项目说明文档

数据结构课程设计

——家谱管理系统

同济大学

Tongji University

目 录

[1 分析 1](#_Toc23872)

[1.1 背景分析 1](#_Toc22965)

[1.2 功能分析 1](#_Toc28123)

[2 设计 1](#_Toc32201)

[2.1 数据结构设计 1](#_Toc28175)

[2.2 成员与操作设计 1](#_Toc28027)

[2.3 系统设计 2](#_Toc11570)

[3 实现 2](#_Toc26633)

[3.1 输入功能的实现 2](#_Toc28098)

[3.1.1 输入功能流程图 2](#_Toc3403)

[3.1.2 输入功能核心代码 3](#_Toc25151)

[3.1.3 输入功能截屏示例 4](#_Toc14248)

[3.2 添加功能的实现 6](#_Toc11731)

[3.2.1 添加功能流程图 6](#_Toc20110)

[3.2.2 添加功能核心代码 6](#_Toc7658)

[3.2.3 插入功能截屏示例 8](#_Toc20873)

[3.3 查找功能的实现 8](#_Toc25144)

[3.3.1 查找功能流程图 8](#_Toc12595)

[3.3.2 查找功能核心代码 9](#_Toc30234)

[3.3.3 查找功能截屏示例 10](#_Toc13717)

[3.4 修改功能的实现 10](#_Toc11385)

[3.4.1 修改功能流程图 10](#_Toc3185)

[3.4.2 修改功能核心代码 11](#_Toc7889)

[3.4.3 查找功能截屏示例 11](#_Toc4166)

[3.5 解散子女功能的实现 12](#_Toc17369)

[3.5.1 解散子女功能流程图 12](#_Toc19514)

[3.5.2 解散子女功能核心代码 12](#_Toc22777)

[3.5.3 解散功能截屏示例 13](#_Toc21383)

[3.6 删除功能的实现 14](#_Toc2607)

[3.6.1 删除功能流程图 14](#_Toc6452)

[3.6.2 删除功能核心代码 14](#_Toc16051)

[3.6.3 排序功能截屏示例 16](#_Toc22584)

[3.7总体系统实现 17](#_Toc23817)

[3.7.1 总体系统流程图 17](#_Toc29444)

[3.7.2总体系统核心代码 18](#_Toc27419)

[4 测试 21](#_Toc12523)

[4.1 功能测试 21](#_Toc176)

[4.1.1 添加功能测试 21](#_Toc27510)

[4.1.2 查找功能测试 22](#_Toc9174)

[4.1.3 修改功能测试 23](#_Toc15313)

[4.1.4 解散功能测试 23](#_Toc28409)

[4.1.5 删除功能测试 24](#_Toc26854)

[4.2 边界测试 24](#_Toc29412)

[4.2.1 输入的结点不存在 24](#_Toc6090)

[4.2.2 输入的操作符非法 25](#_Toc8677)

# 1 分析

## 1.1 背景分析

家谱是一种以表谱形式，记载一个以血缘关系为主体的家族世袭繁衍和重要任务事迹的特殊图书体裁。家谱是中国特有的文化遗产，是中华民族的三大文献（国史，地志，族谱）之一，属于珍贵的人文资料，对于历史学，民俗学，人口学，社会学和经济学的深入研究，均有其不可替代的独特功能。本项目兑对家谱管理进行简单的模拟，以实现查看祖先和子孙个人信息，插入家族成员，删除家族成员的功能。

## 1.2 功能分析

本项目的实质是完成兑家谱成员信息的建立，查找，插入，修改，删除等功能，可以首先定义家族成员数据结构，然后将每个功能作为一个成员函数来完成对数据的操作，最后完成主函数以验证各个函数功能并得到运行结果。

# 

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求大量的增加、删除、修改操作，而且上下节点有关系，符合树的逻辑，而二叉树进行增加、删除等操作十分简便，因此考虑使用左长子—右兄弟的二叉树数据结构。同时，为了实现简易，在树类定义中增加了一个link指针用于前序遍历时节点的储存。

## 2.2 成员与操作设计

树的类

class family

{

private:

string name;

int sonnumber;

public:

family\*link;

family\*son;

family\*brother;

int add(family \*&p1, int &number); //添加子女函数

int remove(string namestring); //删除一个成员函数

int refamily();//解散家庭函数

int find(family \* &head, int &a, string b, family \*&p1); //寻找一个成员函数

int putin(int &a); //添加函数子函数

void show();//显示名字

void showson();//显示所有子女的名字

void changename(string b);//改变该成员的姓名

family(string b,family a);

family(string b);

family() { son = NULL; brother = NULL; sonnumber = 0; }

};

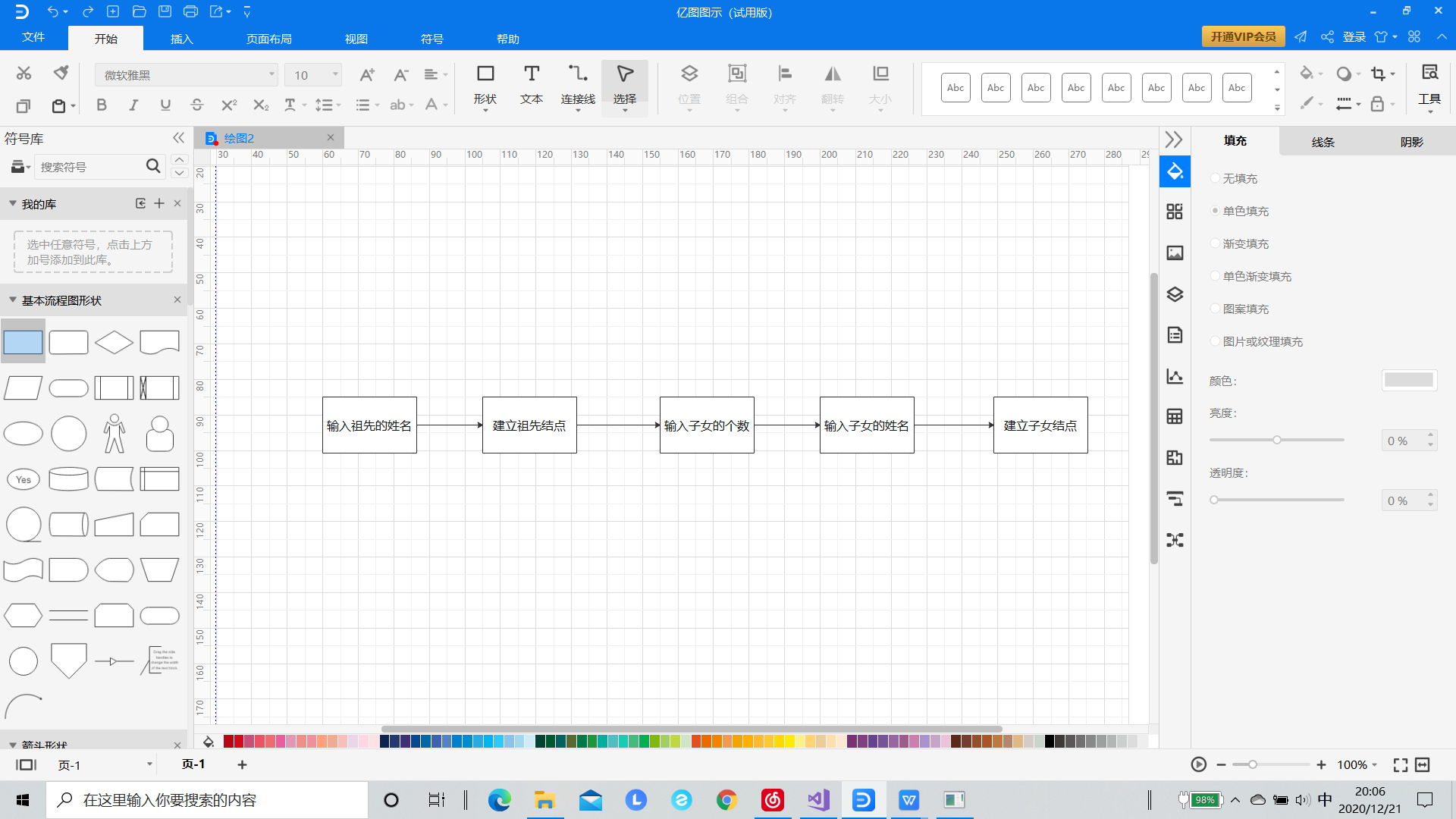
## 2.3 系统设计

系统首先在主函数中实现对家谱祖先与其子女的初始化，再通过for（）和swith语句进行不同功能的选择，通过类函数进行功能的实现

# 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

构造函数family（）

family::family(string b) //一个参数的初始化

{

name = b;

son = NULL; brother = NULL; sonnumber = 0;

}

family::family(string b, family a) //两个参数的初始化

{

family \* head;

head = new family;

family \* current;

int stacknumber = 0;

if (a.find(head, stacknumber, b, current) == 1)

{

}

name = b;

son = NULL; brother = NULL; sonnumber = 0;

}

输入函数putin（）

int family::putin(int &a)

{

sonnumber = a;

if (a >= 1)

{

string namestring1;

getline(cin, namestring1); //getline用于读入带有空格的字符串

istringstream is(namestring1); //istringsteam 用于分隔空格

is >> namestring1; // is 给 namestring1 不断进行赋值

son = new family(namestring1);

family \* p1;

p1 = son;

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

{

is >> namestring1;

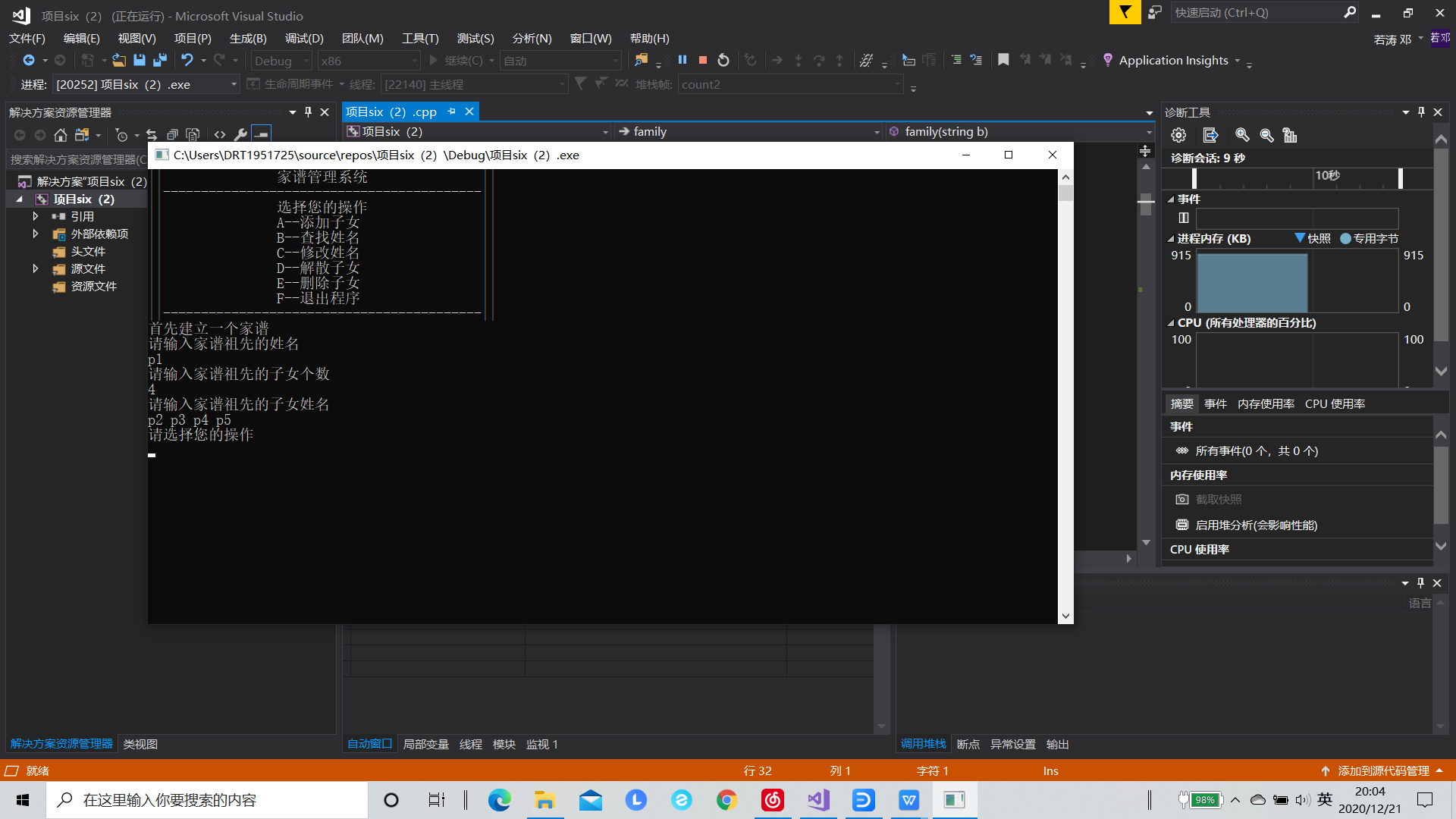
p1->brother = new family(namestring1);

p1 = p1->brother;

}

}

### 3.1.3 输入功能截屏示例



## 3.2 添加功能的实现

### 3.2.1 添加功能流程图

### 

### 3.2.2 添加功能核心代码

成员函数add

int family::add(family \*&p1, int &number) //添加函数

{

if (p1->son == NULL)//当该结点没有子女时

{

p1->putin(number);

}

Else //当该结点有子女时

{

family\* p2;

p2 = son;

for (;;) //不断寻找，直到找到子女的最右边的兄弟

{

if (p2->brother == NULL) { break; }

p2 = p2->brother;

}

string namestring; //添加兄弟

getline(cin, namestring);

istringstream is(namestring);

for (int m = 0; m < number; m++)

{

is >> namestring;

p2->brother = new family(namestring);

p2 = p2->brother;

}

}

return 1;

}

主函数的添加

cout << "输入需要增加子女的成员姓名" << endl;

getline(cin, namestring);

if (root.find(head, stacknumber, namestring, current) == 1) //先看看是否有该结点存在

{

cout << "该点的子女如下" << endl;

current->showson(); //展示结点的子女

cout << "请输入要加入的子女个数" << endl;

int childnumber = 0; cin >> childnumber; cin.ignore(); //添加子女+清除cin中的回车

cout << "请输入要加入的子女姓名" << endl;

if (current->add(current, childnumber) == 1)

{

cout << "该点的子女修改后如下" << endl;

current->showson();

}

}

else

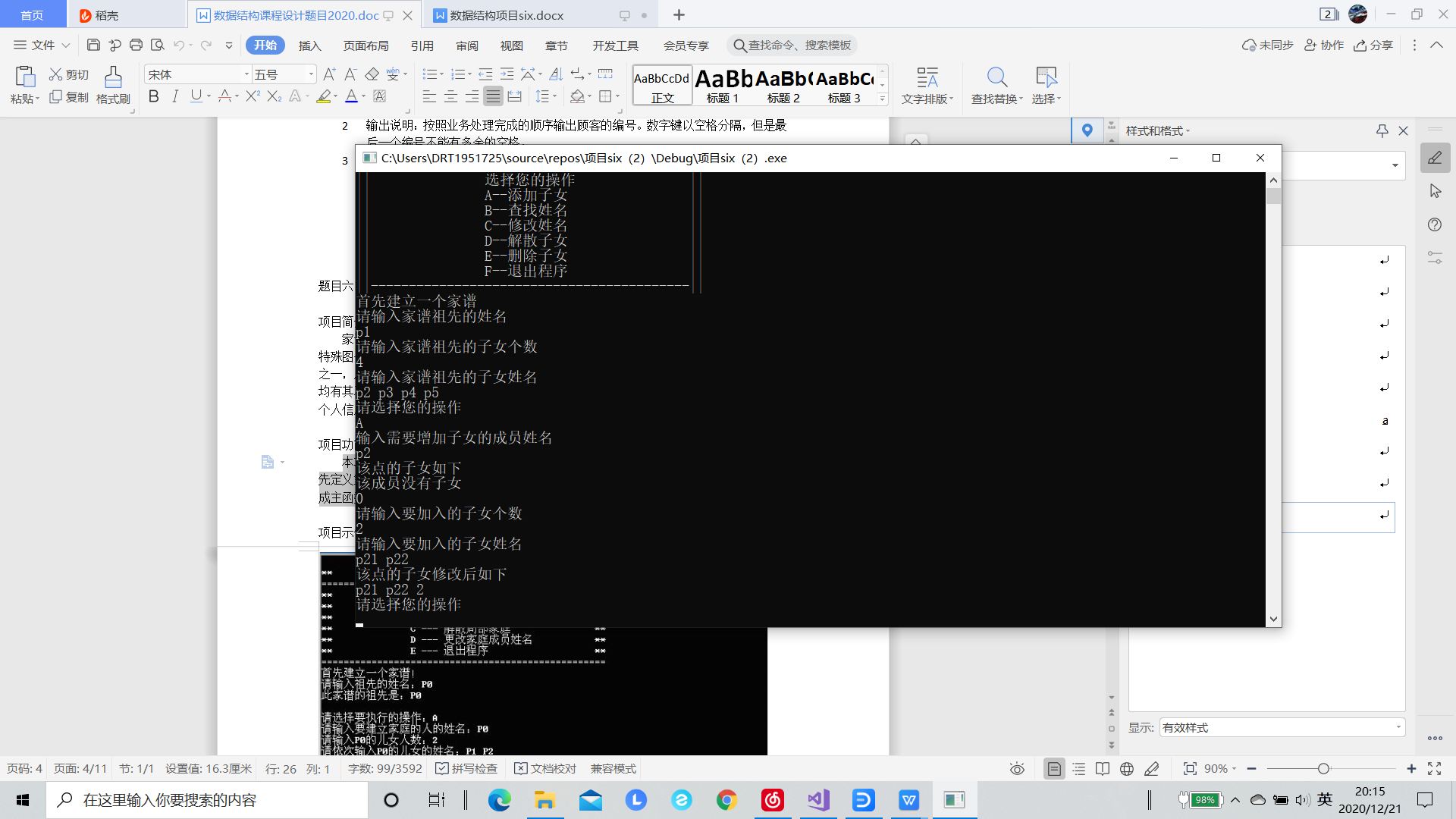
{

cout << "家谱中该成员不存在" << endl;

}

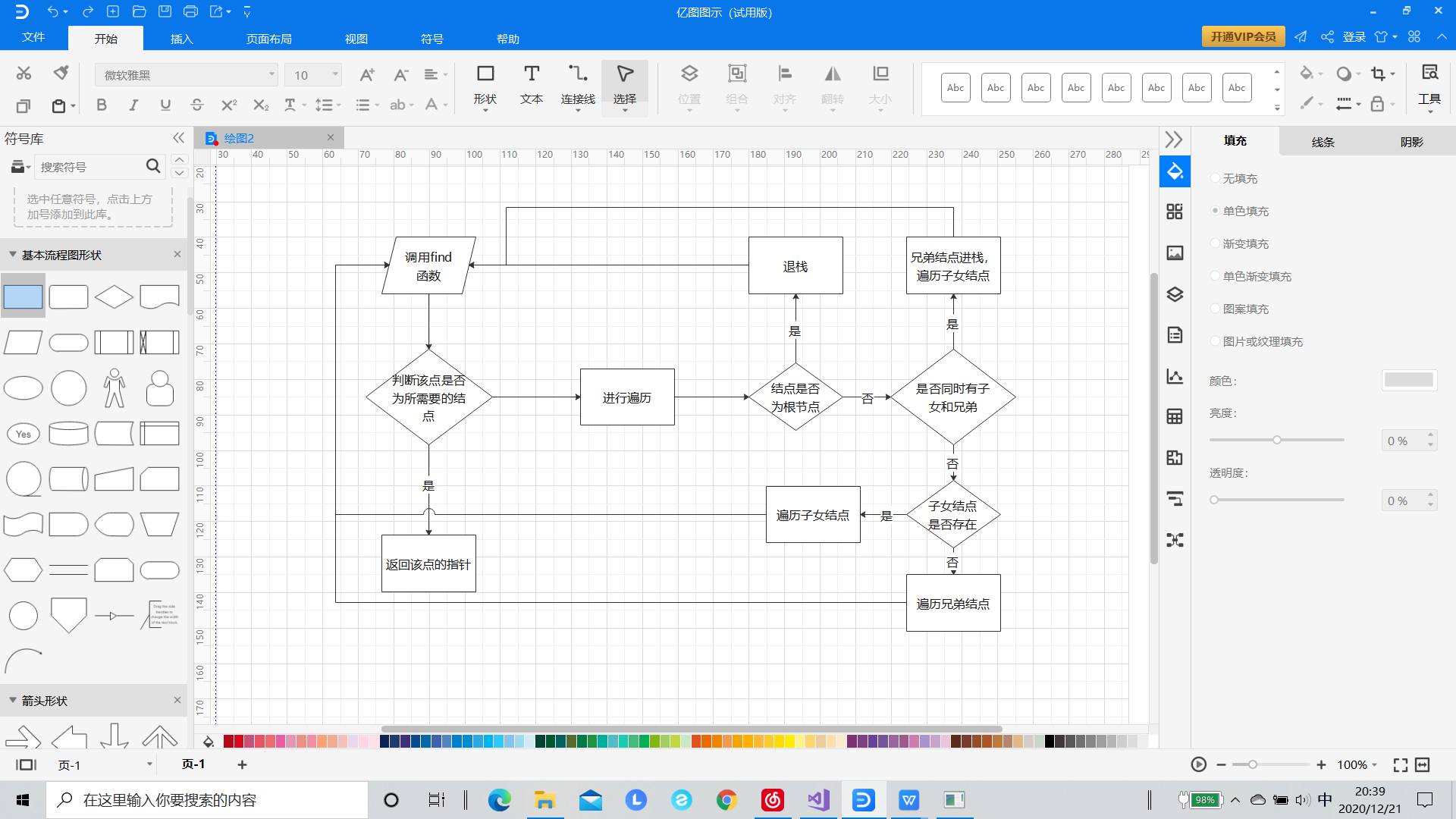
### 

### 3.2.3 插入功能截屏示例



## 3.3 查找功能的实现

### 3.3.1 查找功能流程图



### 3.3.2 查找功能核心代码

类函数find：

int family::find(family \* &head, int &a, string b, family\* &p1) //通过成员的名字来寻找

{

if (name == b) { p1 = this; return 1; } //返回该成员的指针

else

{

if (son == NULL && brother == NULL)

{

if (a != 0)

{

head = head->link; //退栈中的点进行遍历

if (head->find(head, a, b, p1) == 1) { return 1; };

}

else { return 0; }

}

if (son != NULL && brother == NULL)

{

if (son->find(head, a, b, p1) == 1) { return 1; } //先左遍历

}

if (son == NULL && brother != NULL)

{

if (brother->find(head, a, b, p1) == 1) { return 1; } //后右遍历

}

if (son != NULL && brother != NULL) //将该点的兄弟结点放入栈中再进行左遍历

{

a++;

brother->link = head->link;//栈添加

head->link = brother;

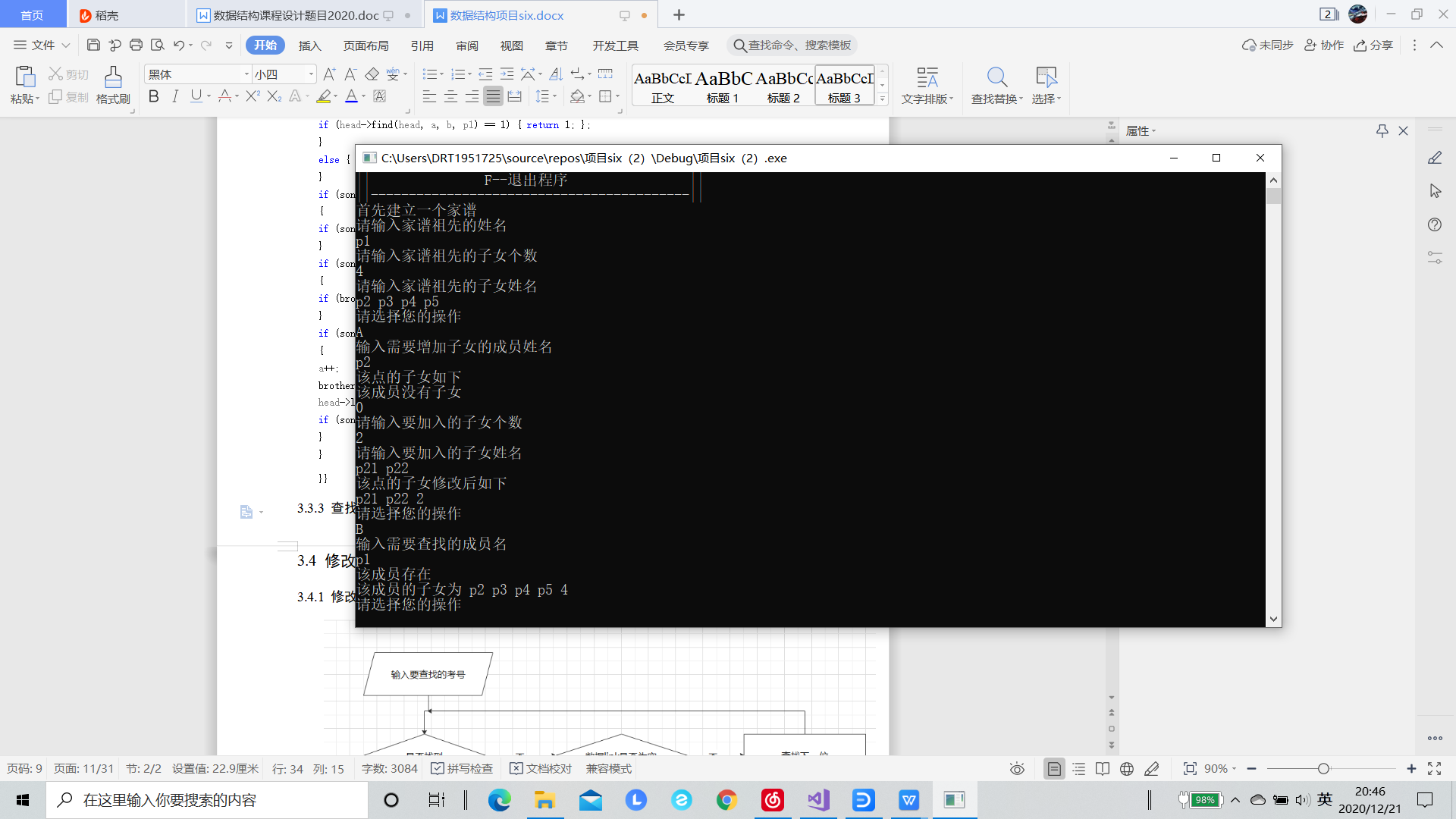
if (son->find(head, a, b, p1) == 1) { return 1; }

}

}

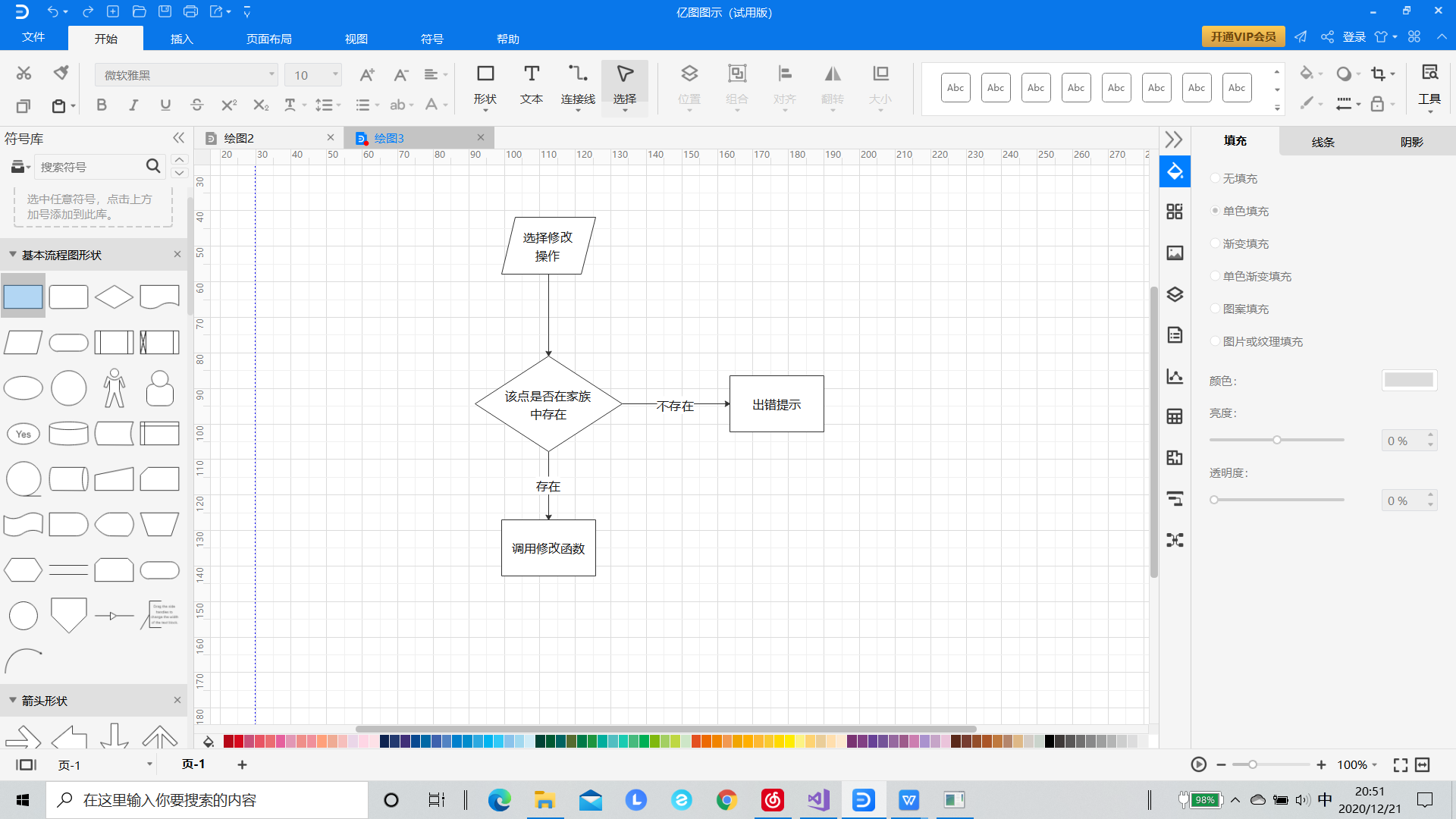
### }}

### 3.3.3 查找功能截屏示例



## 3.4 修改功能的实现

### 3.4.1 修改功能流程图



### 3.4.2 修改功能核心代码

类函数changename

void family::changename(string b)

{

name = b;

}

bool list::seek(int n)

{

主函数中修改片段：

case 'C':

{

cout << "输入需要修改名字的成员名" << endl;

getline(cin, namestring);

if (root.find(head, stacknumber, namestring, current) == 1)

{

cout << "输入修改后的名字" << endl;

getline(cin, namestring);

current->changename(namestring);

}

else

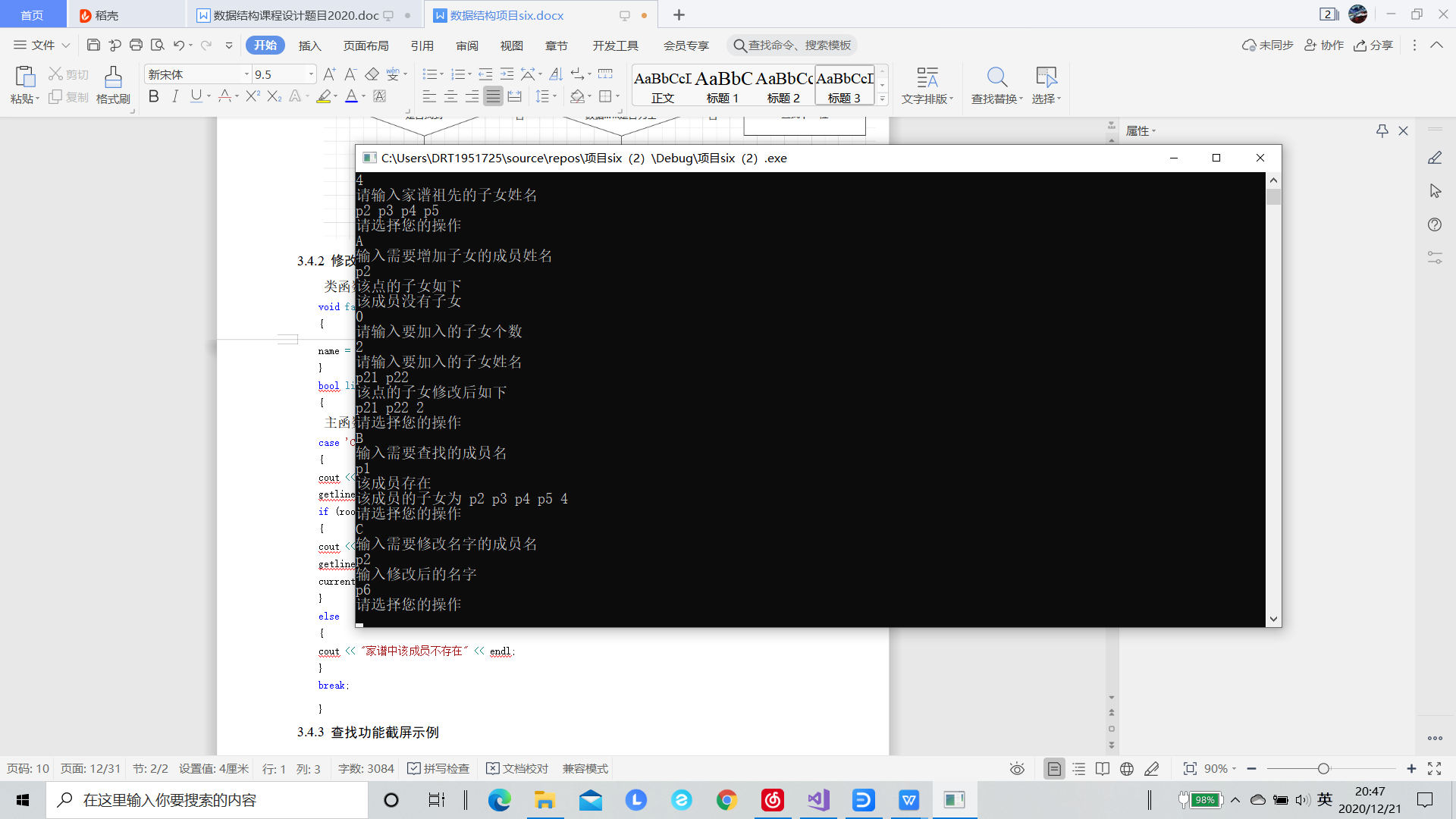
{

cout << "家谱中该成员不存在" << endl;

}

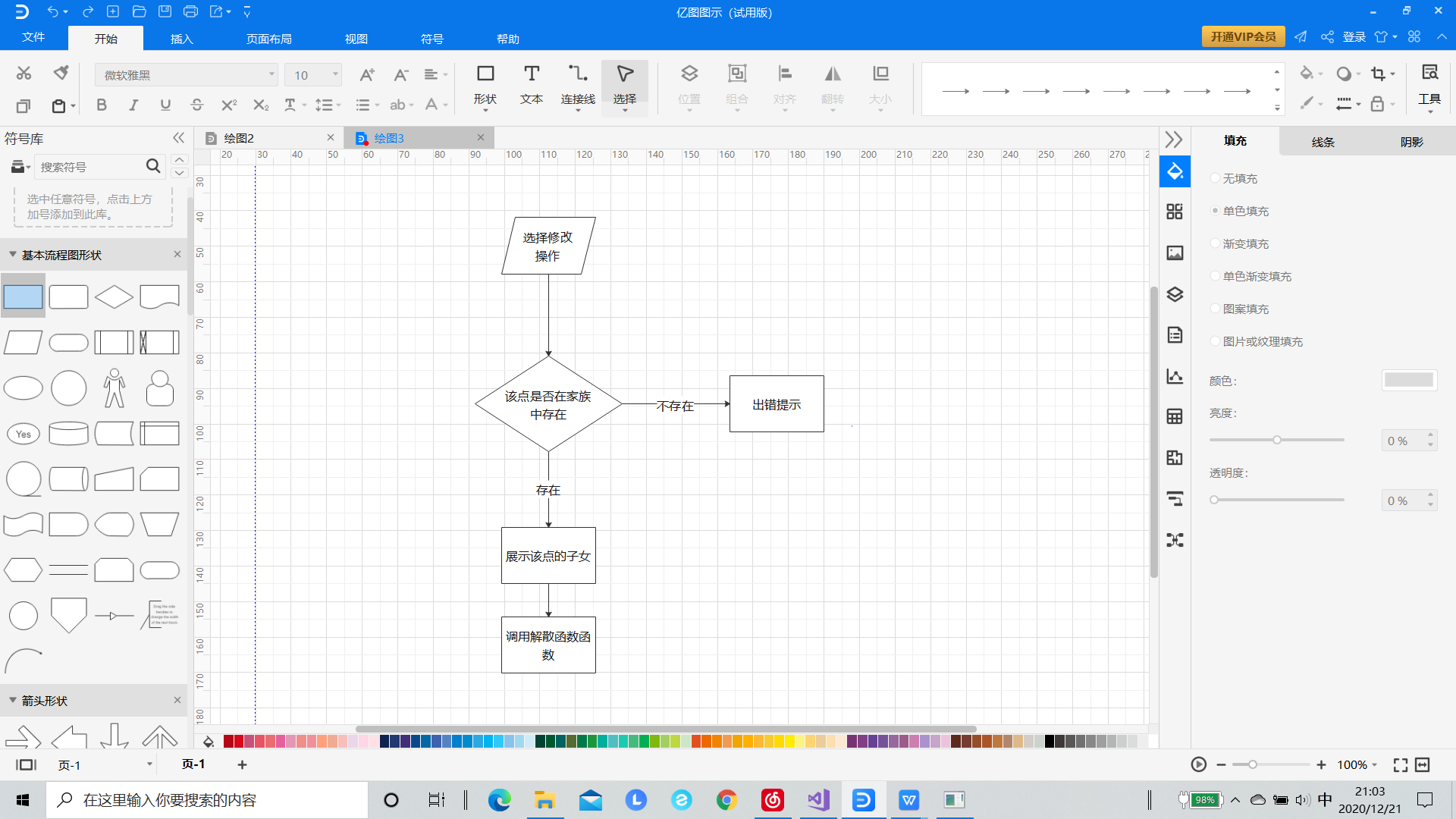
break;

### } 3.4.3 查找功能截屏示例



## 3.5 解散子女功能的实现

### 3.5.1 解散子女功能流程图



### 3.5.2 解散子女功能核心代码

类函数family

int family::refamily()

{

cout << "该成员的子女如下" << endl;

showson();

son = NULL; sonnumber = 0;

cout << "现成员子女如下" << endl;

showson();

return 0;

}

主函数中的解散子女片段

case 'D':

{

cout << "输入需要解散子女的成员名" << endl;

getline(cin, namestring);

if (root.find(head, stacknumber, namestring, current) == 1)

{

current->refamily();

}

else

{

cout << "家谱中该成员不存在" << endl;

}

break;

}

### 3.5.3 解散功能截屏示例



## 3.6 删除功能的实现

### 3.6.1 删除功能流程图

### 

### 3.6.2 删除功能核心代码

类函数中的remove函数

int family::remove(string namestring)

{

if (sonnumber >= 1)

{

family \* p1, \*p2;

p1 = son;

p2 = son;

for (int n = 0; n < sonnumber; n++)

{

if (p1->name == namestring) { break; }

p2 = p1;

p1 = p1->brother;

}

if (p1 == NULL) { return 0; } //p1如果存在为该点删除子女的位置，p2为该子女的上一个结点

else

{ //四种不同的删除情况

if (p1 == son && p1->brother == NULL) { son = NULL; sonnumber--; return 1; }

if (p1 == son && p1->brother != NULL) { son = son->brother; sonnumber--; return 1; }

if (p1 != son && p1->brother == NULL) { p2->brother = NULL; sonnumber--; return 1; }

if (p1 != son && p1->brother != NULL) { p2->brother = p1->brother; sonnumber--; delete p1; return 1; }

}

}

else

{

return 0;

}

}

主函数中的删除功能的片段

case 'E':

{

cout << "输入需要删去子女的双亲名" << endl;

getline(cin, namestring);

if (root.find(head, stacknumber, namestring, current) == 1)

{

cout << "该成员的子女有如下" << endl;

current->showson();

cout << "输入需要删去的子女名字" << endl;

getline(cin, namestring);

if (current->remove(namestring) == 1)

{

current->showson();

}

else

{

cout << "该成员中无该子女" << endl;

}

}

else

{

cout << "家谱中该成员不存在" << endl;

}

break;

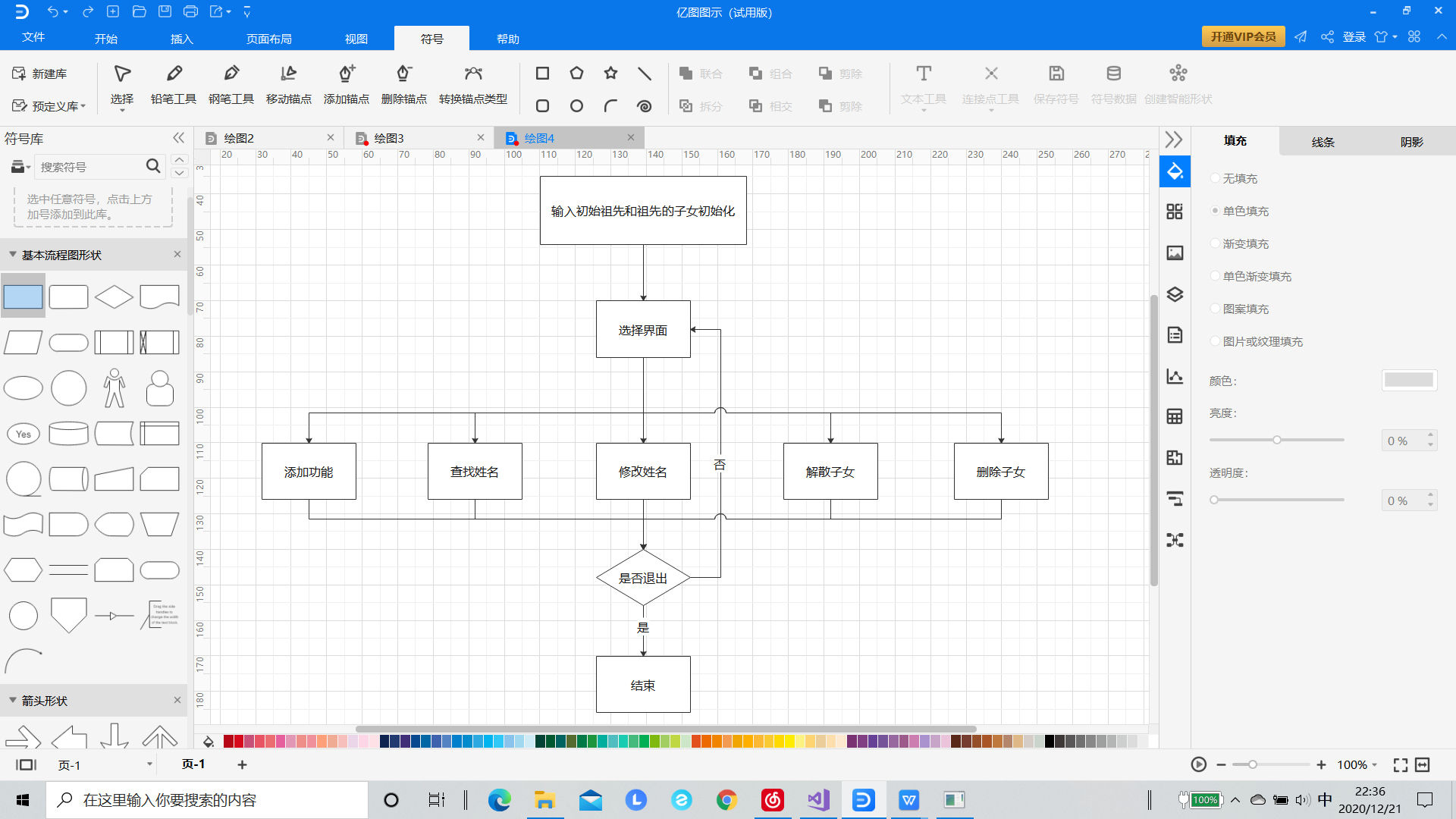
### }

### 3.6.3 排序功能截屏示例

# 

# 3.7总体系统实现

### 3.7.1 总体系统流程图



### 3.7.2总体系统核心代码

for (;;)

{

cout << "请选择您的操作" << endl;

cin >> kind;

cin.ignore();

switch (kind)

{

case 'A':

{

cout << "输入需要增加子女的成员姓名" << endl;

getline(cin, namestring);

if (root.find(head, stacknumber, namestring, current) == 1)

{

cout << "该点的子女如下" << endl;

current->showson();

cout << "请输入要加入的子女个数" << endl;

int childnumber = 0; cin >> childnumber; cin.ignore();

cout << "请输入要加入的子女姓名" << endl;

if (current->add(current, childnumber) == 1)

{

cout << "该点的子女修改后如下" << endl;

current->showson();

}

}

else

{

cout << "家谱中该成员不存在" << endl;

}

break;

}

case 'B':

{

cout << "输入需要查找的成员名" << endl;

getline(cin, namestring);

if (root.find(head, stacknumber, namestring, current) == 1)

{

cout << "该成员存在" << endl;

cout << "该成员的子女为" << " ";

current->showson();

}

else

{

cout << "家谱中该成员不存在" << endl;

}

break;

}

case 'C':

{

cout << "输入需要修改名字的成员名" << endl;

getline(cin, namestring);

if (root.find(head, stacknumber, namestring, current) == 1)

{

cout << "输入修改后的名字" << endl;

getline(cin, namestring);

current->changename(namestring);

}

else

{

cout << "家谱中该成员不存在" << endl;

}

break;

}

case 'D':

{

cout << "输入需要解散子女的成员名" << endl;

getline(cin, namestring);

if (root.find(head, stacknumber, namestring, current) == 1)

{

current->refamily();

}

else

{

cout << "家谱中该成员不存在" << endl;

}

break;

}

case 'E':

{

cout << "输入需要删去子女的双亲名" << endl;

getline(cin, namestring);

if (root.find(head, stacknumber, namestring, current) == 1)

{

cout << "该成员的子女有如下" << endl;

current->showson();

cout << "输入需要删去的子女名字" << endl;

getline(cin, namestring);

if (current->remove(namestring) == 1)

{

cout<<"该点的子女修改后如下"<<endl;

current->showson();

}

else

{

cout << "该成员中无该子女" << endl;

}

}

else

{

cout << "家谱中该成员不存在" << endl;

}

break;

}

case 'F': {end = 1; break; }

default: {cout << "无效字符请重新输入" << endl; break; }

}

if (end == 1) { break; }

}

# 4 测试

## 4.1 功能测试

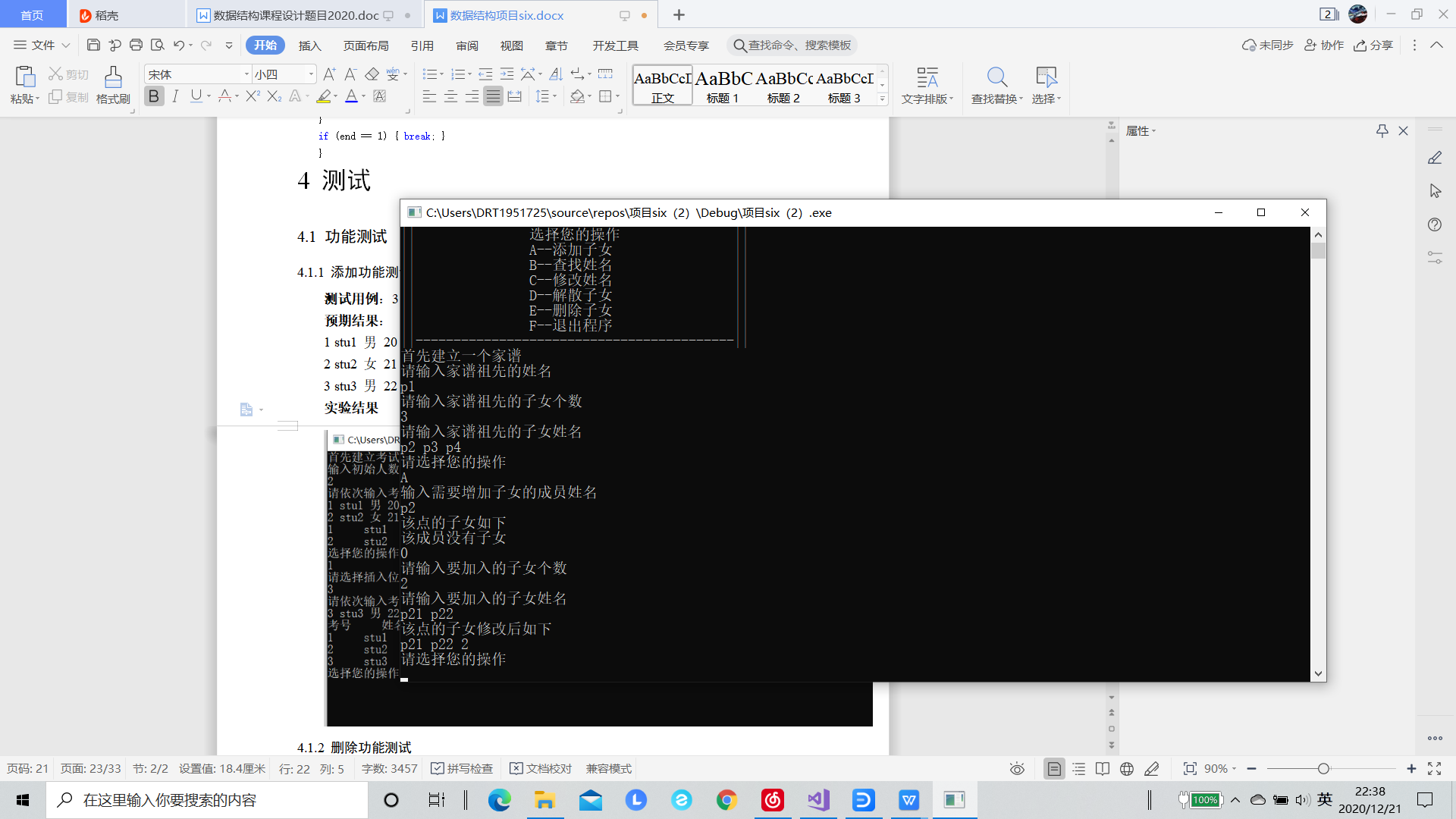
### 4.1.1 添加功能测试

**测试用例**：p2 2 p21 p22

**预期结果**：

**p21 p22**

**实验结果**



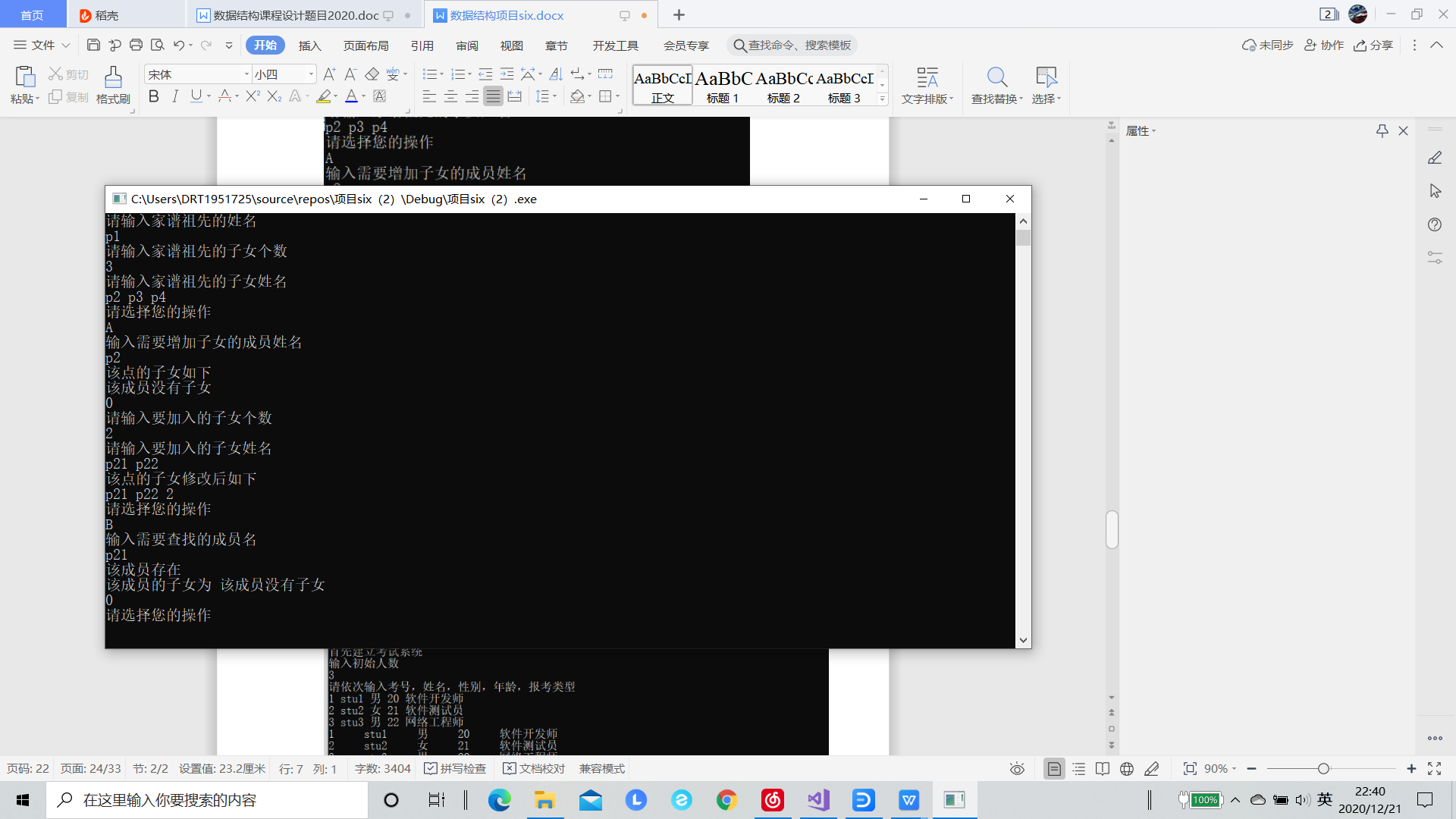
### 4.1.2 查找功能测试

**测试用例：p21**

**预期结果：**

**p21 存在**

**实验结果：**



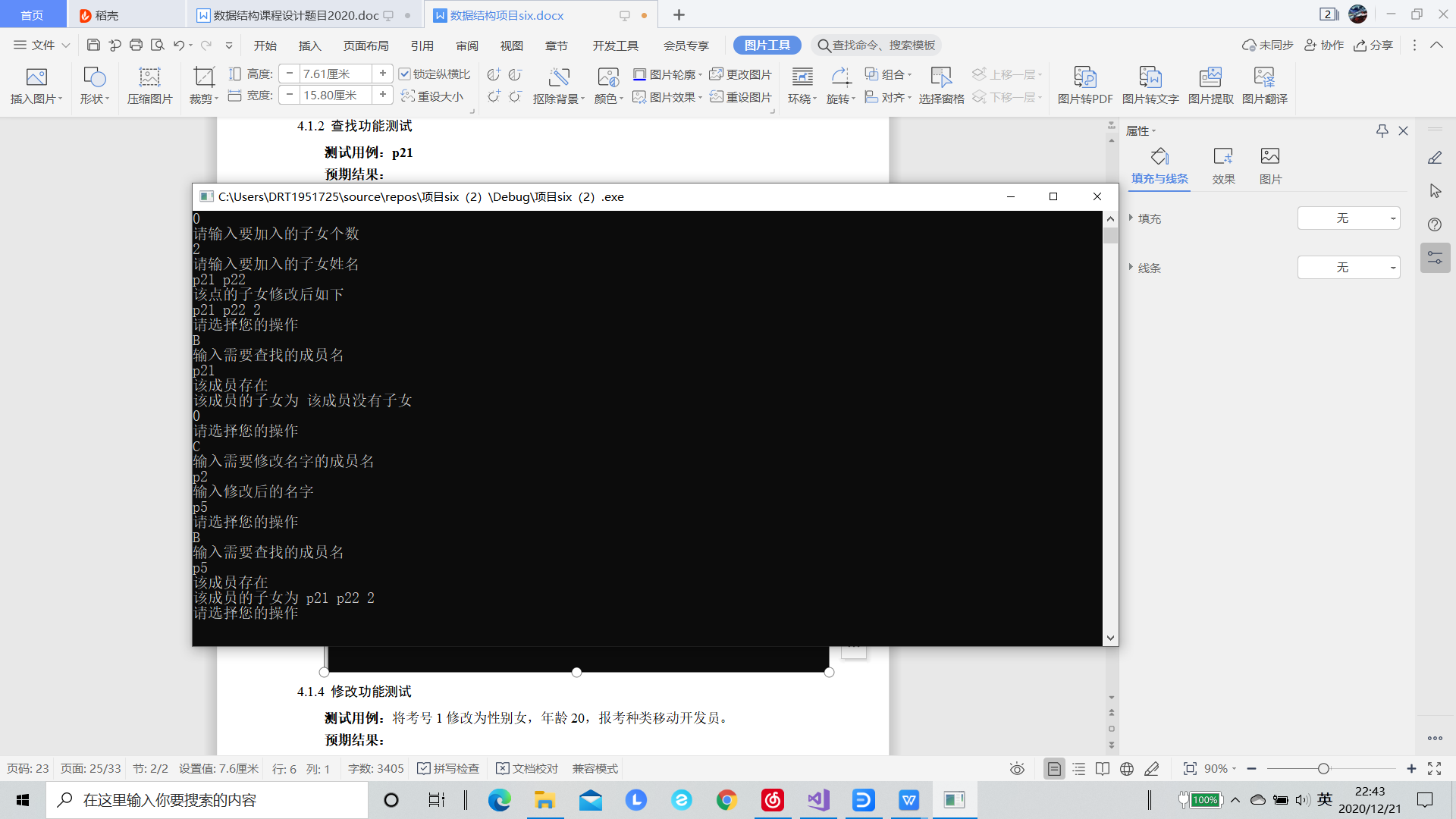
### 4.1.3 修改功能测试

**测试用例：p2 p5 在调用查找函数找p5**

**预期结果：**

**p5存在 子女为p21 p22**

**实验结果：**



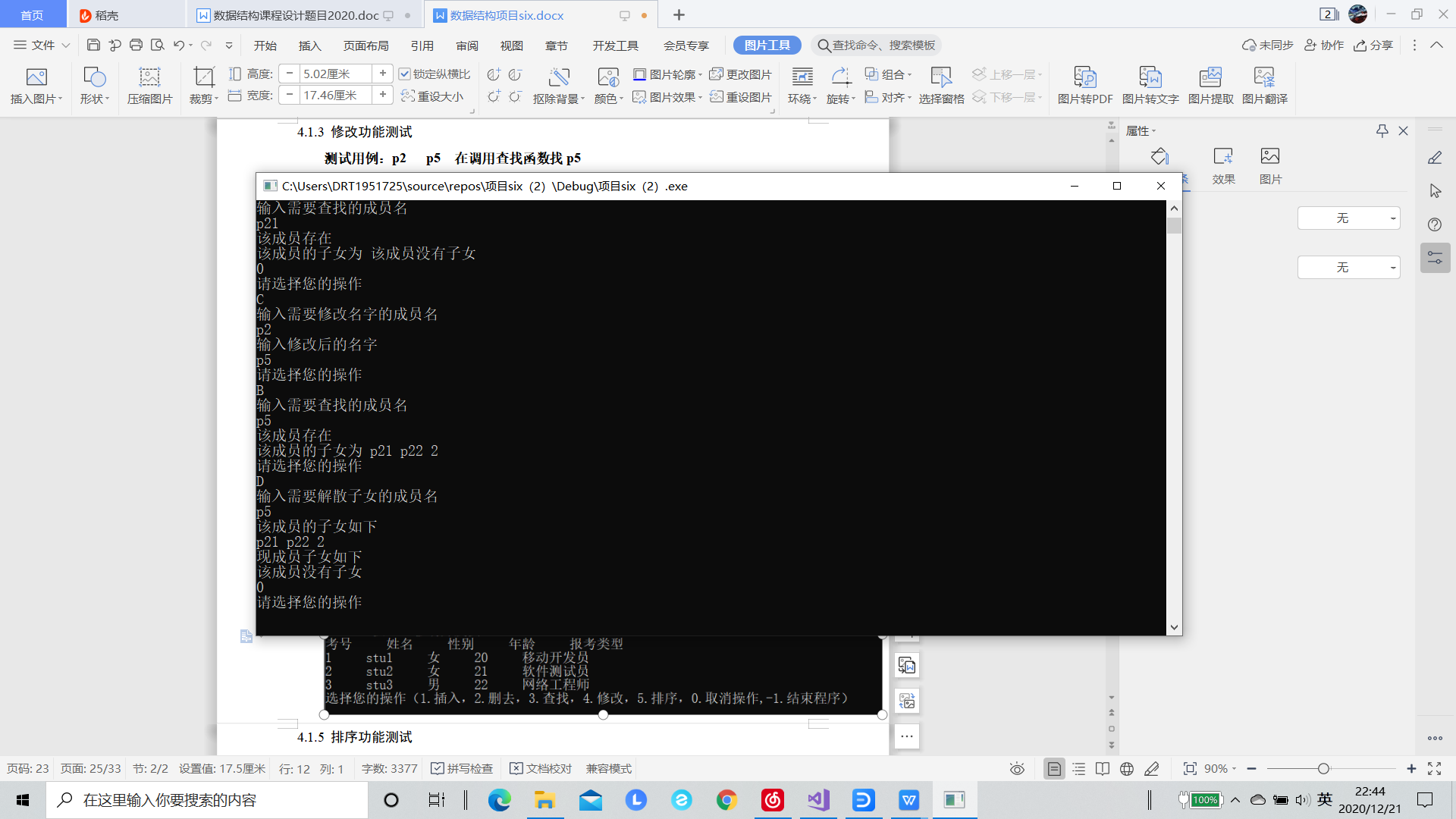
### 4.1.4 解散功能测试

**测试用例：p5**

**预期结果：**

**p5 没有子女**

**实验结果：**



### 4.1.5 删除功能测试

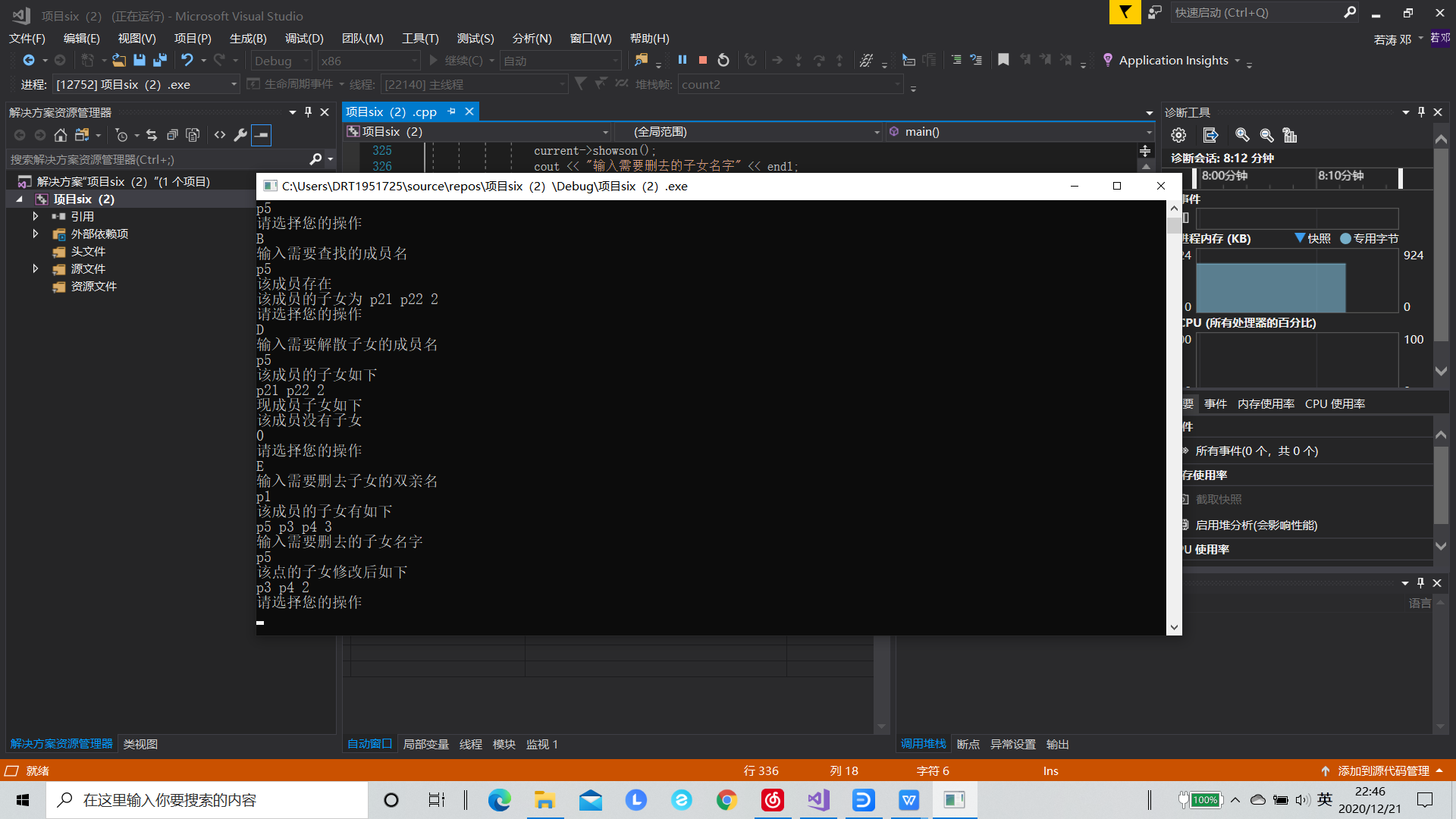
**测试用例：**

**p1 p5**

**预期结果：**

**p1 的子女为p3 p4**

**实验结果：**



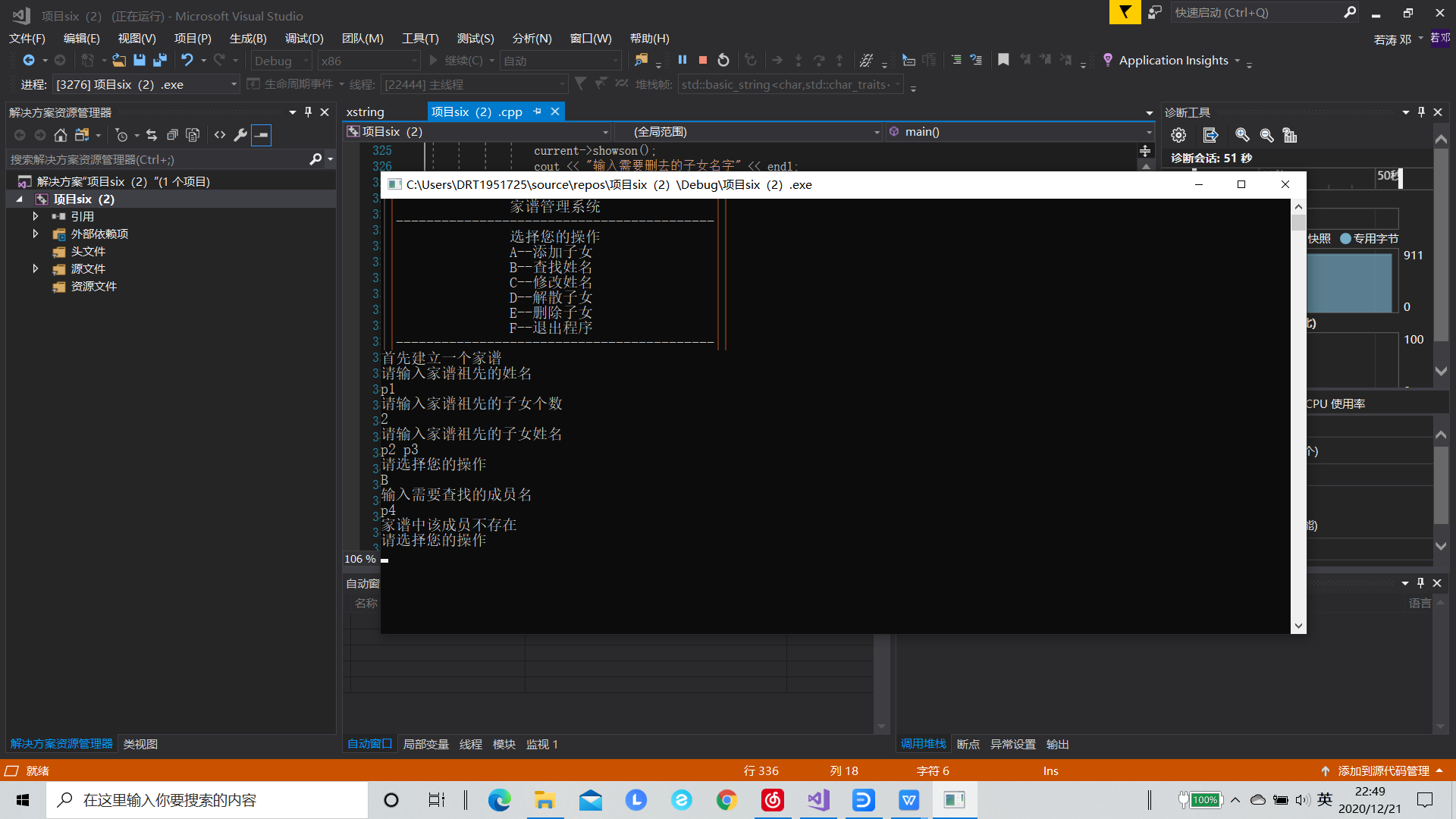
## 4.2 边界测试

### 4.2.1 输入的结点不存在

**测试用例：p4**

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**



### 4.2.2 输入的操作符非法

**测试用例：G**

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

