of object

            cout<<"\n Entered Values : \n";

            c1.display();                       //Displaying user input values

            c2.display();

            cout<<"\n Addition of Real and Imaginary Numbers : \n";

            c3.display();
           getch();
     }
```

Example for Unary Operator Overloading

```
class complex
{
     int a, b, c;
    public:
    complex() { }
    void getvalue()
    {
       cout << "Enter the Two Numbers:";
       cin >> a>>b;
    }
     void operator++()
    { a = ++a; b = ++b; }
    void operator--()
    { a = --a; b = --b; }
    void display()
    { cout << a << “+" << b << "i" << endl; }
 };
```

Main Function Definition

```
void main()
{
    complex obj;
    obj.getvalue();
    obj++;
    cout << "Increment Complex Number\n";
    obj.display();
    obj--;
    cout << "Decrement Complex Number\n";
    obj.display();
    getch();
}
```

Unary operator overloading using Friend function

```
class UnaryFriend
    {
        int a,b,c;

        public:
           void getvalues()
           {
               cout<<"Values of A, B & C\n";
               cin>>a>>b>>c;
           }
           void show()
           {
               cout<<a<<"\n"<<b<<"\n"<<c<<"\n"<<endl;
           }

            friend void operator-(UnaryFriend);      //Pass by reference
    };
    void operator-(UnaryFriend x)
    {
        x.a = -x.a;     //Object name must be used as it is a friend function
        x.b = -x.b;
        x.c = -x.c;
    }
```

```
int main()
    {
        UnaryFriend x1;
        x1.getvalues();
        cout<<"Before Overloading\n";
        x1.show();
        cout<<"After Overloading \n";
        -x1;
         x1.show();
         return 0;
    }
```

- The statement **-x1** invokes the **operator()** function. The object **x1** is created of class **UnaryFriend.** The object itself acts as a source and destination object. This can be accomplished by sending reference of an object. The object **x1** is a reference of object **x.** The values of object **x1** are replaced by itself by applying negation.

Overloading Relational Operator in C++

- We can also overload relational operators like == , != , >= , <= etc. to compare two object of any class. overloading the == operator in the Time class to directly compare two objects of Time class.

```
class Time
 {
    int hr, min, sec;
    public:
    Time(int h, int m, int s)
     { hr=h, min=m; sec=s; }
friend bool operator= =(Time , Time );          //overloading '==' operator
};
```

```
/* Defining the overloading operator function Here we are simply
    comparing the hour, minute and second values of two different Time
    objects to compare their values */
bool operator= =(Time t1, Time t2)
{
return ( t1.hr = = t2.hr && t1.min = = t2.min && t1.sec = = t2.sec );
 }


void main()
{
Time t1(3,15,45);
Time t2(4,15,45);
 if(t1 == t2)
{ cout << "Both the time values are equal"; }
else
{ cout << "Both the time values are not equal"; }
}
```

References:

- www.studytonight.com
- www.tutorialpoint.com
- www.geeksforgeeks.org
- “Object oriented programming in C++”, Robert Lafore
- “Object oriented programming with C++”, E.Balagurusamy