Lecture-27

Class – operator overloading

Sum two complex numbers

```
1  #include <iostream>
2  #include "complex.h"
3  using namespace std;
4  
5  =int main() {
6     complex c1(1.2, 3.2);
    complex c2 (-2.1, 4.5);
    complex c3;
10     c3 = sum(c1, c2);
}
```

To accomplish this we need to overload the operator (+) to handle our user defined data type complex.

Instead of a function call sum(c1,c2), we would like to do sum naturally as c1 + c2 and assign the result to c3.

```
#include <iostream>
#include "complex.h"
using namespace std;

int main() {

complex c1(1.2, 3.2);
complex c2 (-2.1, 4.5);
complex c3;

c3 = c1 + c2;
}
```

```
□complex operator+(complex& c1, complex& c2) {
                                                                 Write a global function named
42
           complex result;
                                                                 operator followed by symbol of
43
           result.Re = c1.Re + c2.Re;
                                                                 the operator. Pass the two
44
           result.Im = c1.Im + c2.Im;
                                                                 complex numbers as arguments
45
           return result;
                                                                 and return the resulting complex
46
                                                                 number by value.
      #ifndef COMPLEX H
      #define COMPLEX H
      #include <iostream>
                                                 Add the function declaration to the class as friend.
 4
      using namespace std;
                                                 The output is same as before. Here the statement
 5
                                                 c3 = c1 + c2 is interpreted as c3 = operator + (c1, c2).
    □class complex {
 7
          private:
 8
              double Re, Im;
 9
              static int counter;
10
          public:
              complex();
                                            // Constructor function
11
12
              complex(double, double); //
                                               Overloading constructor
13
              ~complex();
                                            // Destructor function
14
15
              static int getCounterValue();
16
              friend complex sum (complex& c1, complex& c2);
              friend complex operator+(complex& c1, complex& c2);
17
     L } ;
18
19
20
      #endif // COMPLEX H
```

- The precedence of an operator can not be changed.
- The associativity of an operator can not be changed.
- The arity (binary or unary) of the operator can not be changed.
- No new operators can be defined.
- Meaning of operator remains same.
- Each operator must be overloaded separately.
- Operator overloading can be done using class member and non-member functions
 - (), [], -> and any assignment operator must be overloaded as member function only.
- Member access (.), pointer member access (.*), scope resolution (: :) and ternary (?:) operators can not be overloaded.

22

```
□complex complex::operator+(complex& c1) {
                                                                     Write a member function
50
            complex result;
                                                                     named operator followed by
51
            result.Re = this->Re + c1.Re;
                                                                     symbol of the operator. Pass the
52
            result.Im = this->Im + c1.Im;
                                                                     second complex number as
53
            return result;
                                                                     argument and return the
54
                                                                     resulting complex number by
                                                                     value.
      #ifndef COMPLEX H
 2
      #define COMPLEX H
                                                    We have used the this pointer which the compiler
 3
      #include <iostream>
                                                    implicitly creates for each object. This pointer
 4
      using namespace std;
 5
                                                    points at the address of the calling object.
    □class complex {
 7
          private:
                                                                           Add the function
 8
              double Re, Im;
 9
              static int counter;
                                                                           declaration to the class.
10
          public:
                                                                           The output is same as
                                           // Constructor function
11
              complex();
                                                                           before. Here the
              complex(double, double);
                                          // Overloading constructor
12
                                                                           statement c3 = c1 + c2 is
              ~complex();
                                           // Destructor function
13
14
                                                                           interpreted as c3 =
15
              static int getCounterValue();
                                                                           c1.operator+(c2).
16
              friend complex sum (complex& c1, complex& c2);
               //friend complex operator+(complex& c1, complex& c2);
17
              complex operator+(complex& c1);
18
                                                            Calling object is one on the left of operator
19
20
21
      #endif // COMPLEX H
```

```
#include <iostream>
                                                                         Now we would like to output
          #include "complex.h"
                                                                         the complex numbers in the
          using namespace std;
                                                                         format (Re) + i (Im) by just
    4
                                                                         streaming the object to cout
    5
        □int main() {
                                                                         object.
                complex c1(1.2, 3.2);
                complex c2 (-2.1, 4.5);
    9
                complex c3;
               c3 = c1 + c2;
  10
                cout << "Result is " << c3 << endl;</pre>
others
            X Search results
                             X / Cccc X
                                         Build log
                                                       Build messages X
                                                                        CppCheck/Vera++
 Code::Blocks
             Line
                   Message
                   === Build: Debug in complex (compiler: GNU GCC Compiler) ===
                   In function 'int main()':
CHN-103\Lec...
CHN-103\Lec... 11
                   error: no match for 'operator<<' (operand types are 'std::basic_ostream<char>' and 'co..
Program Fil... 108
                   note: candidate: std::basic_ostream<_CharT, _Traits>::_ostream_type& std::basic_ostre..
```

Error is produced because the *operator*<< does not know how to handle the object on the right. We have to enable it by overloading the operator. The *cout* object should appear on left of operator and our user defined should appear on right. Therefore, it has to be done using non-member function only.

```
56
     ⊟ostream &operator<< (ostream &out, const complex& c) {</p>
            out << "(" << c.Re << ") +i (" << c.Im << ")";
57
58
            return out;
59
                                                   Write a global function named operator <<.
                                                   Pass the ostream& object and complex&
      #ifndef COMPLEX H
                                                   object as arguments and return ostream& as
     #define COMPLEX H
                                                   output. Returning the reference to same
      #include <iostream>
      using namespace std;
                                                   ostream object enable cascading of stream.
    □class complex {
                                                   The global function is added to class
          private:
                                                   definition as a friend to enable accessing the
              double Re, Im;
                                                   private data members.
 9
              static int counter;
          public:
10
11
              complex();
                                            // Constructor function
12
                                           // Overloading constructor
              complex(double, double);
              ~complex();
                                            // Destructor function
13
14
15
              static int getCounterValue();
              friend complex sum (complex& c1, complex& c2);
16
17
              //friend complex operator+(complex& c1, complex& c2);
              complex operator+(complex& c1);
18
19
              friend ostream &operator<<(ostream &out, const complex& c);</pre>
20
21
22
      #endif // COMPLEX H
```

```
#include <iostream>
                                                                Now we would like to input
 2
      #include "complex.h"
                                                                the complex numbers in the
 3
      using namespace std;
                                                                format Re +i Im by just
 4
                                                                streaming input stream from
     □int main(){
                                                                cin to the object.
 6
 7
           complex c1(1.2, 3.2);
           complex c2;
 9
           cout << "Enter the second complex no. as Re +i Im: ";</pre>
10
           cin >> c2;
11
           complex c3;
12
           c3 = c1 + c2;
13
           cout << "Result is " << c3 << endl;</pre>
14
15
                                                                Write a global function
```

named operator>>. Pass the istream& object and complex& object as arguments and return istream& as output. Returning the reference to same istream object enable cascading of stream.

```
"E:\CHN-103\Lectures\L24 Class-static and friend keyword\complex\bin\Debug\complex.exe"
Parameterized Constructor is called for (1.2,3.2) # 1
Default Constructor is called for (0,0) # 2
Enter the second complex no. as Re + i Im: -2.6 + i 1.254
Default Constructor is called for (0,0) # 3
Default Constructor is called for (0,0) # 4
Destructor is called for (-1.4,4.454) # 4
Result is (-1.4) +i (4.454)
Destructor is called for (-1.4,4.454) # 3
Destructor is called for (-2.6,1.254) # 2
Destructor is called for (1.2,3.2) # 1
Process returned 0 (0x0) execution time : 16.463 s
Press any key to continue.
```