

**数学与信息学院学生实验报告**

**实验课程名称：** 算法分析与设计基础 **教师： \_\_**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** |  | | | **实验成绩** |  |
| **学生姓名** |  | **学 号** | **145** | **年级专业班级** |  |
| **小组成员** | **无** | | | **实验日期** | **2019年5月21日** |

# 1. 实验目的和要求

## 1.1 实验目的

理解贪心算法的基本原理，掌握贪心算法设计的基本方法及其应用；

## 1.2 实验软硬件环境

① 操作系统 windows 10

② 编译环境 devc++

## 1.3 实验要求

## （一）加油问题（Problem Set 1702）：

**1.问题描述**

一个旅行家想驾驶汽车从城市A到城市B（设出发时油箱是空的）。给定两个城市之间的距离dis、汽车油箱的容量c、每升汽油能行驶的距离d、沿途油站数n、油站i离出发点的距离d[i]以及该站每升汽油的价格p[i],i=1,2,…,n。设d[1]=0<d[2]<…<d[n]。要花最少的油费从城市A到城市B，在每个加油站应加多少油，最少花费为多少？

**2.具体要求**

# Input

输入的第一行是一个正整数k，表示测试例个数。接下来几行是k个测试例的数据，每个测试例的数据由三行组成，其中第一行含4个正整数，依次为A和B两个城市之间的距离d1、汽车油箱的容量c（以升为单位）、每升汽油能行驶的距离d2、沿途油站数n (1<=n<=200)；第二行含n个实数d1, d2 ,…, dn，表示各油站离出发点的距离（d1=0）；第三行含n个实数p1, p2 ,…, pn，表示各油站每升汽油的价格。同一行的数之间用一个空格隔开。

**Output**

对于每个测试例输出一行，含一个实数，表示从城市Ａ到城市Ｂ所要花费的最少油费（输出的结果精确到小数点后一位）。若问题无解，则输出“No Solution”。

**3.测试数据**

# Sample Input

2

1500 50 10 4

0 300.0 800.0 1200.0

4.0 5.0 4.0 4.5

1000 40 10 3

0 500.0 750.0

4.5 5.0 4.2

### Sample Output

640.0

No Solution

**（二）黑白点的匹配（Problem Set 1714）：**

**1.问题描述**

设平面上分布着n个白点和n个黑点，每个点用一对坐标（x, y）表示。一个黑点b=（xb,yb）支配一个白点w=(xw, yw)当且仅当xb>=xw和yb>=yw。若黑点b支配白点w，则黑点b和白点w可匹配（可形成一个匹配对）。在一个黑点最多只能与一个白点匹配，一个白点最多只能与一个黑点匹配的前提下，求n个白点和n个黑点的最大匹配对数。一对一匹配

**2.具体要求**

# Input

输入的第一行是一个正整数k，表示测试例个数。接下来几行是k个测试例的数据，每个测试例的数据由三行组成，其中第一行含1个正整数n(n<16)；第二行含2n个实数xb1, yb1,xb2, yb2,…, xbn, ybn， (xbi, ybi)，i=1, 2, …, n表示n个黑点的坐标；第三行含2n个实数xw1, yw1,xw2, yw2,…, xwn, ywn，(xwi, ywi)，i=1, 2, …, n表示n个白点的坐标。同一行的实数之间用一个空格隔开。

**Output**

对于每个测试例输出一行，含一个整数，表示n个白点和n个黑点的最大匹配对数。

**3.测试数据**

# Sample Input

1

3

5.0 3.0 5.0 -1.0 4.0 4.0

2.0 3.5 2.0 2.0 -2.0 -2.0

### Sample Output

3

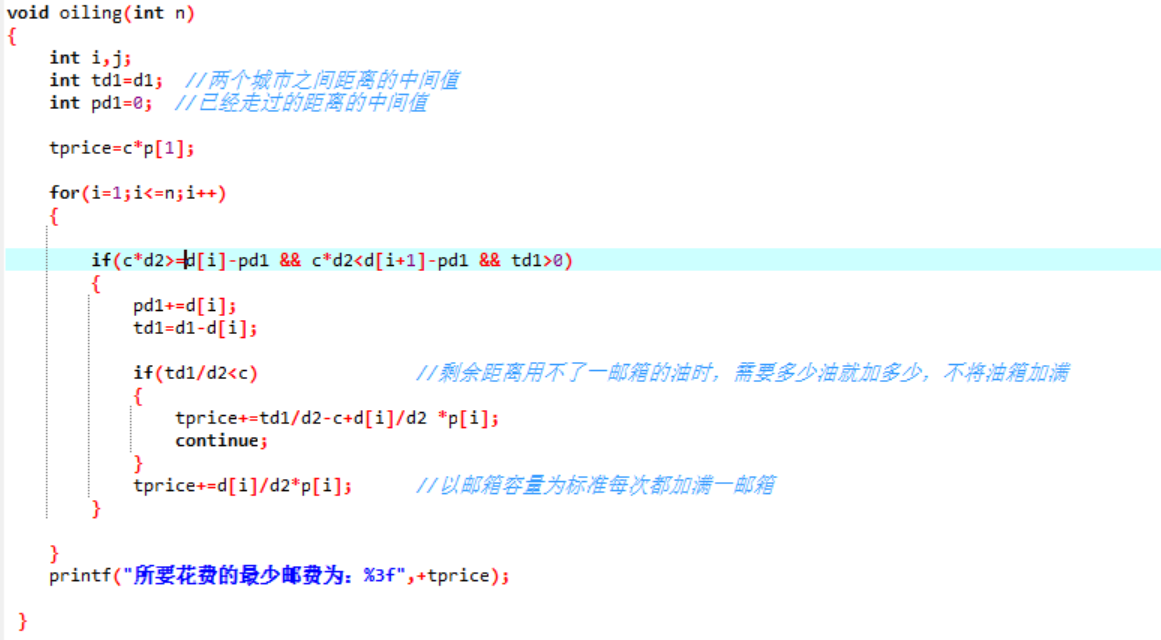
**2.1实验记录**

1. 算法思想：

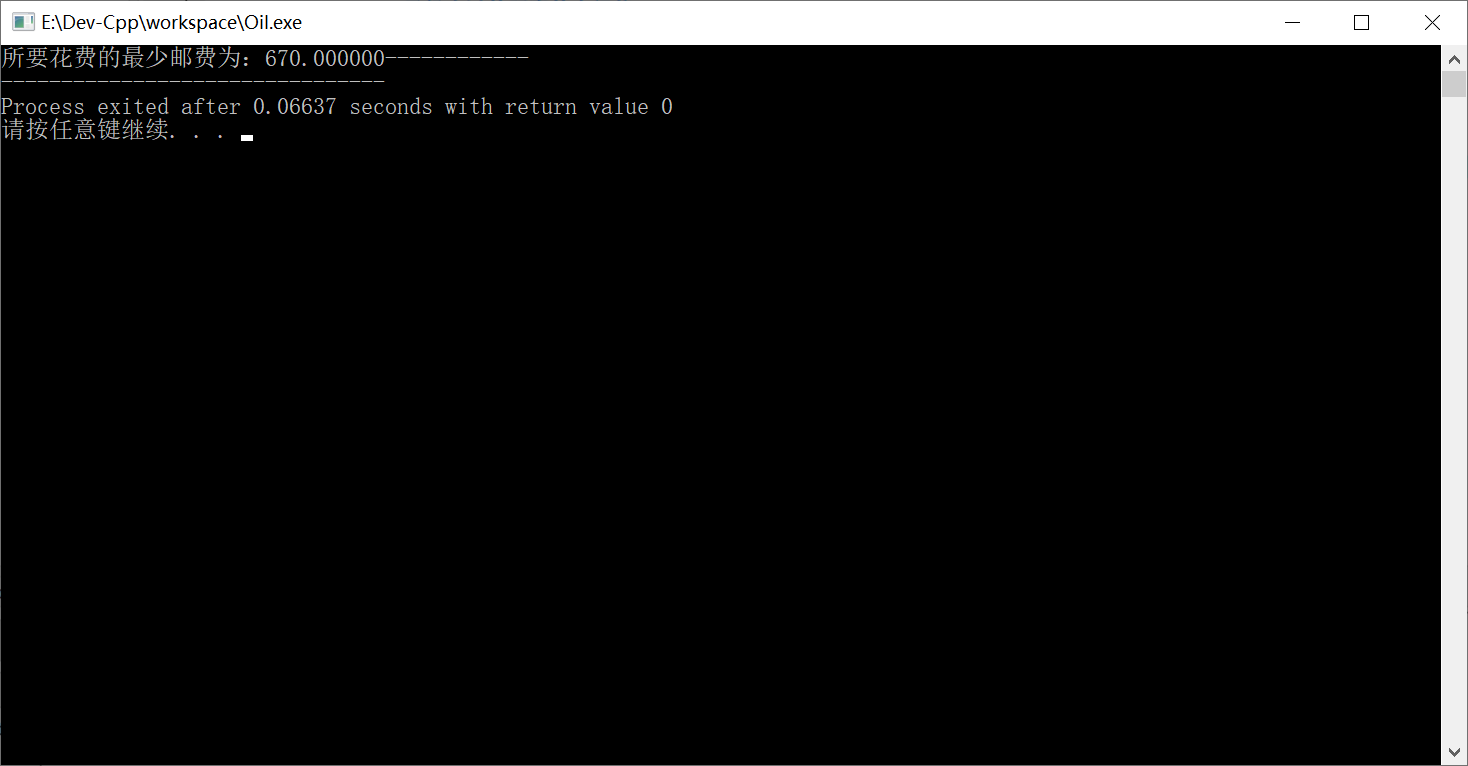
在第i=1个加油点处将油箱加满油，直到邮箱油量可以用到第i个加油点有剩余而不够跑到第i+1加油点，则在第i个加油点将油箱加满，每一次加油时都判断剩余距离td1是否可以用 <= 一个油箱的油量跑完，若可以，则计算剩余距离需要的油量，将邮箱中的油量加到该值（始终保持到达终点时，油箱为空）；若不可以，则继续进行上述计算。另外，每一次都需要在第i个加油点加满油后，判断油量是否能跑到第i+1 个加油点，若不能，则直接输出 no solution !

1. 算法实现过程记录

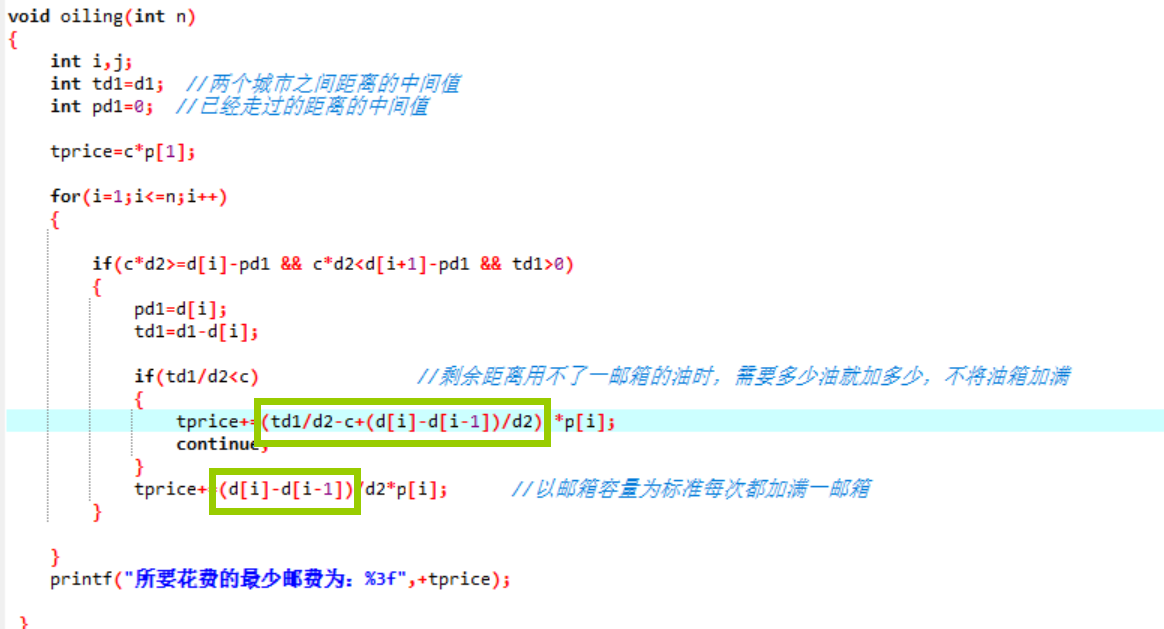
最原始的代码



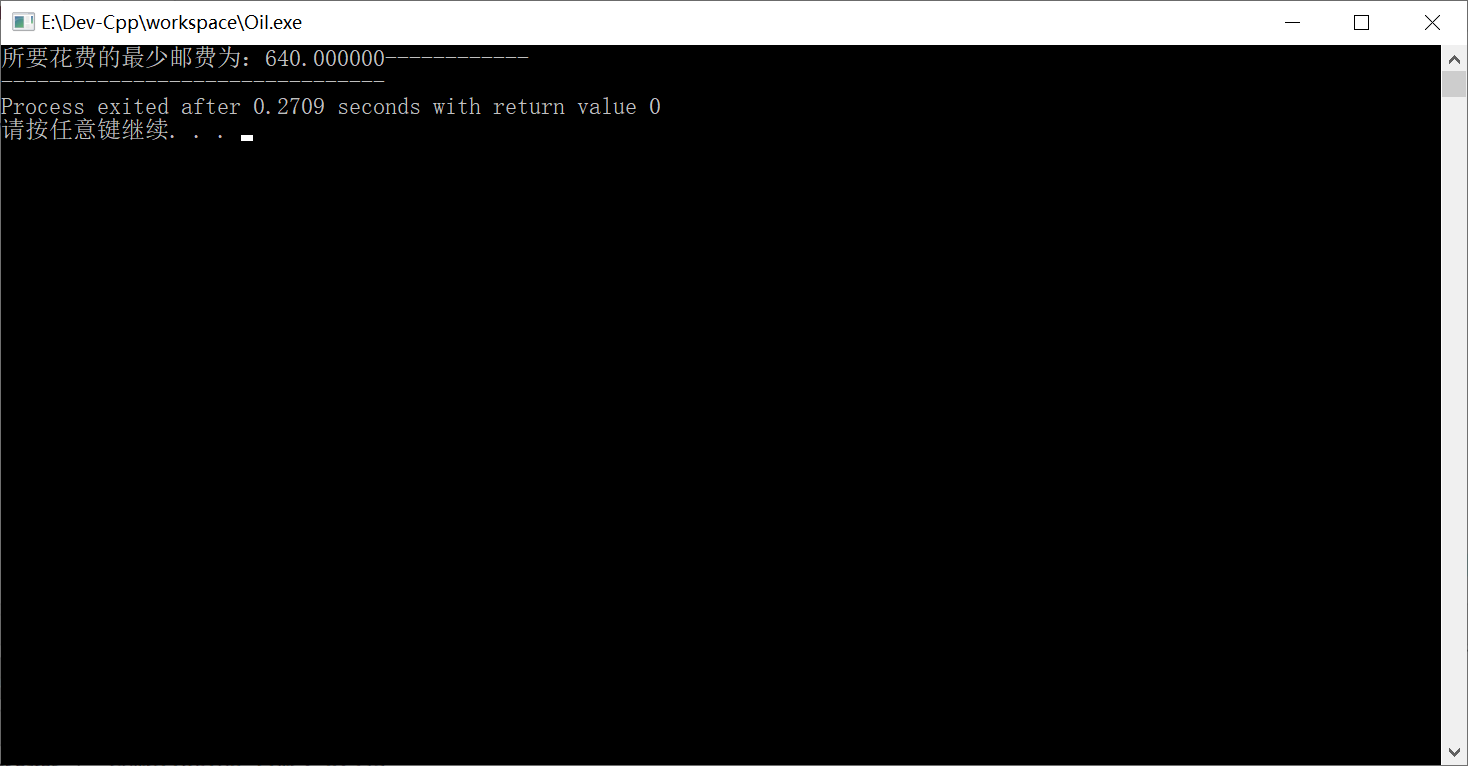
初次实验结果



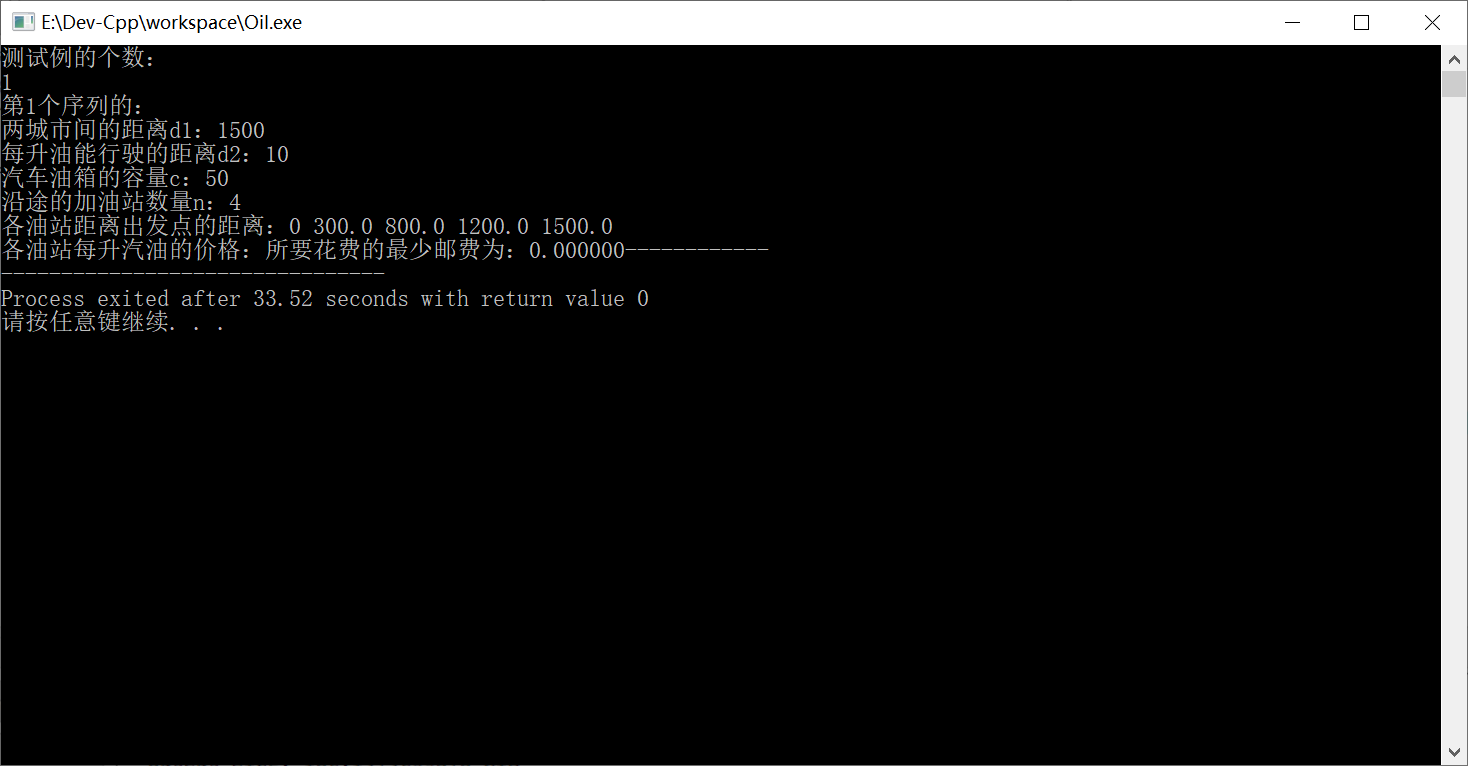
修改之后



正确结果

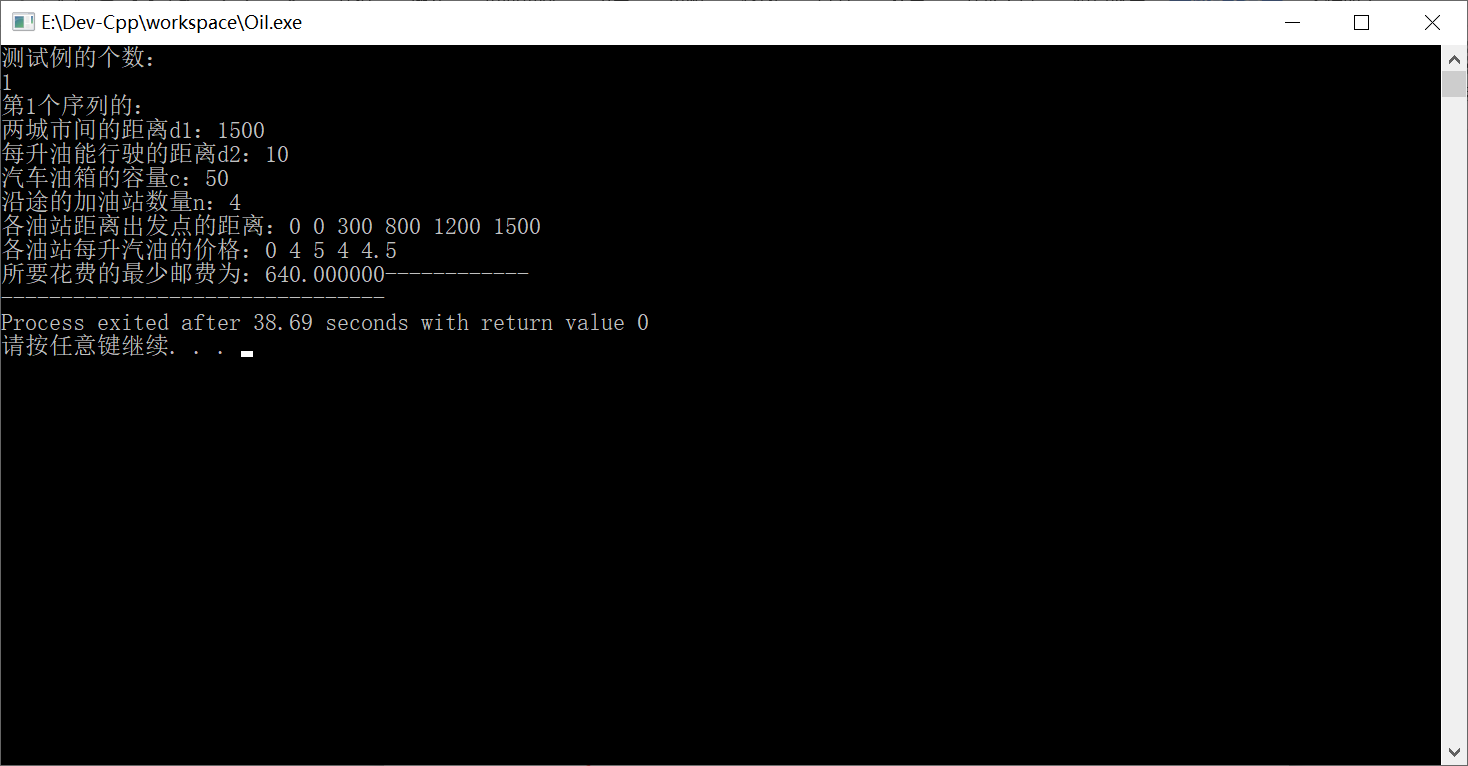


对数组进行输入赋值时

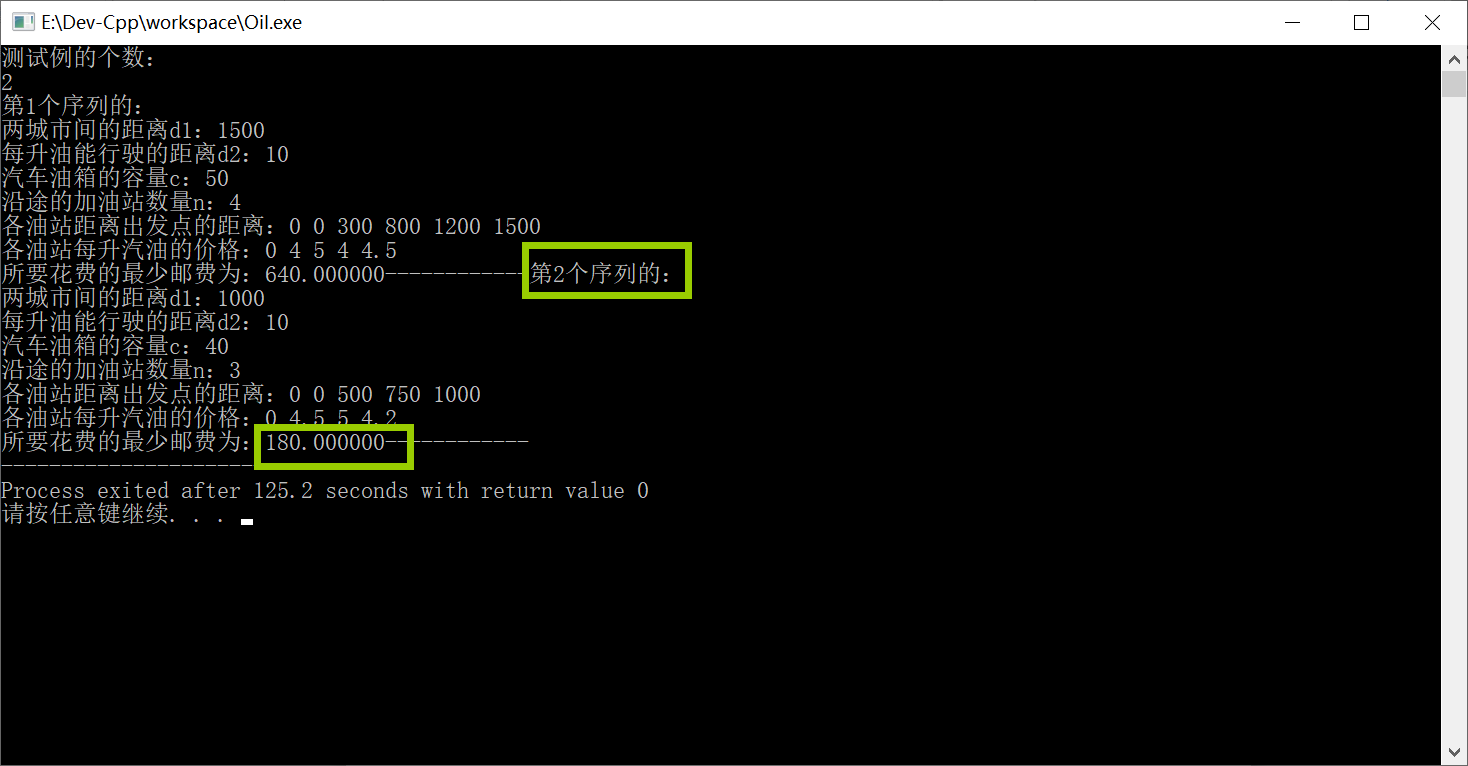


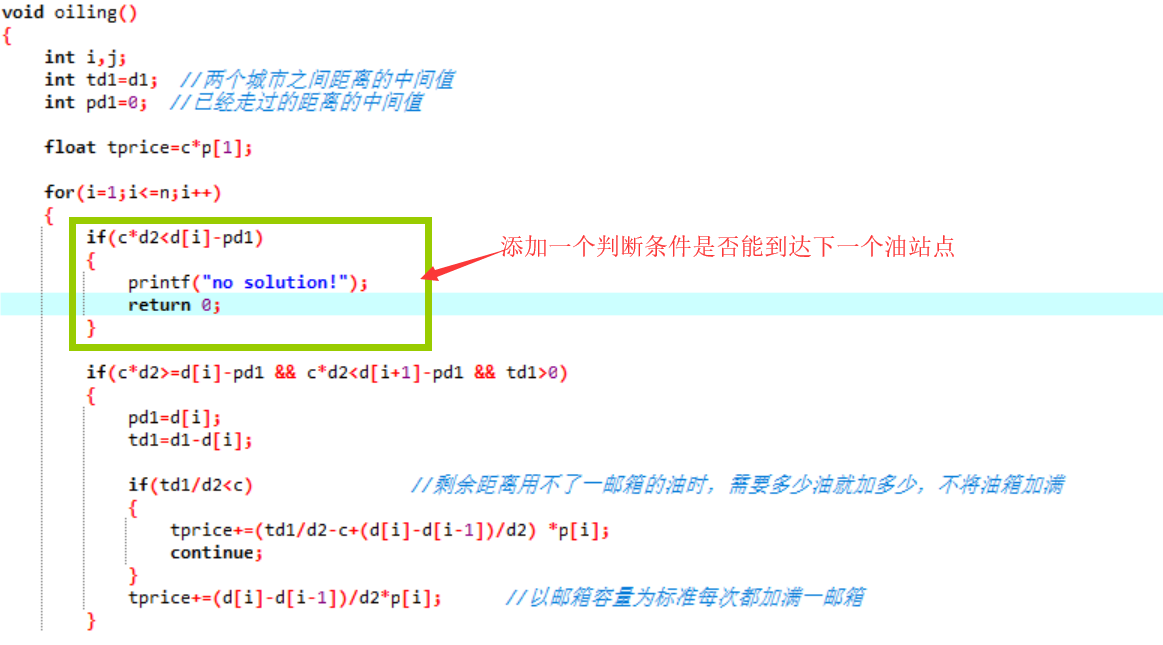


需要重新编译

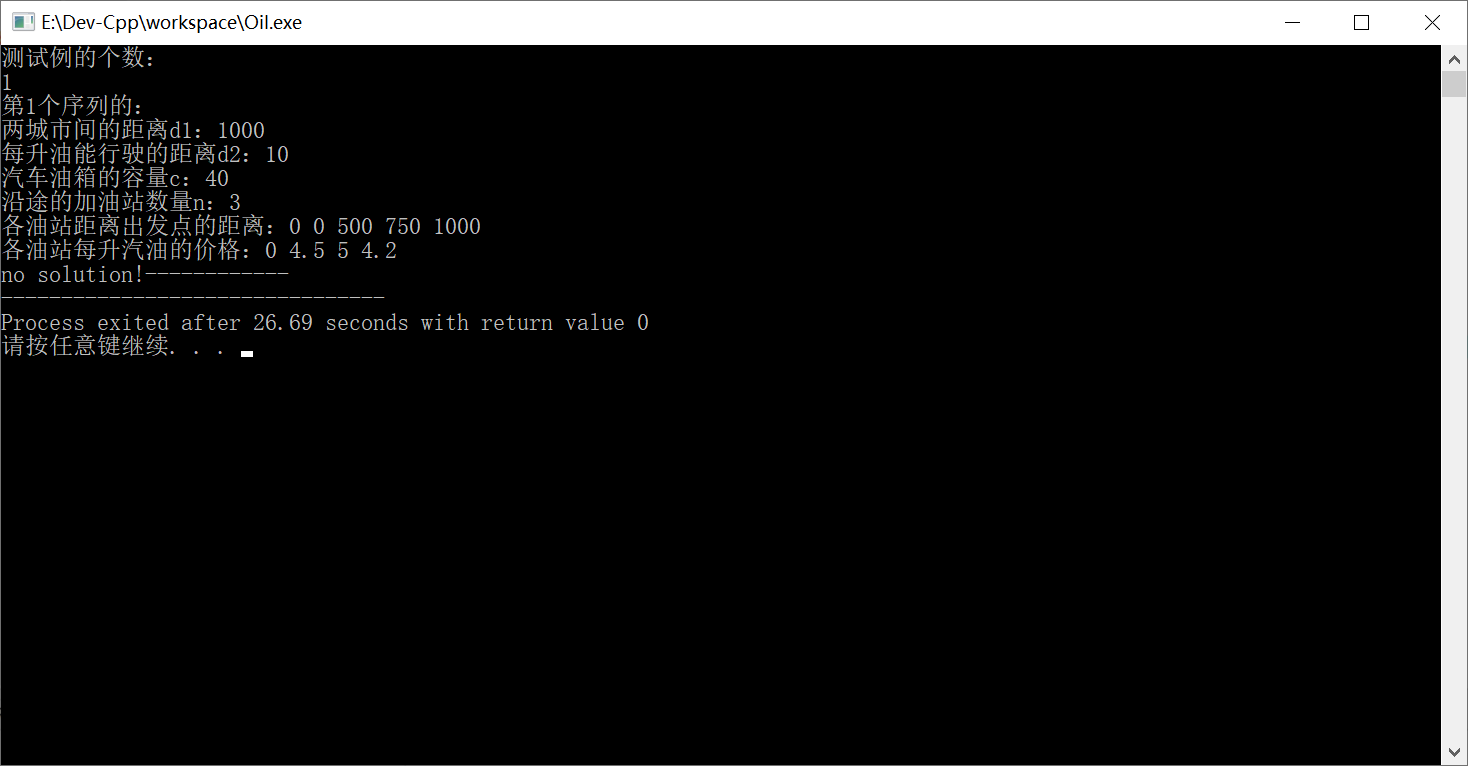


又出现了问题





完美！



# 2.2实验结果

代码1：

#include <stdio.h>

#include <stdlib.h>

#define MAX 200

int d1,c,d2,n; //d1两个城市之间的距离,d2每升汽油能行驶的距离,c汽车油箱的容量,沿途油站数n (1<=n<=200)

float tprice; //tprice需要的总价格

float d[MAX]; //各油站离出发点的距离（d1=0）

float p[MAX]; //各油站每升汽油的价格

//int d1=1500,c=50,d2=10,n=4;

//float tprice;

//float d[6]= {0, 0, 300.0, 800.0 ,1200.0,1500.0};

//float p[5]= {0, 4.0 ,5.0, 4.0, 4.5};

void oiling()

{

int i,j;

int td1=d1; //两个城市之间距离的中间值

int pd1=0; //已经走过的距离的中间值

float tprice=c\*p[1];

for(i=1;i<=n;i++)

{

if(c\*d2<d[i]-pd1)

{

printf("no solution!");

return ;

}

if(c\*d2>=d[i]-pd1 && c\*d2<d[i+1]-pd1 && td1>0)

{

pd1=d[i];

td1=d1-d[i];

if(td1/d2<c) //剩余距离用不了一邮箱的油时，需要多少油就加多少，不将油箱加满

{

tprice+=(td1/d2-c+(d[i]-d[i-1])/d2) \*p[i];

continue;

}

tprice+=(d[i]-d[i-1])/d2\*p[i]; //以邮箱容量为标准每次都加满一邮箱

}

}

printf("所要花费的最少邮费为：%3f",+tprice);

}

int main(int argc, char \*argv[])

{

int m,i,j;

printf("测试例的个数：\n");

scanf("%d",&m);

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

{

printf("第%d个序列的：\n",i+1);

printf("两城市间的距离d1：");

scanf("%d",&d1);

printf("每升油能行驶的距离d2：");

scanf("%d",&d2);

printf("汽车油箱的容量c：");

scanf("%d",&c);

printf("沿途的加油站数量n：");

scanf("%d",&n);

printf("各油站距离出发点的距离：");

for(j=0;j<n+2;j++)

{

scanf("%f",&d[j]);

}

printf("各油站每升汽油的价格：");

for(j=0;j<n+1;j++)

{

scanf("%f",&p[j]);

}

oiling();

printf("\n ------------ \n");

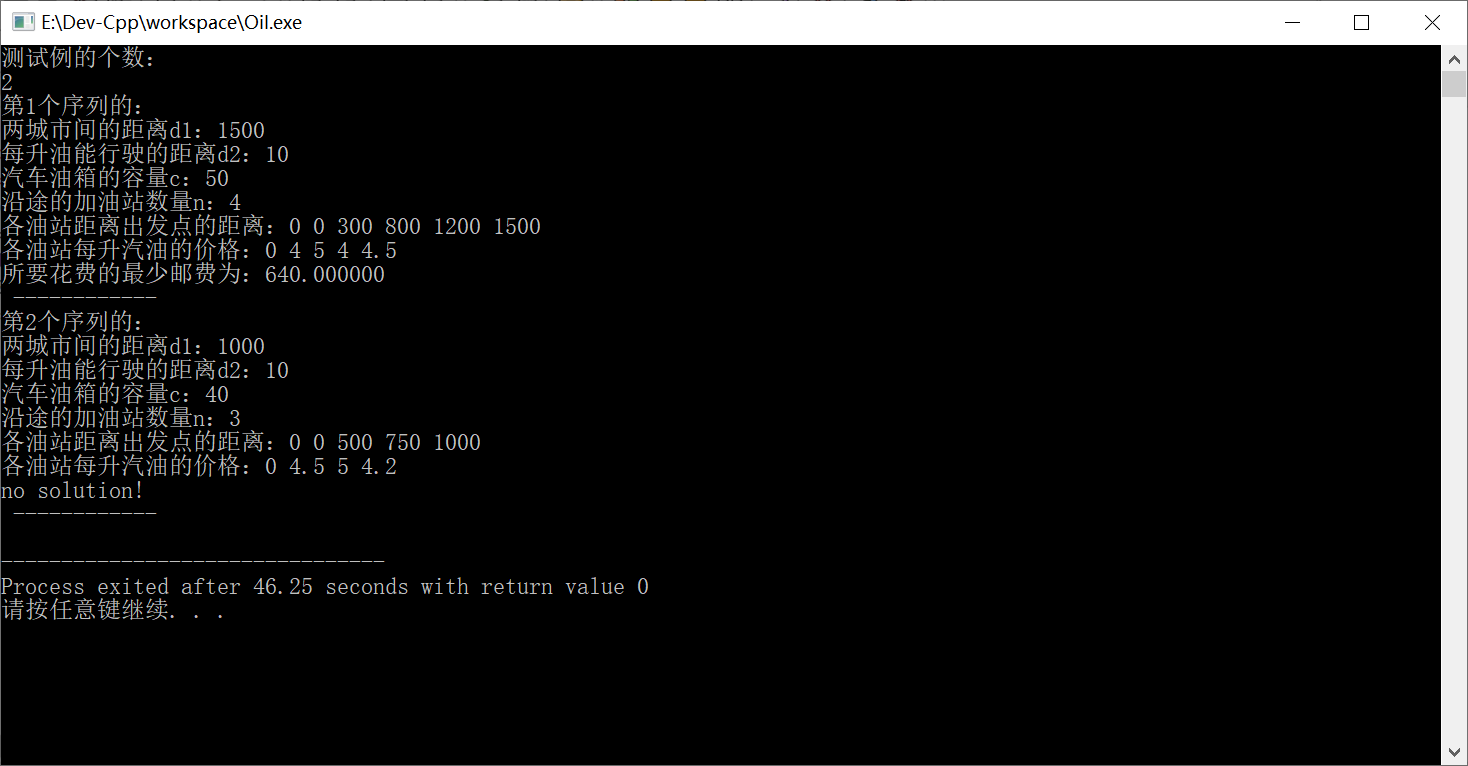
}

return 0;

}

**结果：**

终于成功了！！！！



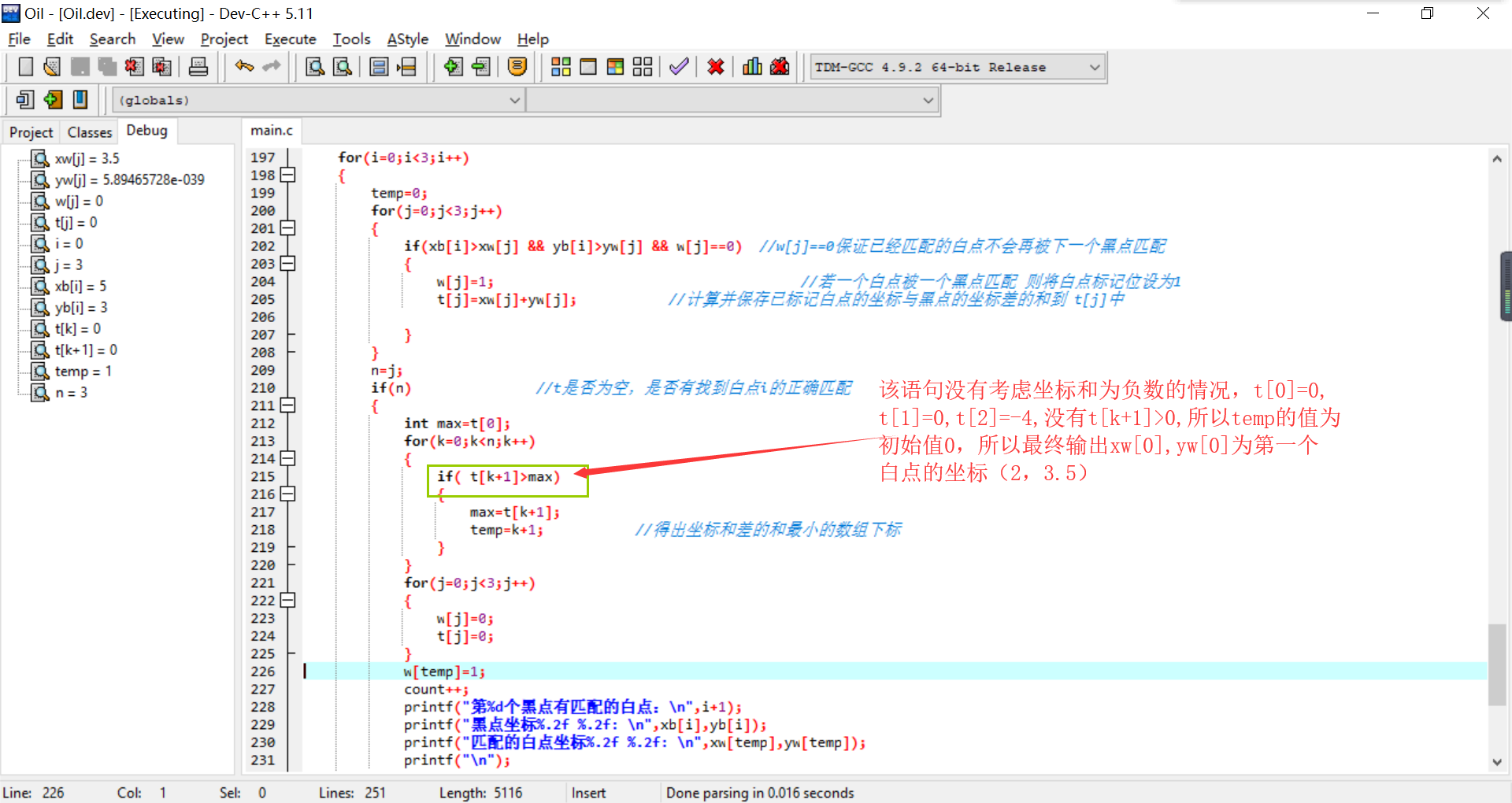
**3.1实验记录**

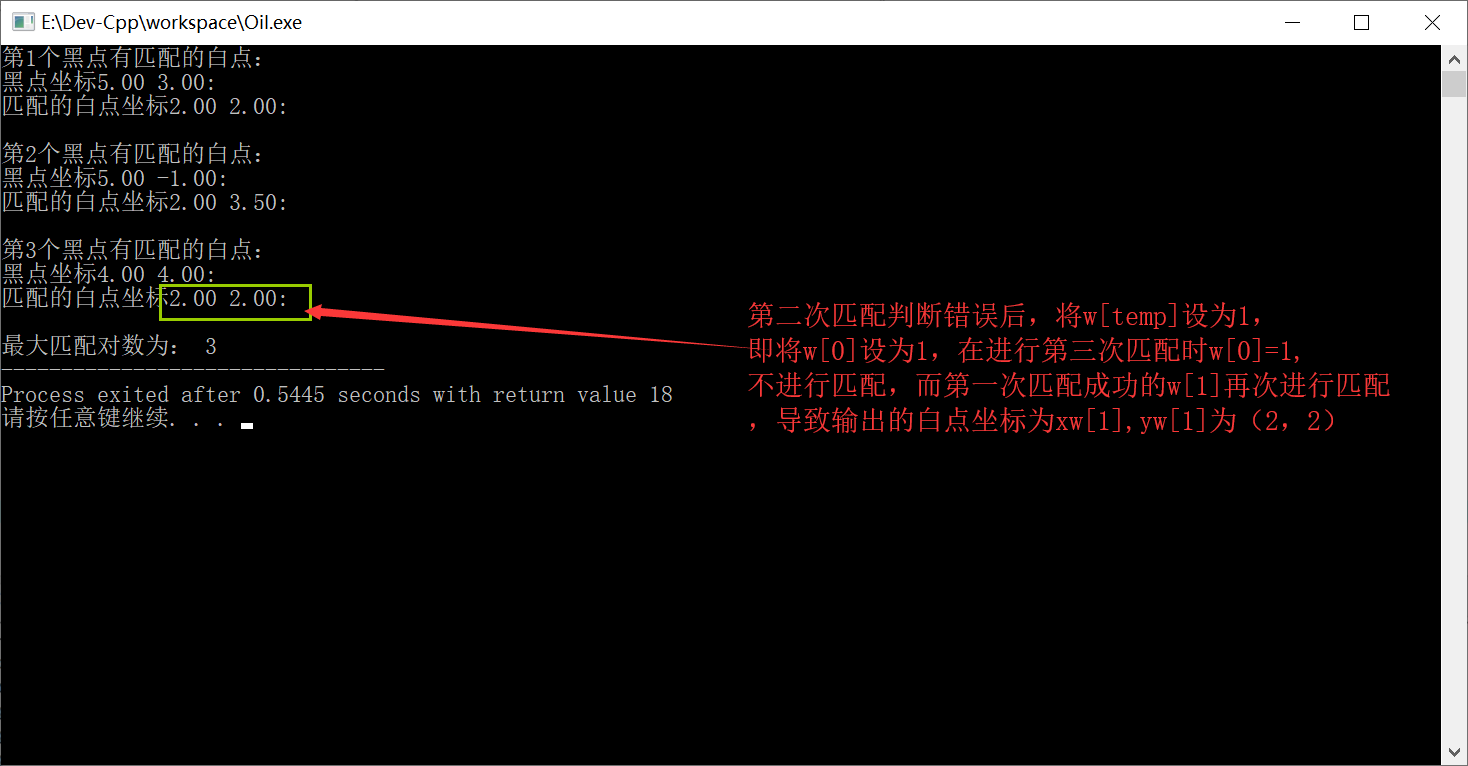
1. 算法思想

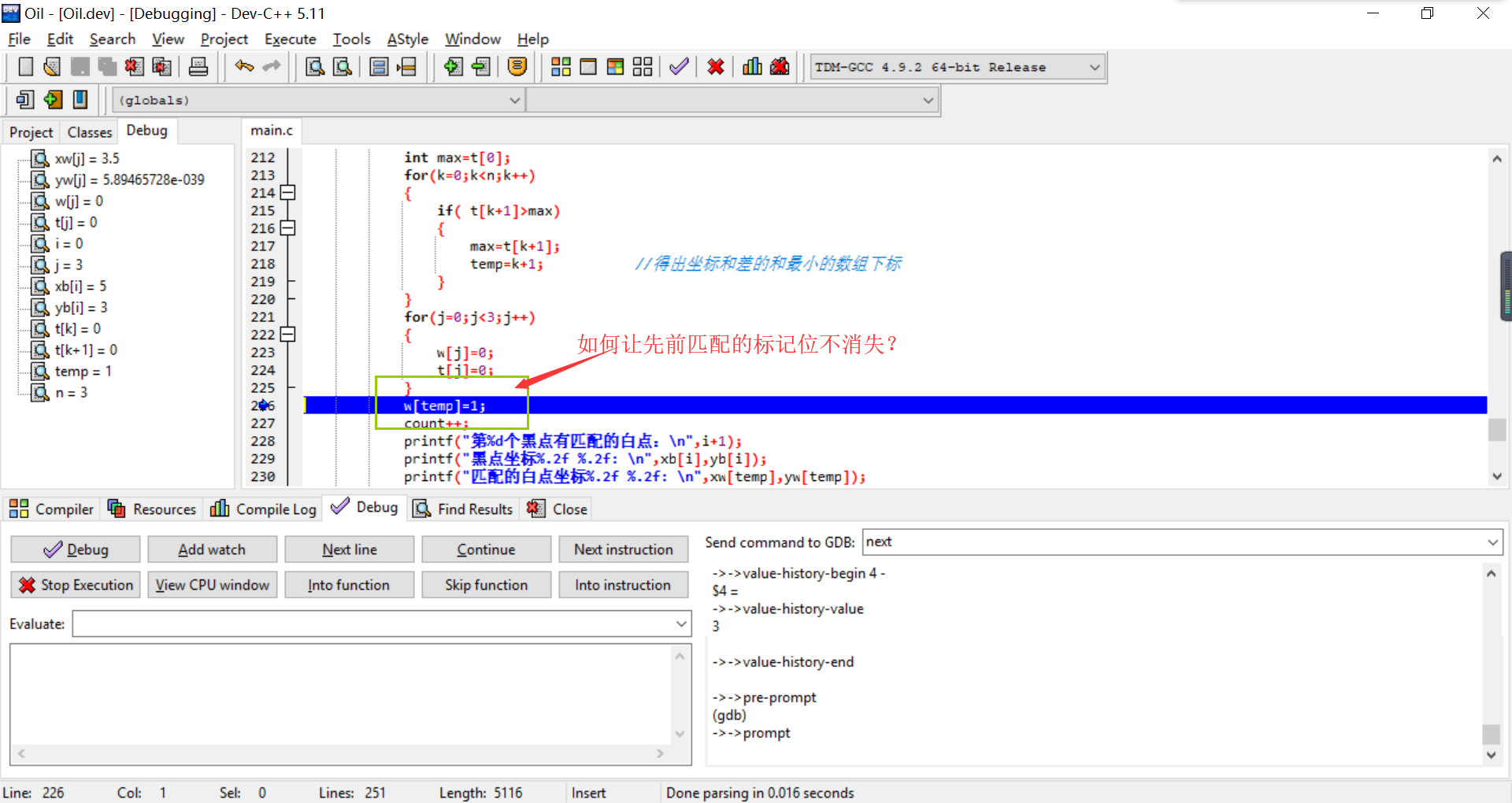
在满足xb>=xw，yb>=yw，w[j]==0(标记位为0表示还未匹配)的条件下，将可以匹配成功的白点的标记位w[j]设为1，通过计算标记位为1且坐标和最大的白点，该点即为与黑点最佳匹配的白点。重置所有标记位为0，将已匹配的的白点的标记位设为1,进行下一个黑点的匹配。

1. 实验记录

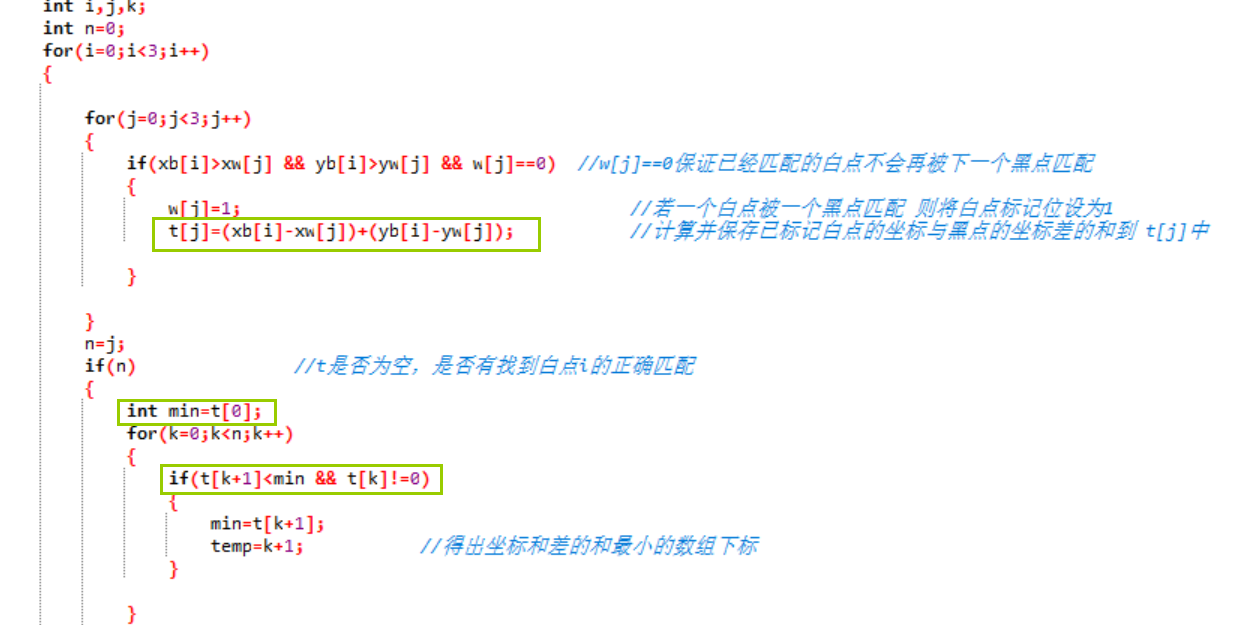
先考虑坐标和最大的情况，结果出现了问题

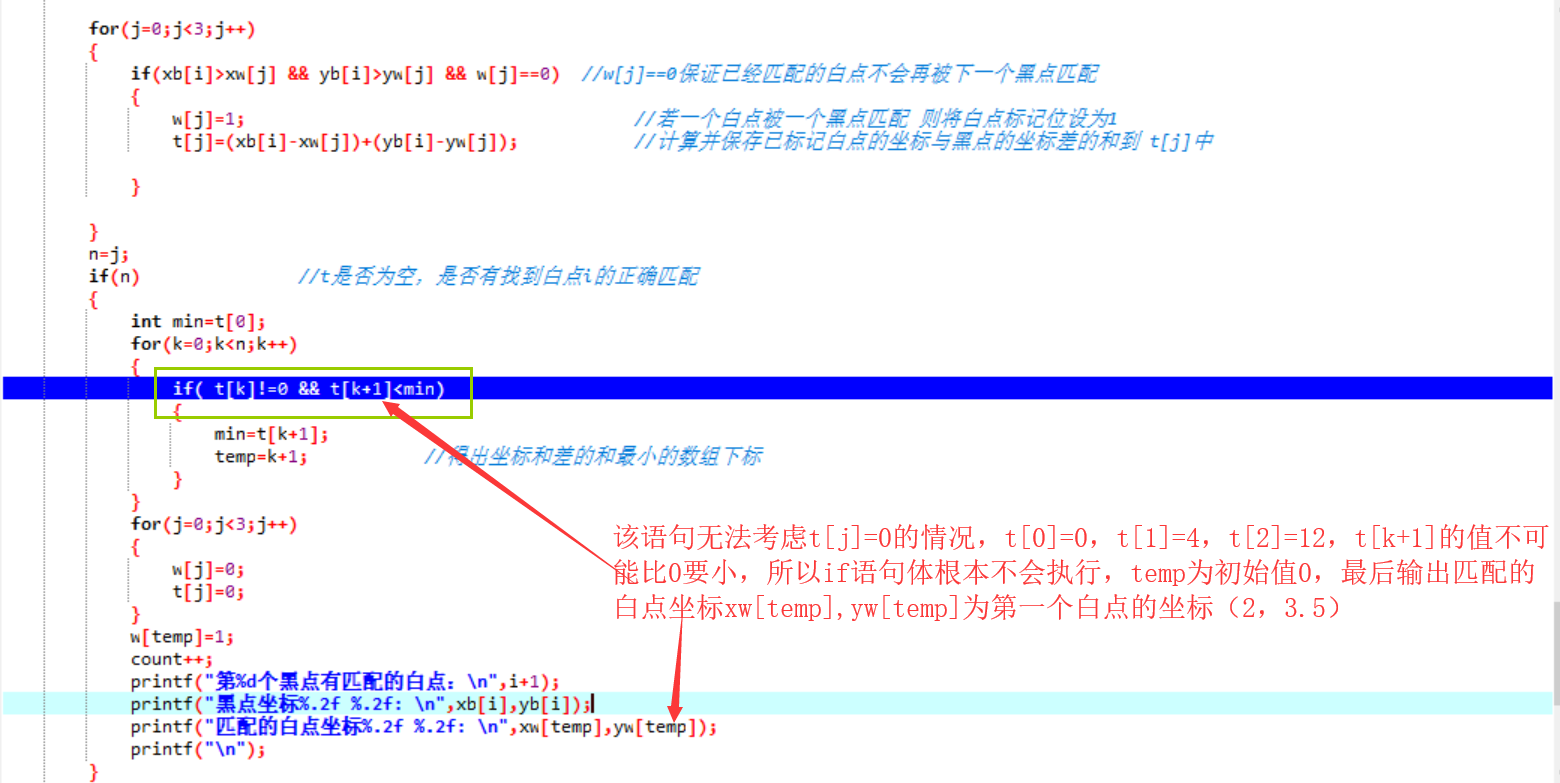


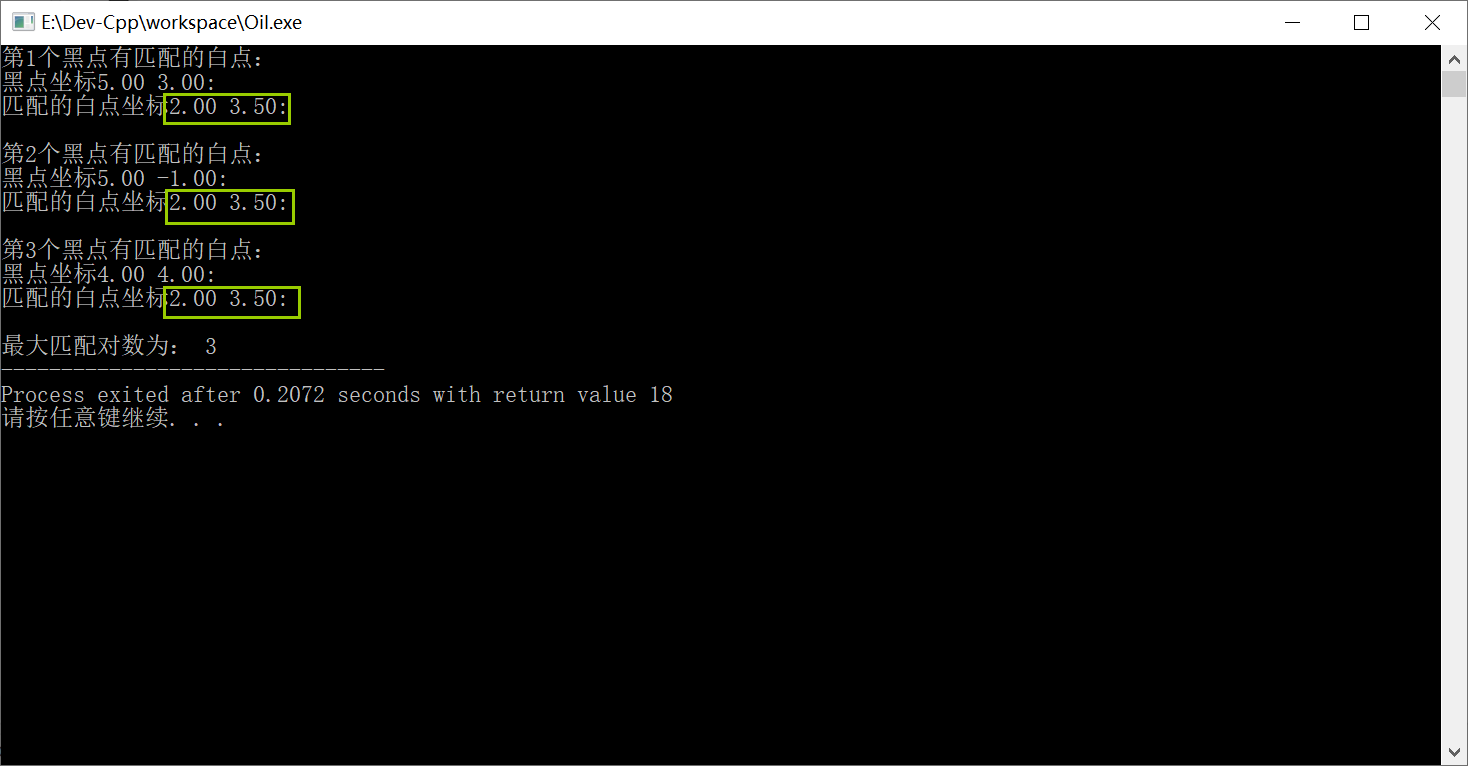




因为算法结果不正确，又考虑了黑点与白点差值最小







结果还是不正确，希望老师可以指点指点

# 3.2实验结果

代码1：

#include<stdio.h>

#include<string.h>

#define MAX 50

float xb[MAX]; //黑点的横坐标

float yb[MAX]; //黑点的纵坐标

float xw[MAX]; //白点的横坐标

float yw[MAX]; //白点的纵坐标

int count=0;

int temp=0;

int n=0;

float w[MAX],t[MAX];

void match()

{

int i,j,k,s;

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

{

temp=0;

for(j=0;j<n;j++)

{

if(xb[i]>xw[j] && yb[i]>yw[j] && w[j]==0) //w[j]==0保证已经匹配的白点不会再被下一个黑点匹配

{

w[j]=1; //若一个白点被一个黑点匹配 则将白点标记位设为1

t[j]=xw[j]+yw[j]; //已标记白点的坐标之和赋值给 t[j]

}

}

s=j;

if(s) //t是否为空，是否有找到白点i的正确匹配

{

int max=t[0];

for(k=0;k<s;k++)

{

if(t[k+1]>max)

{

max=t[k+1];

temp=k+1; //得出坐标和最大的数组下标

}

}

for(j=0;j<n;j++)

{

w[j]=0;

t[j]=0;

}

w[temp]=1;

count++;

printf("第%d个黑点有匹配的白点：\n",i+1);

printf("黑点坐标%.2f %.2f: \n",xb[i],yb[i]);

printf("匹配的白点坐标%.2f %.2f: \n",xw[temp],yw[temp]);

printf("\n");

}

}

printf("最大匹配对数为： %d",+count);

}

void main ()

{

int i,j,m;

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

{

t[i]=0; //白点坐标和数组

w[i]=0; //白点标记位数组

}

printf("测试例的个数：\n");

scanf("%d",&m);

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

{

printf("第%d个序列的个数：\n",i+1);

scanf("%d",&n);

printf("请输入黑点的坐标：\n");

for(j=0;j<n;j++)

{

scanf("%f",&xb[j]);

scanf("%f",&yb[j]);

}

printf("请输入白点的坐标：\n");

for(j=0;j<n;j++)

{

scanf("%f",&xw[j]);

scanf("%f",&yw[j]);

}

printf("\n");

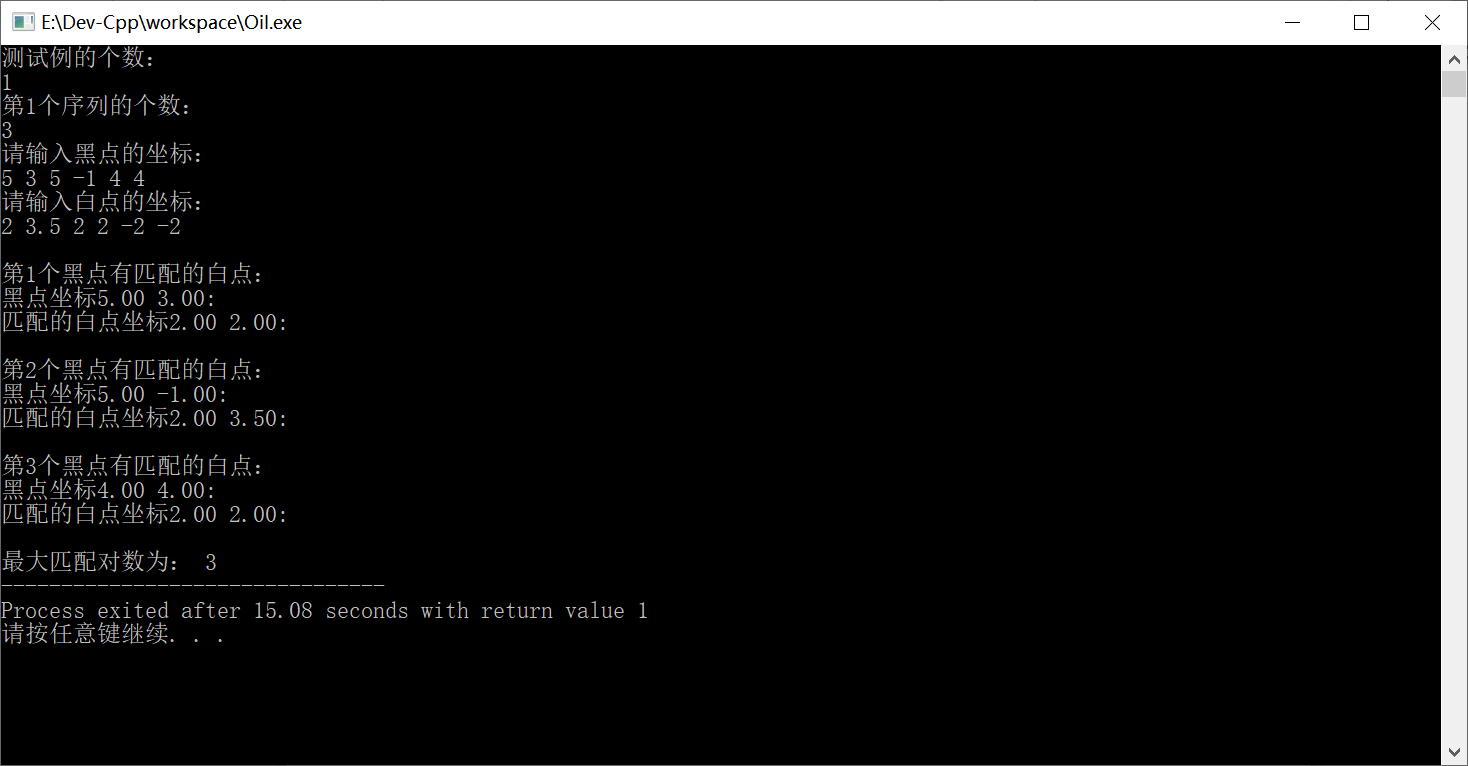
match();

}

return;

}

结果：



# 4.实验总结

1. 第一次在没有参考任何资料的情况下独立做出实验一，实验二（虽然实验二没有成功......），虽然不知道自己编写的算法属不属于贪心算法，但是能做出来已经很开心了!
2. 慢慢发现自己开始对算法思想有了一些领悟，把实际问题----->代码思想----->实际的编程语言，开始对这个转化过程不再害怕，不再苦恼，一步一步来，认真思考，然后发现其实也不是难到无法下手的那种程度，自信心拾起来一点了！
3. 用无数个小时写出来一个错误的算法，也是让我很难受呀。完成算法实验报告的过程就是“在成功中跌倒，在失败中趴下（爬起）”。

遇到的问题：

黑白点匹配问题的算法，中间有个过程不知道怎么修改，不知道如何考虑负数情况和为0的情况，希望老师在看完之后，可以教教我，要怎么修改我的代码就可以得出正确结果！

希望老师可以给我您认为的优秀的代码，让我参考参考，让我长点见识，谢谢老师！