Operator overloading

operator overloading provides a flexible option for the creation of new definition for most of the c++ operators

We can overload all the c++ operators except the following:

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

The process of overloading involves the following steps:

1.Create a class that defines the data type that is to be used in the overloading operation.

2.Declare the operator function op( ) in the public part of the class. It may be either a member function or friend function.

3.Define the operator function to implement the required operations.

Overloading unary operators

```
# include < iostream.h>
Class space
{
   int x, y, z;
   public:
   void getdata ( int a, int b, int c);
   void display (void);
   void operator – ( ); // overload unary minus
};
 void space:: getdata (int a, int b, int c)
{
  x=a;
  y=b;
  z=c;
}
Void space:: display( void)
{ cout<<x<<“ ”;
    cout<<y<<“ ”;
   cout<<z<<“\n”;
 }
```

```
 void space :: operator – ( )
{
  x= -x;
  y= -y;
  z= -z;
}
Int main( )
{   space S;
     S.getdata (10,-20,30);
     cout << “S :”;
     S.display( );
    -S;
    cout<<“ S:”;
    S.display( );
 return 0;
 }
```

output

```
S: 10 -20 30
S: -10 20 -30
```

Overloading binary operators

```
# include < iostream.h>
 class complex
{  float x, y;
    public :
    complex ( ) { }
   complex ( float real, float imag)
   { x= real, y= imag; }
  complex operator + (complex);
  void display (void) ;
  }
 complex complex : : operation + (complex c)
  (
    complex temp ;
    temp.x = x+ c.x
    temp.y = y+ c.y
    return (temp);
 }
```

```
 void complex :: display (void)
{
   cout<< x<< "+i" << y <<"\n";
}
  int main( )
{
  complex c1,c2,c3;
   c1= complex (2.5, 3.5);
   c2= complex (1.6,2.7);
   c3= c1 + c2;
   cout << "c1 =";c1.display( );
   cout<< "c2 =";c2.display( );
   cout << "c3=";c3.display( );
  return 0;
}
```

Output:

```
 c1 = 2.5+ i 3.5
 c2 = 1.6 + i 2.7
 c3= 4.1 +  i 6.2
------------------------
 c3 = c1 + c2
 c3= c1.operator+( c2)
```

Overloading with friend functions

Syntax:

friend Return Type operator operator symbol (arg1,[arg2])

friend function play a very important role in operator overloading. They allow overloading of stream operators (<< or>>) for stream computation on user defined data types.

Input/output stream classes defined is iostream.h

friend { ostream or istream} & operator {<< or >>}

(ostream & out , istream & in }, arg)

cin / cout

User defined object

```
# include < iostream.h >
  const size = 3;
 class vector
  {
     int v[ size];
    public:
        vector ( );
        vector ( int * x );
       friend vector operator * ( int a, vector b);
       friend vector operator* (vector b, int a);
       friend istream & operator >> ( istream & , vector &) ;
} ;
 vector :: vector ( )
 { for ( int i=0, i< size; i++ )
         v[i] = 0;
}
 vector : : vector ( int * x)
```

```
{ for ( int i=0, i<size, i++)
     v[i] = x[i];
}
 vector operator * ( int a, vector b)
{ vector c;
   for ( int i=0, I < size, i++)
     c.v[i] = a* b.v[i];
   return c;
}
  vector operator * ( vector b, int a)
{ vector c;
  for ( int i=0, i< size; i++)
    c.v[i]= b.v[i]*a;
  return c;
}
istream & operator >> ( istream & din, vector & b)
```

```
{
  for ( int i=0;i< size; i++)
      din>> b.v[i];
   return ( din);
}
 ostream & operator << ( ostream & dout, vector & b)
{
  dout << " ( "<< b.v[0];
  for ( int i= 1; i< size; i++)
      dout << "," << b.v[i];
  dout << " )";
  return ( dout);
}
  int x [size ]= {2,4,6};
  int main ( )
{ vector m; // invokes constructor 1
   vector n= x;  // invokes constructor 2
   cout << " enter elements of vector m "<< "\n";
   cin >> m; // invokes operator >> ( ) function
```

```
 cout << "\n";
 cout <<" m=" << m<< "\n"; // invokes operator << ( )
 vector p,q ;
  p= 2*m //invokes friend 1
  q = n* 2 // invokes friend 2
  cout << "\n";
 cout <<"p="<<p<<"\n"; // invokes operator << ( )
 cout << "q =" << q<< " \n";
  return 0;
}
```

Output:

```
enter elements of vector m
 5 10 15
 m=( 5,10,15)
 p= ( 10,20,30)
 q= (4,8,12)
```