项目6说明文档

数据结构课程设计

——家谱管理系统

作 者 姓 名： 谭欢秘

学 号： 1954185

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](#_Toc60232584)

[1.1 背景分析 1](#_Toc60232585)

[1.2 功能分析 1](#_Toc60232586)

[2 设计 1](#_Toc60232587)

[2.1 数据结构设计 1](#_Toc60232588)

[2.2 类结构设计 2](#_Toc60232589)

[2.3 成员与操作设计 2](#_Toc60232590)

[3.1.1 FamilyNode类成员： 2](#_Toc60232591)

[3.1.2 FamilyTree类成员： 2](#_Toc60232592)

[2.4 主函数设计 3](#_Toc60232593)

[3 实现 4](#_Toc60232594)

[3.1 定位成员功能的实现 4](#_Toc60232595)

[3.1.1 定位成员功能流程图 4](#_Toc60232596)

[3.1.2 定位成员功能核心代码 4](#_Toc60232597)

[3.2 定位成员的最后一个孩子功能的实现 5](#_Toc60232598)

[3.2.1 定位最后一个孩子功能流程图 5](#_Toc60232599)

[3.2.2 定位最后一个孩子功能核心代码 5](#_Toc60232600)

[3.3 输出成员的所有子女功能的实现 6](#_Toc60232601)

[3.3.1 输出所有子女功能流程图 6](#_Toc60232602)

[3.3.2 输出所有子女功能核心代码 6](#_Toc60232603)

[3.4 成员建立家庭功能的实现 7](#_Toc60232604)

[3.4.1 建立家庭功能流程图 7](#_Toc60232605)

[3.4.2 建立家庭功能核心代码 7](#_Toc60232606)

[3.5 成员添加子女成员功能的实现 9](#_Toc60232607)

[3.5.1 添加子女功能流程图 9](#_Toc60232608)

[3.6.2 搜索家庭成员功能核心代码 9](#_Toc60232609)

[3.6 搜索家庭成员功能的实现 10](#_Toc60232610)

[3.6.1 搜索家庭成员功能流程图 10](#_Toc60232611)

[3.6.2 搜索家庭成员功能核心代码 11](#_Toc60232612)

[3.7 删除家庭成员功能的实现 11](#_Toc60232613)

[3.7.1 删除家庭成员功能流程图 11](#_Toc60232614)

[3.7.2 删除家庭成员功能核心代码 12](#_Toc60232615)

[3.7.3 remove()函数实现时需要注意的细节 13](#_Toc60232616)

[3.8 修改家庭成员信息功能的实现 13](#_Toc60232617)

[3.8.1 修改成员信息功能流程图 13](#_Toc60232618)

[3.8.2 修改成员信息功能核心代码 14](#_Toc60232619)

[3.9 初始化家谱系统功能的实现 14](#_Toc60232620)

[3.9.1 初始化家谱系统功能核心代码 14](#_Toc60232621)

[3.9.2 initialize()函数实现时需要注意的细节 15](#_Toc60232622)

[3.10 运行家谱系统功能的实现 15](#_Toc60232623)

[3.10.1 运行家谱系统功能核心代码 15](#_Toc60232624)

[3.10.2 run()函数实现时需要注意的细节 15](#_Toc60232625)

[3.11 主函数的实现 16](#_Toc60232626)

[3.11.1 主函数核心代码 16](#_Toc60232627)

[3.11.2 总体系统截屏示例 16](#_Toc60232628)

[4 测试 17](#_Toc60232629)

[4.1 功能测试 17](#_Toc60232630)

[4.1.1 建立家庭功能测试 17](#_Toc60232631)

[4.1.2 插入家庭成员功能测试 17](#_Toc60232632)

[4.1.3 搜索成员功能测试 18](#_Toc60232633)

[4.1.4 删除成员功能测试 18](#_Toc60232634)

[4.1.4 修改成员信息功能测试 18](#_Toc60232635)

[4.2 边界测试 19](#_Toc60232636)

[4.2.1 对家谱中不存在的成员做操作 19](#_Toc60232637)

[4.2.2尝试插入已存在的成员 20](#_Toc60232638)

[4.2.3 尝试删除家族祖先 20](#_Toc60232639)

[5 总结 21](#_Toc60232640)

[5.1 遇到的错误 21](#_Toc60232641)

[5.1.1 xx函数中 21](#_Toc60232642)

[5.2 改进与优化 21](#_Toc60232643)

[5.3 项目心得 21](#_Toc60232644)

# 1 分析

## 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。家谱管理系统是对家谱资料电子化、信息化的必要手段，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、为某个家庭成员建立家庭等功能。

## 1.2 功能分析

作为一个家谱管理系统，其基本的功能就是首先通过输入祖先名字来创立一个家族。在此基础上，可以为家庭成员建立小家庭，即一次性为某成员添加若干名子女。此外，还应具有查询家族成员、插入家族成员、删除家族成员、修改家族成员信息等等功能。最后，家谱管理系统还应该确保程序可以正常关闭，停止对家谱的写入、读取操作。

综上所述，一个家谱管理系统至少应该具有创建家族、建立家庭、添加成员、删除成员、查询成员的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，一个成员可以有多个孩子、但只能有一个父母，即父母与孩子之间是一对多的结构。因此，纯线性的链表结构不再适用于本项目，而应转为使用“一对多”的树结构。在树结构中我们较为擅长使用的多是二叉树，但一个人可以有许多的孩子，所以在设计树的节点时应采用“左长子 右兄弟”的存储结构、便于存储。

## 2.2 类结构设计

经典的树一般包括两个抽象数据类型（ADT）——树结点类（LNode）与树类（LinkList），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。为方便处理，本系统将树类FamilyTree类作为节点类FamilyNode的友元，这样便于FamilyTree类访问结点FamilyNode类的私有成员。

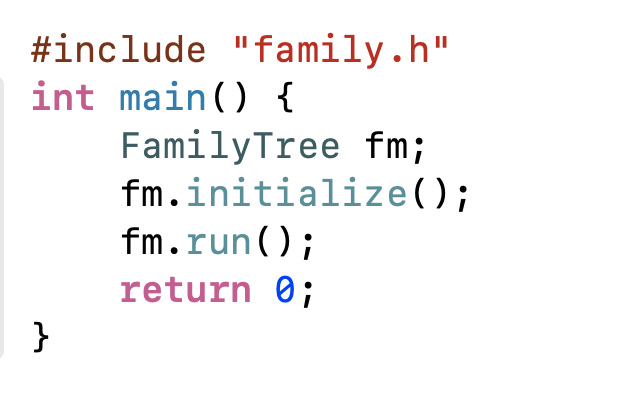
## 2.3 成员与操作设计

### 3.1.1 FamilyNode类成员：

### 3.1.2 FamilyTree类成员：



## 2.4 主函数设计

声明一个FamilyTree类的对象ft。首先调用ft.initialize()函数实现家谱系统的初始化，完成对家谱祖先的创建。然后调用ft.run()，然后根据用户所选择的操作（operation变量）执行树结构中相应的操作。

# 3 实现

## 3.1 定位成员功能的实现

### 3.1.1 定位成员功能流程图

### 3.1.2 定位成员功能核心代码

## 3.2 定位成员的最后一个孩子功能的实现

### 3.2.1 定位最后一个孩子功能流程图

### 3.2.2 定位最后一个孩子功能核心代码

## 3.3 输出成员的所有子女功能的实现

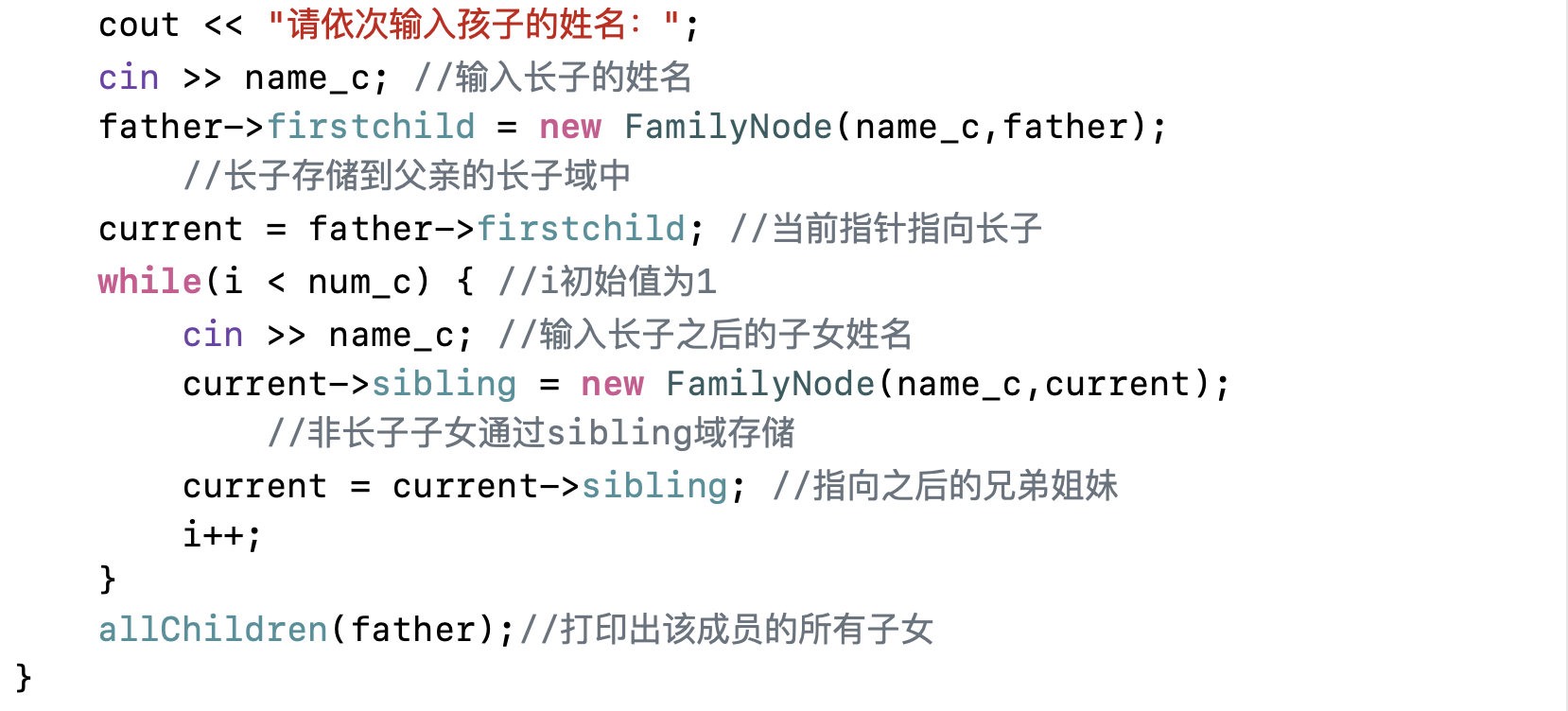
### 3.3.1 输出所有子女功能流程图

### 3.3.2 输出所有子女功能核心代码

## 3.4 成员建立家庭功能的实现

### 3.4.1 建立家庭功能流程图

### 3.4.2 建立家庭功能核心代码

注意边界保护部分，首先要检查名叫name\_f的人是否存在于家谱系统中，若不存在则不能为他建立家庭、并要求用户重新输入一个名字。在输入子女个数时也应做好边界保护，要求子女个数必须为正整数。

## 3.5 成员添加子女成员功能的实现

### 3.5.1 添加子女功能流程图

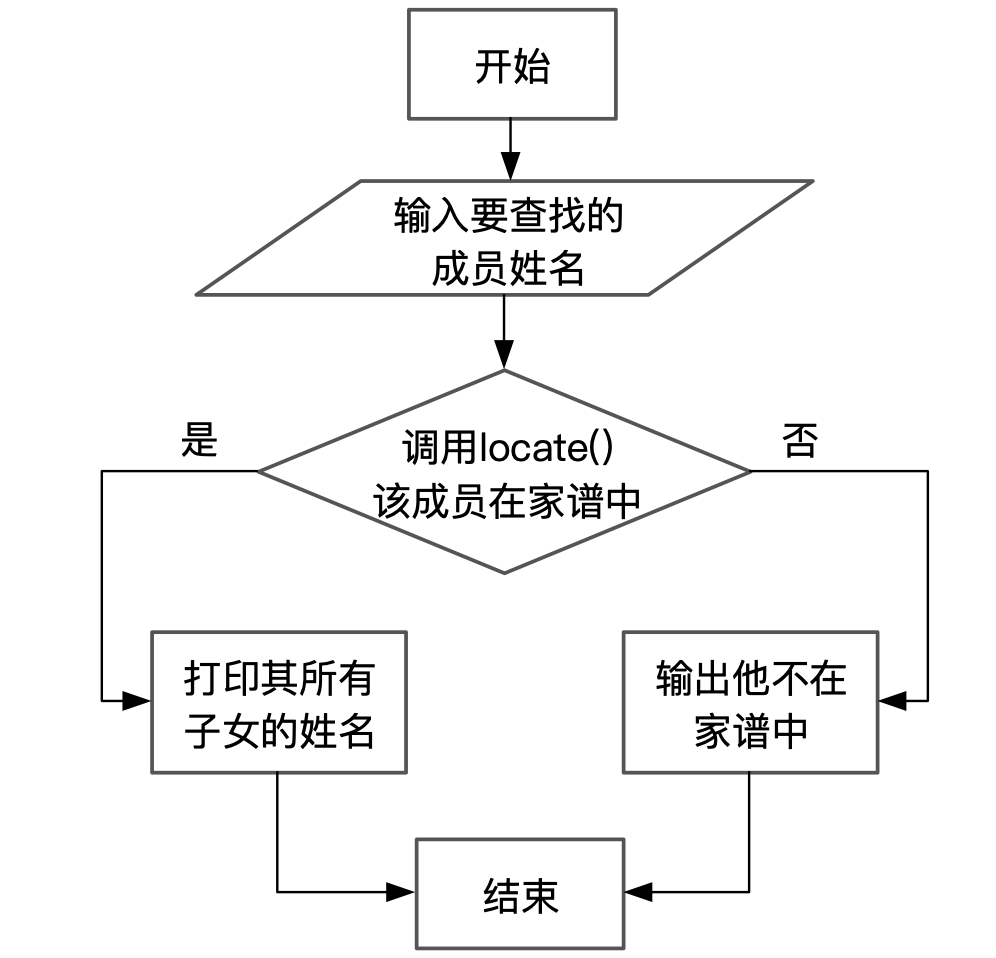
### 3.6.2 搜索家庭成员功能核心代码





## 3.6 搜索家庭成员功能的实现

### 3.6.1 搜索家庭成员功能流程图



### 3.6.2 搜索家庭成员功能核心代码

## 3.7 删除家庭成员功能的实现

### 3.7.1 删除家庭成员功能流程图

### 3.7.2 删除家庭成员功能核心代码

### 3.7.3 remove()函数实现时需要注意的细节

本函数在实现过程中，同样要注意边界保护、要检查删除成员是否存在于家谱系统中。此外，按照严谨的工程思维来处理问题，在删除家庭成员时，修改了他前驱节点的指针域虽然已经是从逻辑上删除了，但他和他子女的内存还未回收。所以还需对他的子女做检查，将其和其子女的内存逐一回收。完成这个函数时，不仅要考虑表面的效果、更应保护好底层的内存，才是真正实现了删除操作。

## 3.8 修改家庭成员信息功能的实现

### 3.8.1 修改成员信息功能流程图

### 3.8.2 修改成员信息功能核心代码

## 3.9 初始化家谱系统功能的实现

### 3.9.1 初始化家谱系统功能核心代码

### 3.9.2 initialize()函数实现时需要注意的细节

家族树首先需要一个根节点，所以初始化函数首先应做的是接受祖先姓名、创建根节点。随后输出本家谱管理系统支持的操作菜单即可。

## 3.10 运行家谱系统功能的实现

### 3.10.1 运行家谱系统功能核心代码

### 3.10.2 run()函数实现时需要注意的细节

如果输入了A～F之外的操作，利用switch的default向用户报错、并重新输入；循环在操作码为F时停止，系统运行结束。

## 3.11 主函数的实现

### 3.11.1 主函数核心代码

### 3.11.2 总体系统截屏示例

# 4 测试

## 4.1 功能测试

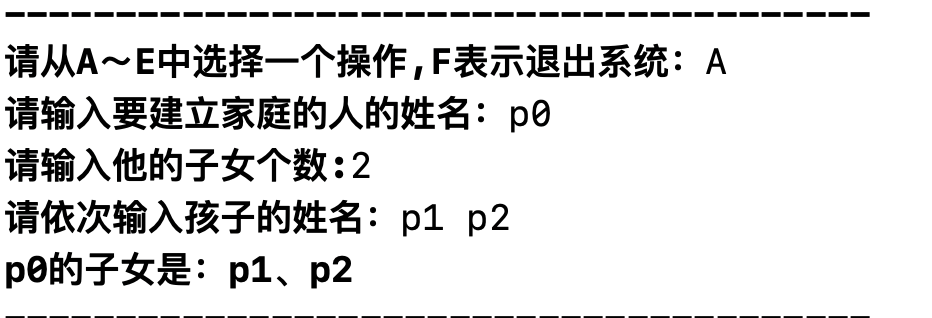
### 4.1.1 建立家庭功能测试

**测试用例**：

为p0（祖先）建立家庭，有2个子女 p1、p2

**预期结果**：

p0将有两个子女，分别是p1、p2

**实验结果：**

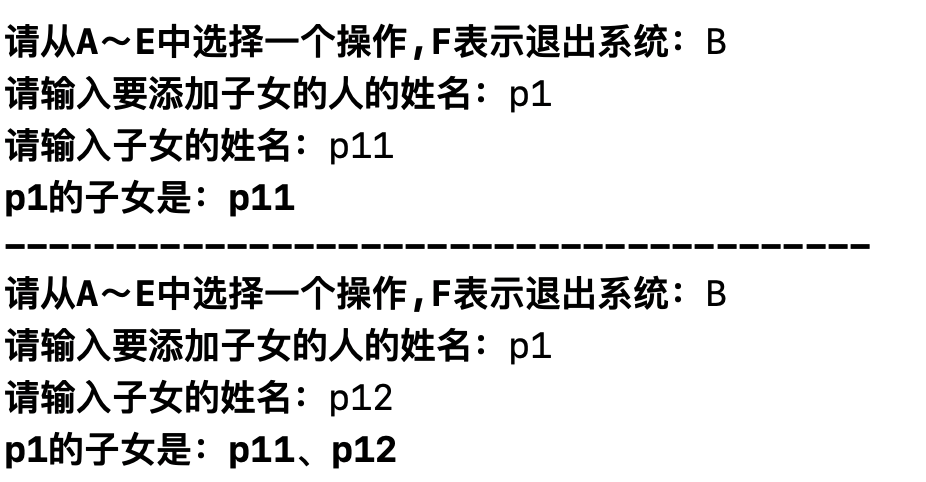
### 4.1.2 插入家庭成员功能测试

**测试用例：**

为p1插入两个子女p11、p12

**预期结果：**

p1将拥有两个子女p11、p12

**实验结果：**

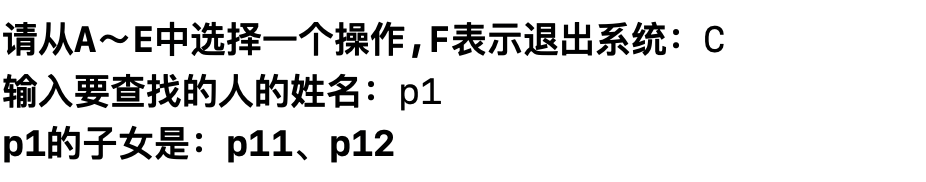
### 4.1.3 搜索成员功能测试

**测试用例：**

搜索p1及其子女情况

**预期结果：**

能在家谱中找到p1，并且输出p1的子女为上一步添加的p11、p12

**实验结果：**

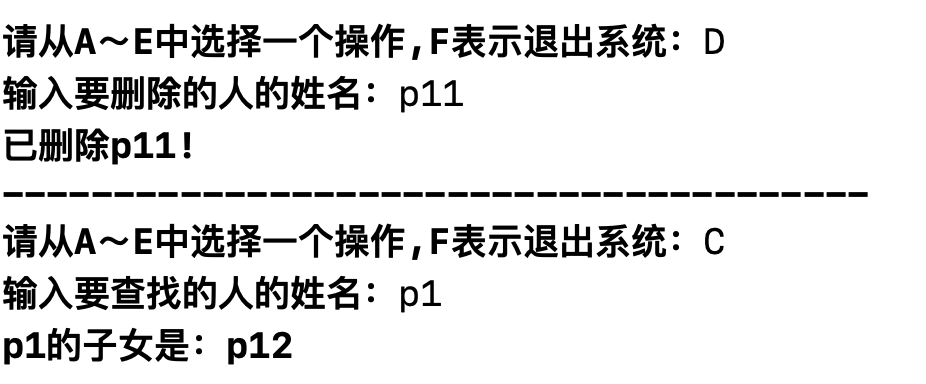
### 4.1.4 删除成员功能测试

**测试用例：**

将删除p1的子女p11

**预期结果：**

p11被删除，p12成为p1的长子

**实验结果：**

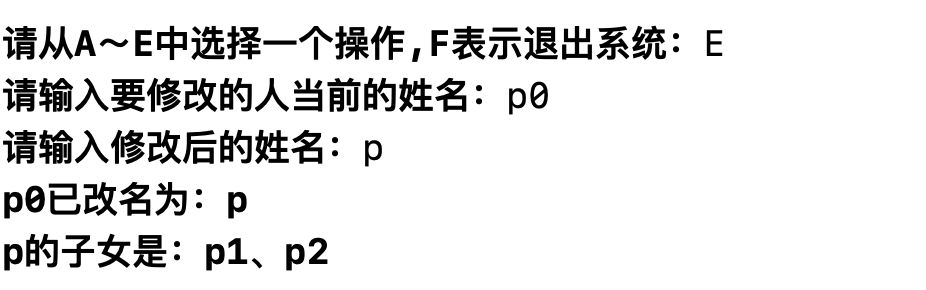
### 4.1.4 修改成员信息功能测试

**测试用例：**

将修改祖先p0的名字，改为p

**预期结果：**

祖先p0的名字被改为p

**实验结果：**

## 4.2 边界测试

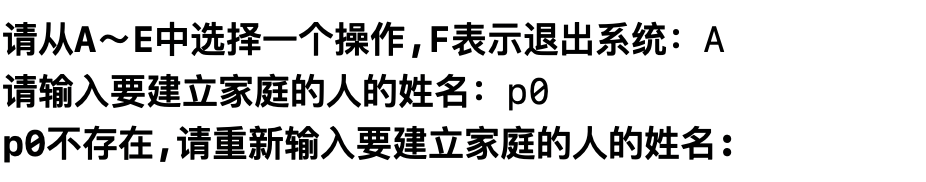
### 4.2.1 对家谱中不存在的成员做操作

**测试用例1：**

尝试为不存在的成员p0建立家庭（已有p p1 p2 p12）

**预期结果1：**

立即给出错误提示，并要求用户重新输入一个姓名，程序运行正常不崩溃。

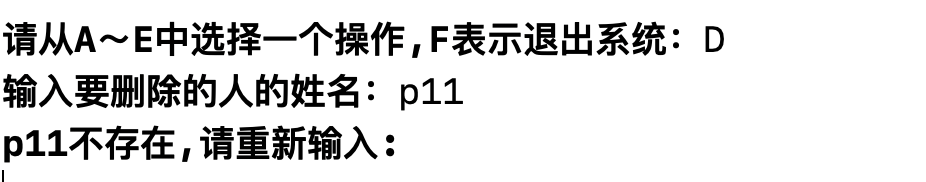
**实验结果1：**

**测试用例2：**

尝试删除不存在的成员p11（已有p p1 p2 p12 p21）

**预期结果2：**

立即给出错误提示，并要求用户重新输入一个姓名，程序运行正常不崩溃。

**实验结果2：**

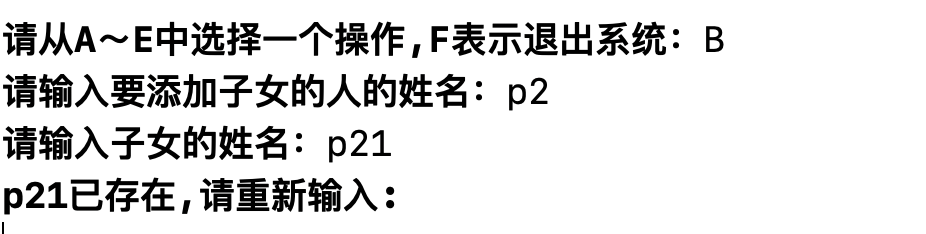
### 4.2.2尝试插入已存在的成员

**测试用例：**

尝试插入已存在的成员p21（已有p p1 p2 p12 p21）

**预期结果：**

立即给出错误提示，并要求用户重新输入一个姓名，程序运行正常不崩溃。

**实验结果：**

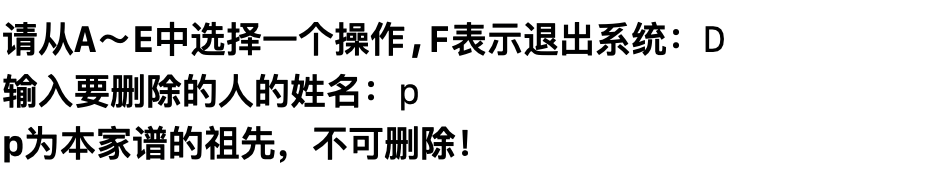
### 4.2.3 尝试删除家族祖先

**测试用例1：**

尝试删除祖先p（已有p p1 p2 p12 p21）

**预期结果2：**

立即给出错误提示，结束删除操作，程序运行正常不崩溃。

**实验结果2：**

# 5 总结

## 5.1 遇到的错误

### 5.1.1 xx函数中

## 5.2 改进与优化

## 5.3 项目心得