// ResistanceCount.cpp : 定义控制台应用程序的入口点。

//

/\*

题目：第三周 第四题

4) 题目：

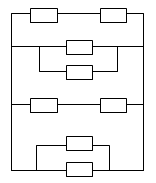
编写程序，确定由4个并联的子电路组成的电路的总电阻。

子电路由俩个串联或并联电阻器组成，

由键盘输入，向屏幕输出总电阻。

使用虚类和多态性，

对所有的子电路使用一个循环计算总电阻。



编译环境：Visual Studio 2015

作者：杨政权

\*/

#include "stdafx.h"

#include<iostream>

#include <string>

using namespace std;

class Resistance

{

protected:

double Resist1, Resist2;

public:

//获取用户输入电阻

void InputResistance()

{

cout << "请输入两个电阻，中间用空格隔开，回车键结束:" << endl;

cin >> Resist1>>Resist2;

}

virtual double ResistanceCount() = 0;

};

//串联电路

class SeriesResistance :public Resistance

{

public:

//串联电路的计算

double ResistanceCount()

{

return (Resist1 + Resist2);

}

};

//并联电路

class ParallelResistance :public Resistance

{

public:

//并联电路的计算

double ResistanceCount()

{

return (Resist1\*Resist2) / (Resist1 + Resist2); //并联计算公式

}

};

int main()

{

int i=0;

char choice;

double Denom=0.0;

double TotalRes=0.0;

Resistance \*PArray[4];

for (; i<4; i++)

{

do

{

cout << "请输入要串联(s)还是并联(p):" << endl;

cin >> choice;

if (choice == 's') //选择两电阻串联

{

PArray[i] = new SeriesResistance;

PArray[i]->InputResistance();

}

else if(choice=='p')//选择两电阻并联

{

PArray[i] = new ParallelResistance;

PArray[i]->InputResistance();

}

else //命令不正确，进行处理

{

cout << "您输入的命令有误：请重新输入：" << endl;

}

} while (choice != 's'&&choice!='p');

}

for (i = 0; i < 4; i++)

{

Denom += 1/(PArray[i]->ResistanceCount());

}

//计算最后的总电阻

TotalRes = 1 / Denom;

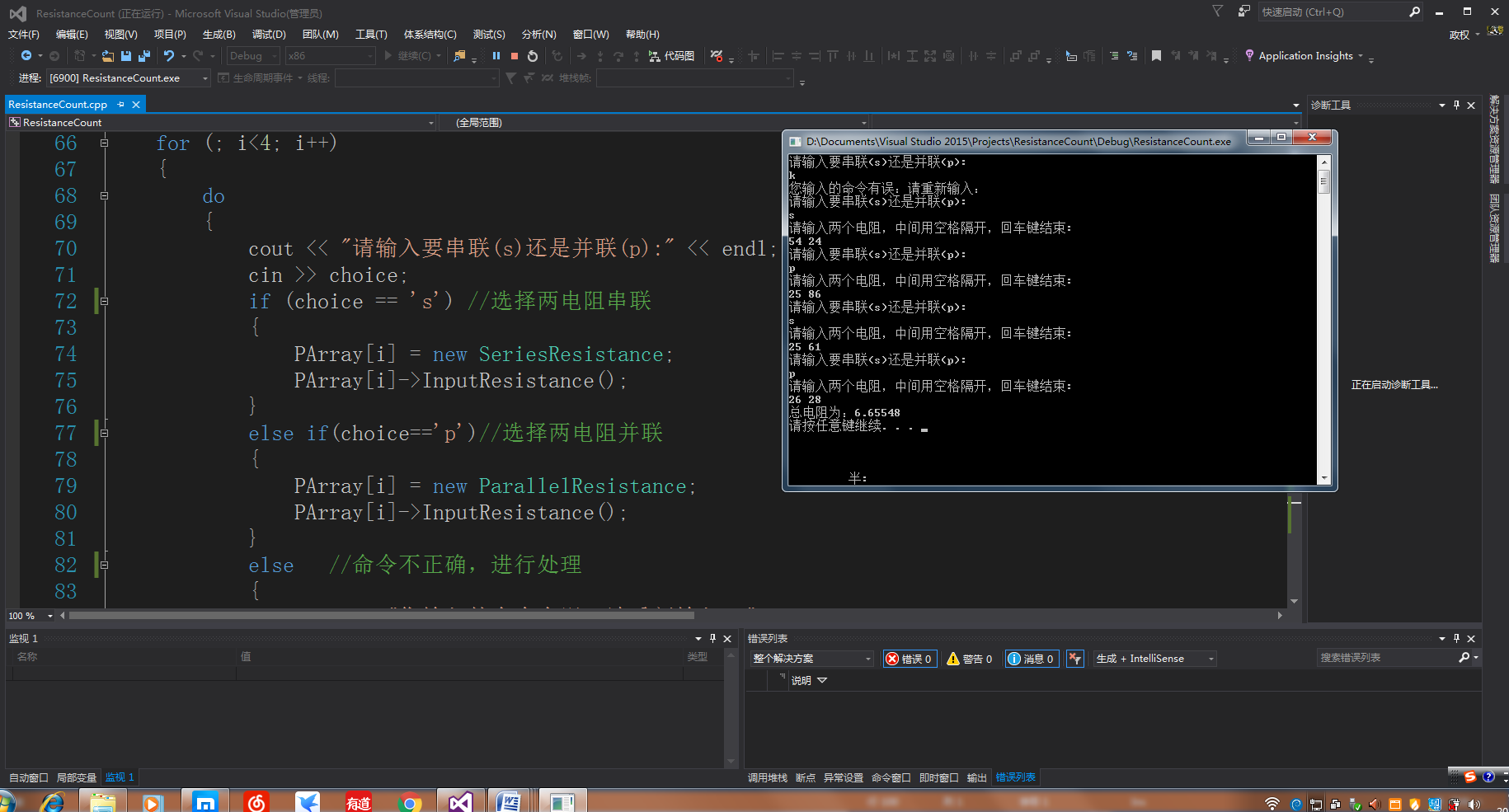
//将总电阻打印出来

cout << "总电阻为：" << TotalRes << endl;

system("pause");

}

//程序运行截图



//实验总结：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* 这个实验主要用到类的多态与虚函数

\*\*\* 本身难度不大，就是有时候会把串并联电阻公式搞错

\*\*\* 只要掌握了该电路的总电路电阻公式

\*\*\* 问题就迎刃而解了

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/