项目说明文档

数据结构课程设计

——家谱管理系统

作 者 姓 名： 罗佳瑞

学 号： 1952528

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc11394)

[1.1 背景分析 1](#_Toc29967)

[1.2 功能分析 1](#_Toc28216)

[2 设计 2](#_Toc31740)

[2.1 数据结构设计 2](#_Toc20968)

[2.2 类结构设计 2](#_Toc20914)

[2.3 成员与操作设计 2](#_Toc4122)

[2.4 系统设计 3](#_Toc5182)

[3 实现 4](#_Toc8108)

[3.1 创建功能的实现 4](#_Toc29699)

[3.1.1 创建功能流程图 4](#_Toc6279)

[3.1.2 创建功能核心代码 4](#_Toc30225)

[3.1.3 创建功能截屏示例 5](#_Toc11549)

[3.2 查找功能的实现 5](#_Toc17965)

[3.2.1 查找功能流程图 5](#_Toc30932)

[3.2.2 查找功能核心代码 6](#_Toc32270)

[3.3 完善家庭功能的实现 7](#_Toc4804)

[3.3.1 完善家庭功能流程图 7](#_Toc13218)

[3.3.2 完善家庭功能核心代码 7](#_Toc3960)

[3.3.3 完善家庭功能截屏示例 9](#_Toc28245)

[3.4 添加家庭成员功能的实现 10](#_Toc12684)

[3.4.1 添加家庭成员功能流程图 10](#_Toc25519)

[3.4.2 添加家庭成员功能核心代码 10](#_Toc18524)

[3.4.3 添加家庭成员功能截图示例 11](#_Toc7623)

[3.5 解散局部家庭功能的实现 12](#_Toc29991)

[3.5.1 解散局部家庭功能流程图 12](#_Toc597)

[3.5.2 解散局部家庭功能核心代码 12](#_Toc4690)

[3.5.3 解散局部家庭功能截屏示例 14](#_Toc10101)

[3.6 更改姓名功能的实现 15](#_Toc5548)

[3.6.1 更改姓名功能流程图 15](#_Toc2900)

[3.6.2 更改姓名功能核心代码 17](#_Toc19938)

[3.5.3 更改姓名功能截屏示例 17](#_Toc11863)

[3.6 总体系统的实现 18](#_Toc7301)

[3.6.1 总体系统流程图 18](#_Toc19336)

[3.6.2 总体系统核心代码 18](#_Toc6477)

[3.6.3 总体系统截屏示例 19](#_Toc2370)

[4 测试 20](#_Toc12424)

[4.1 功能测试 20](#_Toc3192)

[4.2 边界测试 20](#_Toc7628)

[4.2.1 完善家庭的成员已有家庭 20](#_Toc27412)

[4.2.2 解散局部家庭的成员无家庭 20](#_Toc28004)

[4.2.3 解散根节点家庭 20](#_Toc7725)

[4.3 出错测试 21](#_Toc16921)

[4.3.1操作码错误 21](#_Toc23346)

[4.3.2 要完善家庭的成员不存在 21](#_Toc19007)

[4.3.3 要添加成员的成员不存在 21](#_Toc13120)

[4.3.4 要解散家庭的成员不存在 21](#_Toc26160)

[4.3.5 要修改姓名的成员不存在 22](#_Toc4660)

# 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。

随着人员数量和世代数量的日益庞大，家谱数据的管理显得极为复杂，传统的手工管理工作量大且容易出错，而使用计算机在管理家谱报名系统方面，具有手工管理所无法比拟的优势。这些优点能够极大地便利家谱的完善与传承，也是中华文化遗产走向信息化、科学化、国际化的重要条件之一。因此，开发一套家谱管理系统具有十分重要的意义。

## 1.2 功能分析

本项目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，修改成员信息，删除家族成员的功能。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，该系统要求大量的插入、删除、修改操作，且各节点之间有很强的次序与关联关系，而链表进行插入、删除等操作十分简便，因此考虑使用链表构成树的数据结构。同时，为了实现简易，除了储存信息外另增加了父节点指针、首子女指针、兄弟指针，便于家谱树的增加或者删除过程中的遍历，使得程序更为简洁。

## 2.2 类结构设计

家谱树结点类（Person）与家谱树类（genealogyt）两个类之间的耦合关系采用嵌套定义，使得家谱树类（genealogyt）可以访问树结点。同时，将家谱树类的一些功能操作设为私有，由唯一公有函数（run）调用，使结点访问更为安全。

## 2.3 成员与操作设计

**家谱树结点类（Person）**

**公有成员：**

string name;//家庭成员姓名

Person\* parent;//父节点指针

Person\* firstChild;//首子女结点指针

Person\* nextSibling;//下一兄弟结点指针

Person\* lastSibling;//上一兄弟结点指针

**公有操作：**

Person::Person();//默认构造函数

Person::Person(string n, Person\* P = NULL, Person\* F = NULL, Person\* N = NULL, Person\* L = NULL);//构造函数

**家谱类（genealogy）**

**私有成员：**

Person\* ancestor;

**私有操作：**

void genealogy::printChild(Person\* member);//打印第一代子女

void genealogy::construct();//建立家谱

Person\* genealogy::find(Person\* current, string n);//查找家庭成员

void genealogy::constructFamily();//建立家庭

void genealogy::addChild();//添加家庭成员

void genealogy::deleteMember(Person\* member);//删除操作

void genealogy::deleteFamily();//解散局部家庭

void genealogy::changeName();//改变家庭成员姓名

**公有操作：**

void genealogy::run();//运行函数

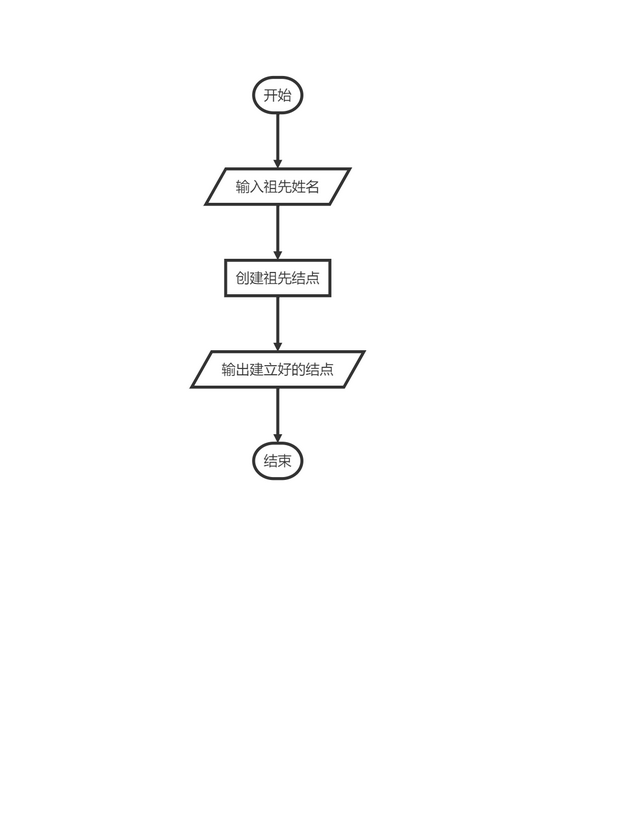
## 2.4 系统设计

系统首先提示用户输入，完成对家谱树T的创建和输入数据工作，然后根据用户所输入的操作码（op）执行家谱树T对应的成员函数。

# 3 实现

## 3.1 创建功能的实现

### 3.1.1 创建功能流程图



### 3.1.2 创建功能核心代码

void genealogy::construct()//建立家谱

{

ancestor = new Person;//为祖先结点申请地址

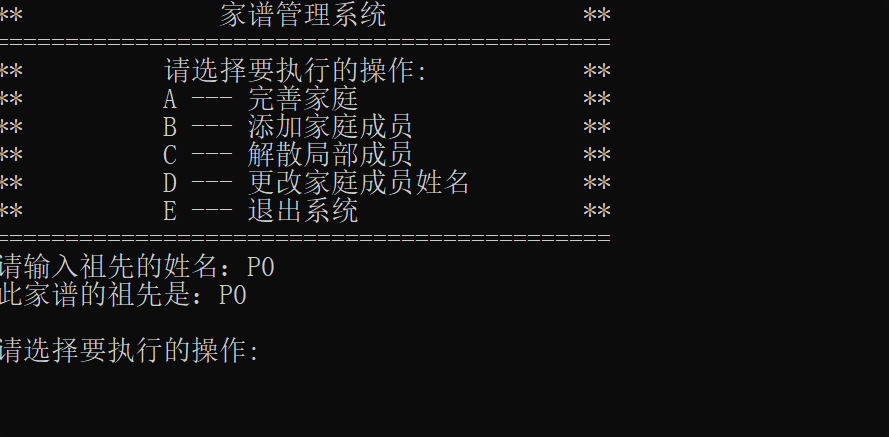
cout << "请输入祖先的姓名：";

cin >> ancestor->name;

cout << "此家谱的祖先是：" << ancestor->name << endl << endl;

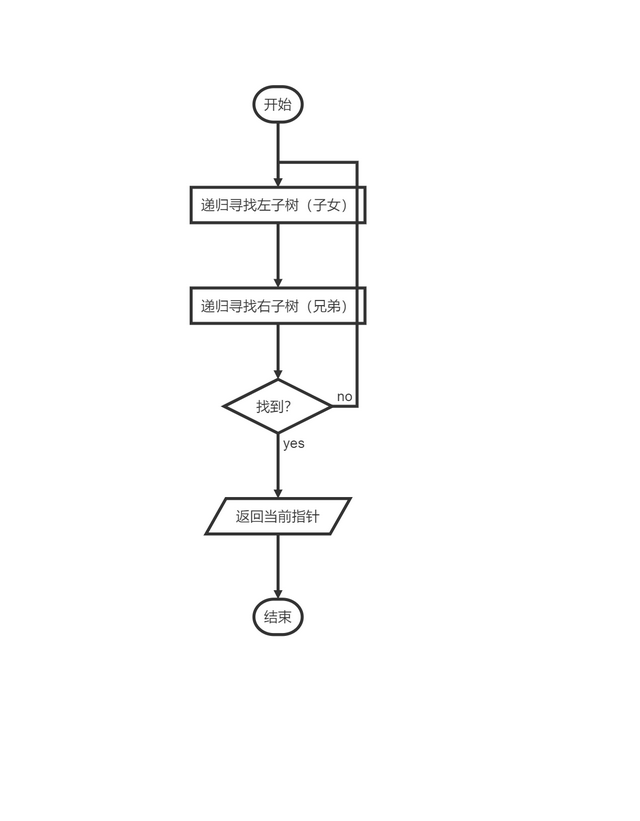
}

### 3.1.3 创建功能截屏示例



## 3.2 查找功能的实现

### 3.2.1 查找功能流程图



### 3.2.2 查找功能核心代码

Person\* genealogy::find(Person\* current, string n)//查找家庭成员

{

if (current == NULL)return NULL;//递归到底未找到

if (current->name == n)return current;//找到返回当前指针

Person\* temp;

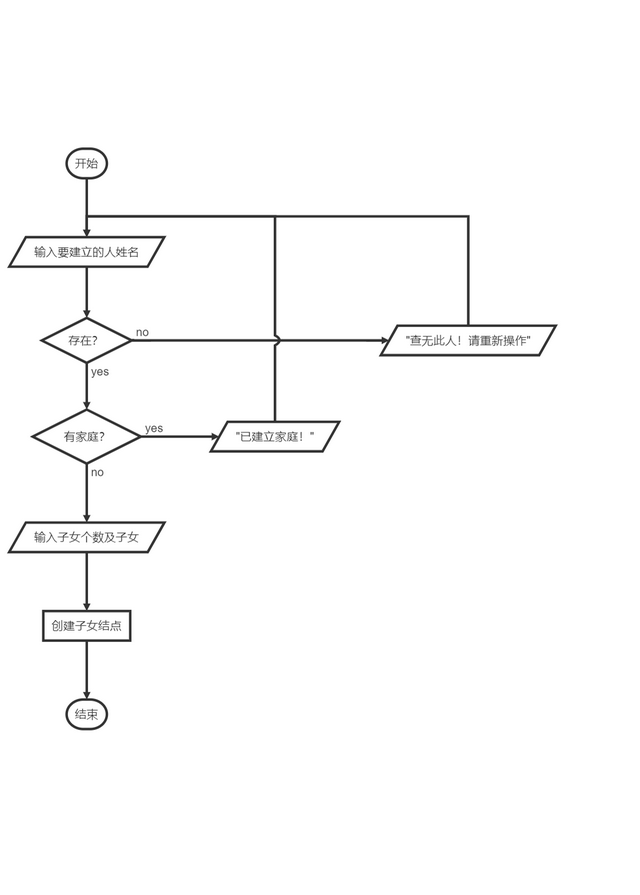
if ((temp = find(current->firstChild, n)) != NULL)return temp;//递归寻找左子树（子女）

else return find(current->nextSibling, n);//递归寻找右子树（兄弟）

}

## 3.3 完善家庭功能的实现

### 3.3.1 完善家庭功能流程图



### 3.3.2 完善家庭功能核心代码

void genealogy::constructFamily()//建立家庭

{

string s;

cout << "请输入要建立家庭人的姓名：";

cin >> s;

Person\* member = find(ancestor, s);

if (member == NULL)

{

cout << "查无此人！请重新操作" << endl;

return;

}

if (member->firstChild != NULL)

{

cout << "已建立家庭！" << endl;

return;

}

int n;

cout << "请输入" << member->name << "的儿女人数：";

cin >> n;

string\* child = new string[n];

cout << "请依次输入" << member->name << "的儿女的姓名：";

Person\* temp = member->firstChild;

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

{

cin >> child[i];

Person\* newChild = new Person(child[i], member);

if (i == 0)

{

member->firstChild = newChild;

temp = member->firstChild;

}

else

{

newChild->lastSibling = temp;

temp->nextSibling = newChild;

temp = temp->nextSibling;

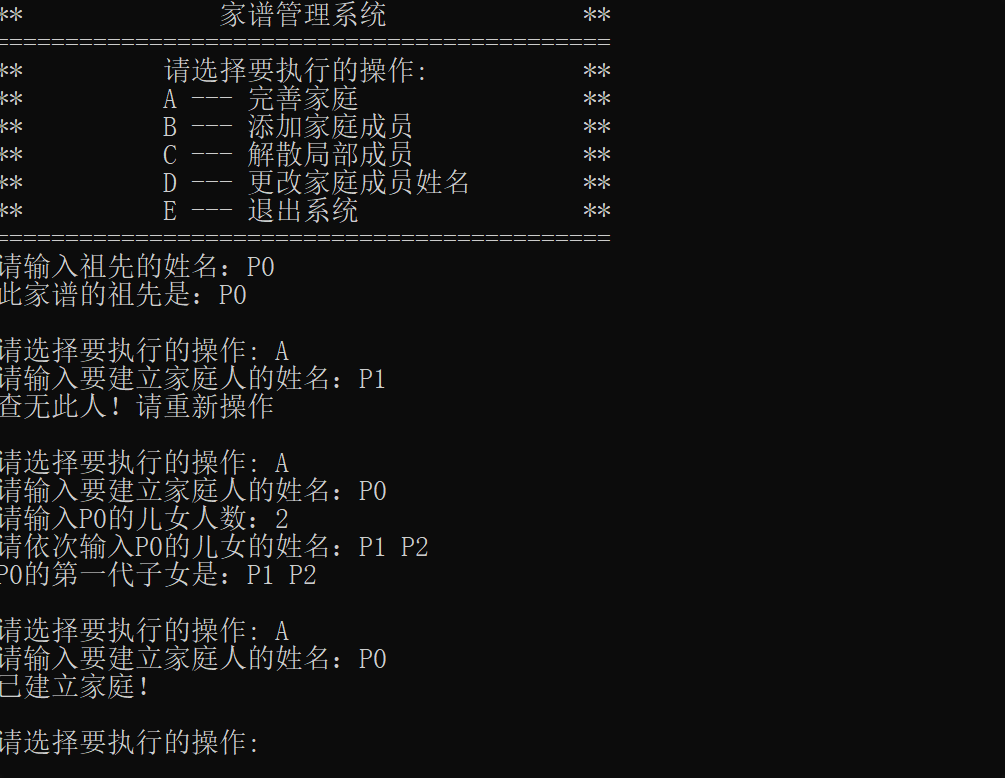
}

}

printChild(member);

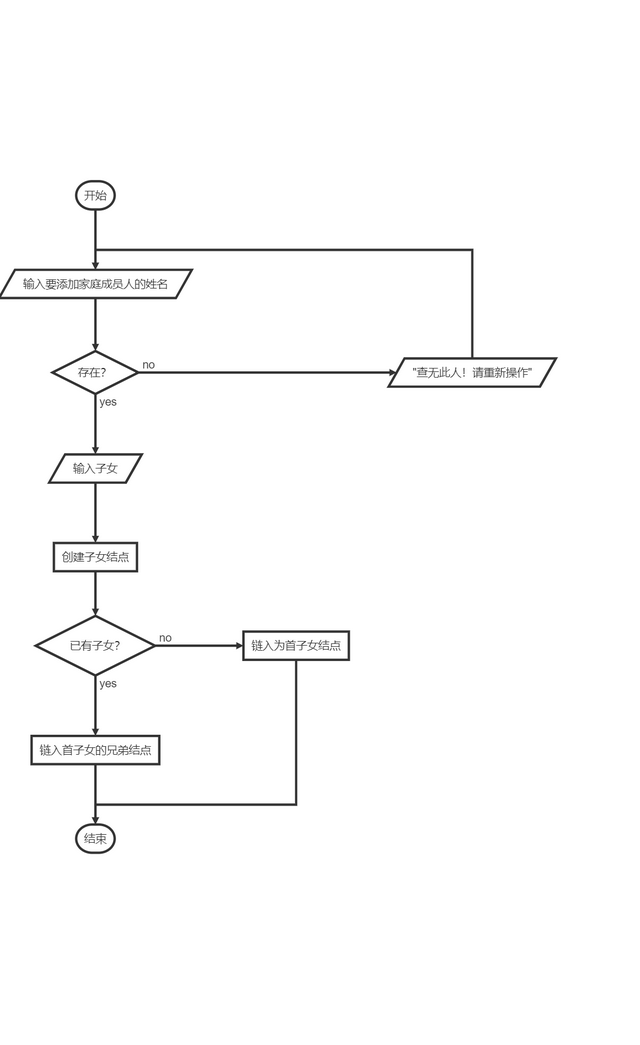
}

### 3.3.3 完善家庭功能截屏示例



## 3.4 添加家庭成员功能的实现

### 3.4.1 添加家庭成员功能流程图



### 3.4.2 添加家庭成员功能核心代码

void genealogy::addChild()//添加家庭成员

{

string p, c;

cout << "请输入要添加儿女的人的姓名：";

cin >> p;

Person\* member = find(ancestor, p);

if (member == NULL)

{

cout << "查无此人！请重新操作" << endl;

return;

}

cout << "请输入" << p << "新添加的儿女的姓名：";

cin >> c;

Person\* newChild = new Person(c, member);

Person\* temp = member->firstChild;

if (member->firstChild == NULL)

member->firstChild = newChild;

else

{

while (temp->nextSibling != NULL)

temp = temp->nextSibling;

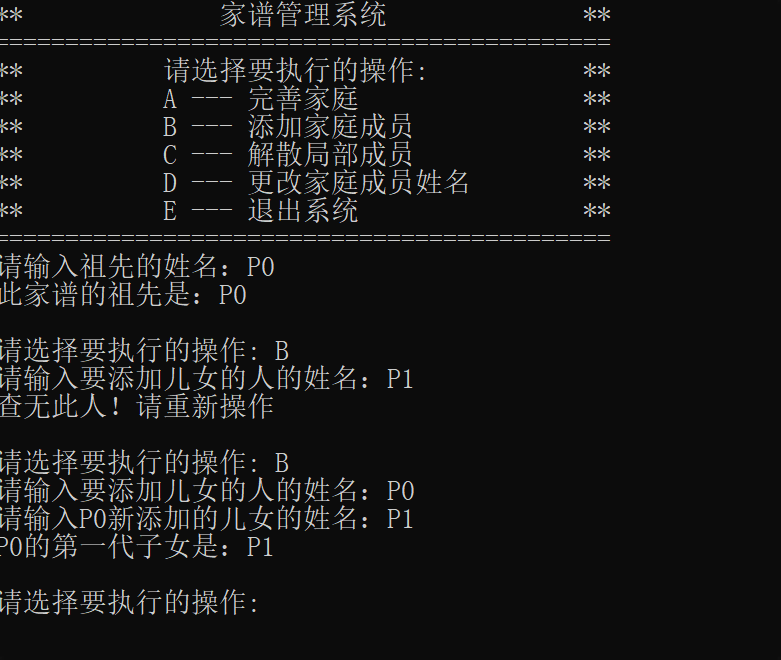
newChild->lastSibling = temp;

temp->nextSibling = newChild;

}

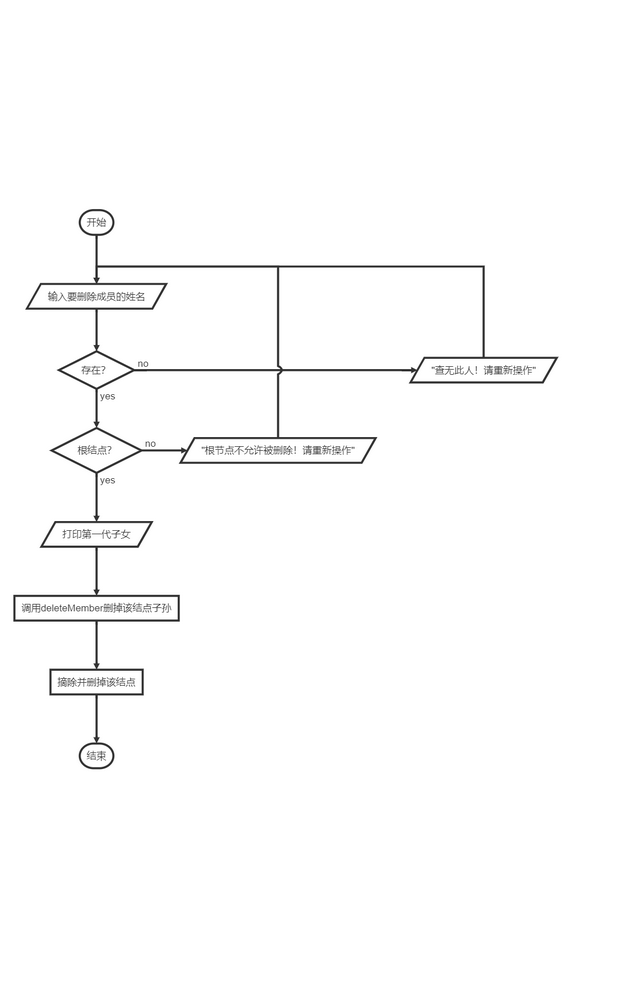
printChild(member);}

### 3.4.3 添加家庭成员功能截图示例



## 3.5 解散局部家庭功能的实现

### 3.5.1 解散局部家庭功能流程图



### 3.5.2 解散局部家庭功能核心代码

void genealogy::deleteMember(Person\* member)//删除操作

{

Person\* temp = member;

while (temp->nextSibling != NULL)//先循兄弟链到底

temp = temp->nextSibling;

if (temp->lastSibling != NULL) {//向回把后一个兄弟截下来,都为没有兄弟节点的树

temp = temp->lastSibling;

(temp->nextSibling)->lastSibling = NULL;

deleteMember(temp->nextSibling);

}

if (temp->firstChild != NULL)//处理截下来的兄弟子链

deleteMember(temp->firstChild);

delete member;

member = NULL;//都处理完释放该结点

return;

}

void genealogy::deleteFamily()//解散局部家庭

{

string s;

cout << "请输入要解散家庭的人的姓名：";

cin >> s;

Person\* member =find(ancestor, s);

if (member == NULL)

{

cout << "查无此人！请重新操作" << endl;

return;

}

if (member == ancestor)

{

cout << "根节点不允许被删除！请重新操作" << endl;

return;

}

printChild(member);

if (member->firstChild != NULL)//删掉该结点的子孙

deleteMember(member->firstChild);

if (member->lastSibling != NULL && member->nextSibling != NULL) {

(member->lastSibling)->nextSibling = member->nextSibling;

(member->nextSibling)->lastSibling = member->lastSibling;

}

else if (member->lastSibling != NULL && member->nextSibling == NULL) {

(member->lastSibling)->nextSibling = NULL;

}

else if (member->lastSibling == NULL && member->nextSibling != NULL) {

(member->nextSibling)->lastSibling = NULL;

(member->parent)->firstChild = member->nextSibling;

}

else if (member->lastSibling == NULL && member->nextSibling == NULL) {

(member->parent)->firstChild = NULL;

}

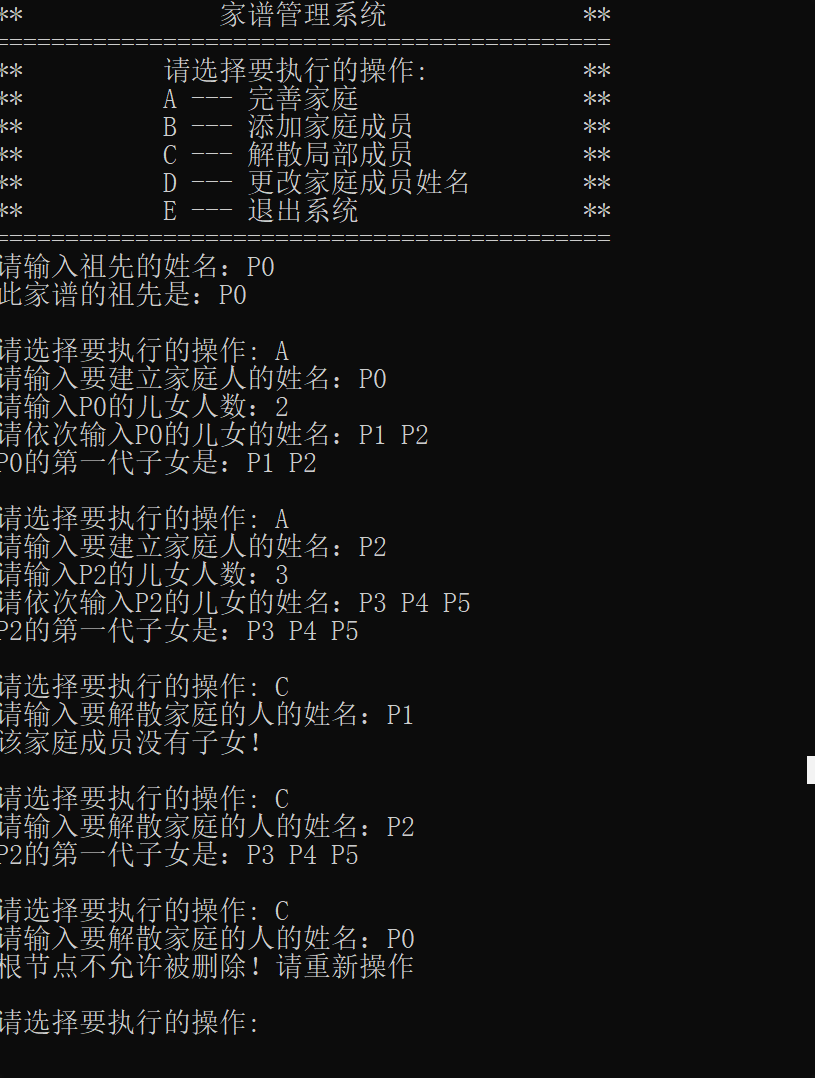
delete member;

member = NULL;//删掉该结点

return;

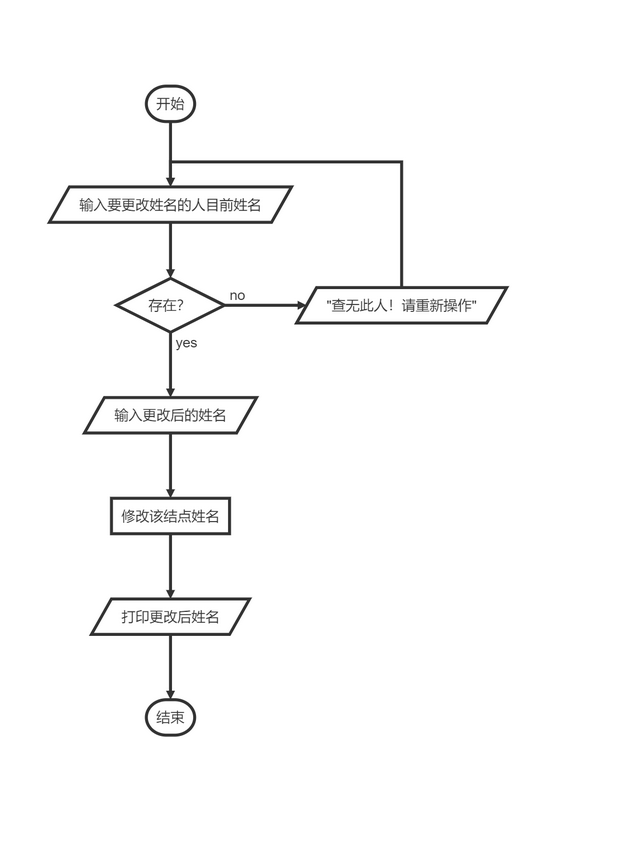
}

### 3.5.3 解散局部家庭功能截屏示例



## 3.6 更改姓名功能的实现

### 3.6.1 更改姓名功能流程图



### 3.6.2 更改姓名功能核心代码

void genealogy::changeName()//改变家庭成员姓名

{

string s1, s2;

cout << "请输入要更改姓名的人目前姓名：";

cin >> s1;

Person\* member =find(ancestor, s1);

if (member == NULL)

{

cout << "查无此人！请重新操作" << endl;

return;

}

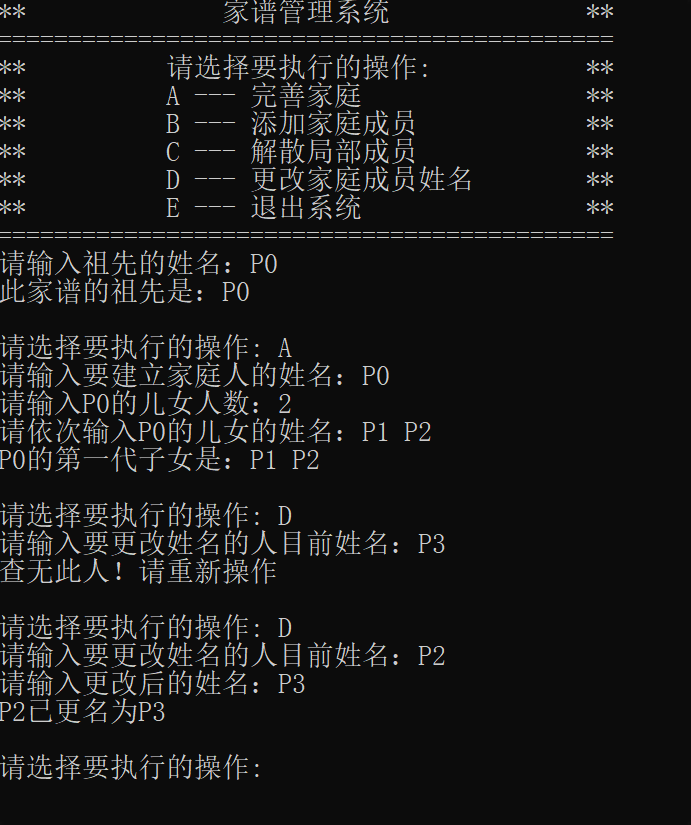
cout << "请输入更改后的姓名：";

cin >> s2;

member->name = s2;

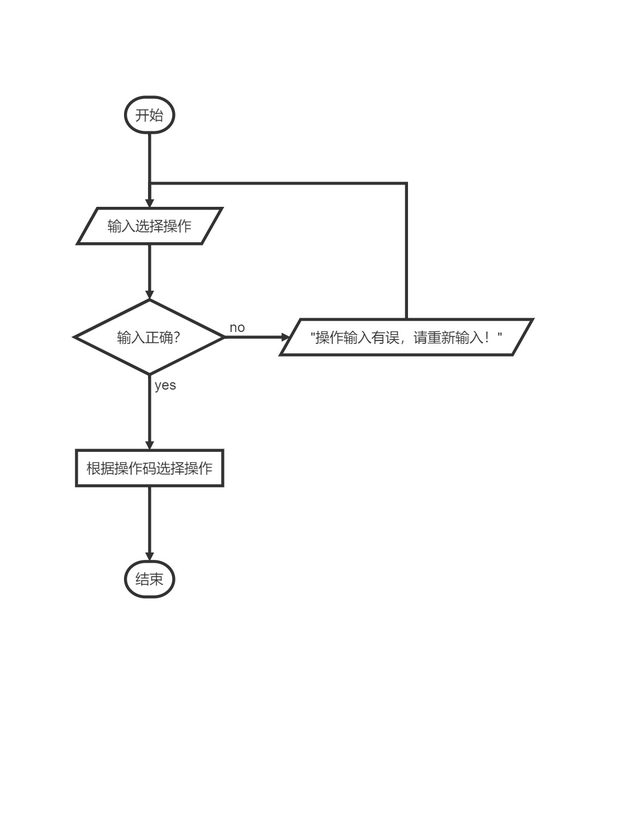
cout << s1 << "已更名为" << s2 << endl;}

### 3.5.3 更改姓名功能截屏示例



## 3.6 总体系统的实现

### 3.6.1 总体系统流程图



### 3.6.2 总体系统核心代码

void genealogy::run() {

cout << "\*\* 家谱管理系统 \*\*" << endl

<< "============================================" << endl

<< "\*\* 请选择要执行的操作: \*\*" << endl

<< "\*\* A --- 完善家庭 \*\*" << endl

<< "\*\* B --- 添加家庭成员 \*\*" << endl

<< "\*\* C --- 解散局部成员 \*\*" << endl

<< "\*\* D --- 更改家庭成员姓名 \*\*" << endl

<< "\*\* E --- 退出系统 \*\*" << endl

<< "============================================" << endl;

construct();

while (1) {

char op;

cout << "请选择要执行的操作: ";

cin >> op;

if (!cin.good()) {

cin.clear();

op = -1; // 输入有误

}

cin.ignore(INT32\_MAX, '\n');

switch (op) {

case 'A':case'a':constructFamily();break;

case 'B':case'b':addChild();break;

case 'C':case'c':deleteFamily();break;

case 'D':case'd':changeName();break;

case 'E':case'e':cout << "正在退出..." << endl;return;

default:cout << "操作输入有误，请重新输入！" << endl << endl;continue;

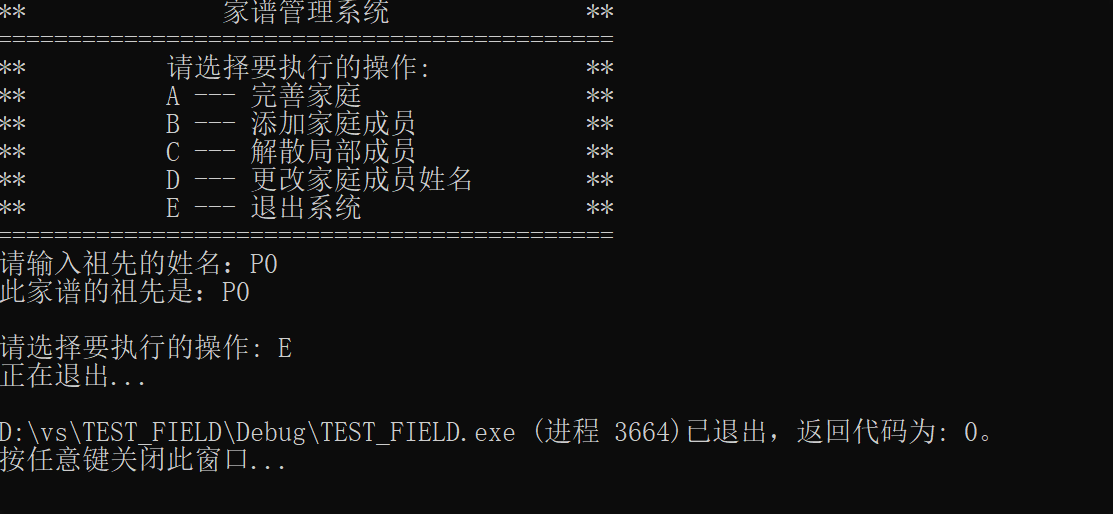
}

cout << endl;

}

}

### 3.6.3 总体系统截屏示例



# 4 测试

## 4.1 功能测试

**见各功能截屏示例**

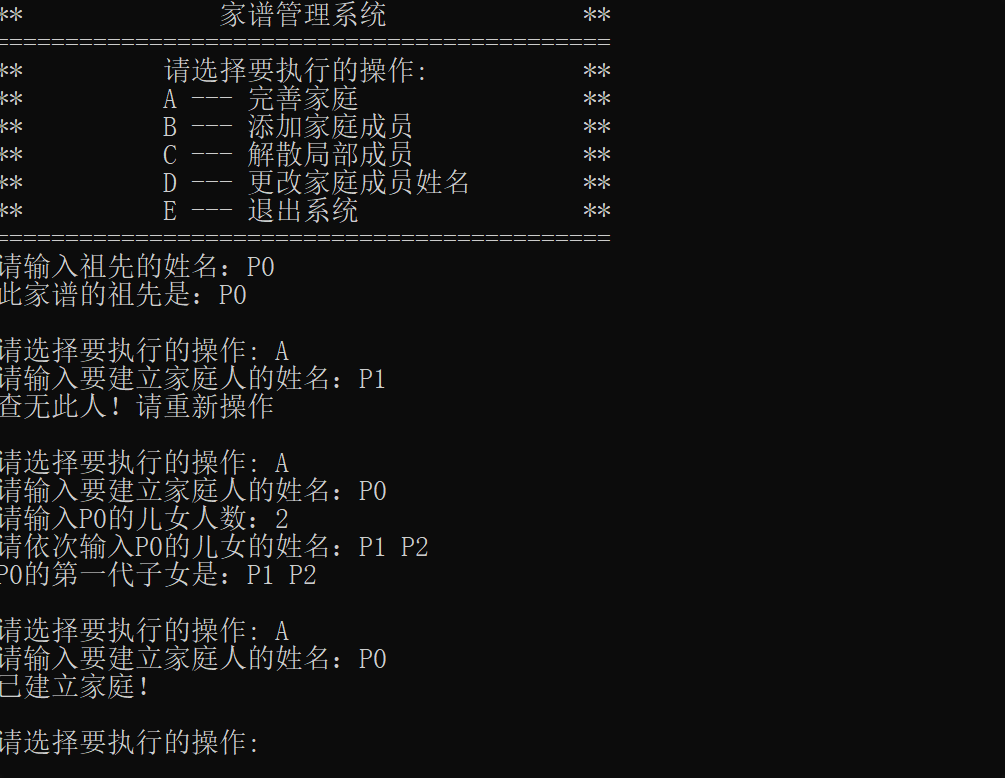
## 4.2 边界测试

### 4.2.1 完善家庭的成员已有家庭

**测试用例：**已建立家庭的结点

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**

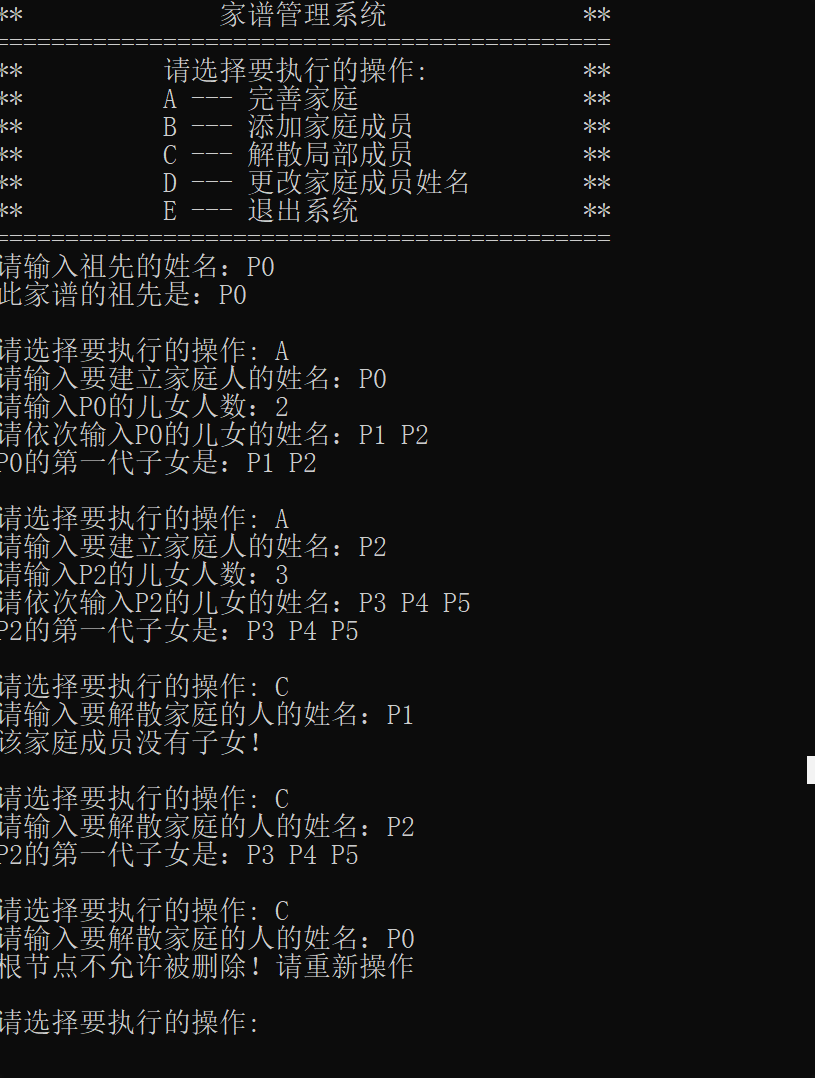


### 4.2.2 解散局部家庭的成员无家庭

**测试用例：**删除无子女结点

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

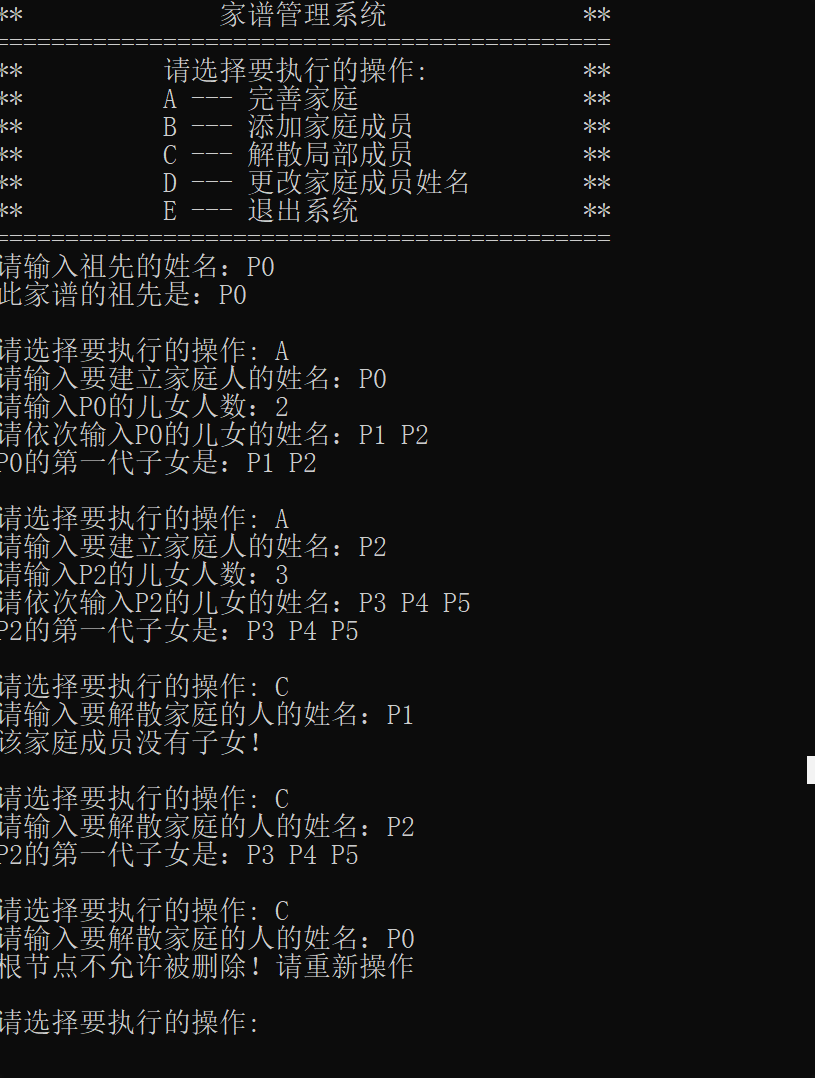


### 4.2.3 解散根节点家庭

**测试用例：**删除根节点

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**



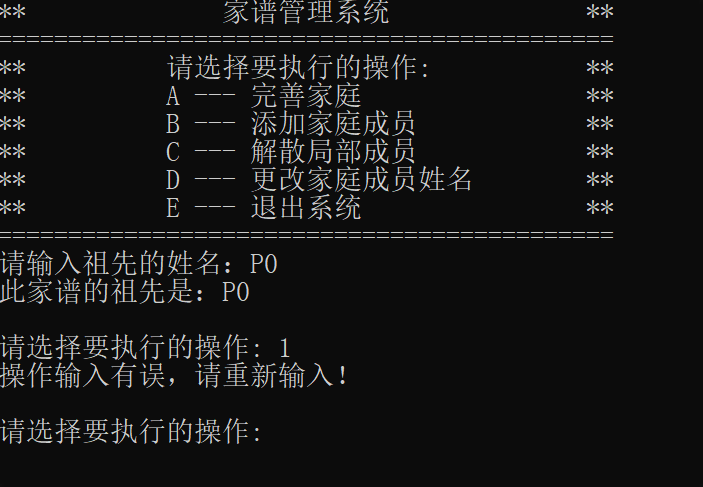
## 4.3 出错测试

### 4.3.1操作码错误

**测试用例：**输入操作码错误

**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：**

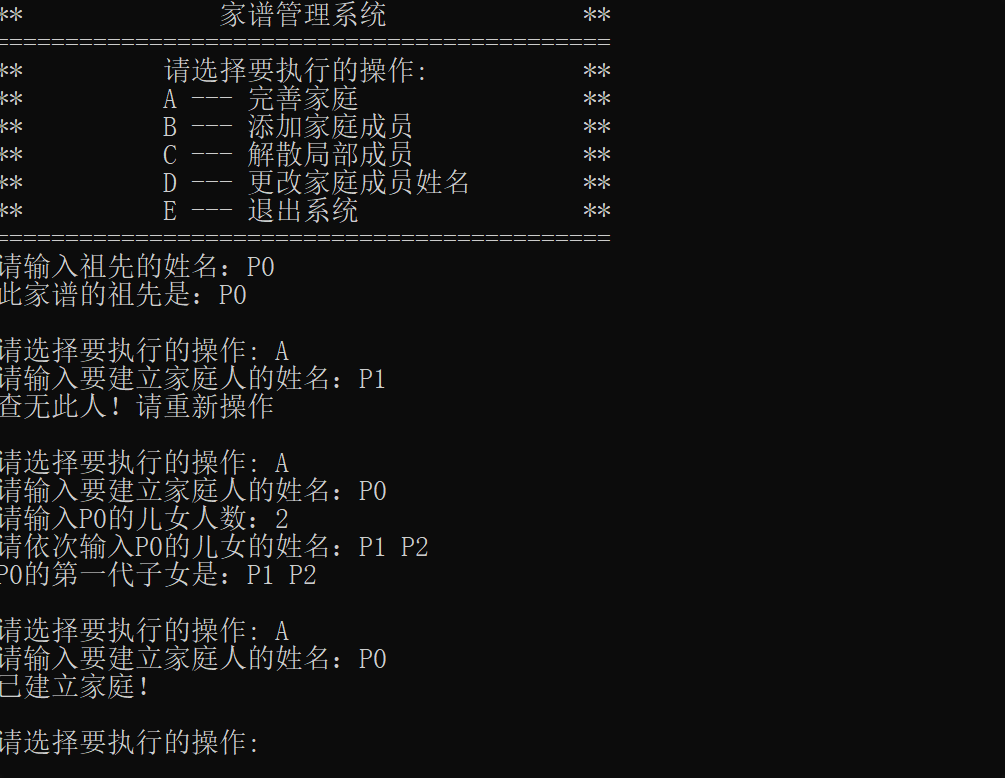
****

### 4.3.2 要完善家庭的成员不存在

**测试用例：**不存在的成员姓名

**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：**

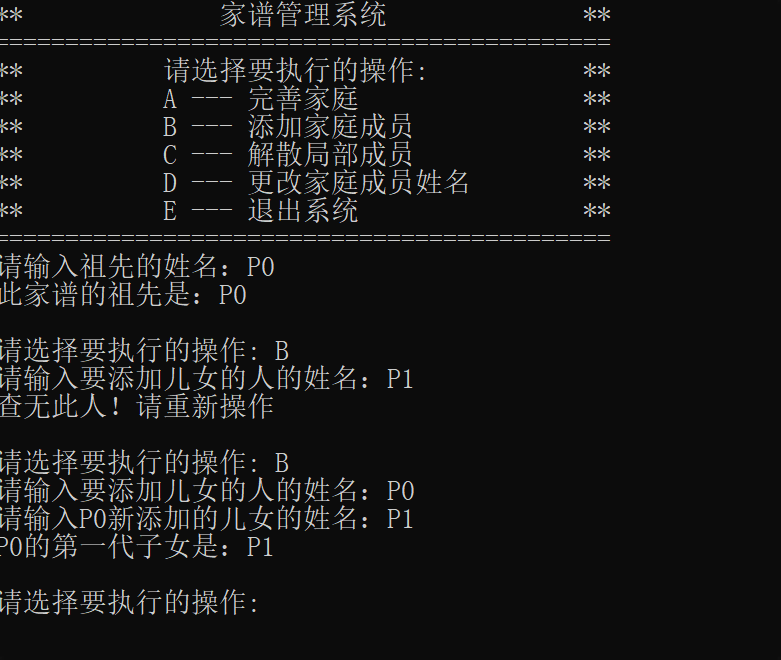


### 4.3.3 要添加成员的成员不存在

**测试用例：**不存在的成员姓名

**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：**

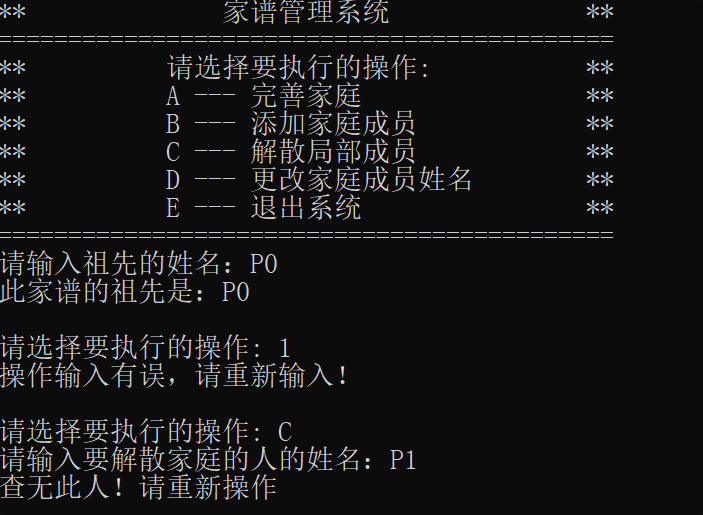


### 4.3.4 要解散家庭的成员不存在

**测试用例：**不存在的成员姓名

**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：**

****

### 4.3.5 要修改姓名的成员不存在

**测试用例：**不存在的成员姓名

**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：**

