六2021.11.26周五上机

1.#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);

}

alist = new int[length];

length = leng;

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;

}

int main()

{

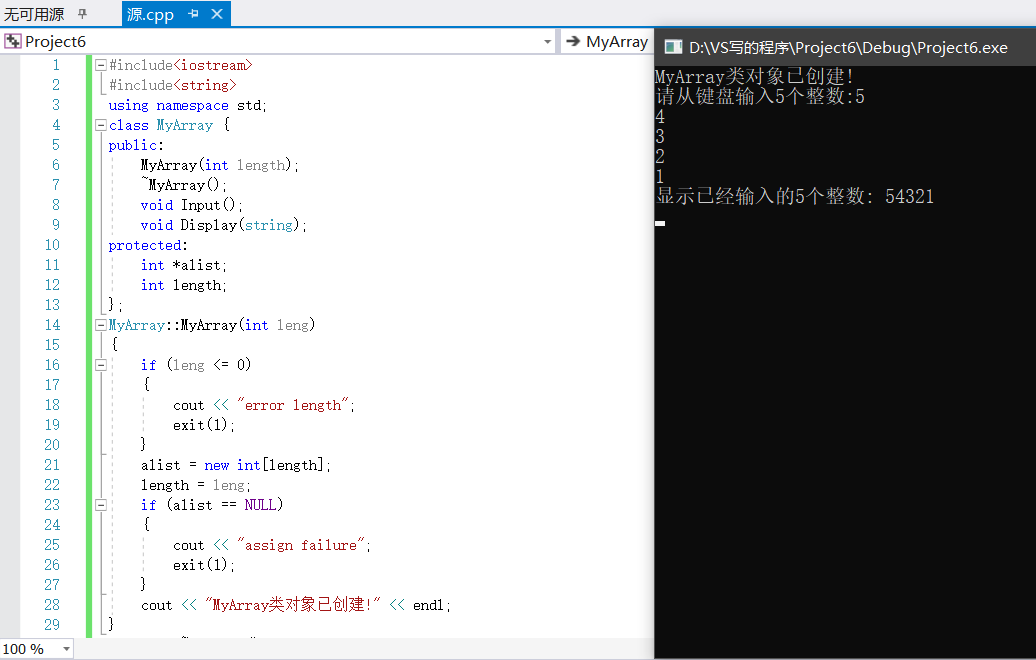
MyArray a(5);

a.Input();

a.Display("显示已经输入的");

return 0;

}



2.#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);

}

alist = new int[length];

length = leng;

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:

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++)

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();

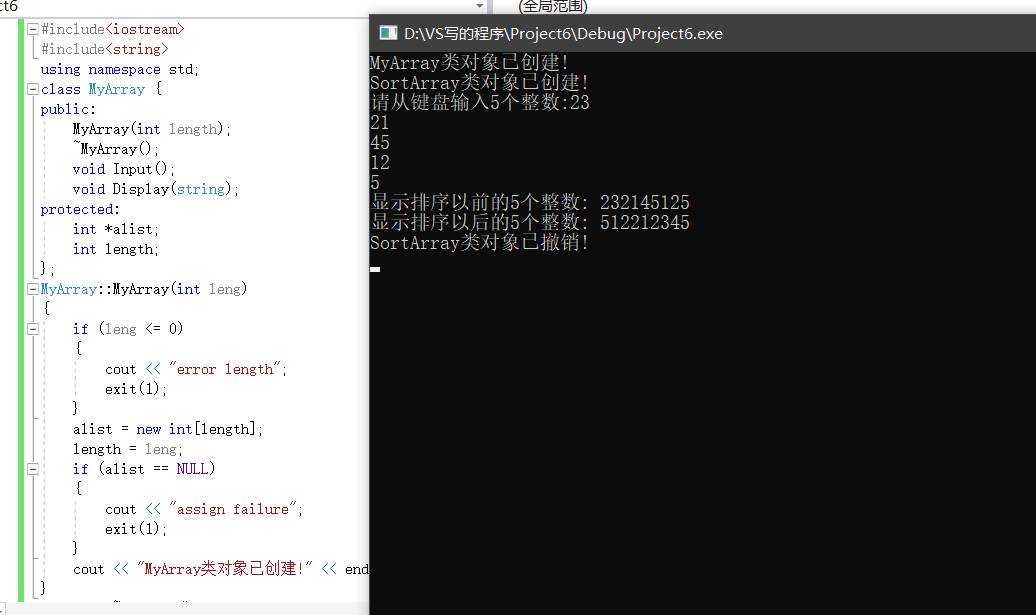
s.Display("显示排序以前的");

s.Sort();

s.Display("显示排序以后的");

return 0;

}



3.实验心得

1.派生类构造函数的一般形式为：  
派生类构造函数名（总参数表）：基类构造函数名（参数表）  
{  
   派生类中新增加数据成员初始化语句  
}  
2.在建立一个对象时，执行构造函数的顺序是：  
a.派生类构造函数先调用基类构造函数；  
b.再执行派生类构造函数本身（即派生类构造函数的函数体）  
3.在派生类对象释放时，先执行派生类析构函数，再执行其基类析构函数