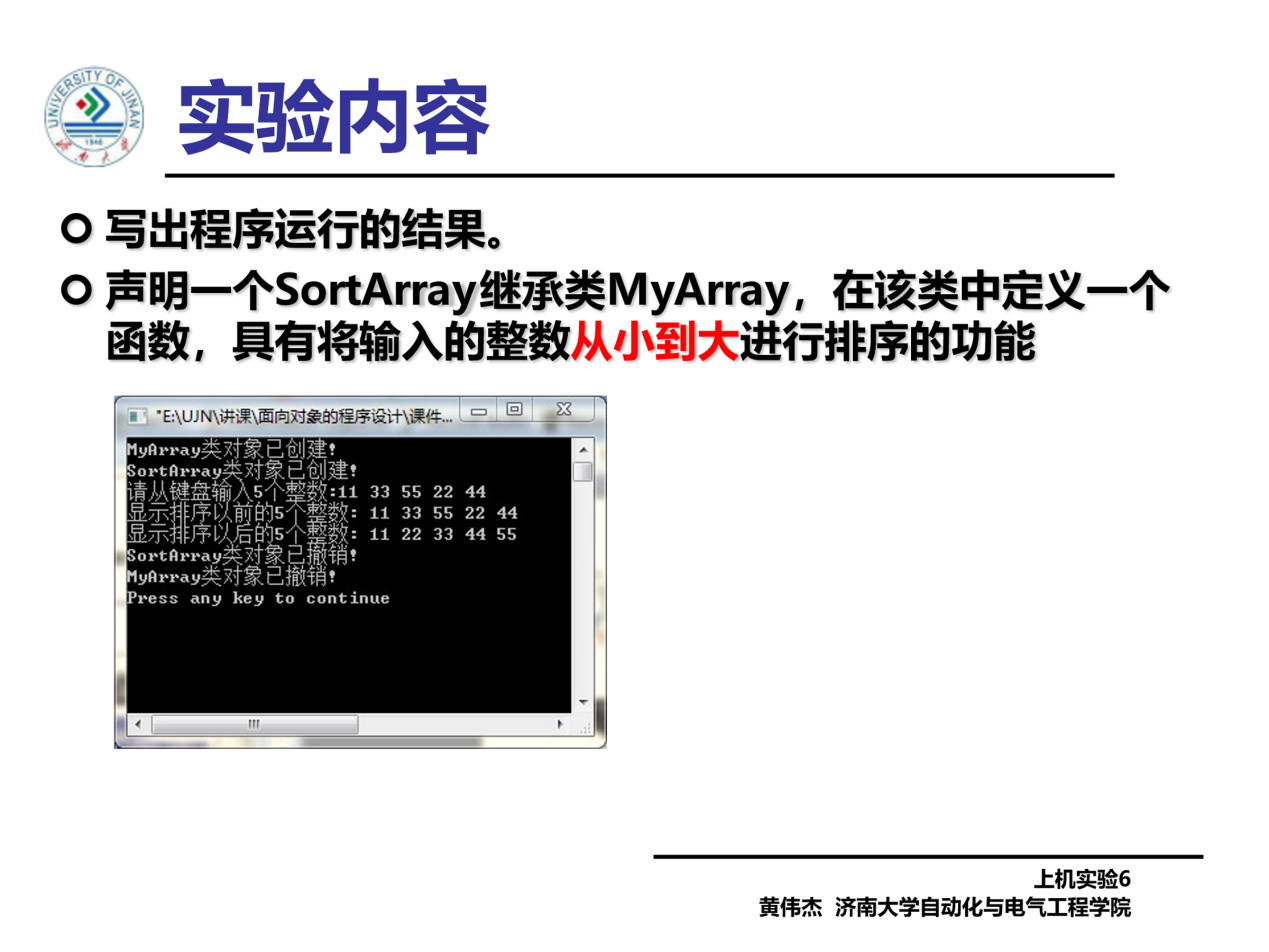
Copyright ©2021-2099 XuhanZhang. All rights reserved

**实验要求：**



**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;

}

**实验程序输出结果：**



**实验总结：**

完成程序的编写，决不意味着万事大吉。你认为万无一失的程序，实际上机运行时可能不断出现麻烦。如编译程序检测出一大堆错误。有时程序本身不存在语法错误，也能够顺利运行，但是运行结果显然是错误的。开发环境所提供的编译系统无法发现这种程序逻辑错误，只能靠自己的上机经验分析判断错误所在。

**出错现象:**

1.执行delete语句时，程序直接弹窗，崩溃。

2.执行delete语句时，程序卡死。将delete语句注释掉，又正常了，但发生了内存泄露

**解决方法：**

使用普通数组替换指针。

**认识到了继承的好处：**

（1）代码重用，减少创建类的成本，每个子类都拥有父类的方法和属性；

（2）子类与父类基本相似，但又与父类有所区别；

（3）提高代码的可扩展性。