C++语言作业

17 计基 杨添宝 320170941671

复数类

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | /\* Complex2.h \*/  #ifndef \_COMPLEX2\_H\_  #define \_COMPLEX2\_H\_  #include <iostream>  class Complex {  private:  float m\_real; // real part  float m\_imag; // imaginary part  public:  // constructors  Complex(float, float);  Complex(float = 0); // conversion constructor  Complex(const Complex&); // copy constructor  // getters, setters  void setReal(float);  float getReal() const;  void setImag(float);  float getImag() const;  // operator overloading  explicit operator bool() const; // safe bool, C++11 support is required  float operator() () const; // get mold  bool operator== (const Complex&) const;  bool operator!= (const Complex&) const;  Complex operator+ (const Complex&) const; // addition  Complex operator- (const Complex&) const; // substract  Complex operator\* (const Complex&) const; // multiplication  Complex operator/ (const Complex&) const; // division  Complex operator+ () const; // plus  Complex operator- () const; // minus  Complex& operator+= (const Complex&);  Complex& operator-= (const Complex&);  Complex& operator\*= (const Complex&);  Complex& operator/= (const Complex&);  // friend function  friend std::ostream& operator<< (std::ostream&, const Complex&); // output  friend Complex operator+ (float, const Complex&);  friend Complex operator- (float, const Complex&);  friend Complex operator\* (float, const Complex&);  friend Complex operator/ (float, const Complex&);  };  #endif /\* \_COMPLEX2\_H\_ \*/ |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128 | /\* Complex.cpp \*/  #include "Complex2.h"  #include <math.h>  Complex::Complex(float real, float imag)  : m\_real(real)  , m\_imag(imag)  {}  Complex::Complex(float real)  : m\_real(real)  , m\_imag(0)  {}  Complex::Complex(const Complex& right)  : m\_real(right.m\_real)  , m\_imag(right.m\_imag)  {}  void Complex::setReal(float real) { m\_real = real; }  float Complex::getReal() const { return m\_real; }  void Complex::setImag(float imag) { m\_imag = imag; }  float Complex::getImag() const { return m\_imag; }  Complex::operator bool() const {  return m\_real != 0 || m\_imag != 0;  }  float Complex::operator() () const {  return sqrt(m\_real \* m\_real + m\_imag \* m\_imag);  }  bool Complex::operator== (const Complex& right) const {  return m\_real == right.m\_real && m\_imag == right.m\_imag;  }  bool Complex::operator!= (const Complex& right) const {  return m\_real != right.m\_real || m\_imag != right.m\_imag;  }  Complex Complex::operator+ (const Complex& right) const {  Complex left(right);  left.m\_real += m\_real;  left.m\_imag += m\_imag;  return left;  }  Complex Complex::operator- (const Complex& right) const {  Complex left;  left.m\_real = m\_real - right.m\_real;  left.m\_imag = m\_imag - right.m\_imag;  return left;  }  Complex Complex::operator\* (const Complex& right) const {  Complex left;  left.m\_real = m\_real \* right.m\_real - m\_imag \* right.m\_imag;  left.m\_imag = m\_real \* right.m\_imag + m\_imag \* right.m\_real;  return left;  }  Complex Complex::operator/ (const Complex& right) const {  float denominator = right.m\_real \* right.m\_real + right.m\_imag \* right.m\_imag;  if(denominator == 0)  throw "mold of 'right' cannot be zero";  Complex left;  left.m\_real = (m\_real \* right.m\_real + m\_imag \* right.m\_imag) / denominator;  left.m\_imag = (m\_imag \* right.m\_real - m\_real \* right.m\_imag) / denominator;  return left;  }  Complex Complex::operator+ () const {  return Complex(\*this);  }  Complex Complex::operator- () const {  return Complex(-this->m\_real, -this->m\_imag);  }  Complex& Complex::operator+= (const Complex& right) {  Complex left = \*this + right;  m\_real = left.m\_real;  m\_imag = left.m\_imag;  return \*this;  }  Complex& Complex::operator-= (const Complex& right) {  Complex left = \*this - right;  m\_real = left.m\_real;  m\_imag = left.m\_imag;  return \*this;  }  Complex& Complex::operator\*= (const Complex& right) {  Complex left = \*this \* right;  m\_real = left.m\_real;  m\_imag = left.m\_imag;  return \*this;  }  Complex& Complex::operator/= (const Complex& right) {  Complex left = \*this / right;  m\_real = left.m\_real;  m\_imag = left.m\_imag;  return \*this;  }  std::ostream& operator<< (std::ostream &os, const Complex &c) {  if (c.m\_real != 0) {  os << c.m\_real;  if (c.m\_imag > 0) os << '+';  }  if (c.m\_imag != 0) {  if (c.m\_imag == 1) ;  else if (c.m\_imag == -1) os << '-';  else os << c.m\_imag;  os << 'i';  }  return os;  }  Complex operator+ (float left, const Complex &right) {  Complex temp(right);  temp.m\_real += left;  return temp;  }  Complex operator- (float left, const Complex &right) {  Complex temp(right);  temp.m\_real -= left;  temp.m\_imag = -temp.m\_imag;  return temp;  }  Complex operator\* (float left, const Complex &right) {  Complex temp(right);  temp.m\_real \*= left;  temp.m\_imag \*= left;  return temp;  }  Complex operator/ (float left, const Complex &right) {  // float denominator = right.m\_real \* right.m\_real + right.m\_imag \* right.m\_imag;  // if(denominator == 0)  // throw "mold of 'right' cannot be zero";  // Complex temp(right);  // temp.m\_real /= (denominator / left);  // temp.m\_imag /= (-denominator / left);  // return temp;  return Complex(left) / right;  } |