项目说明文档

数据结构课程设计

——银行业务

同济大学

Tongji University

目 录

[1 分析 1](#_Toc1083)

[1.1 背景分析 1](#_Toc8084)

[1.2 功能分析 1](#_Toc9897)

[2 设计 1](#_Toc889)

[2.1 数据结构设计 1](#_Toc24208)

[2.2 成员与操作设计 1](#_Toc25001)

[3 实现 2](#_Toc8304)

[3.1 输入功能的实现 2](#_Toc15447)

[3.1.1 输入功能流程图 2](#_Toc3743)

[3.1.2 输入功能核心代码 2](#_Toc22976)

[3.2 重复判断功能的实现 4](#_Toc6335)

[3.2.1 重复判断功能流程图 4](#_Toc28132)

[3.2.2 重复判断功能核心代码 4](#_Toc21156)

[3.3 输出功能的实现 5](#_Toc1561)

[3.3.1 输出功能流程图 5](#_Toc22201)

[3.3.2 输出功能核心代码 5](#_Toc15825)

[3.4总体系统实现 7](#_Toc3420)

[3.4.1 总体系统流程图 7](#_Toc7624)

[3.7.2总体系统核心代码 7](#_Toc26201)

[4 测试 11](#_Toc16467)

[4.1 功能测试 11](#_Toc29649)

[4.1.1 奇数表数字更多 11](#_Toc16326)

[4.1.2 偶数表数字更多 11](#_Toc26761)

[4.2 出错测试 12](#_Toc18571)

[4.2.1人数错误 12](#_Toc22810)

[4.2.2 编号错误 12](#_Toc5983)

[4.2.3 数字有重复 13](#_Toc18346)

# 1 分析

## 1.1 背景分析

通过利用两个链表，一个用于存放奇数，一个用于存放偶数，在输入的同时还要进行错误的判断，包括数量是否为正数，输入的数是否为正整数，输入的数是否有重复的情况出现

## 1.2 功能分析

需要建立两个链表，一段检验错误的代码段，一段有序输出的代码段，所以需要输入功能，错误检验功能，输出功能

# 

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求使用两个链表，这样可以灵活的开辟动态的空间进行储存

## 2.2 成员与操作设计

类定义

class stack

{

public:

int number;

stack\* link;

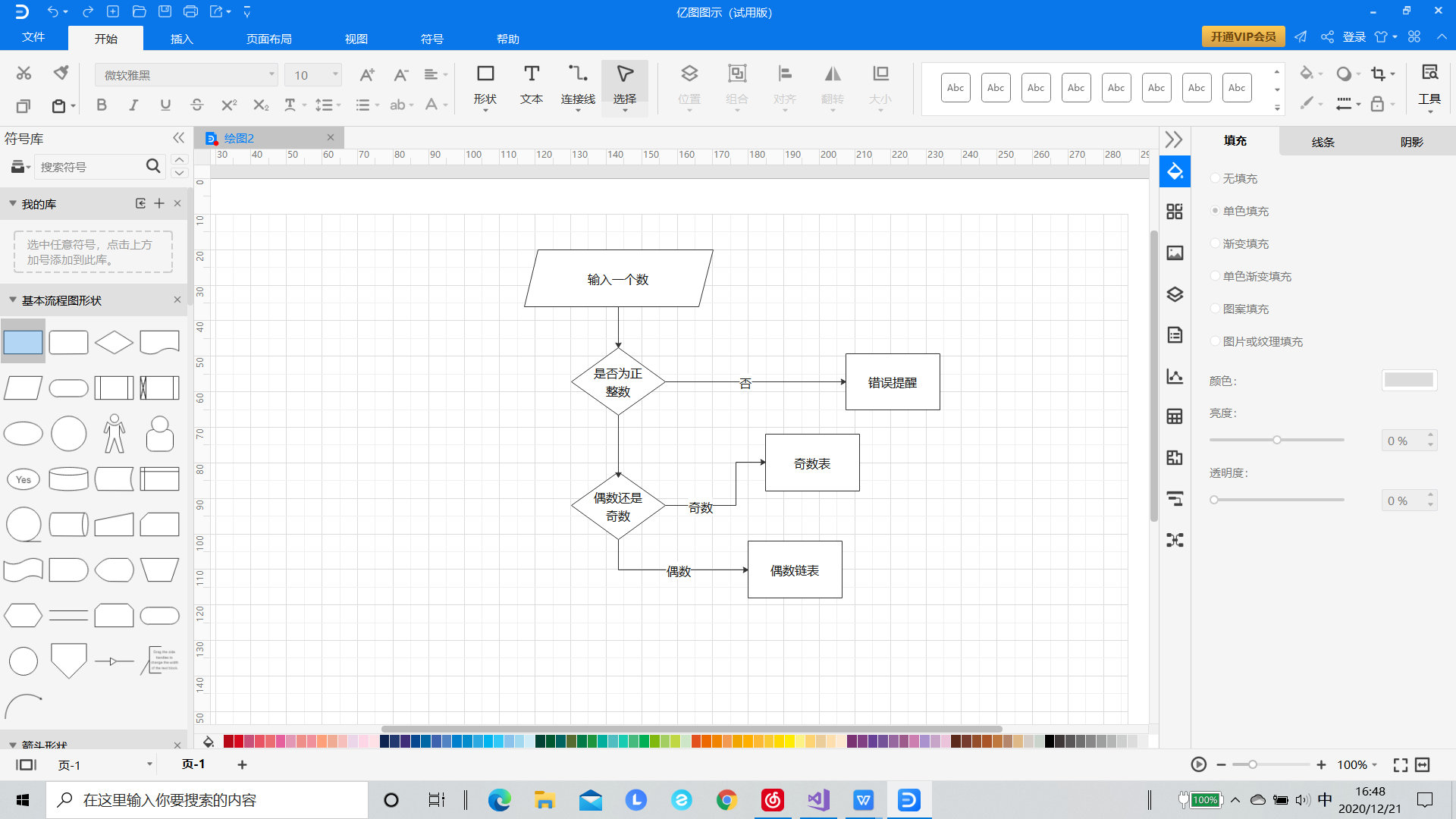
stack() { number = 0; }

};

# 3 实现

## 3.1 输入功能的实现

### 3.1.1 输入功能流程图



### 3.1.2 输入功能核心代码

current1 = new stack;

current2 = new stack;

head1 = current1;

head2 = current2;

cin >> total;

if (total <= 0) { cout << "总数不能小于0" << endl; }

else

{

for (int n = 0; n <= total - 1; n++)

{

cin >> putin[n];

if (putin[n] <= 0) { cout << "数列中有负数或零" << endl; wrong++; break; }

else

{

if (putin[n] % 2 != 0)

{

current1->link = new stack;

current1 = current1->link;

current1->number = putin[n];

a++;

}

else

{

current2->link = new stack;

current2 = current2->link;

current2->number = putin[n];

b++;

}

}

### }

## 3.2 重复判断功能的实现

### 3.2.1 重复判断功能流程图

### 

### 3.2.2 重复判断功能核心代码

if (wrong == 0)

{

current1 = head1;

current2 = head2;

for (int n = 1; n <= a; n++)//检验是否奇数表中有重复的数

{

current1 = current1->link;

stack \*check;

check = current1;

for (int m = 1; m <= a - n; m++)

{

check = check->link;

if (check->number == current1->number) { checka++; break; }

}

if (checka != 0) { break; }

}

for (int n = 1; n <= b; n++)//检验是否偶数表中有重复的数

{

current2 = current2->link;

stack \*check;

check = current2;

for (int m = 1; m <= b - n; m++)

{

check = check->link;

if (check->number == current2->number) { checkb++; break; }

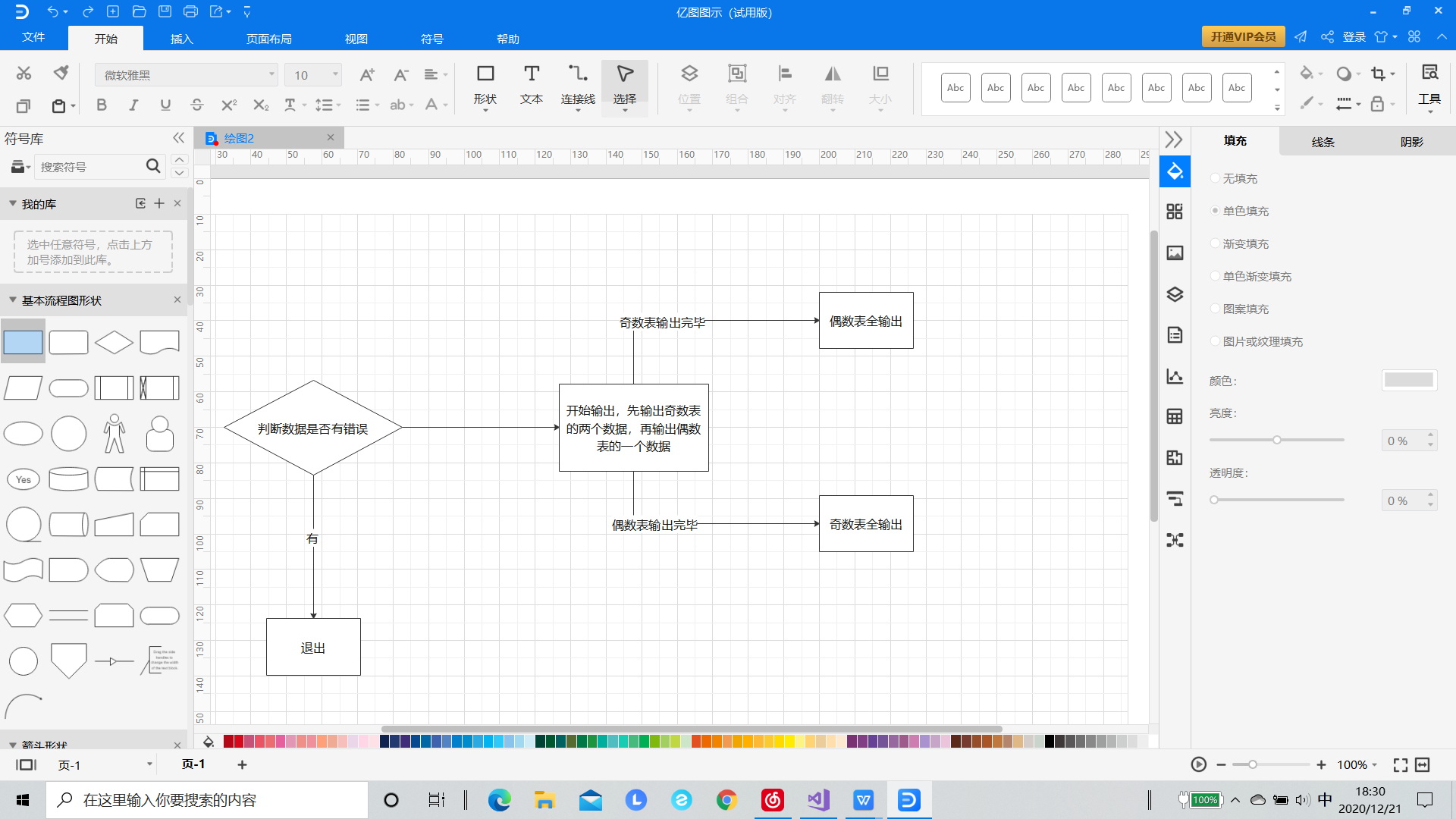
}

if (checkb != 0) { break; }

}

## 3.3 输出功能的实现

### 3.3.1 输出功能流程图



### 3.3.2 输出功能核心代码

for (;;)

{

if (a >= 2)

{

cout << head1->link->number << " ";

head1 = head1->link;

cout << head1->link->number << " ";

head1 = head1->link;

a--; a--;

}

else

{

for (int n = 1; n <= a; n++)

{

cout << head1->link->number << " ";

head1 = head1->link;

}

a = 0;

}

if (b != 0)

{

cout << head2->link->number << " ";

head2 = head2->link;

b--;

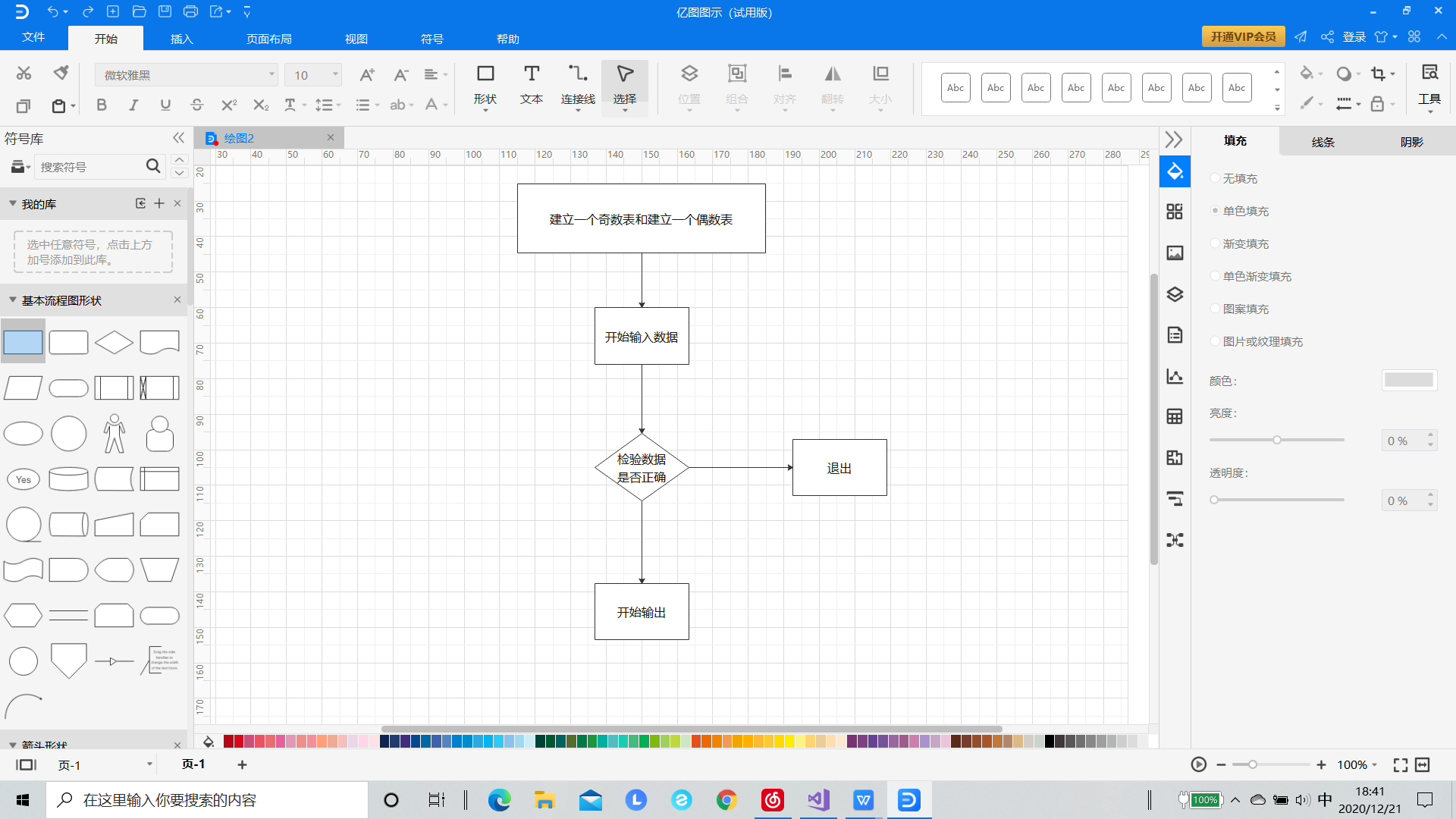
}

if (a == 0 && b == 0) { break; }

### }

# 3.4总体系统实现

### 3.4.1 总体系统流程图



### 3.7.2总体系统核心代码

int main()

{

int total = 0;

int putin[1000] = {0};

int a, b;

stack\* head1, \* head2;

stack\* current1, \* current2;

int checka, checkb, wrong;

for (;;)

{

a = b = 0;

wrong = checka = checkb = 0;

current1 = new stack;

current2 = new stack;

head1 = current1;

head2 = current2;

cin >> total;

if (total <= 0) { cout << "总数不能小于0" << endl; }

else

{

for (int n = 0; n <= total - 1; n++)

{

cin >> putin[n];

if (putin[n] <= 0) { cout << "数列中有负数或零" << endl; wrong++; break; }//判断是否为输入的数据有错误

else

{

if (putin[n] % 2 != 0) //分别加入到奇数表中和偶数表中

{

current1->link = new stack;

current1 = current1->link;

current1->number = putin[n];

a++;

}

else

{

current2->link = new stack;

current2 = current2->link;

current2->number = putin[n];

b++;

}

}

}

//check

if (wrong == 0)

{

current1 = head1;

current2 = head2;

for (int n = 1; n <= a; n++)

{

current1 = current1->link;

stack \*check;

check = current1;

for (int m = 1; m <= a - n; m++)

{

check = check->link;

if (check->number == current1->number) { checka++; break; }

}

if (checka != 0) { break; }

}

for (int n = 1; n <= b; n++)

{

current2 = current2->link;

stack \*check;

check = current2;

for (int m = 1; m <= b - n; m++)

{

check = check->link;

if (check->number == current2->number) { checkb++; break; }

}

if (checkb != 0) { break; }

}

if (checka == 0 && checkb == 0)

{

for (;;)

{

if (a >= 2)

{

cout << head1->link->number << " ";

head1 = head1->link;

cout << head1->link->number << " ";

head1 = head1->link;

a--; a--;

}

else

{

for (int n = 1; n <= a; n++)

{

cout << head1->link->number << " ";

head1 = head1->link;

}

a = 0;

}

if (b != 0)

{

cout << head2->link->number << " ";

head2 = head2->link;

b--;

}

if (a == 0 && b == 0) { break; }

}

}

else

{

cout << "输入中有重复的数字" << endl;

}

}

}

cout << "重复实验?" << " " << "(Y,N)";

char k;

cin >> k;

for (;;)

{

if (k == 'N' || k == 'Y') { break; }

else { cout << "请输入正确字符" << endl; }

}

if (k == 'N') { break; }

}

# 4 测试

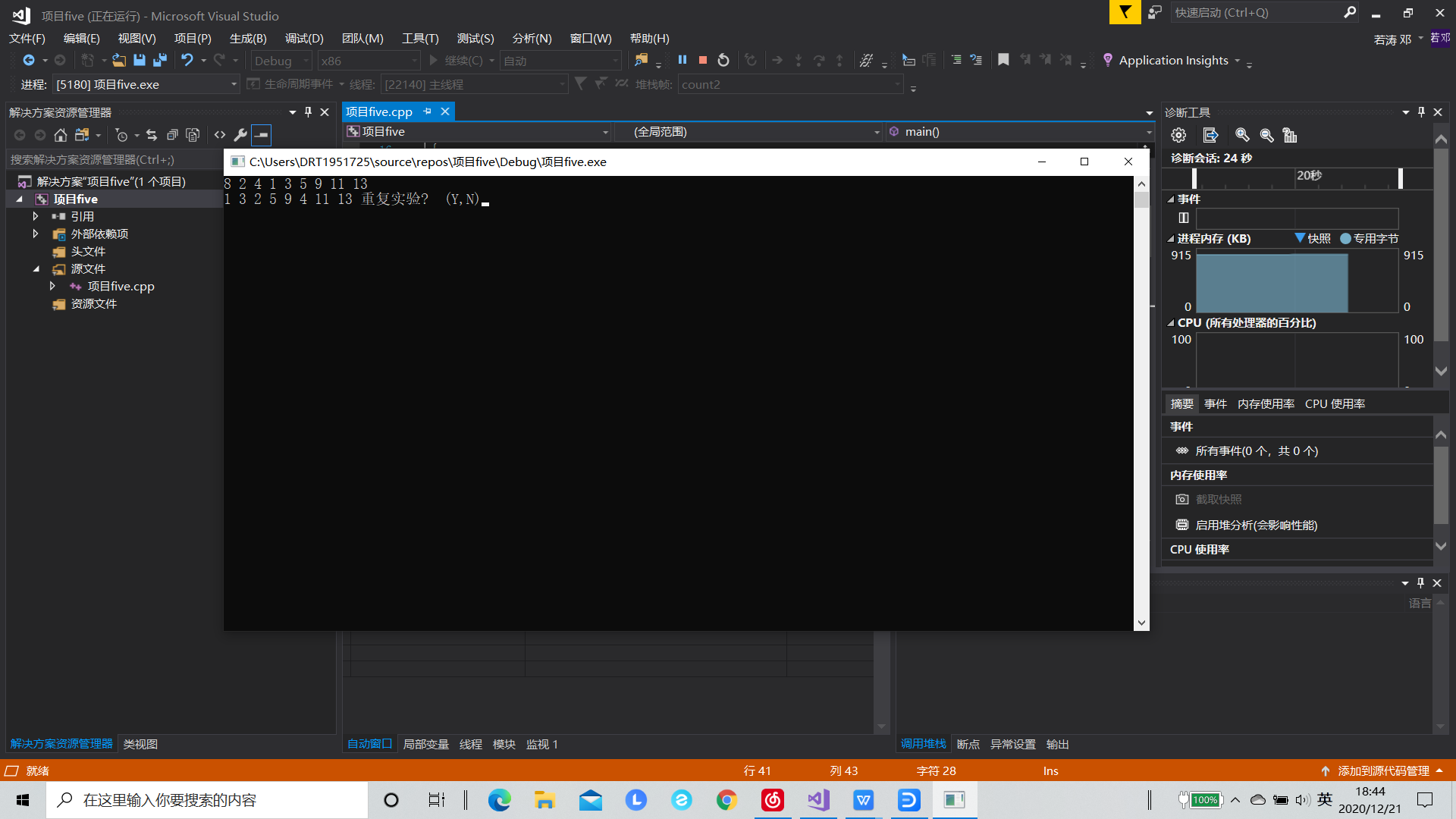
## 4.1 功能测试

### 4.1.1 奇数表数字更多

**测试用例：8 2 4 1 3 5 9 11 13**

**预期结果：1 3 2 5 9 4 11 13**

**实验结果：**

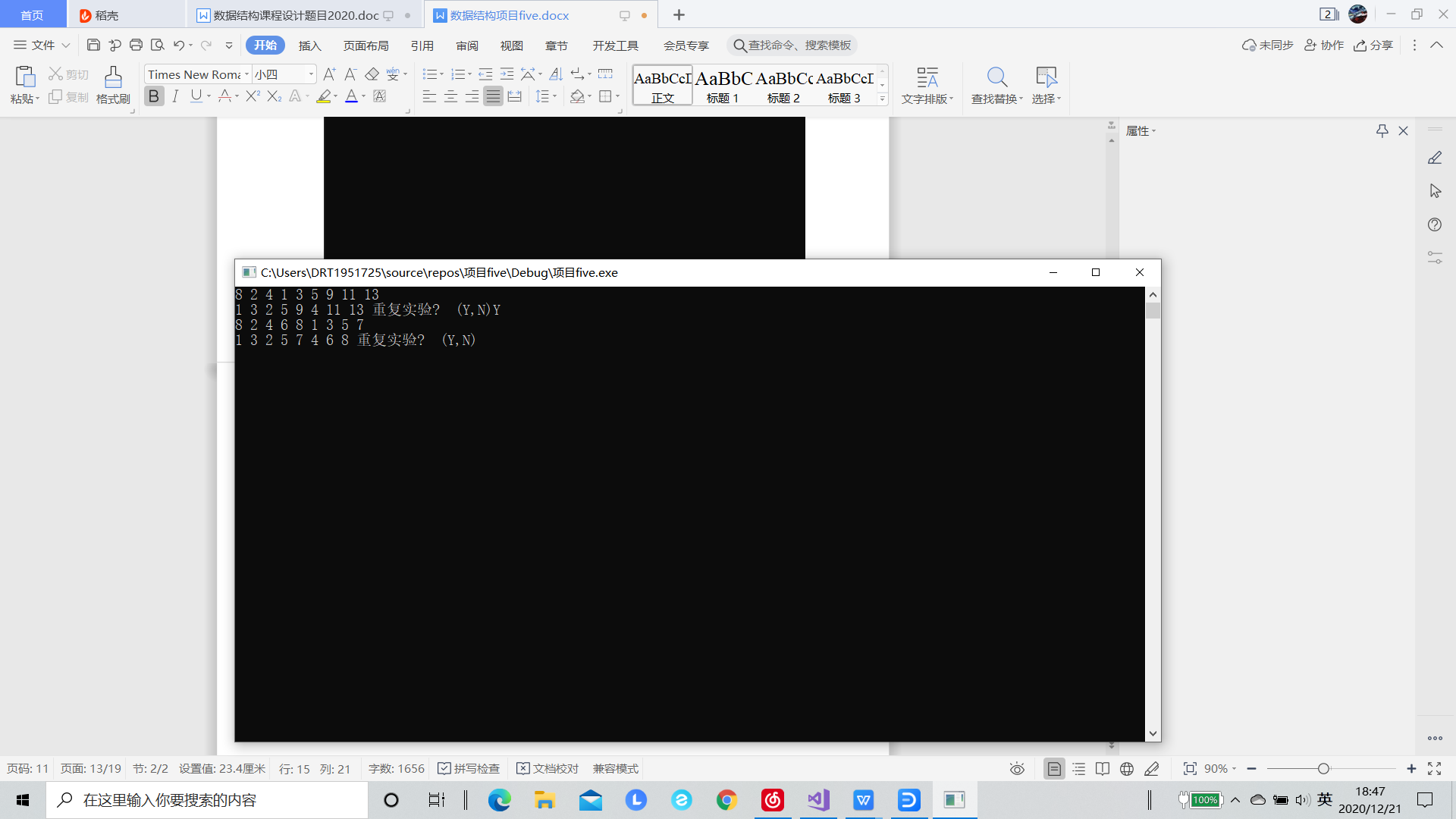


### 4.1.2 偶数表数字更多

**测试用例：8 2 4 6 8 1 3 5 7**

**预期结果：1 3 2 5 7 4 6 8**

**实验结果：**



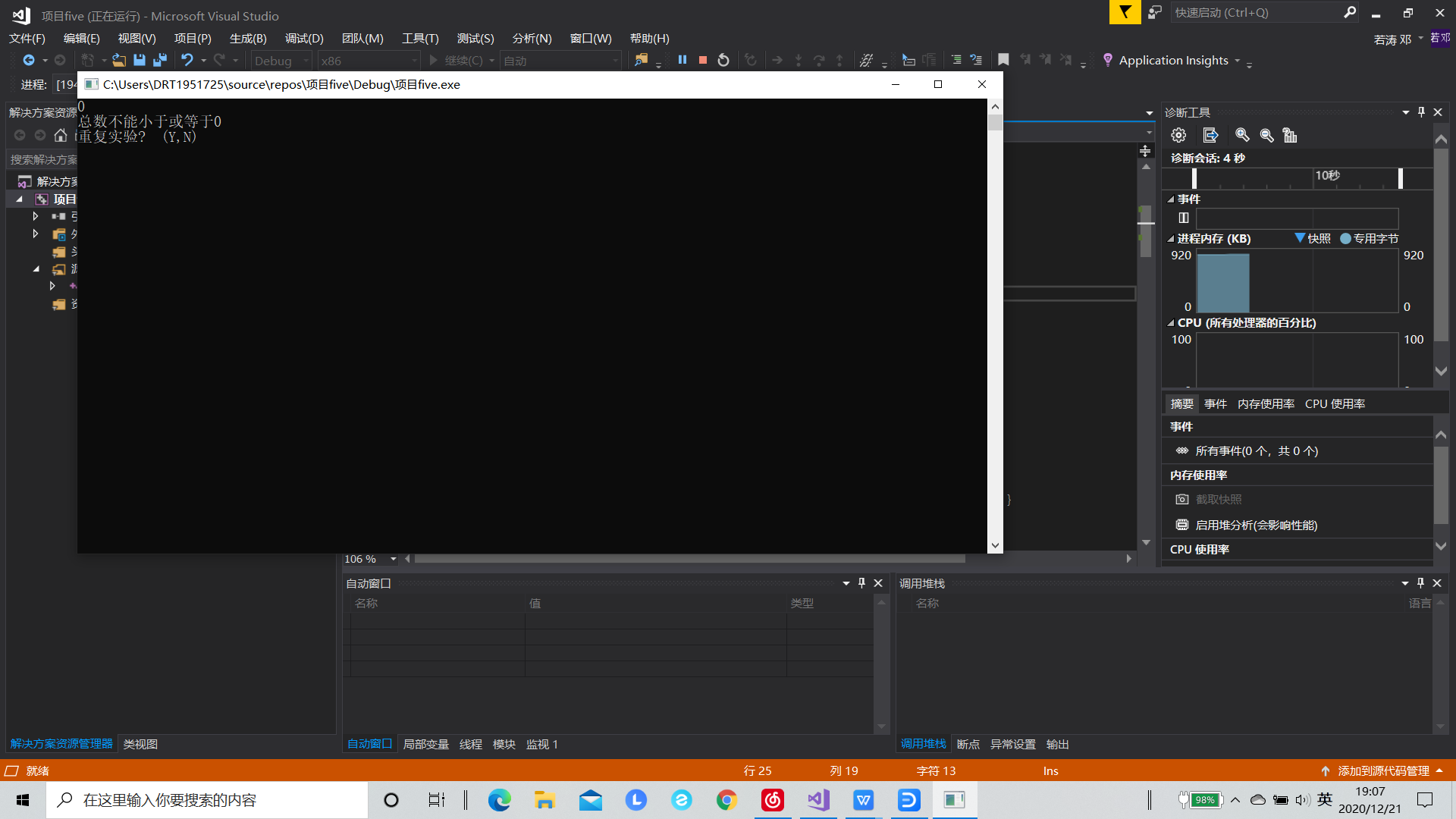
## 4.2 出错测试

### 4.2.1人数错误

**测试用例：**输入的数字小于或等于0

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

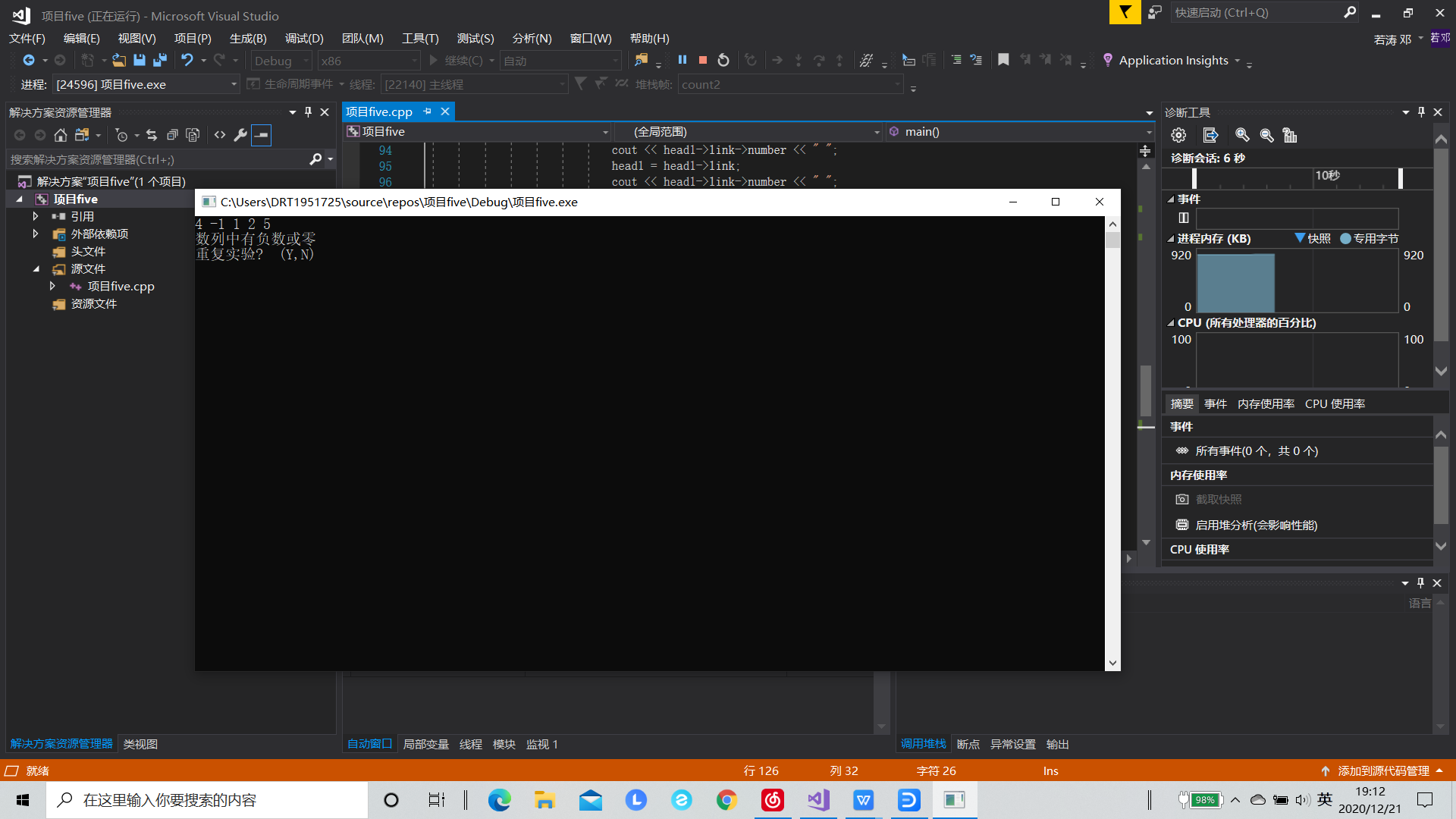


### 4.2.2 编号错误

**测试用例：4 -1 1 2 5**

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**



### 4.2.3 数字有重复

**测试用例：5 1 1 3 4 5**

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

