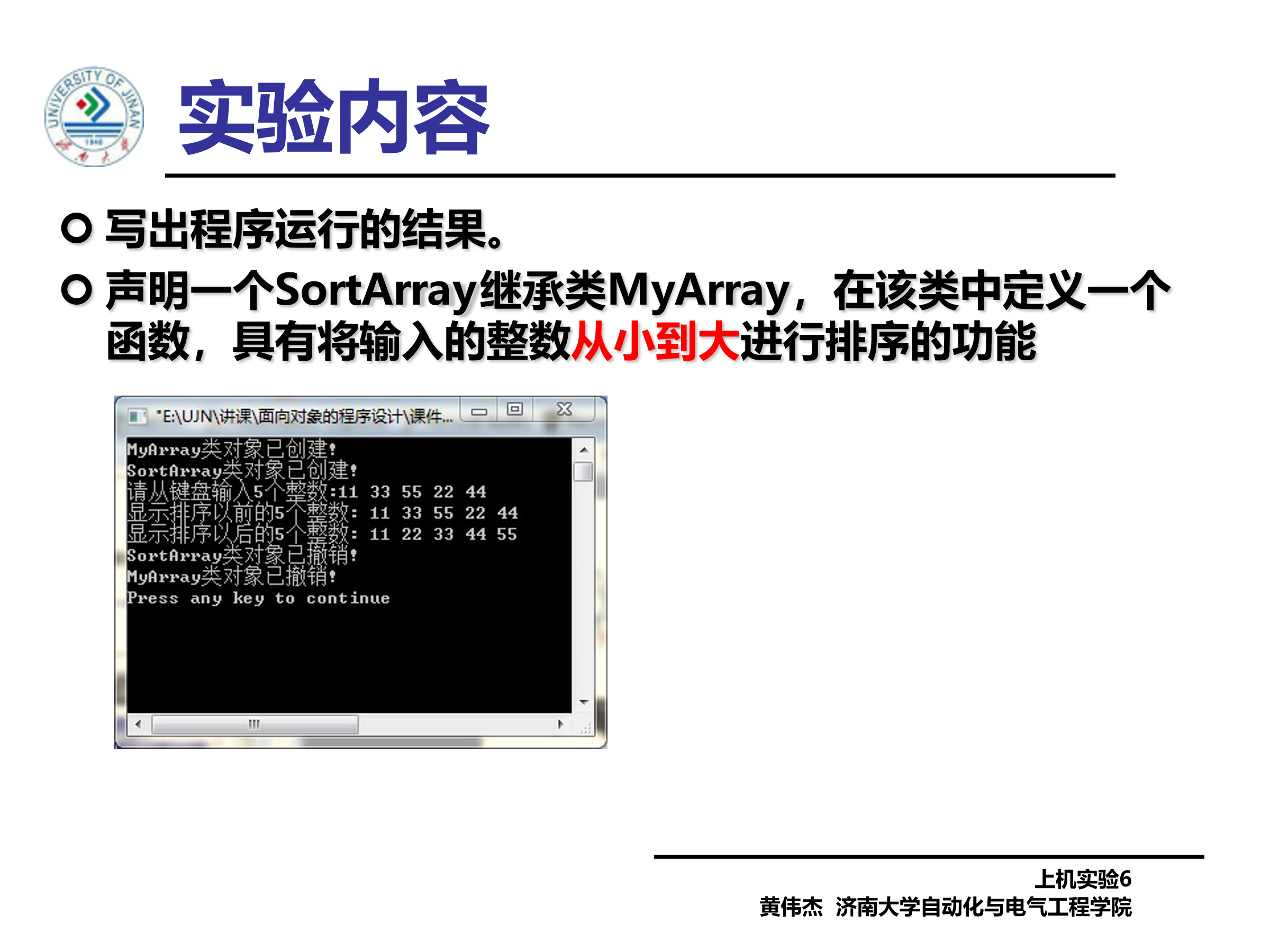
**实验内容：**



**程序：**

#include <iostream>

#include <string>

using namespace std;

class MyArray {

public:

MyArray(int length);

~MyArray();

void Input();

void Display(string);

protected:

int alist[20];

int length;

};

class SortArray :public MyArray {

public:

SortArray(int a);

~SortArray();

void sort();

};

void SortArray::sort()

{

int i, x, j;

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

{

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

{

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

{

x = alist[i];

alist[i] = alist[j];

alist[j] = x;

}

}

}

}

MyArray::MyArray(int leng)

{

if (leng <= 0)

{

cout << "error length";

exit(1);

}

length = leng;

cout << "MyArray类对象已创建!" << endl;

}

SortArray::SortArray(int a) :MyArray(a)

{

cout << "SortArray类对象已创建!" << endl;

}

SortArray::~SortArray()

{

cout << "SortArray类对象已撤销!" << endl;

}

MyArray::~MyArray()

{

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;

}

int main()

{

SortArray a(5);

a.Input();

a.Display("显示排序以前的");

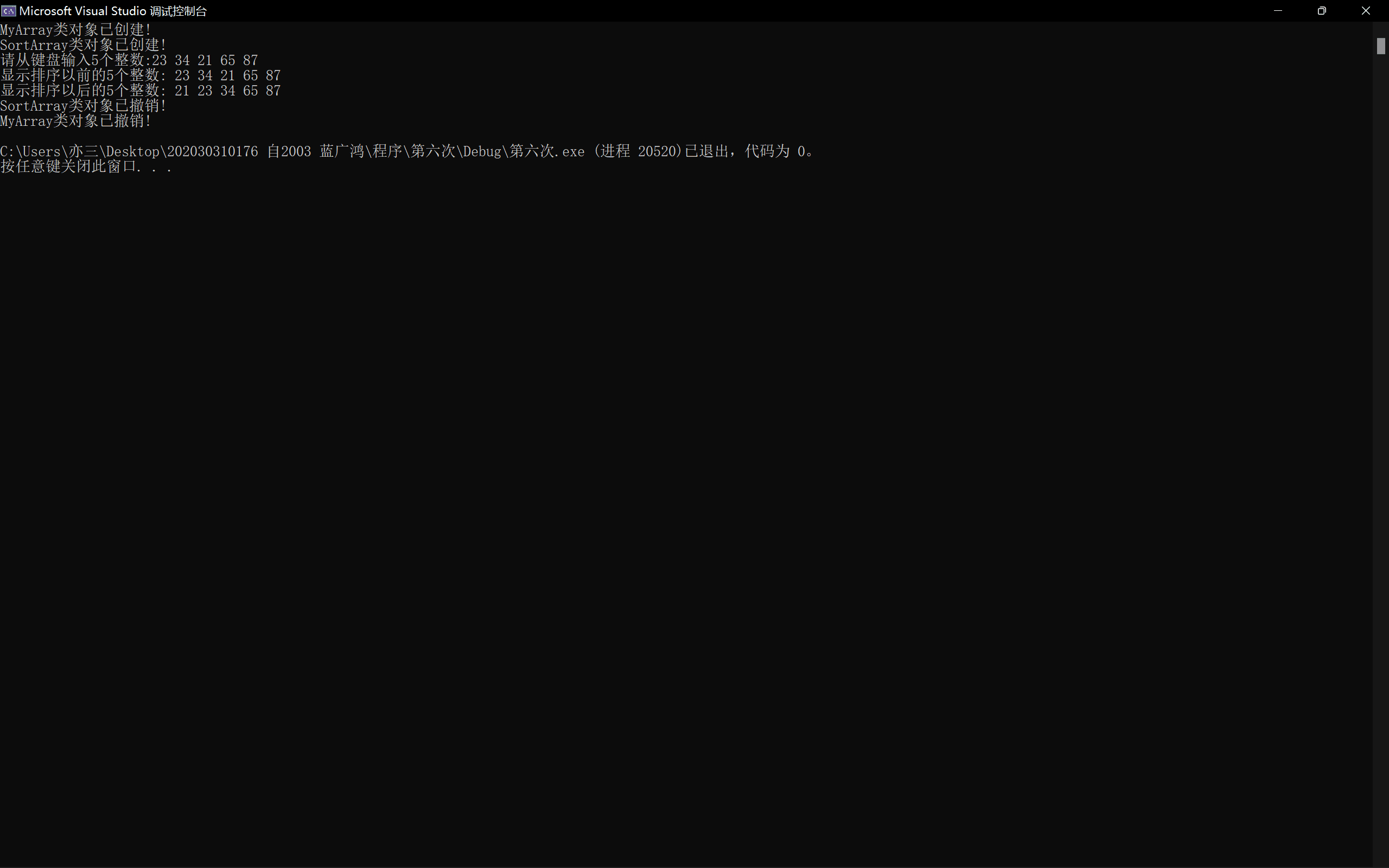
a.sort();

a.Display("显示排序以后的");

return 0;

}

**运行结果：**

****

**总结：**

**编写程序时，给我一个很深的印象就是方便，跟一般的程序编写相比，这种方式的编写更快，其中原因在于一些重复代码可以直接使用，不需要再次输入。经过翻阅课本，了解到继承的一个很好的优点，就是提高代码的重用性。**