Copyright ©2021-2099 SongyunMo. All rights reserved

第七次上机

1.#include<iostream>

using namespace std;

class Complex{

public:

double real;

double imag;

Complex(double r =0,double i=0)

{

   real=r; imag=i;

}

};

Complex operator+(Complex co1,Complex co2)

{

Complex temp;

temp.real = co1.real + co2.real;

temp.imag = co1.imag + co2.imag;

return temp;

}

int main()

{

Complex com1(1.1,2.2),com2(3.3,4.4),total1,total2;

total1 = operator+(com1,com2);

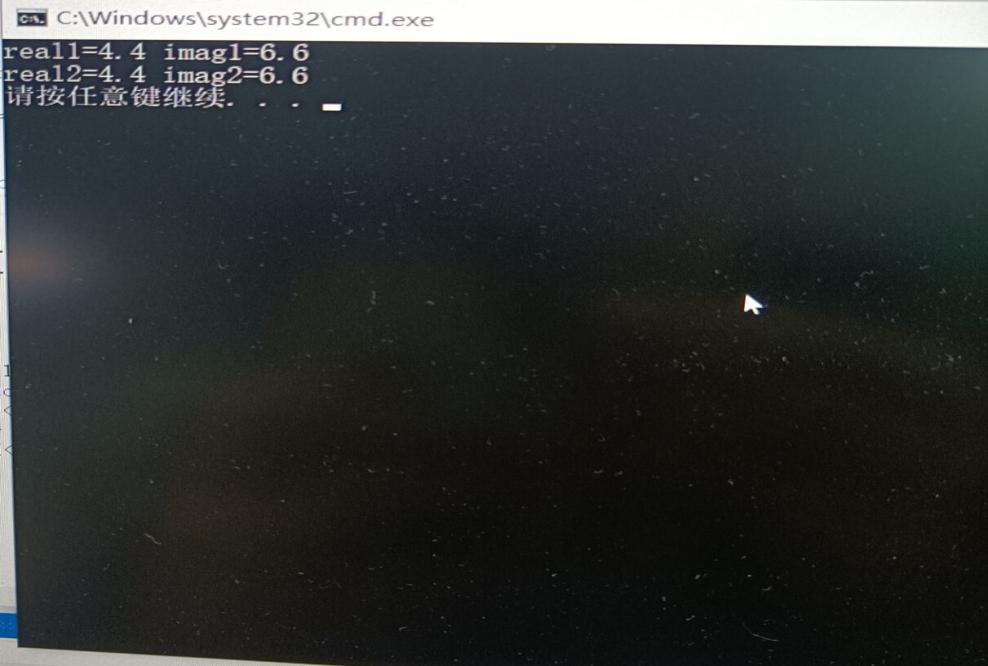
cout<<"real1="<<total1.real<<" "<<"imag1="<<total1.imag<<endl;

total2 = com1 + com2;

    cout<<"real2="<<total2.real<<" "<<"imag2="<<total2.imag<<endl;

return 0;

}



2.#include<iostream>

using namespace std;

class Complex{

double real;

double imag;

public:

Complex(double r =0.0,double i=0.0);

void print();

         Complex operator+(Complex c);

};

Complex::Complex(double r,double i)

{

real = r; imag = i;

}

Complex Complex::operator+(Complex c)

 {

      Complex temp;

  temp.real = real + c.real;

  temp.imag = imag + c.imag;

  return temp;

  }

void Complex::print()

{ cout<<"total real="<<real<<" "<<" total imag="<<imag<<endl;  }

int main()

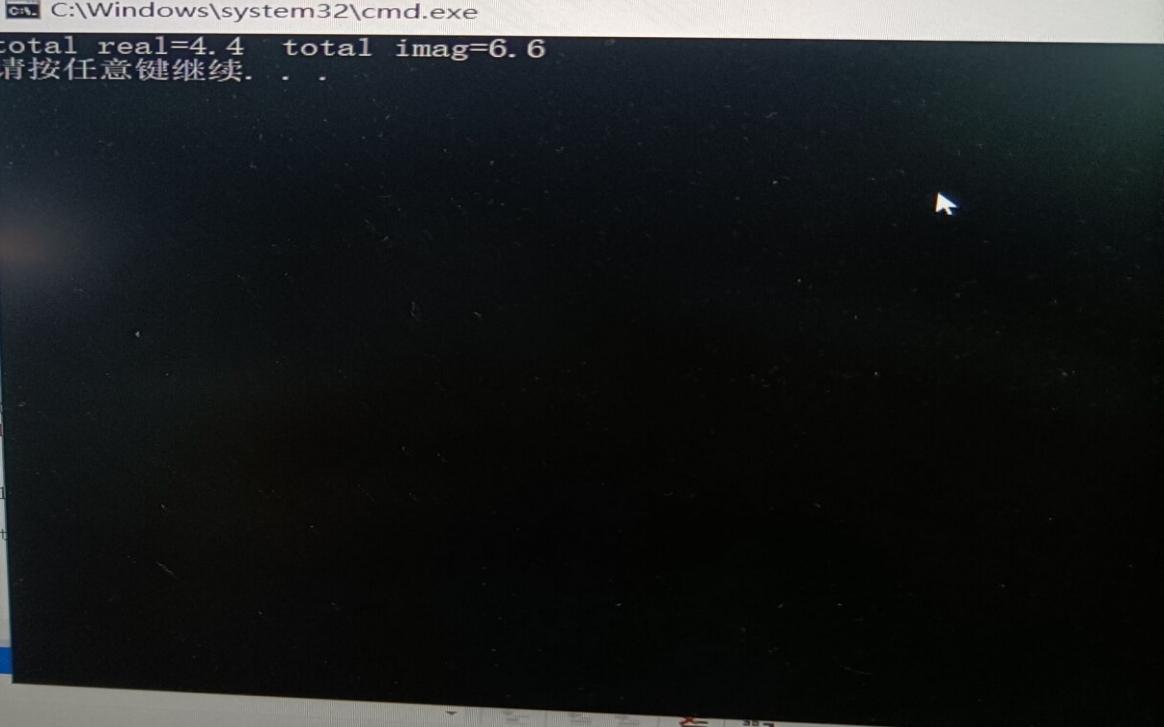
{ Complex com1(2.3,4.6),com2(3.6,2.8),total1;

total1 = com1 + com2;

        total1.print();

return 0;

}



3.#include<iostream>

#include<string>

using namespace std;

class MyArray{

public:

     MyArray(int length);

     ~MyArray();

     void Input();

     void Display(string);

protected:

     int \*alist;

     int length;

};

MyArray::MyArray(int leng)

{

     if(leng<=0)

     {

         cout<<"error length";

         exit(1);

     }

     alist = new int [length];

     length = leng;

     if(alist == NULL)

     {    cout<<"assign failure";

           exit(1);

     }

          cout<<"MyArray类对象已创建!"<<endl;

}

MyArray::~MyArray()

{

         delete [ ] alist;

         cout<<"MyArray类对象已撤销!"<<endl;

}

class SortArray: public MyArray{

public:

    void Sort();

    SortArray(int leng):MyArray(leng)

    {

      cout<<"SortArray类对象已创建!"<<endl;

     }

    ~SortArray();

};

SortArray::~SortArray()

{

    cout<<"SortArray类对象已撤销!"<<endl;

}

void SortArray::Sort()

{

     int i,j,temp;

     for(i=0;i<length-1;i++)

        for(j=0;j<length-i-1;j++)

        {

          if(alist[j]>alist[j+1])

          {

             temp=alist[j];

             alist[j]=alist[j+1];

             alist[j+1]=temp;

           }

        }

}

int main()

{

    SortArray s(5);

    s.Input();

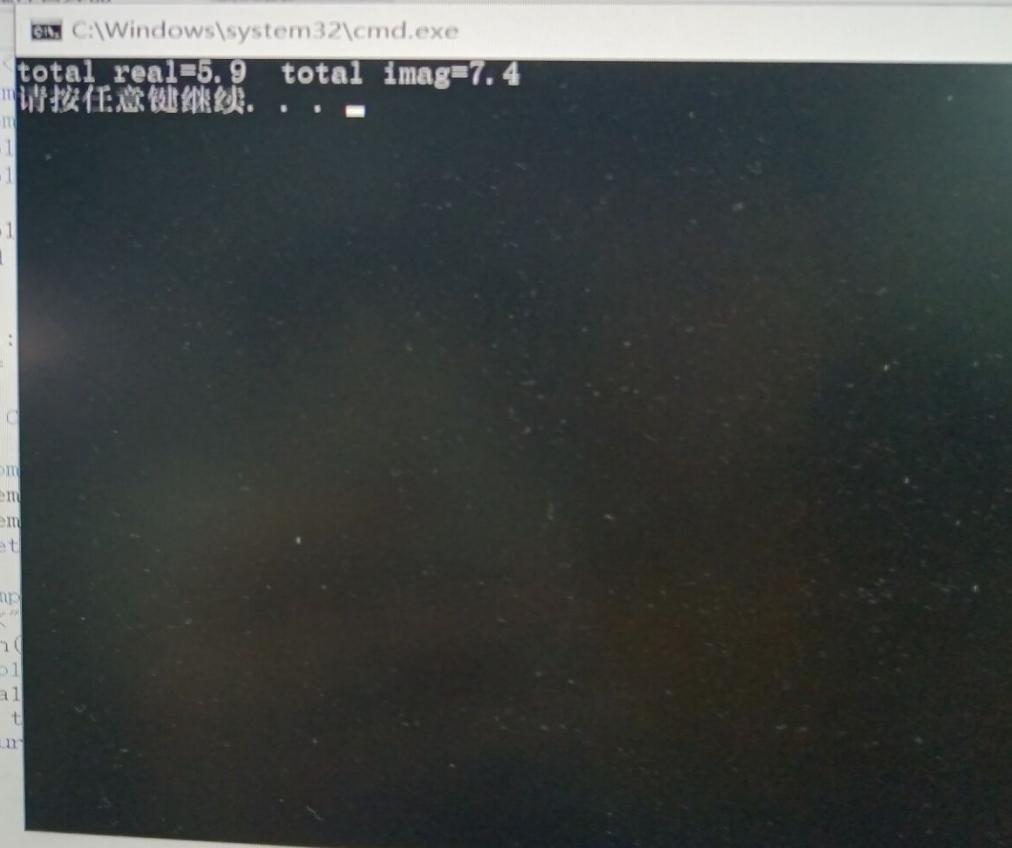
    s.Display("显示排序以前的");

    s.Sort();

    s.Display("显示排序以后的");

    return 0;

}



感想心得：

通过重载类上的标准算符，你可以发掘类的用户的直觉。使得用户程序所用的语言是面向问题的，而不是面向机器的。

从开发角度而言，运算符重载为了提高开发效率，增加代码的可复用性，很多时候简化了问题。