C++语言作业

17 计基 杨添宝 320170941671

一、复数类

```
   /* Complex.h */
 1 #ifndef _COMPLEX_H_
 2 #define _COMPLEX_H_
 3
 4 class Complex {
 5 private:
 6     double real;    // real part
 7     double imaginary;   // imaginary part
 8 public:
 9     Complex(double = 0, double = 0);
10     Complex(const Complex&);    // copy constructor
11     //getter, setter
12     void setRealPart(double);
13     double getRealPart() const;
14     void setImaginaryPart(double);
15     double getImaginaryPart() const;
16     Complex Add(const Complex&) const;
17     Complex Sub(const Complex&) const;
18     Complex Mul(const Complex&) const;
19     Complex Div(const Complex&) const;
20     inline double Mold() const;
21     //operator overloading
22     inline bool operator== (const Complex&) const;
23     inline bool operator!= (const Complex&) const;
24     inline Complex operator+ (const Complex&);
25     inline Complex operator- (const Complex&);
26     inline Complex operator* (const Complex&);
27     inline Complex operator/ (const Complex&);
28     Complex& operator+= (const Complex&);
29     Complex& operator-= (const Complex&);
30     Complex& operator*= (const Complex&);
31     Complex& operator/= (const Complex&);
32 };
33
34 #endif
```

```
   /* Complex.cpp */
 1 #include "Complex.h"
 2 #include <math.h>
 3
 4 Complex::Complex(double real_part, double
   imaginary_part)
 5                 : real(real_part)
 6                 , imaginary(imaginary_part)
 7 {}
 8 Complex::Complex(const Complex& a)
 9                 : real(a.real)
10                 , imaginary(a.imaginary)
11 {}
12 void Complex::setRealPart(double r) { real = r; }
13 double Complex::getRealPart() const { return real; }
14 void Complex::setImaginaryPart(double i) { imaginary
   = i; }
15 double Complex::getImaginaryPart() const { return
   imaginary; }
16 Complex Complex::Add(const Complex& a) const {
17     Complex b(a);
18     b.real += real;
19     b.imaginary += imaginary;
20     return b;
21 }
22 Complex Complex::Sub(const Complex& a) const {
23     Complex b;
24     b.real = real - a.real;
25     b.imaginary = imaginary - a.imaginary;
26     return b;
27 }
28 Complex Complex::Mul(const Complex& a) const {
29     Complex b;
30     b.real = real * a.real - imaginary *
   a.imaginary;
31     b.imaginary = real * a.imaginary + imaginary *
   a.real;
32     return b;
```

```
33 }
34 Complex Complex::Div(const Complex& a) const {
35     int denominator = a.real * a.real + a.imaginary
   * a.imaginary;
36     // if(denominator == 0)
37     //     throw "mold of 'a' cannot be zero";
38     Complex b;
39     b.real = (real * a.real + imaginary *
   a.imaginary) / denominator;
40     b.imaginary = (imaginary * a.real - real *
   a.imaginary) / denominator;
41     return b;
42 }
43 double Complex::Mold() const {
44     return sqrt(real * real + imaginary *
   imaginary);
45 }
46 bool Complex::operator== (const Complex& a) const {
47     return real == a.real && imaginary ==
   a.imaginary;
48 }
49 bool Complex::operator!= (const Complex& a) const {
50     return real != a.real || imaginary !=
   a.imaginary;
51 }
52 Complex Complex::operator+ (const Complex& a)
   { return Add(a); }
53 Complex Complex::operator- (const Complex& a)
   { return Sub(a); }
54 Complex Complex::operator* (const Complex& a)
   { return Mul(a); }
55 Complex Complex::operator/ (const Complex& a)
   { return Div(a); }
56 Complex& Complex::operator+= (const Complex& a) {
57     Complex temp = this->Add(a);
58     real = temp.real;
59     imaginary = temp.imaginary;
60     return *this;
```

```
61 }
62 Complex& Complex::operator-= (const Complex& a) {
63     Complex temp = this->Sub(a);
64     real = temp.real;
65     imaginary = temp.imaginary;
66     return *this;
67 }
68 Complex& Complex::operator*= (const Complex& a) {
69     Complex temp = this->Mul(a);
70     real = temp.real;
71     imaginary = temp.imaginary;
72     return *this;
73 }
74 Complex& Complex::operator/= (const Complex& a) {
75     Complex temp = this->Div(a);
76     real = temp.real;
77     imaginary = temp.imaginary;
78     return *this;
79 }
```

二、链表类

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

```cpp
/* 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
```

```
38      if(!p->next || i > pos - 1) return false;
39      q = p->next;
40      p->next = q->next;
41      delete q;
42      return true;
43  }
44  bool LinkList::Delete(char c) {
45      NODE *p = head, *q;
46      while(p->next && p->next->ch != c) { p =
    p->next; }
47      if(!p->next) return false;
48      q = p->next;
49      p->next = q->next;
50      delete q;
51      return true;
52  }
53  void LinkList::Display() {
54      NODE *p = head->next;
55      while(p) { cout << p->ch; p = p->next; }
56  }
```