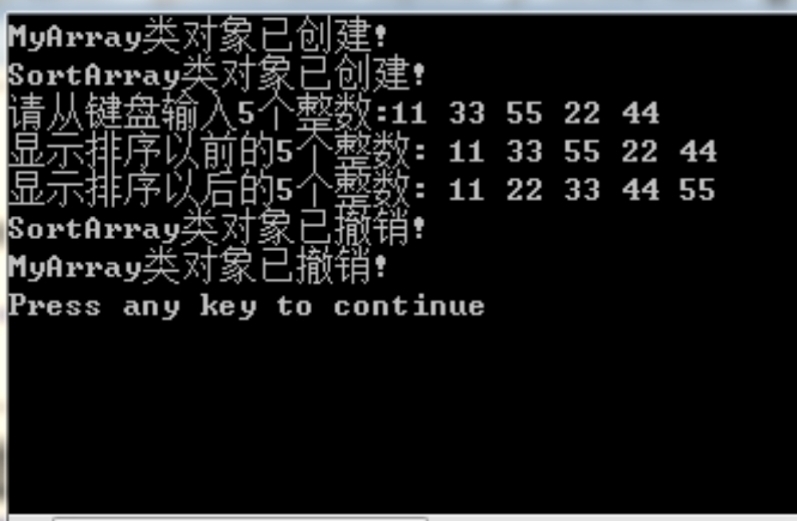
实验报告6

程序代码及结果：

#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 << "assiign failure";  
exit(1);  
}  
cout << "MyArray类对象已创建!" << endl;  
}  
MyArray::~MyArray()  
{  
delete[] alist;  
cout << "MuArray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("显示已输入!");  
return 0;  
}  
  
#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 << "assiign failure";  
exit(1);  
}  
cout << "MyArray类对象已创建!" << endl;  
}  
MyArray::~MyArray()  
{  
delete[] alist;  
cout << "MuArray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("显示已输入!");  
return 0;  
}

结果：



心得体会：

派生类继承了基类的全部数据成员和除了构造函数和析构函数之外的全部成员函数，一般需要在派生类中定义新的构造函数和析构函数，派生类不能对接受基类的成员进行选择但是可以对这些成员进行某些调整。

Copyright   
2021-2099 zhouchengping All rights reserved.