实验报告六

#include<iostream>

//#include<cmath>

#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::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;

}

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;

}

int main()

{

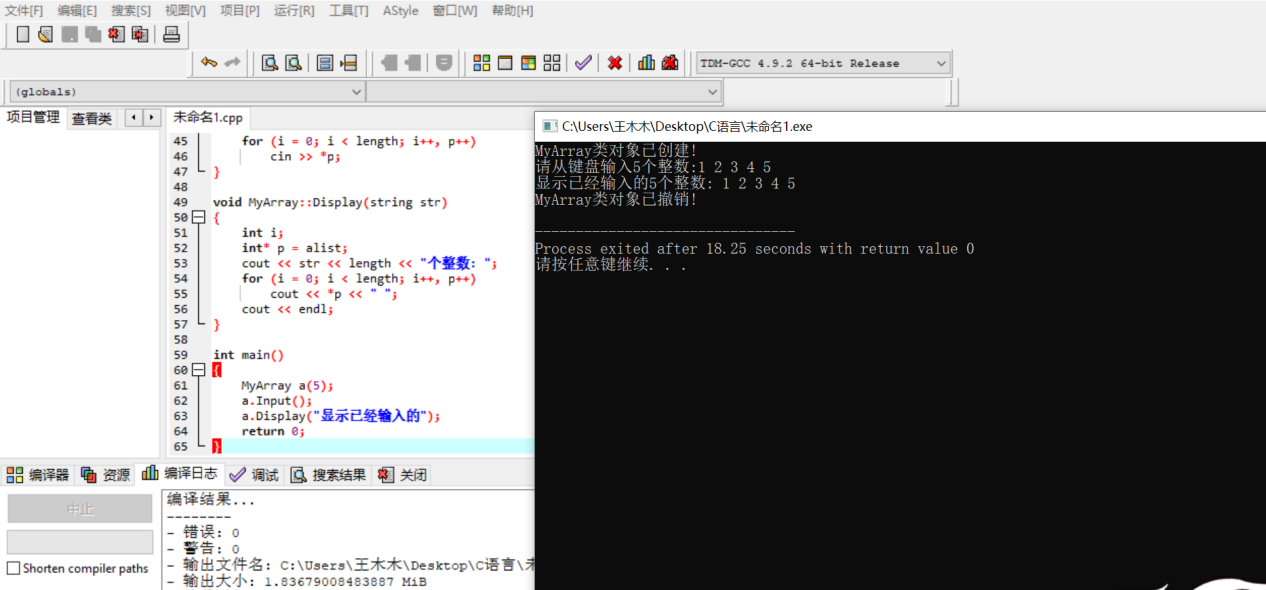
MyArray a(5);

a.Input();

a.Display("显示已经输入的");

return 0;

}



**• 吸收基类的成员：不论是数据成员还是函数成员，派生类吸收除基类的构造函数和析构函数之外的全部成员。**

**• 改造基类函数：在派生类中声明一个或多个与其（某个）基类中的成员函数同名的成员函数，并将它（们）根据新的需求进行重写**

**• 发展新的成员：在派生类中添加新的成员变量和成员函数，其中新添加的成员要求必须和基类中的成员不同名，并且应当保证新添加的成员会使派生类在功能上相比其基类有所发展**

**• 重写派生类的构造函数和析构函数**