1. **实验代码**

**实验一**

#include<iostream>

#include<string>

using namespace std;

class MyArray {

public:

MyArray(int);

~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[leng];

length = leng;

if (alist == NULL)

{

cout << "assign failure";

exit(1);

}

cout << "MyArray类的对象以创建!" << 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;

}

int main()

{

MyArray a(5);

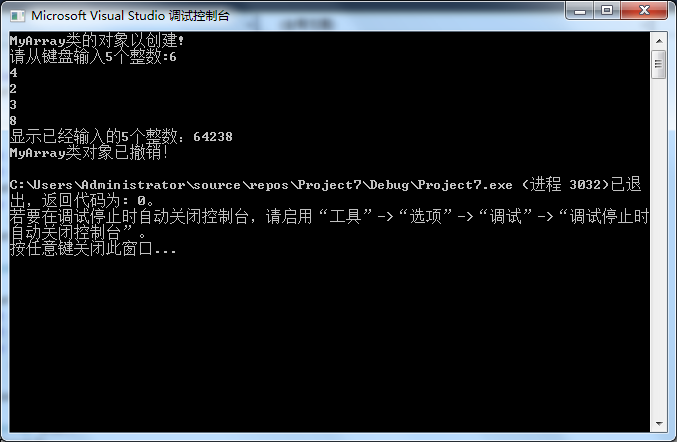
a.Input();

a.Display("显示已经输入的");

return 0;

}

**代码结果**



实验二

#include<iostream>

#include<string>

using namespace std;

class MyArray {

public:

MyArray(int leng);

~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[leng];

length = leng;

if (alist == NULL)

{

cout << "assign failure";

exit(1);

}

cout << "MyArray类的对象以创建!" << 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 paixu :public MyArray {

public:

paixu(int leng) :MyArray(leng)

void px()

{

int\* p = alist;

cout << "显示排序前整数： ";

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

cout << \*p << " ";

cout << endl;

p = alist;

for (int i = 0; i < length - 1; i++) //进行4次比较

{

for (int j = 0; j < length - i - 1;j++) //在每次中进行5-i-1次比较

{

if (\*(p+j) > \*(p+j + 1)) //判断相邻元素的大小

{

int k;

k = \*(p + j);

\*(p + j) = \*(p + j + 1);

\*(p + j + 1) = k; //借助中间变量进行值传递

}

}

}

cout << "显示排序后整数： ";

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

cout << \*p << " ";

cout << endl;

}

};

int main()

{

paixu b(5);

b.Input();

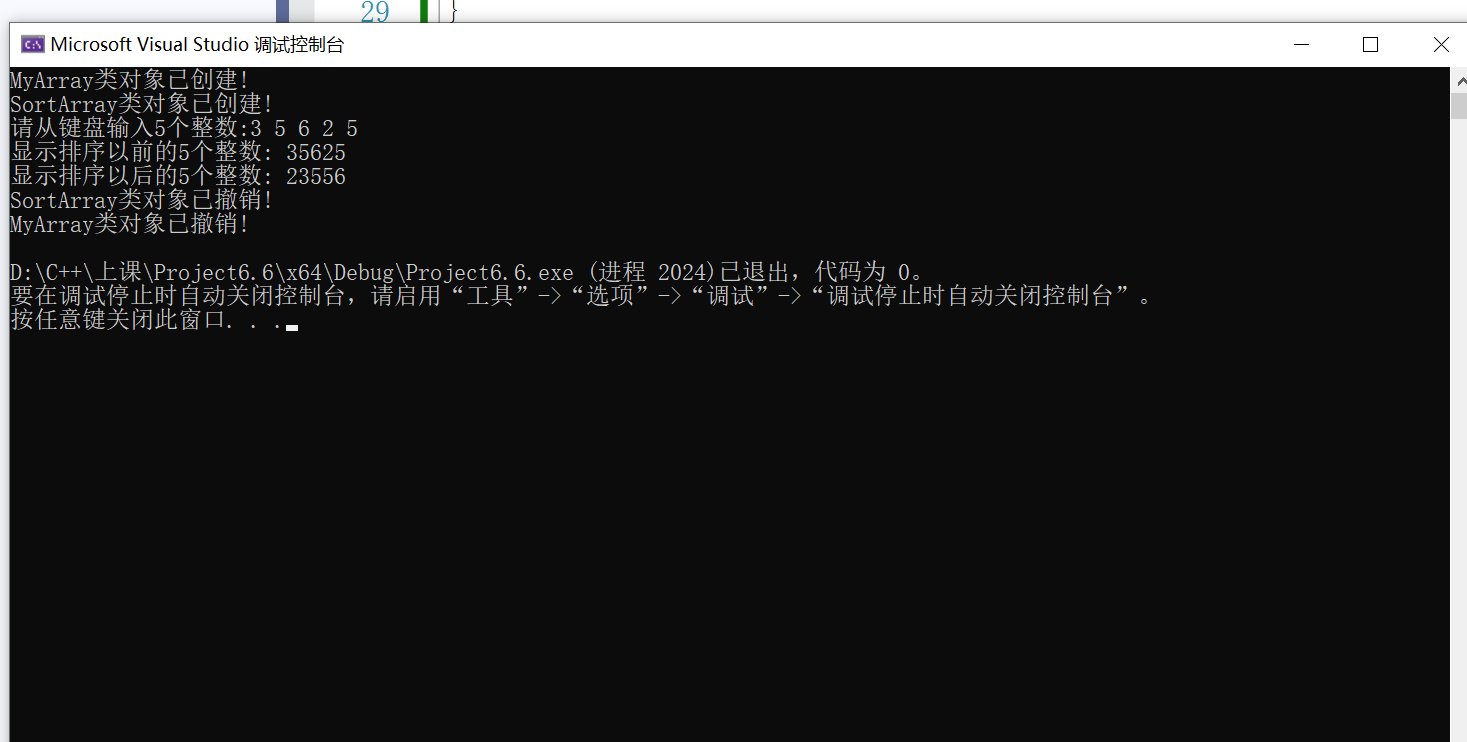
b.Display("显示已经输入的");

b.px();

return 0;

}

**代码结果**

****

3．实验心得

通过本次上机实验，我主要明白了派生类的构造函数与析构函数的构造规则，当基类构造函数没有参数或者没有显式定义时，派生类的构造函数无需传递参数，甚至无需定义。

Copyright ©2021-2099 ZhangKangNian. All rights reserved