项目说明文档

数据结构课程设计

——家谱管理系统

作 者 姓 名： 赵子昱

学 号： 1951459

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](#_Toc59806128)

[1.1 背景分析 1](#_Toc59806129)

[1.2 功能分析 1](#_Toc59806130)

[2 设计 2](#_Toc59806131)

[2.1 数据结构设计 2](#_Toc59806132)

[2.2 类结构设计 2](#_Toc59806133)

[2.3 成员与操作设计 2](#_Toc59806134)

[2.4 系统设计 3](#_Toc59806135)

[3 实现 4](#_Toc59806136)

[3.1 从指定成员开始按姓名查找**Find(…)**功能的实现 4](#_Toc59806137)

[3.1.1 从指定成员开始按姓名查找功能流程图 4](#_Toc59806138)

[3.1.2 从指定成员开始按姓名查找功能代码 5](#_Toc59806139)

[3.2 按姓名查找成员功能的实现 5](#_Toc59806140)

[3.2.1 按姓名查找成员功能代码 5](#_Toc59806141)

[3.3 输出指定成员的子女信息功能**ShowChildren(…)**的实现 5](#_Toc59806142)

[3.3.1 输出指定成员的子女信息功能流程图 6](#_Toc59806143)

[3.3.2 输出指定成员的子女信息功能代码 6](#_Toc59806144)

[3.4 对指定成员建立家庭功能**SetChildren()**的实现 7](#_Toc59806145)

[3.4.1 对指定成员建立家庭功能流程图 7](#_Toc59806146)

[3.4.2 对指定成员建立家庭功能代码 8](#_Toc59806147)

[3.4.3 对指定成员建立家庭功能截屏示例 9](#_Toc59806148)

[3.5 对指定成员添加子女功能**AddChildren(…)**的实现 9](#_Toc59806149)

[3.5.1 对指定成员添加子女功能流程图 10](#_Toc59806150)

[3.5.2 对指定成员添加子女功能代码 11](#_Toc59806151)

[3.5.3 对指定成员添加子女功能截屏示例 11](#_Toc59806152)

[3.6 解散家庭**DeleteFamily()**的实现 12](#_Toc59806153)

[3.6.1 解散家庭功能代码 12](#_Toc59806154)

[3.6.2 解散家庭功能截屏示例 12](#_Toc59806155)

[3.7 对指定成员改名功能**ChangeName()**的实现 13](#_Toc59806156)

[3.7.1 对指定成员改名功能流程图 13](#_Toc59806157)

[3.7.2 对指定成员改名功能代码 14](#_Toc59806158)

[3.7.3 对指定成员改名功能截屏示例 14](#_Toc59806159)

[3.8 查找成员**Search()**的实现 14](#_Toc59806160)

[3.8.1 查找成员功能代码 15](#_Toc59806161)

[3.9 初始化家谱功能**Build()**的实现 15](#_Toc59806162)

[3.9.1 初始化家谱功能代码 15](#_Toc59806163)

[3.9.2 初始化家谱截屏示例 16](#_Toc59806164)

[3.10 **test()**的实现 16](#_Toc59806165)

[3.10.1 **test()**流程图 16](#_Toc59806166)

[3.10.2 解散家庭功能代码 17](#_Toc59806167)

[3.10.3 总体系统截屏示例 18](#_Toc59806168)

[3.11 主函数的实现 18](#_Toc59806169)

[3.11.1 主函数代码 18](#_Toc59806170)

[3.11.2 主函数截屏示例 18](#_Toc59806171)

[4 测试 19](#_Toc59806172)

[4.1 功能测试 19](#_Toc59806173)

[4.1.1 建立功能测试 19](#_Toc59806174)

[4.1.2 插入功能测试 19](#_Toc59806175)

[4.1.3 删除功能测试 21](#_Toc59806176)

[4.1.4 查找功能测试 21](#_Toc59806177)

[4.1.5 修改功能测试 22](#_Toc59806178)

[4.2 边界测试 23](#_Toc59806179)

[4.2.1 要建立家庭的人不在家谱中 23](#_Toc59806180)

[4.2.2 要增加子女的人不在家谱中 23](#_Toc59806181)

[4.2.3 要解散家庭的人不在家谱中 24](#_Toc59806182)

[4.2.4 要更改姓名的人不在家谱中 24](#_Toc59806183)

[4.2.5 要查询的人不在家谱中 25](#_Toc59806184)

[4.3 出错测试 25](#_Toc59806185)

[4.3.1 操作码错误 25](#_Toc59806186)

[4.3.2 添加人女数错误 25](#_Toc59806187)

# 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

家谱的结构类似树，所以项目中选择用树结构来存储家族成员信息。但实际中家谱中每个人可以有多个子女，因此结构为多叉树，为了操作简便，我们采用左长子右兄弟的二叉树结构来存储。

## 2.2 类结构设计

项目中，定义了家谱类FamilyTree，封装了家谱信息与各个操作。还定义了家庭成员类Member，用于作为家谱树的节点。

## 2.3 成员与操作设计

**家谱类（FamilyTree）**

1. **class** FamilyTree {
2. **public**:
3. FamilyTree() :Ancestor(NULL) {};
4. ~FamilyTree() { Destory(Ancestor); }
6. **void** Build();
7. //初始化建立家谱
9. **void** SetChildren();
10. //对指定成员建立家庭
12. **void** AddChildren();
13. //对指定成员添加子女
15. **void** DeleteFamily();
16. //对指定成员解散家庭
18. **void** ChangeName();
19. //对指定成员改名
21. **void** Search();
22. //按姓名查找某个成员并输出其信息
24. **void** test();
25. //用于主函数调用
27. **private**:
28. Member \*Ancestor;
29. //祖先节点
31. Member \* Find(string & name);
32. //按姓名查找成员
34. Member \* Find(Member \*start, string &name);
35. //从指定成员开始按姓名查找
37. **void** ShowChildren(Member \* currentMem);
38. //输出指定成员的子女信息
40. **void** Destory(Member \* ¤tMem);
41. //用于析构函数中
43. };

**成员类（Member）**

1. **class** Member {
2. **friend** **class** FamilyTree;
3. **private**:
4. string m\_Name;
5. //姓名
7. Member \*firstChild, \*nextSibling;
8. //长子指针，兄弟指针
10. **public**:
11. Member(string &name) :m\_Name(name), firstChild(NULL), nextSibling(NULL) {};
12. };

## 2.4 系统设计

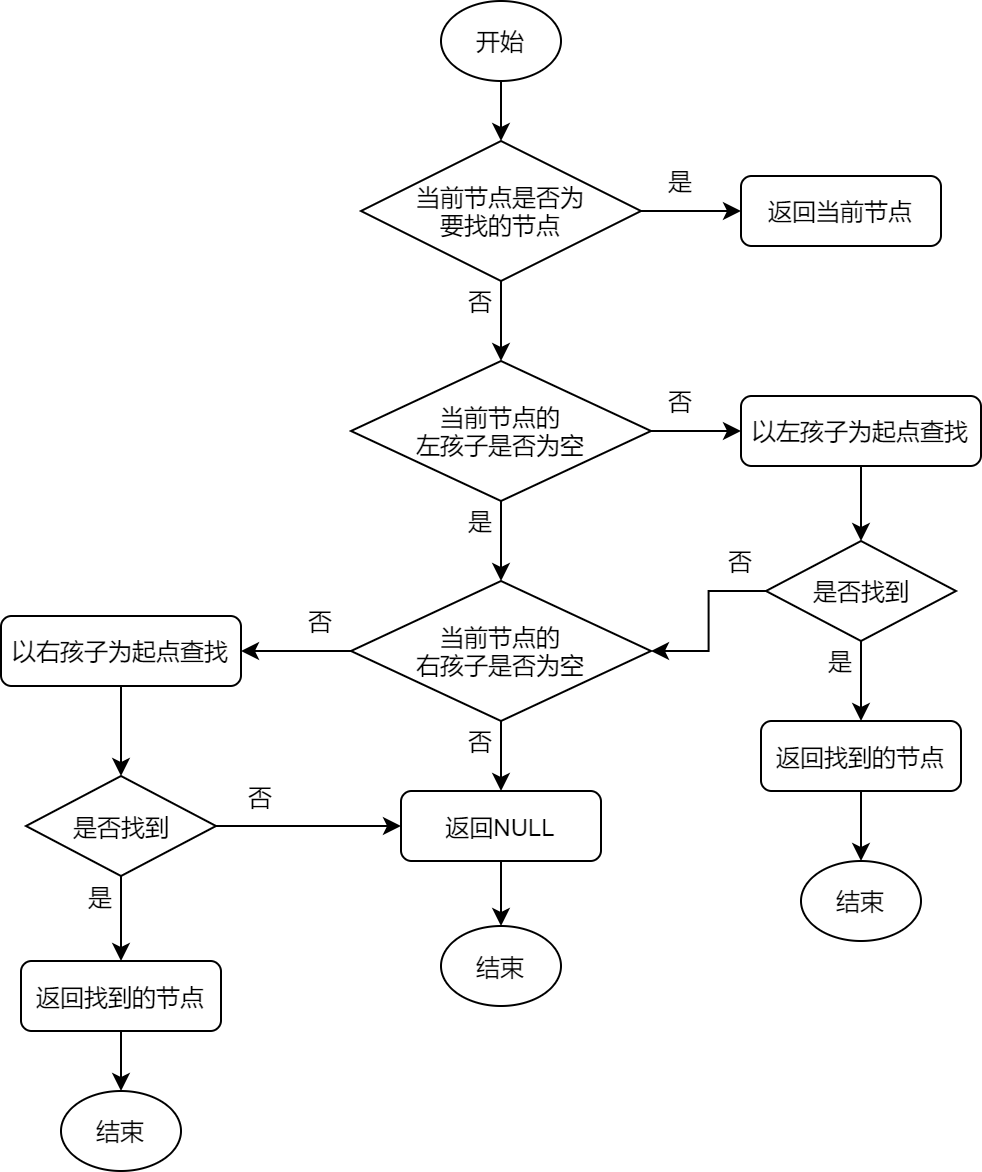
系统首先建立一个家谱对象，完成家谱的建立，然后调用test()根据用户所输入的操作码执行家谱的相应操作。

# 3 实现

## 3.1 从指定成员开始按姓名查找**Find(…)**功能的实现

* start为起点，name为查找目标
* 返回找到的节点地址/NULL
* 被**Find()**函数调用

### 3.1.1 从指定成员开始按姓名查找功能流程图



### 3.1.2 从指定成员开始按姓名查找功能代码

1. Member \* FamilyTree::Find(Member \*start, string &name) {
2. **if** (start->m\_Name == name) {
3. **return** start;
4. }
5. **if** (start->firstChild) {
6. //以左孩子为起点查找
7. Member \* tmpMem = Find(start->firstChild, name);
8. **if** (tmpMem) { **return** tmpMem; }
9. }
10. **if** (start->nextSibling) {
11. //以右兄弟为起点查找
12. Member \* tmpMem = Find(start->nextSibling, name);
13. **if** (tmpMem) { **return** tmpMem; }
14. }
15. **return** NULL;
16. }

## 3.2 按姓名查找成员功能的实现

* name为查找目标
* 返回找到的节点地址/NULL
* 被**SetChildren()、AddChildren()、DeleteFamily()、ChangeName()**函数调用

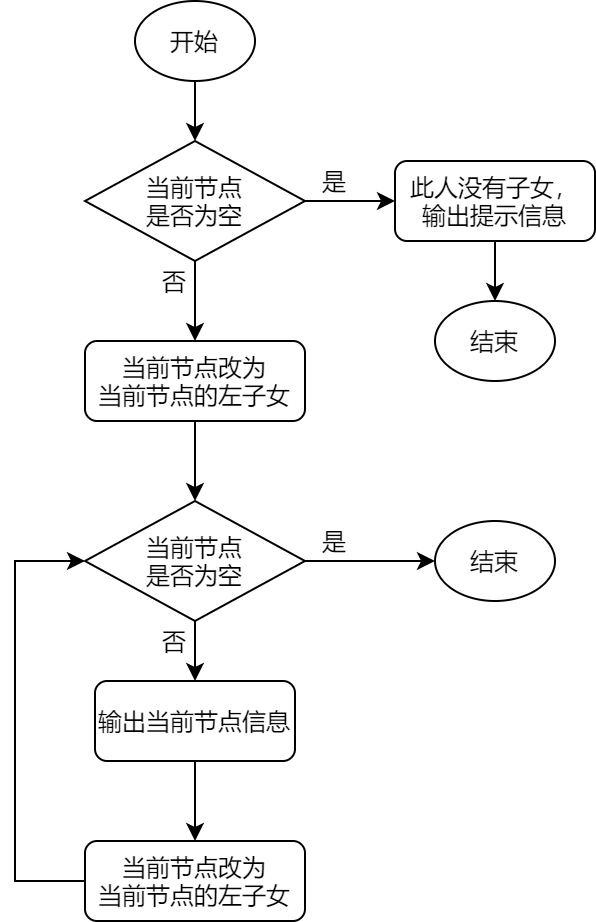
### 3.2.1 按姓名查找成员功能代码

1. Member \* FamilyTree::Find(string & name) {
2. **return** Find(Ancestor, name);
3. }

## 3.3 输出指定成员的子女信息功能**ShowChildren(…)**的实现

* 被**ShowChildren()、AddChildren()、DeleteFamily()、SetChildren()**函数调用

### 3.3.1 输出指定成员的子女信息功能流程图



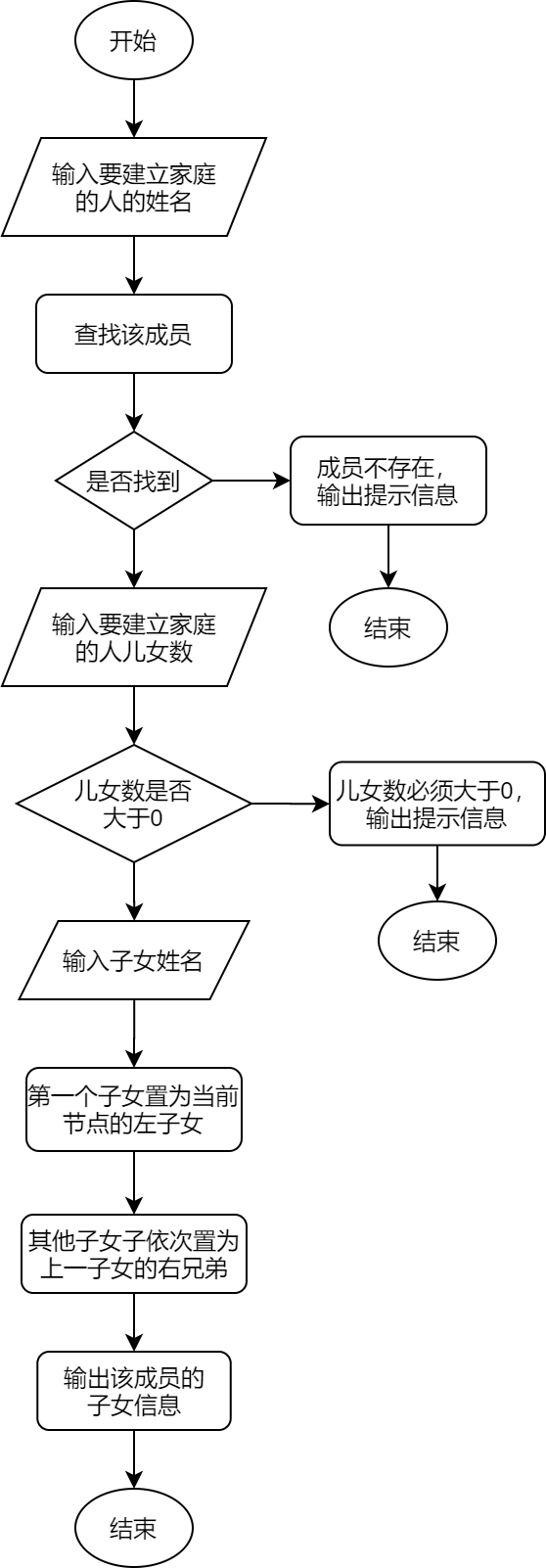
### 3.3.2 输出指定成员的子女信息功能代码

1. **void** FamilyTree::ShowChildren(Member \* Parent) {
2. Member \* currentMem = Parent->firstChild;
3. **if** (!currentMem) {
4. cout << Parent->m\_Name << "此人没有孩子!" << endl;
5. **return**;
6. }
7. cout << Parent->m\_Name << "的第一代子孙是：";
8. **while** (currentMem) {
9. cout << ' ' << currentMem->m\_Name;
10. currentMem = currentMem->nextSibling;
11. }
12. cout << endl;
13. }

## 3.4 对指定成员建立家庭功能**SetChildren()**的实现

* 为指定成员创建子女
* 被**test()**函数调用

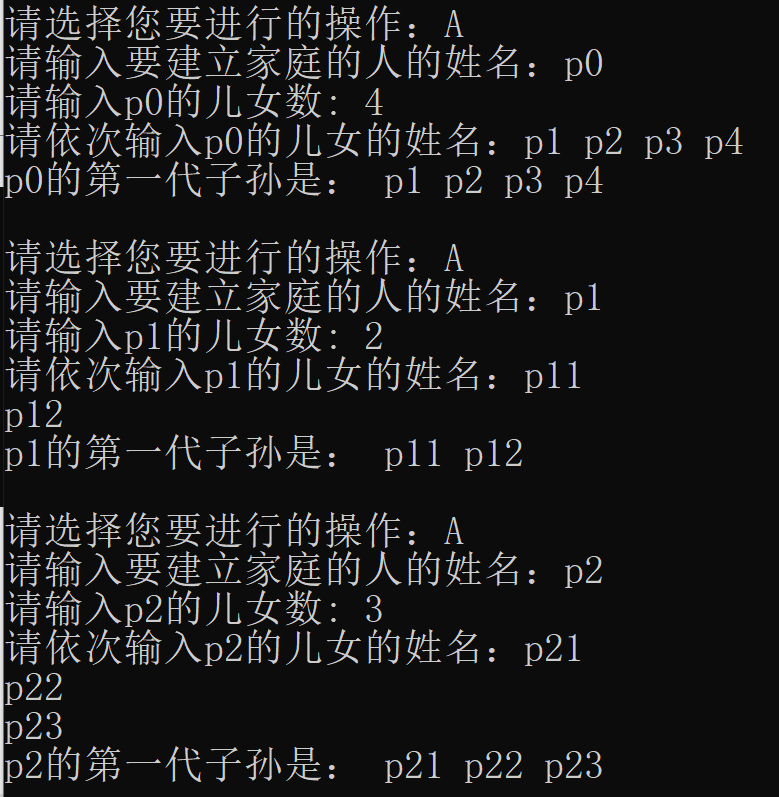
### 3.4.1 对指定成员建立家庭功能流程图



### 3.4.2 对指定成员建立家庭功能代码

1. **void** FamilyTree::SetChildren() {
2. string name;
3. cout << "请输入要建立家庭的人的姓名：";
4. cin >> name;
5. Member \*Parent = Find(name);
6. **if** (!Parent) {
7. cout << "该成员不存在! " << endl;
8. **return**;
9. }
10. **int** ChildNum;
11. cout << "请输入" << name << "的儿女数: ";
12. cin >> ChildNum;
13. **while** (!cin.good()) {
14. cin.clear();
15. cin.ignore(1024, '\n');
16. cout << "非法输入，请重新输入 ：";
17. cin >> ChildNum;
18. }
19. **if** (ChildNum <= 0) {
20. cout << "儿女数必须大于零！" << endl;
21. **return**;
22. }
23. cout << "请依次输入" << name << "的儿女的姓名：";
24. string ChildName;
25. cin >> ChildName;
26. Member \*currentMem = Parent->firstChild = **new** Member(ChildName);
27. **for** (**int** i = 1; i < ChildNum; i++) {
28. cin >> ChildName;
29. currentMem->nextSibling = **new** Member(ChildName);
30. currentMem = currentMem->nextSibling;
31. }
32. ShowChildren(Parent);
33. }

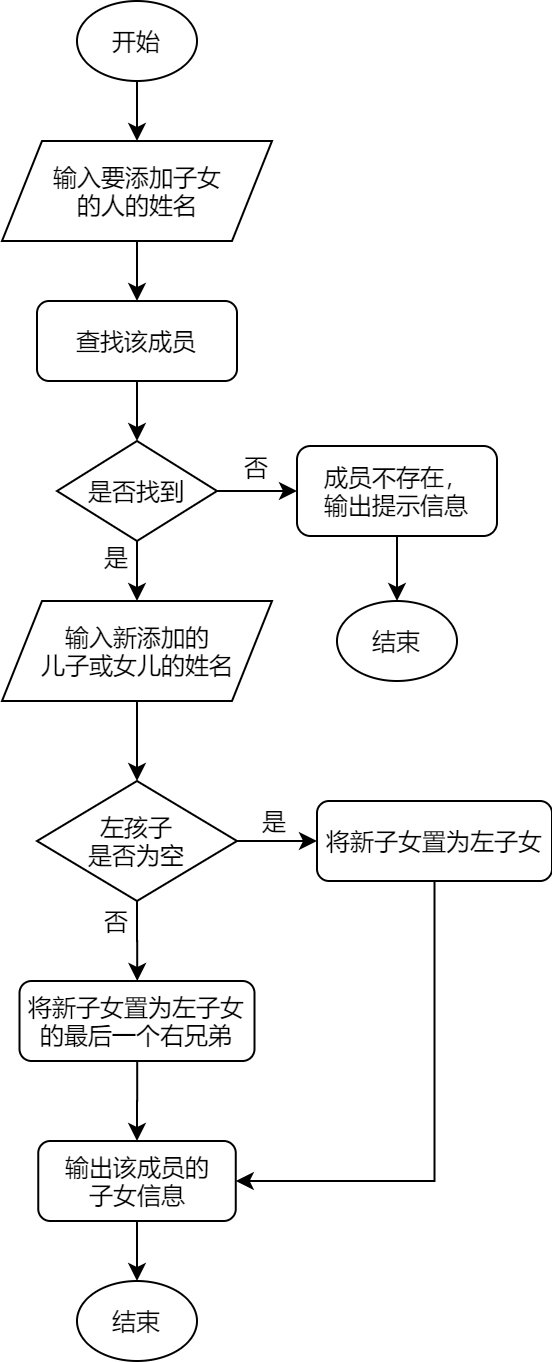
### 3.4.3 对指定成员建立家庭功能截屏示例



## 3.5 对指定成员添加子女功能**AddChildren(…)**的实现

* 为指定成员添加一个子女
* 被**test()**函数调用

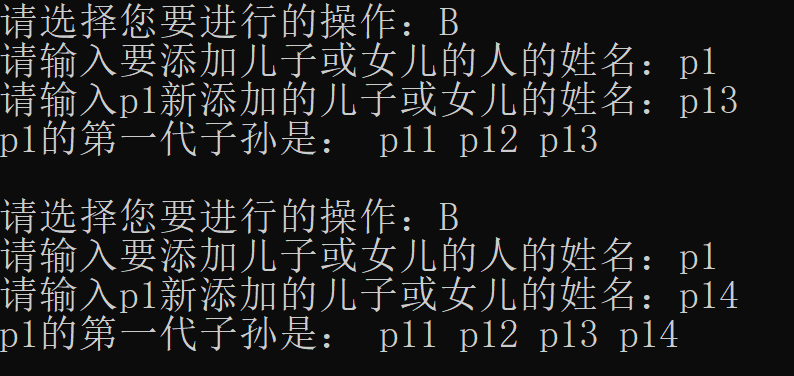
### 3.5.1 对指定成员添加子女功能流程图



### 3.5.2 对指定成员添加子女功能代码

1. **void** FamilyTree::AddChildren() {
2. string name;
3. cout << "请输入要添加儿子或女儿的人的姓名：";
4. cin >> name;
5. Member \*Parent = Find(name);
6. **if** (!Parent) {
7. cout << "该成员不存在! " << endl;
8. **return**;
9. }
10. cout << "请输入" << name << "新添加的儿子或女儿的姓名：";
11. string ChildName;
12. cin >> ChildName;
13. Member \*currentMem = Parent->firstChild;
14. **if** (!currentMem) { Parent->firstChild = **new** Member(ChildName); }
15. **else** {
16. **while** (currentMem->nextSibling) { currentMem = currentMem->nextSibling; }
17. currentMem->nextSibling = **new** Member(ChildName);
18. }
19. ShowChildren(Parent);
20. }

### 3.5.3 对指定成员添加子女功能截屏示例



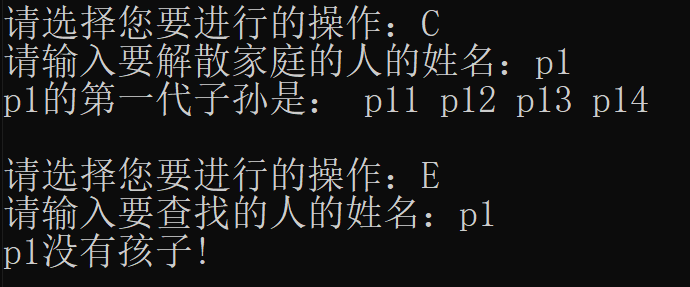
## 3.6 解散家庭**DeleteFamily()**的实现

* 将指定成员的全部子女删除
* 被**test()**函数调用

### 3.6.1 解散家庭功能代码

1. **void** FamilyTree::DeleteFamily() {
2. string name;
3. cout << "请输入要解散家庭的人的姓名：";
4. cin >> name;
5. Member \*Parent = Find(name);
6. **if** (!Parent) {
7. cout << "该成员不存在! " << endl;
8. **return**;
9. }
10. ShowChildren(Parent);
11. **if** (Parent->firstChild) {
12. Destory(Parent->firstChild);
13. }
14. }

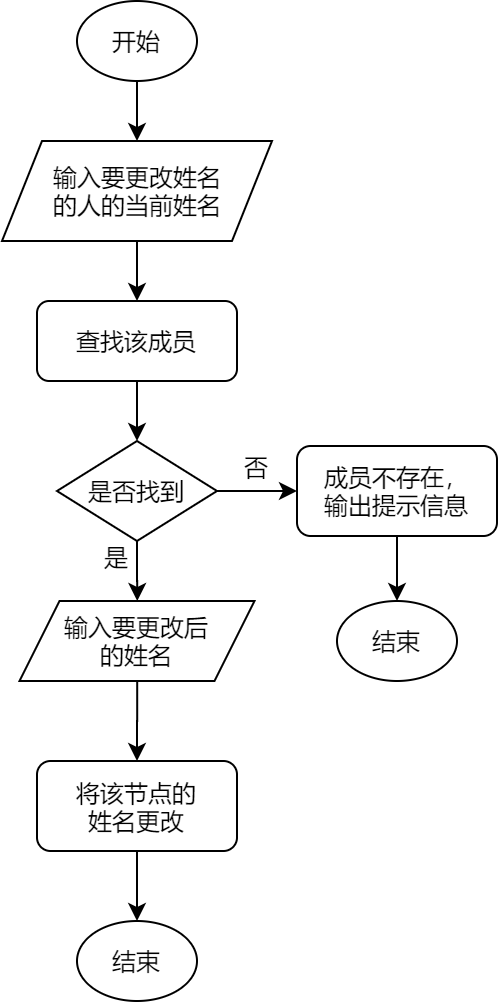
### 3.6.2 解散家庭功能截屏示例



## 3.7 对指定成员改名功能**ChangeName()**的实现

* 被**test()**函数调用

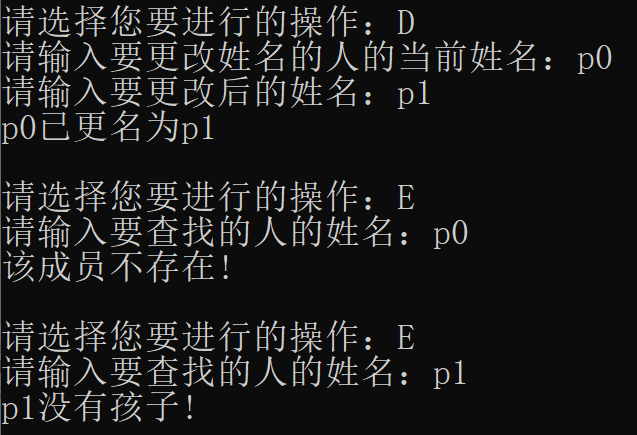
### 3.7.1 对指定成员改名功能流程图



### 3.7.2 对指定成员改名功能代码

1. **void** FamilyTree::ChangeName() {
2. string currentName,newName;
3. cout << "请输入要更改姓名的人的当前姓名：";
4. cin >> currentName;
5. Member \*Mem = Find(currentName);
6. **if** (!Mem) {
7. cout << "该成员不存在! " << endl;
8. **return**;
9. }
10. cout << "请输入要更改后的姓名：";
11. cin >> newName;
12. Mem->m\_Name = newName;
13. cout << currentName << "已更名为" << newName << endl;
14. }

### 3.7.3 对指定成员改名功能截屏示例



## 3.8 查找成员**Search()**的实现

* 查找指定成员并输出其信息
* 被**test()**函数调用

### 3.8.1 查找成员功能代码

1. **void** FamilyTree::Search() {
2. string name;
3. cout << "请输入要查找的人的姓名：";
4. cin >> name;
5. Member \*Mem = Find(name);
6. **if** (!Mem) {
7. cout << "该成员不存在! " << endl;
8. **return**;
9. }
10. ShowChildren(Mem);
11. }

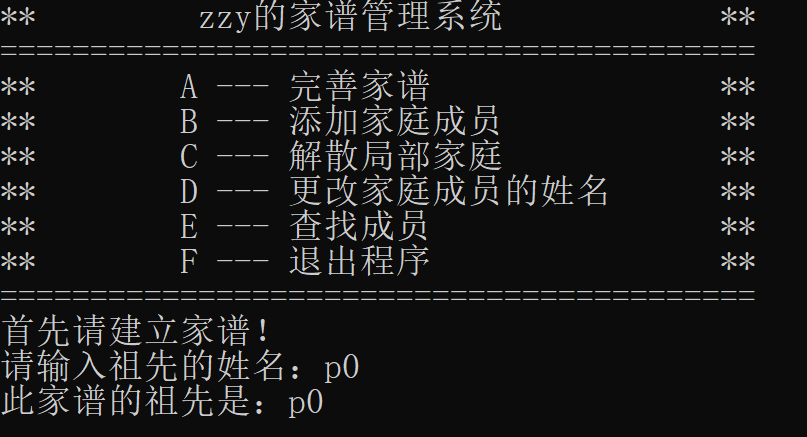
## 3.9 初始化家谱功能**Build()**的实现

* 添加祖先
* 被**test()**函数调用

### 3.9.1 初始化家谱功能代码

1. **void** FamilyTree::Build() {
2. string name;
3. cout << "首先请建立家谱！" << endl;
4. cout << "请输入祖先的姓名：";
5. cin >> name;
6. Ancestor = **new** Member(name);
7. cout << "此家谱的祖先是：" << name << endl;
8. }

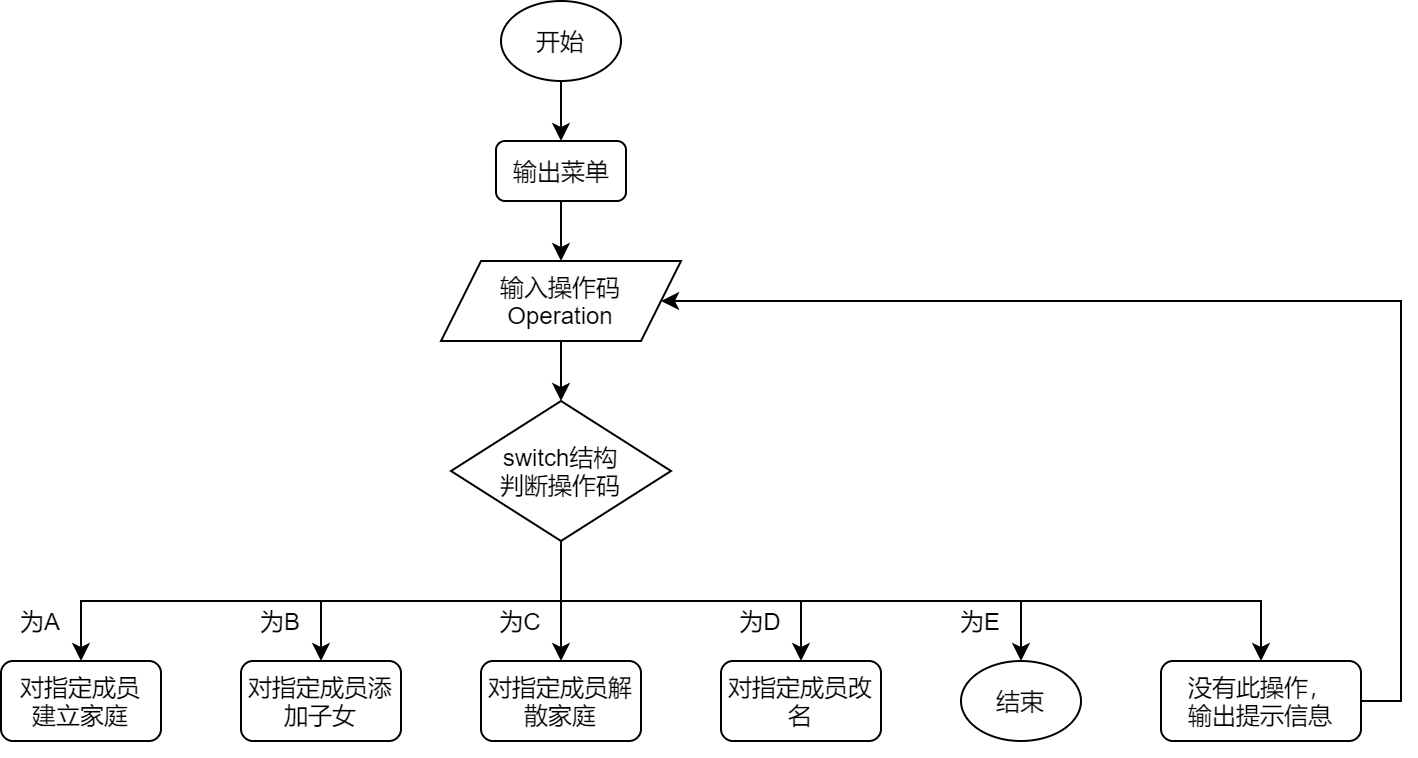
### 3.9.2 初始化家谱截屏示例



## 3.10 **test()**的实现

* 作用类似于主函数
* 被主函数调用

### 3.10.1 **test()**流程图



### 3.10.2 解散家庭功能代码

1. **void** FamilyTree::test() {
2. cout << "\*\*         zzy的家谱管理系统        "<<'\t'<<"\*\*" << endl;
3. cout << "==========================================" << endl;
4. cout << "\*\*        A --- 完善家谱            " << '\t' << "\*\*" << endl;
5. cout << "\*\*        B --- 添加家庭成员        " << '\t' << "\*\*" << endl;
6. cout << "\*\*        C --- 解散局部家庭        " << '\t' << "\*\*" << endl;
7. cout << "\*\*        D --- 更改家庭成员的姓名   " << '\t' << "\*\*" << endl;
8. cout << "\*\*        E --- 查找成员            " << '\t' << "\*\*" << endl;
9. cout << "\*\*        F --- 退出程序            " << '\t' << "\*\*" << endl;
10. cout << "==========================================" << endl;
11. Build();
12. cout << endl;
13. **while** (1) {
14. cout << "请选择您要进行的操作：";
15. **char** Operation;
16. cin >> Operation;
17. **switch** (Operation)
18. {
19. **case** 'A':
20. SetChildren();
21. cout << endl;
22. **break**;
23. **case** 'B':
24. AddChildren();
25. cout << endl;
26. **break**;
27. **case** 'C':
28. DeleteFamily();
29. cout << endl;
30. **break**;
31. **case** 'D':
32. ChangeName();
33. cout << endl;
34. **break**;
35. **case** 'E':
36. Search();
37. cout << endl;
38. **break**;
39. **case** 'F':
40. **return**;
41. **default**:
42. cout << "没有此操作！" << endl;
43. cin.ignore(1024, '\n');
44. cout << endl;
45. **break**;
46. }
47. }
48. }

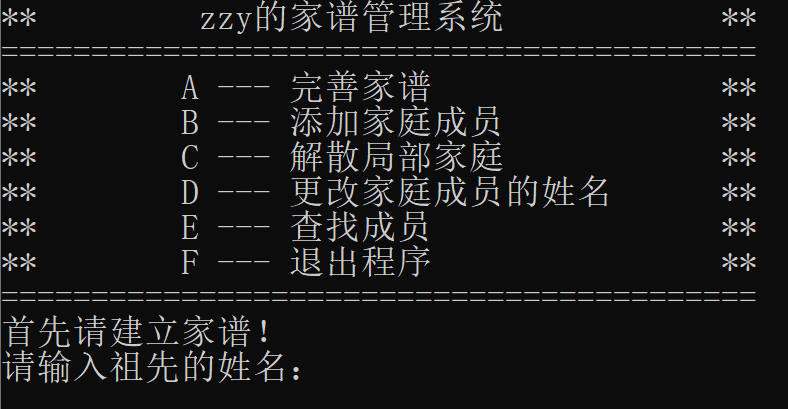
### 3.10.3 总体系统截屏示例

## 3.11 主函数的实现

### 3.11.1 主函数代码

1. **int** main() {
2. FamilyTree myTree;
3. myTree.test();
4. system("pause");
5. **return** 0;
6. }

### 3.11.2 主函数截屏示例



# 4 测试

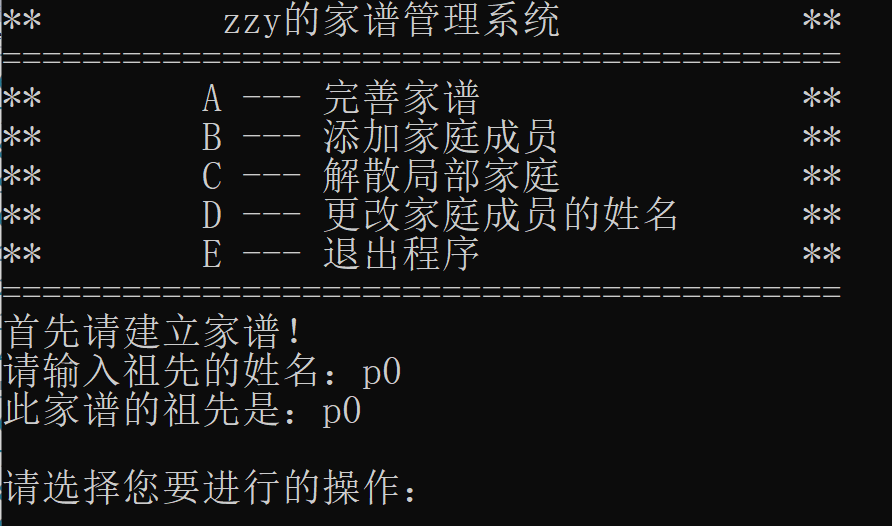
## 4.1 功能测试

### 4.1.1 建立功能测试

**测试用例**：

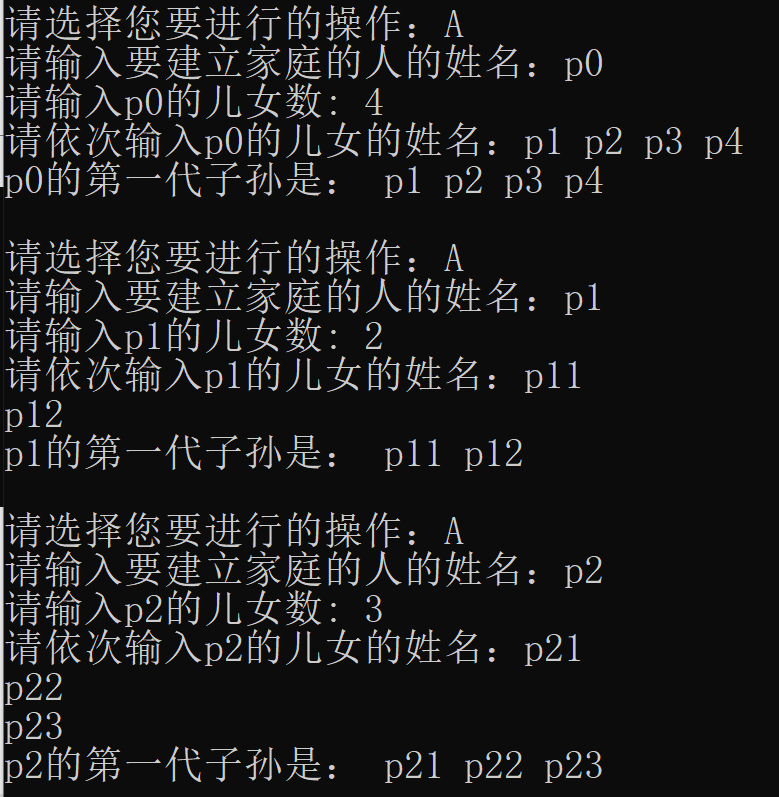
P0

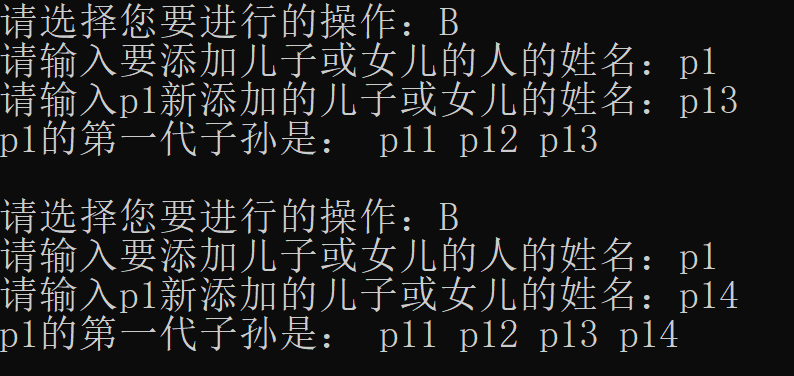
**实验结果**



### 4.1.2 插入功能测试

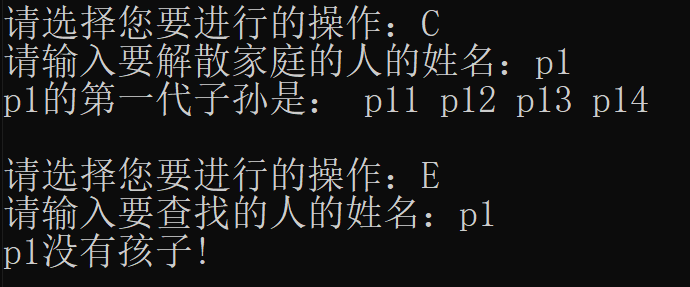
**实验结果：**





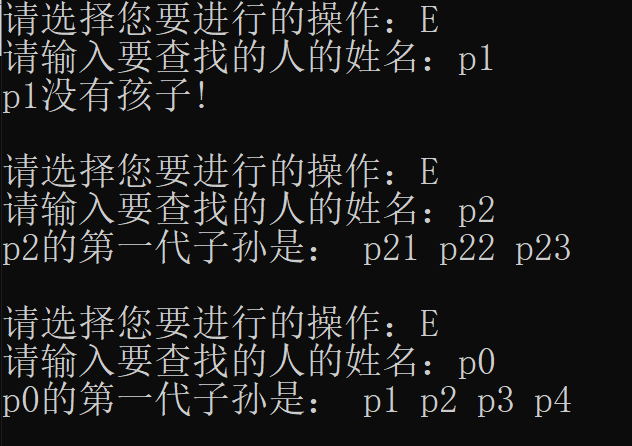
### 4.1.3 删除功能测试

**实验结果：**



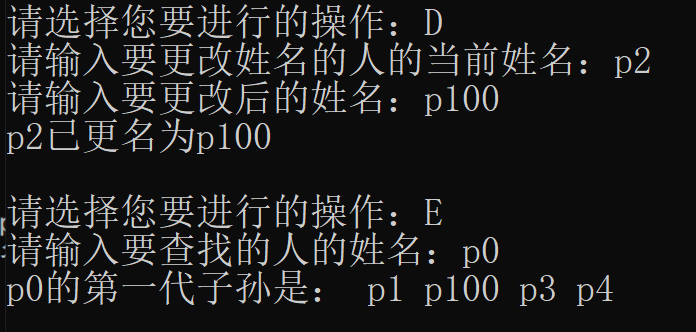
### 4.1.4 查找功能测试

**实验结果：**



### 4.1.5 修改功能测试

**实验结果：**

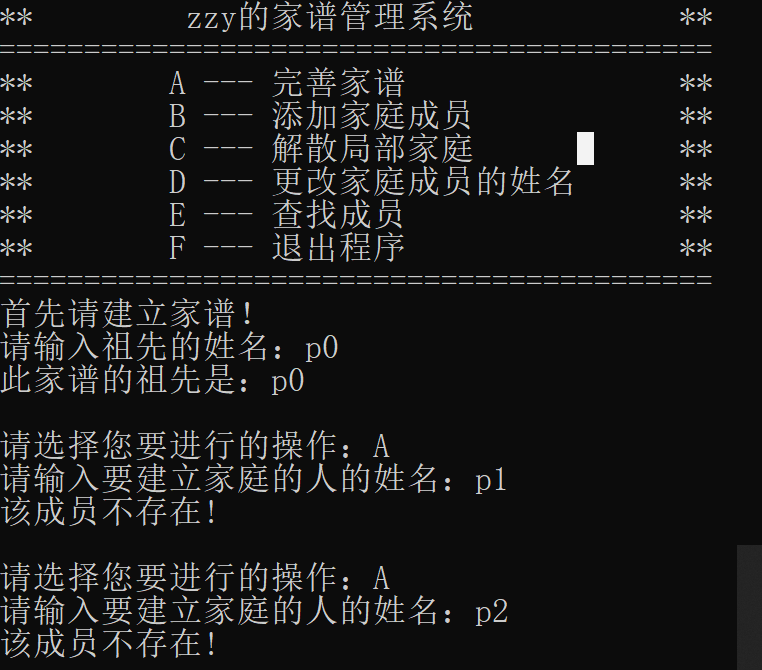


## 4.2 边界测试

### 4.2.1 要建立家庭的人不在家谱中

**预期结果：**给出错误提示，程序运行正常不崩溃。

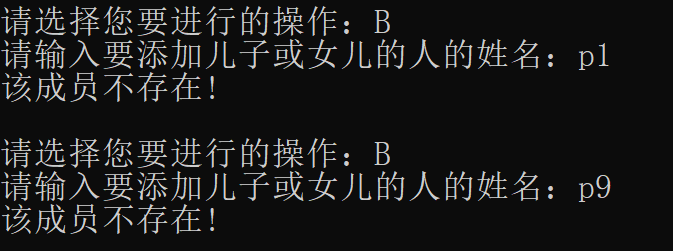
**实验结果：**



### 4.2.2 要增加子女的人不在家谱中

**预期结果：**程序正常运行，不崩溃。

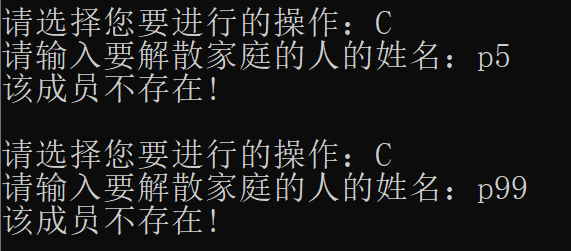
**实验结果：**



### 4.2.3 要解散家庭的人不在家谱中

**预期结果：**程序正常运行，不崩溃。

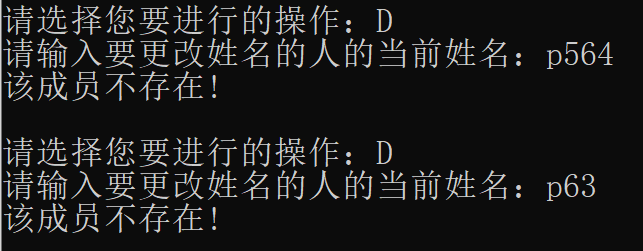
**实验结果：**



### 4.2.4 要更改姓名的人不在家谱中

**预期结果：**程序正常运行，不崩溃。

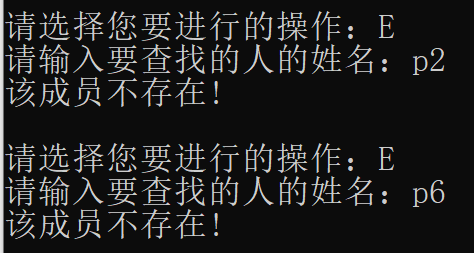
**实验结果：**



### 4.2.5 要查询的人不在家谱中

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

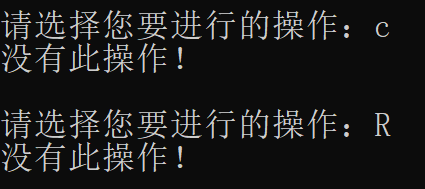


## 4.3 出错测试

### 4.3.1 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。

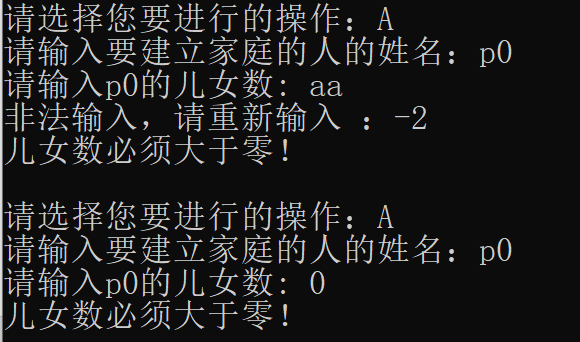
**实验结果：**

****

### 4.3.2 添加人女数错误

**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：**

****