Chapter 11

Operator Overloading

11.1 Introduction

Operator overloading is an important feature of C++ language. Using this feature, we can add two variables of user-defined types with the same way as we do with basic data types.

The mechanism of giving special meanings to an operator for a data type is known as operator overloading.

All the operators in C++ can be overload except the following:

- Class member access operators (., .*)
- Scope resolution operator (::)
- Size operator (sizeof)
- Conditional operator (?:)

When we overload an operator, its original meaning remains same. For example, if we overload + operator to add two matrices, can still be used to add two numbers.

11.2 The operator function

To give additional meaning to an operator, we use a special function called operator function. The prototype of the operator function is :

```
return_type class_name :: operator op(arguments list)
{
        function body
}
```

Where operator is a keyword and op is an operator to be overload.

Operator function must be either member function or friend function of a class. The basic difference between them is that member function takes no argument for unary operators and one argument for binary operators while friend function takes one argument for unary operators and two arguments for binary operators.

11.3 Overloading unary operators using member function

Let us consider the postfix increment operator (++). It takes just one operand and increment its value by one , when applied to basic data types. We will overload this operator so that it can be applied to an object and it increments each data item of that object by one.

Program 11.1 Overloading postfix increment operator

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

```
{
        int x,y;
public:
         void getdata(int a, int b)
         {
                  x=a;
                  y=b;
         }
         void show(void)
         {
                  cout<<"x="<<x;
                  cout<<"y="<<y<<"\n";
         }
         void operator++(int)
         {
                  x++;
                  y++;
         }
};

int main()
{
 point p;
 p.getdata(5,8);
 cout<<"p:";
 p.show();
 p++;            // invoke operator function
 cout<<"p++:";
 p.show();
return 0;
}
```

The output of the program 11.1 would be:
p: x=5 y=8
p++: x=6 y=9

The int in the operator function is used to indicate that we are overloading postfix increment operator not prefix increment operator.

11.4 Overloading unary operator using friend function

Let us consider prefix decrement operator (--) that takes one operand and decrement the value of operand by one , when applied to basic data types. We

will overload this operator so that it can be applied to an object and decrement the value of each data of that object by one.

Program 11.2: Overloading prefix decrement operator

```
#include<iostream>
using namespace std;
class point
{
        int x,y;
public:
         void getdata(int a, int b)
         {
                  x=a;
                  y=b;
         }
         void show(void)
         {
                  cout<<"x="<<x;
                  cout<<"y="<<y<<"\n";
         }
         friend void operator--(point &s)
         {
                  s.x=s.x-1;
                  s.y=s.y-1;
         }
};

int main()
{
 point p;
 p.getdata(7,10);
cout<<"p:";
 p.show();
 --p;
cout<<"--p:";
 p.show();
return 0;
}
```

The output of the program 11.2 would be:

```
p: x=7 y=10
p++: x=6 y=9
```

Note that the argument in operator function is passed by reference. If we pass argument by value it will not work because the changes made in operator function will not reflect in main function.
We can't overload the following operators using friend function.

- = Assignment operator
- () Function call operator
- [] Subscripting operator
- -> Class member access operator

11.5 Overloading binary operators using member function

Let us consider the binary + operator that takes two operands and add them, when applied to basic data types. We will overload this operator to add two matrices.

Program 11.3: Overloading binary + operator

```
#include<iostream>
using namespace std;
class matrix
{
        int mat[2][2];
public:
         void getmatrix(void);
         matrix operator+(matrix);
         void showmatrix(void);
};
void matrix::getmatrix(void)
 {
  for(int i=0; i<2; i++)
    for(int j=0; j<2; j++)
      {
        cout<<"Enter the number:";
        cin>>mat[i][j];
        }
 }
matrix matrix::operator+(matrix m)
{
  matrix temp;
  for(int i=0; i<2; i++)
     for(int j=0; j<2; j++)
       temp.mat[i][j]=mat[i][j]+m.mat[i][j];
  return temp;
}
```

```
void matrix::showmatrix(void)
{
  for(int i=0; i<2; i++)
    {
      for(int j=0; j<2; j++)
      cout<<mat[i][j]<<"\t";
      cout<<"\n";
    }
}

int main()
{
 matrix m1,m2,m3;
 m1.getmatrix();
 m2.getmatrix();
 m3=m1+m2;
 cout<<"matrix m1:\n";
 m1.showmatrix();
 cout<<"matrix m2:\n";
 m2.showmatrix();
 cout<<"Resultant matrix:\n";
  m3.showmatrix();
return 0;
}
```

The output of the program 11.3 would be:

```
Enetr the number: 2
Enetr the number: 3
Enetr the number: 1
Enetr the number: 4
Enetr the number: 6
Enetr the number: 7
Enetr the number: 8
Enetr the number: 9
matrix m1:
2       3
1       4
matrix m2:
6       7
8       9
Resultant matrix:
8       10
```

9 13
In the above program the operator function takes only one argument of matrix type and that is second operand of binary + operator. The first operand m1 is used to invoking the operator function. So, the data members of m1 are directly accessed by the operator function. The following statement
m3=m1+m2;
is equivalent to
m3 =m1.operator+(m2);
For binary operators, the left-side operand is used to invoke the operator function and the right-side operand is passed as an argument.

11.6 Overloading binary operators using friend function

The following program overload the binary + operator to add two complex numbers using friend function.
Program 6.4: Overloading binary + operator using friend function

```
#include<iostream>
using namespace std;
class complex
{
    float real;
    float imag;
pubic:
        void input(float x, float y)
{
        real=x;
        imag=y;
}
friend complex operator + (complex a, complex b)
{
        complex c;
        c.real=a.real+b.real;
        c.imag=a.imag+b.imag;
        return c;
}
void show(void)
{
        cout<<real<<"+i"<<imag<<"\n";
}
};
int main()
{
        complex c1,c2,c3;
```

```
        c1.input(1.6,6.2);
        c2.input(2.3,3.4);
        c3=c1+c2;              //invoke operator function
        cout<<"C1=";
        c1.show();
        cout<<"C2=";
        c2.show();
        cout<<"C3=";
        c3.show();
        return 0;
}
```

The output of the program 11.4 would be:

C1=1.6+i6.2;
C2=2.3+i3.4;
C3=3.9+i9.6;

In the above program, the operator function takes two arguments of complex type explicitly and return a resultant complex number. The following statement

c3=c1+c2;

is equivalent to

c3=operator+(c1,c2);

Important Points

- The mechanism of giving special meanings to an operator for a data type is known as operator overloading.
- When we overload an operator, its original meaning remains same.
- To give additional meaning to an operator, we use a special function called operator function.
- Operator function must be either member function or friend function of a class.

Practice Questions

Objective type questions:

Q.1 Operator that can be overload is

A. Scope resolution operator (::)
B. Class member access operators (., .*)
C. Binary plus operator (+)
D. Conditional operator (?:)

Q.2 To overload a binary operator, operator function as member function will take

A. Two arguments B. One argument
C. Zero argument D. None of these

Q.3 To overload a unary operator, operator function as friend function will take

A. Two arguments B. One argument
C. Zero argument D. None of these

Q.4 Operator that can't be overload using friend function.

A. = Assignment operator B.() Function call operator
C. [] Subscripting operator D. All of these

Very Short Answer Type Questions

Q.1 What is operator overloading?

Q.2 Write the prototype of the operator function.

Q.3 What are the operators that can't be overloaded?

Short Answer Type Questions

Q.1 Explain the difference between the operator function implemented as member function and friend function.

Essay Type Questions

Q.1 Write a program to overload unary minus operator to negate an object of a class using friend function.

Q.2 Write a program to overload binary plus operator to concatenate two strings using member function.

Answer Key

1. C 2. B 3. B 4. D