Chapter 10

Operator Overloading

Chapter 10 Topics(part 1)

- Operator Overloaded as Member Functions
  - Binary Operator Overloading
  - Unary Operator Overloading
- Operator Overloaded as Friend Functions
  - Binary Operator Overloading
  - Unary Operator Overloading
- Member Function vs. Friend Function

Binary Operator Overloading

SYNTAX

```
class ClassName{
 public:
    DataType operator @ (Parameter List);
 ...
 };
DataType ClassName::operator @ (Parameter List)
{
 ...
}
```

INVOKING

```
aa @ bb    //or
aa . operator @ (bb)
```

Add Two Complex Numbers

```
#include <iostream>
using namespace std;
class Complex{
public:
    Complex( ){real=0;imag=0;}
    Complex(double r,double i){real=r;imag=i;}
    Complex operator + (Complex & c);
    void display( );
private:
    double real;
    double imag;
};
Complex Complex::operator + (Complex & c)
{
    Complex temp;
    temp.real=real+c.real;
    temp.imag=imag+c.imag;
    return temp;
}
```

```
void Complex::display( )
{
    cout<<"( "<<real<<" , "<<imag<<"i )"
        <<endl;
}
int main( )
{   Complex c1(3 , 4), c2(5 , -10), c3,c4;
    c3=c1+c2;
    c4=c1.operator+(c2);
    cout<<"c1=";
    c1.display( );
    cout<< "c2=";
    c2.display( );
    cout<< "c1+c2=";
    c3.display( );
    cout<< "c1.operator+(c2)=";
    c4.display( );
    return 0;
}
```

Unary Operator Overloading

SYNTAX

```
class ClassName{
 public:
    DataType operator @ (  );
 ...
 };
DataType ClassName::operator @ (  )
{
 ...
}
```

INVOKING

```
@ aa    //or
aa.operator @ ( )
```

Prefix Increment Operator

```
#include <iostream>
using namespace std;
class Complex{
public:
   Complex( ){real=0;imag=0;}
   Complex(double r,double i){real=r;imag=i;}
   Complex operator ++ ( );
   void display( );
private:
   double real;
   double imag;
};
Complex Complex::operator ++ ( )
{
   ++real;
   ++imag;
   return *this;
}
```

```
void Complex::display( )
{
   cout<<"( "<<real<<" , "<<imag<<"i )"
         <<endl;
}
int main( )
{
   Complex c(3 , 4);
   cout<<"c: ";
   c.display( );
   ++c;
   cout<< "++c: ";
   c.display( );
   c.operator++( );
   cout<< "c.operator++( ) : ";
   c.display( );
   return 0;
}
```

Postfix Increment Operator

```
#include <iostream>
using namespace std;
class Complex{
public:
    Complex( ){real=0;imag=0;}
    Complex(double r,double i){real=r;imag=i;}
    Complex operator ++ ( int );
    void display( );
private:
    double real;
    double imag;
};
Complex Complex::operator ++ ( int )
{
    Complex temp(*this);
    real ++;
    imag ++;
    return temp;
}
```

```
void Complex::display( )
{
    cout<<"( "<<real<<" , "<<imag<<"i )"
        <<endl;
}
int main( )
{
    Complex c1(3 , 4),c2;
    cout<<"c1: ";
    c1.display( );
    c2=c1++; //or c2=c1.operator++(0);
    cout<< "c1 ++ : ";
    c1.display( );
    cout<<"c2: ";
    c2.display( );
    return 0;
}
```

Chapter 10 Topics(part 1)

- Operator Overloaded as Member Functions
  - Binary Operator Overloading
  - Unary Operator Overloading
- Operator Overloaded as Friend Functions
  - Binary Operator Overloading
  - Unary Operator Overloading
- Member Function vs. Friend Function

Binary Operator Overloading

SYNTAX

```
class ClassName{
 public:
    friend DataType operator @ (Parameter List);
 ...
 };
DataType operator @ (Parameter List)
{
 ...
}
```

INVOKING

```
aa @ bb    //or
operator @ (aa , bb)
```

Add Two Complex Numbers

```
#include <iostream>
using namespace std;
class Complex{
public:
    Complex( ){real=0;imag=0;}
    Complex(double r,double i){real=r;imag=i;}
   friend Complex operator+(Complex &,Complex &);
    void display( );
private:
    double real;
    double imag;
};
Complex operator+(Complex &c1, Complex &c2)
{
    Complex temp;
    temp.real=c1.real+c2.real;
    temp.imag=c1.imag+c2.imag;
    return temp;
}
```

```
void Complex::display( )
{
    cout<<"( "<<real<<" , "<<imag<<"i )"
          <<endl;
}
int main( )
{   Complex c1(3 , 4), c2(5 , -10), c3,c4;
    c3=c1+c2;
    c4=operator+(c1, c2);
    cout<<"c1=";
    c1.display( );
    cout<< "c2=";
    c2.display( );
    cout<< "c1+c2=";
    c3.display( );
    cout<< "operator+(c1,c2)=";
    c4.display( );
    return 0;
}
```

Unary Operator Overloading

SYNTAX

```
class ClassName{
 public:
    friend DataType operator @ (Parameter List);
 ...
 };
DataType operator @ (Parameter List )
{
 ...
}
```

INVOKING

```
@ aa    //or
operator @ (aa )
```

Prefix Increment Operator

```
#include <iostream>
using namespace std;
class Complex{
public:
   Complex( ){real=0;imag=0;}
   Complex(double r,double i){real=r;imag=i;}
   friend Complex operator ++ (Complex & );
   void display( );
private:
   double real;
   double imag;
};
Complex operator ++ (Complex &c )
{
   ++c.real;
   ++c.imag;
   return c;
}
```

```
void Complex::display( )
{
   cout<<"( "<<real<<" , "<<imag<<"i )"
         <<endl;
}
int main( )
{
   Complex c(3 , 4);
   cout<<"c: ";
   c.display( );
   ++c;
   cout<< "++c: ";
   c.display( );
   operator++( c );
   cout<< "operator++(c ) : ";
   c.display( );
   return 0;
}
```

Postfix Increment Operator

```
#include <iostream>
using namespace std;
class Complex{
public:
    Complex( ){real=0;imag=0;}
    Complex(double r,double i){real=r;imag=i;}
    friend Complex operator ++ ( Complex & , int );
    void display( );
private:
    double real;
    double imag;
};
Complex operator ++ (Complex & c, int )
{
    Complex temp(c);
    c.real ++;
    c.imag ++;
    return temp;
}
```

```
void Complex::display( )
{
    cout<<"( "<<real<<" , "<<imag<<"i )"
        <<endl;
}
int main( )
{
    Complex c1(3 , 4),c2;
    cout<<"c1: ";
    c1.display( );
    c2=c1++;     //or c2=operator++(c1, 0);
    cout<< "c1 ++ : ";
    c1.display( );
    cout<<"c2: ";
    c2.display( );
    return 0;
}
```

Chapter 10 Topics(part 1)

- Operator Overloaded as Member Functions
  - Binary Operator Overloading
  - Unary Operator Overloading
- Operator Overloaded as Friend Functions
  - Binary Operator Overloading
  - Unary Operator Overloading
- Member Function vs. Friend Function

Parameter List

Number of Parameters	Member Function	Friend Function
Binary Operator	1	2
Unary Operator	0	1

Invoking

Implicit Format	Explicit Format	
	Member Function	Friend Function
a+b	a.operator+(b)	operator+(a,b)
++a	a.operator++( )	operator++(a)
a++	a.operator++(0)	operator++(a,0)

Rules for Operator Overloading

- ✓ can not define new operators
- ✓ can not change the number of operands
- ✓ can not change the precedence
- ✓ can not change the associativity（结合性）
- ✓ can not have default parameters
- ✓ “=”, “( )”, “[ ]” can only be overloaded as member functions
- ✓ “.”, “.*”, “::”, “sizeof”, “?:” can not be overloaded