C++语言作业

17 计基 杨添宝 320170941671

一、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 | /\* Complex.h \*/  #ifndef \_COMPLEX\_H\_  #define \_COMPLEX\_H\_  class Complex {  private:  double real; // real part  double imaginary; // imaginary part  public:  Complex(double = 0, double = 0);  Complex(const Complex&); // copy constructor  //getter, setter  void setRealPart(double);  double getRealPart() const;  void setImaginaryPart(double);  double getImaginaryPart() const;  Complex Add(const Complex&) const;  Complex Sub(const Complex&) const;  Complex Mul(const Complex&) const;  Complex Div(const Complex&) const;  inline double Mold() const;  //operator overloading  inline bool operator== (const Complex&) const;  inline bool operator!= (const Complex&) const;  inline Complex operator+ (const Complex&);  inline Complex operator- (const Complex&);  inline Complex operator\* (const Complex&);  inline Complex operator/ (const Complex&);  Complex& operator+= (const Complex&);  Complex& operator-= (const Complex&);  Complex& operator\*= (const Complex&);  Complex& operator/= (const Complex&);  };  #endif |
| 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 | /\* Complex.cpp \*/  #include "Complex.h"  #include <math.h>  Complex::Complex(double real\_part, double imaginary\_part)  : real(real\_part)  , imaginary(imaginary\_part)  {}  Complex::Complex(const Complex& a)  : real(a.real)  , imaginary(a.imaginary)  {}  void Complex::setRealPart(double r) { real = r; }  double Complex::getRealPart() const { return real; }  void Complex::setImaginaryPart(double i) { imaginary = i; }  double Complex::getImaginaryPart() const { return imaginary; }  Complex Complex::Add(const Complex& a) const {  Complex b(a);  b.real += real;  b.imaginary += imaginary;  return b;  }  Complex Complex::Sub(const Complex& a) const {  Complex b;  b.real = real - a.real;  b.imaginary = imaginary - a.imaginary;  return b;  }  Complex Complex::Mul(const Complex& a) const {  Complex b;  b.real = real \* a.real - imaginary \* a.imaginary;  b.imaginary = real \* a.imaginary + imaginary \* a.real;  return b;  }  Complex Complex::Div(const Complex& a) const {  int denominator = a.real \* a.real + a.imaginary \* a.imaginary;  // if(denominator == 0)  // throw "mold of 'a' cannot be zero";  Complex b;  b.real = (real \* a.real + imaginary \* a.imaginary) / denominator;  b.imaginary = (imaginary \* a.real - real \* a.imaginary) / denominator;  return b;  }  double Complex::Mold() const {  return sqrt(real \* real + imaginary \* imaginary);  }  bool Complex::operator== (const Complex& a) const {  return real == a.real && imaginary == a.imaginary;  }  bool Complex::operator!= (const Complex& a) const {  return real != a.real || imaginary != a.imaginary;  }  Complex Complex::operator+ (const Complex& a) { return Add(a); }  Complex Complex::operator- (const Complex& a) { return Sub(a); }  Complex Complex::operator\* (const Complex& a) { return Mul(a); }  Complex Complex::operator/ (const Complex& a) { return Div(a); }  Complex& Complex::operator+= (const Complex& a) {  Complex temp = this->Add(a);  real = temp.real;  imaginary = temp.imaginary;  return \*this;  }  Complex& Complex::operator-= (const Complex& a) {  Complex temp = this->Sub(a);  real = temp.real;  imaginary = temp.imaginary;  return \*this;  }  Complex& Complex::operator\*= (const Complex& a) {  Complex temp = this->Mul(a);  real = temp.real;  imaginary = temp.imaginary;  return \*this;  }  Complex& Complex::operator/= (const Complex& a) {  Complex temp = this->Div(a);  real = temp.real;  imaginary = temp.imaginary;  return \*this;  } |

二、链表类

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | /\* LinkList.h \*/  #ifndef \_LINKLIST\_H\_  #define \_LINKLIST\_H\_  struct NODE {  char ch;  struct NODE \*next;  };  class LinkList {  private:  NODE \*head;  public:  LinkList();  ~LinkList();  bool Insert(int, char); //insert before i, i start from 1  bool Delete(int); //start from 1  bool Delete(char);  void Display();  };  #endif |

|  |  |
| --- | --- |
| 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 | /\* LinkList.cpp \*/  #include "LinkList.h"  #include <iostream>  using namespace std;  LinkList::LinkList() {  head = new NODE;  head->next = NULL;  }  LinkList::~LinkList() {  NODE \*pWork;  while(head != NULL)  {  pWork = head;  head = head->next;  delete pWork;  }  }  bool LinkList::Insert(int pos, char c) {  int i = 0;  NODE \*p = head, \*s;  while(p && i < pos - 1) {  p = p->next;  i++;  } //find the i-1 node  if(!p || i > pos - 1) return false;  s = new NODE;  s->ch = c;  s->next = p->next;  p->next = s;  return true;  }  bool LinkList::Delete(int pos) {  NODE \*p = head, \*q;  int i = 0;  while(p->next && i < pos - 1) {  p = p->next;  i++;  } //find the i node  if(!p->next || i > pos - 1) return false;  q = p->next;  p->next = q->next;  delete q;  return true;  }  bool LinkList::Delete(char c) {  NODE \*p = head, \*q;  while(p->next && p->next->ch != c) { p = p->next; }  if(!p->next) return false;  q = p->next;  p->next = q->next;  delete q;  return true;  }  void LinkList::Display() {  NODE \*p = head->next;  while(p) { cout << p->ch; p = p->next; }  } |