C++语言作业

17 计基 杨添宝 320170941671

复数类

```
   /* Complex2.h */
 1 #ifndef _COMPLEX2_H_
 2 #define _COMPLEX2_H_
 3 #include <iostream>
 4
 5 class Complex {
 6 private:
 7     float m_real;   // real part
 8     float m_imag;   // imaginary part
 9 public:
10     // constructors
11     Complex(float, float);
12     Complex(float = 0);         // conversion
   constructor
13     Complex(const Complex&);    // copy constructor
14     // getters, setters
15     void setReal(float);
16     float getReal() const;
17     void setImag(float);
18     float getImag() const;
19     // operator overloading
20     explicit operator bool() const; // safe bool,
   C++11 support is required
21     float operator() () const;  // get mold
22     bool operator== (const Complex&) const;
23     bool operator!= (const Complex&) const;
```

```
24      Complex operator+ (const Complex&) const;   //
    addition
25      Complex operator- (const Complex&) const;   //
    substract
26      Complex operator* (const Complex&) const;   //
    multiplication
27      Complex operator/ (const Complex&) const;   //
    division
28      Complex operator+ () const; // plus
29      Complex operator- () const; // minus
30      Complex& operator+= (const Complex&);
31      Complex& operator-= (const Complex&);
32      Complex& operator*= (const Complex&);
33      Complex& operator/= (const Complex&);
34      // friend function
35      friend std::ostream& operator<< (std::ostream&,
    const Complex&);    // output
36      friend Complex operator+ (float, const
    Complex&);
37      friend Complex operator- (float, const
    Complex&);
38      friend Complex operator* (float, const
    Complex&);
39      friend Complex operator/ (float, const
    Complex&);
40  };
41
42  #endif /* _COMPLEX2_H_ */
```

```
   /* Complex.cpp */
 1 #include "Complex2.h"
 2 #include <math.h>
 3
 4 Complex::Complex(float real, float imag)
 5     : m_real(real)
 6     , m_imag(imag)
 7 {}
 8 Complex::Complex(float real)
 9     : m_real(real)
10     , m_imag(0)
11 {}
12 Complex::Complex(const Complex& right)
13     : m_real(right.m_real)
14     , m_imag(right.m_imag)
15 {}
16 void Complex::setReal(float real) { m_real = real; }
17 float Complex::getReal() const { return m_real; }
18 void Complex::setImag(float imag) { m_imag = imag; }
19 float Complex::getImag() const { return m_imag; }
20 Complex::operator bool() const {
21     return m_real != 0 || m_imag != 0;
22 }
23 float Complex::operator() () const {
24     return sqrt(m_real * m_real + m_imag * m_imag);
25 }
26 bool Complex::operator== (const Complex& right)
   const {
27     return m_real == right.m_real && m_imag ==
   right.m_imag;
28 }
29 bool Complex::operator!= (const Complex& right)
   const {
30     return m_real != right.m_real || m_imag !=
   right.m_imag;
31 }
32 Complex Complex::operator+ (const Complex& right)
   const {
```

```
33      Complex left(right);
34      left.m_real += m_real;
35      left.m_imag += m_imag;
36      return left;
37  }
38  Complex Complex::operator- (const Complex& right)
    const {
39      Complex left;
40      left.m_real = m_real - right.m_real;
41      left.m_imag = m_imag - right.m_imag;
42      return left;
43  }
44  Complex Complex::operator* (const Complex& right)
    const {
45      Complex left;
46      left.m_real = m_real * right.m_real - m_imag *
    right.m_imag;
47      left.m_imag = m_real * right.m_imag + m_imag *
    right.m_real;
48      return left;
49  }
50  Complex Complex::operator/ (const Complex& right)
    const {
51      float denominator = right.m_real * right.m_real
    + right.m_imag * right.m_imag;
52      if(denominator == 0)
53          throw "mold of 'right' cannot be zero";
54      Complex left;
55      left.m_real = (m_real * right.m_real + m_imag *
    right.m_imag) / denominator;
56      left.m_imag = (m_imag * right.m_real - m_real *
    right.m_imag) / denominator;
57      return left;
58  }
59  Complex Complex::operator+ () const {
60      return Complex(*this);
61  }
62  Complex Complex::operator- () const {
```

```
63      return Complex(-this->m_real, -this->m_imag);
64  }
65  Complex& Complex::operator+= (const Complex& right)
    {
66      Complex left = *this + right;
67      m_real = left.m_real;
68      m_imag = left.m_imag;
69      return *this;
70  }
71  Complex& Complex::operator-= (const Complex& right)
    {
72      Complex left = *this - right;
73      m_real = left.m_real;
74      m_imag = left.m_imag;
75      return *this;
76  }
77  Complex& Complex::operator*= (const Complex& right)
    {
78      Complex left = *this * right;
79      m_real = left.m_real;
80      m_imag = left.m_imag;
81      return *this;
82  }
83  Complex& Complex::operator/= (const Complex& right)
    {
84      Complex left = *this / right;
85      m_real = left.m_real;
86      m_imag = left.m_imag;
87      return *this;
88  }
89  std::ostream& operator<< (std::ostream &os, const
    Complex &c) {
90      if (c.m_real != 0) {
91          os << c.m_real;
92          if (c.m_imag > 0) os << '+';
93      }
94      if (c.m_imag != 0) {
95          if (c.m_imag == 1) ;
```

```
96          else if (c.m_imag == -1) os << '-';
97          else os << c.m_imag;
98          os << 'i';
99      }
100     return os;
101 }
102 Complex operator+ (float left, const Complex
    &right) {
103     Complex temp(right);
104     temp.m_real += left;
105     return temp;
106 }
107 Complex operator- (float left, const Complex
    &right) {
108     Complex temp(right);
109     temp.m_real -= left;
110     temp.m_imag = -temp.m_imag;
111     return temp;
112 }
113 Complex operator* (float left, const Complex
    &right) {
114     Complex temp(right);
115     temp.m_real *= left;
116     temp.m_imag *= left;
117     return temp;
118 }
119 Complex operator/ (float left, const Complex
    &right) {
120 //    float denominator = right.m_real *
    right.m_real + right.m_imag * right.m_imag;
121 //    if(denominator == 0)
122 //        throw "mold of 'right' cannot be zero";
123 //    Complex temp(right);
124 //    temp.m_real /= (denominator / left);
125 //    temp.m_imag /= (-denominator / left);
126 //    return temp;
127     return Complex(left) / right;
128 }
```