# 实验六 继承下的构造函数与析构函数

程序代码：

#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 << "请从键盘上输入" << length << "个整数:";

int i;

int\* p = alist;

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

cin >> \*p;

}

class SortArray : public MyArray {

public:

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; 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5个整数：");

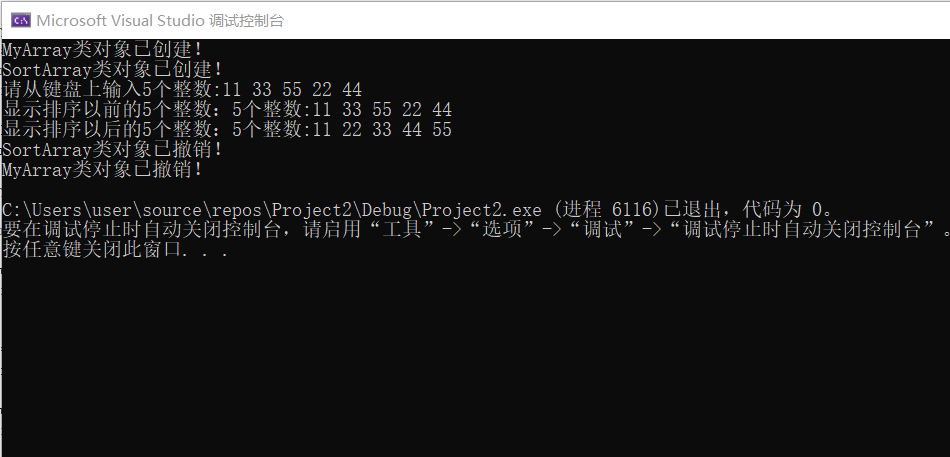
s.Sort();

s.Display("显示排序以后的5个整数：");

return 0;

}

运行结果：



心得体会：

经过此次实验，我掌握了构造函数与析构函数的执行顺序。

先执行基类的构造函数，然后执行对象成员的构造函数，随后再执行派生类的构造函数，析构函数的调用顺序刚好与构造函数的执行顺序相反。

Copyright ©2021-2099 Tianhang Li. All rights reserved