#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);

}

length = leng;

alist = new int[length];

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;

}

void MyArray::Display(string str)

{

int i;

int \*p = alist;

cout << str << length << "个整数: ";

for (i = 0; i < length; i++, p++)

cout << \*p << "";

cout << endl;

}

void MyArray::Input()

{

cout << "请从键盘输入" << length << "个整数:";

int i;

int \*p = alist;

for (i = 0; i < length; i++, p++)

cin >> \*p;

}

class SortArray :public MyArray

{

public:

SortArray(int leng);

~SortArray();

void paixu();

};

SortArray::SortArray(int leng) :MyArray(leng)

{

if (leng <= 0)

{

cout << "error length";

exit(1);

}

length = leng;

alist = new int[length];

if (alist == NULL)

{

cout << "assign failure";

exit(1);

}

cout << "SortArray类对象已创建!" << endl;

}

SortArray::~SortArray()

{

delete[] alist;

cout << "SortArray类对象已撤销!" << endl;

}

void SortArray::paixu()

{

int i, t, j;

int \*p = alist;

for (i = 0; i < length-1; i++)

{

for (j = i+1; j < length; j++)

{

if (alist[i] > alist[j])

{

t = p[i];

p[i] = p[j];

p[j] = t;

}

}

}

cout<<"排序后结果为：" ;

for(i = 0; i < length; i++)

cout << p[i] << " ";

}

int main()

{

// MyArray a(5);

SortArray b(5);

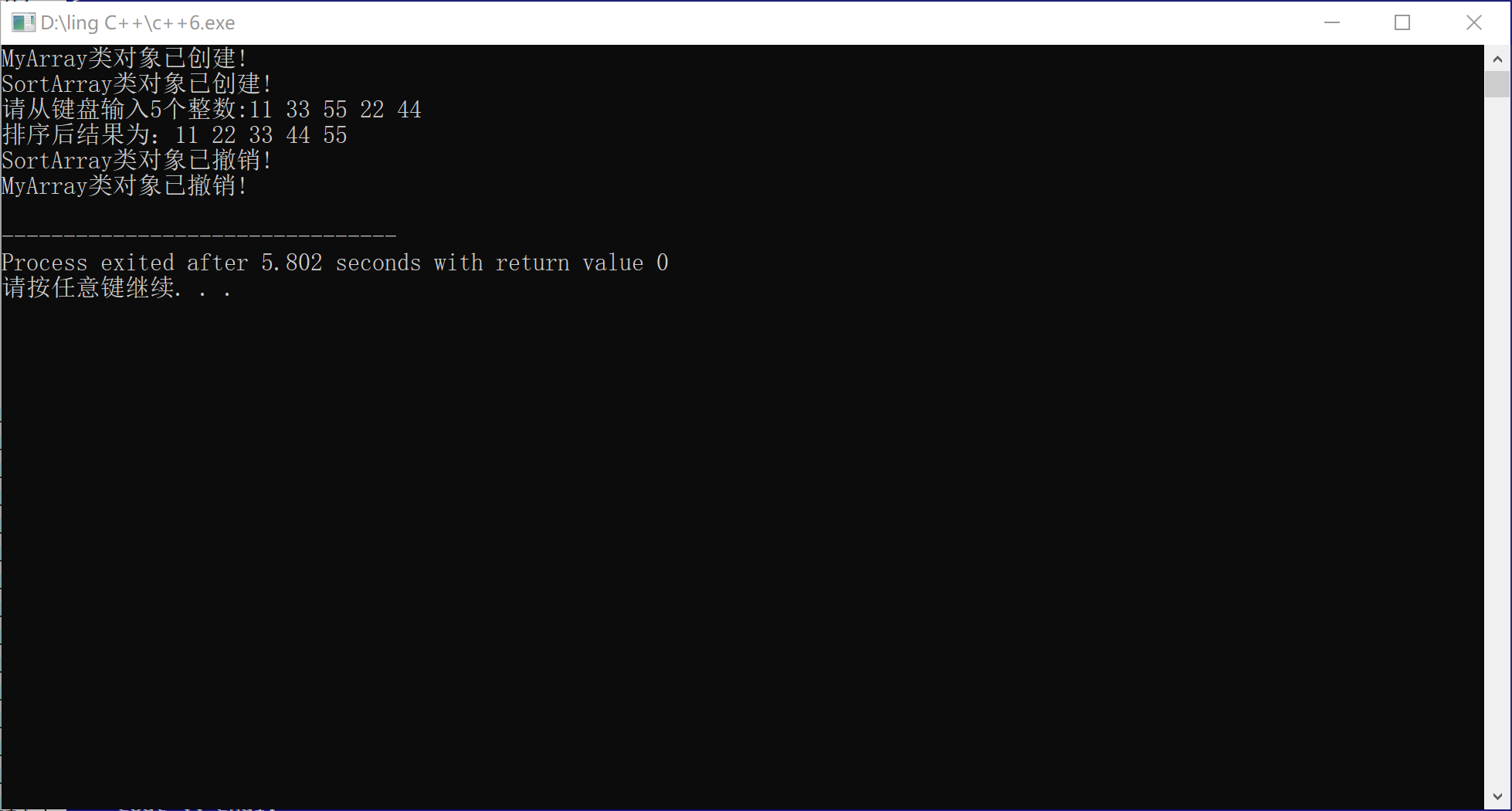
b.Input();

b.paixu();

cout << endl;

return 0;

}



感想：本次上机内容比较复杂，其实验目的是掌握派生类的声明方法和派生类构造函数的定义方法，还有不同方式下构造函数和析构函数的执行顺序和构造规则。继承方式也需要理解继承后的成员函数类型，保证调用时不会出现无法访问的情况。