同濟大學
TONGJI UNIVERSITY

项目说明文档

家谱管理系统

指导教师：张颖

1751984 王舸飞

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。本项目的目的为对家谱管理进行简单的模拟。

1.2 功能分析

在一个简单的家谱管理模型中，应至少具备创建、完善、添加成员、解散局部家庭、更改姓名和错误提示几个基本功能，本程序通过输入不同的操作码来实现这些不同的功能。

2.设计

2.1 数据结构设计

查找一些现成的家谱资料，可以发现大多数家谱都以树的形式组织，这种处理方式一方面可以清晰的表达出亲子之间的上下级关系，一方面可以体现出兄弟之间的分支关系。

另外由于本系统要实现的主要操作包括了大量的插入和删除，所以链表组成数据结构也是必要的选择。

综上，本次采取了长子——兄弟表示法来存储所需要的数据。

2.2 类结构设计

经典的树结构一般包括两个抽象数据类型（ADT）—树结点类（FamilyMember）与树类（FamilyTree），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。本程序中将树节点类作为树类的友元类，使得家谱树可以访问家庭成员。

2.3 成员与操作设计

树节点类（**FamilyMember**）

受保护的成员：

```
protected:FamilyMember *next_sibling;//代表它的兄弟成员
    FamilyMember *first_child;//代表它的长子成员
    string name;//代表它的名字
};
```

公有操作：

```
public:
    FamilyMember();//init without parameters
    FamilyMember(FamilyMember*,FamilyMember*,string); //init with parameters
    void DisplayChild();//show this member's first generation child
    bool HasChild(){return first_child==NULL?false:true;}//判断是否存在长子
    bool HasSibling(){return next_sibling==NULL?false:true;}//判断是否存在兄弟
```

树类（**FamilyTree**）

受保护的成员：

```
protected:FamilyMember *first;//该家谱树的头指针
```

公有操作：

```
public:
    FamilyTree():first(NULL){}
    void InitTree();//Init the tree, will command to input the ancestor's name
    void PerfectTree();//完善树的函数
    bool Find(string p_name,FamilyMember **current,FamilyMember *root);//在root兄弟子女中查找pname，找到将指向它的指针送给current
    void AddMember();//添加成员的函数
    void DeleteMember();//解散家庭的函数
    void ChangeName();//改名的函数
```

其他用到的函数：

```
void Init(){
    cout<<"**                  家谱管理系统                  **"<<endl;
    cout<<"----------------------------------------"<<endl;
    cout<<"**                请选择要执行的操作              **"<<endl;
    cout<<"**                  A --- 完善家谱                **"<<endl;
    cout<<"**                B --- 添加家庭成员              **"<<endl;
    cout<<"**                C --- 解散局部家庭              **"<<endl;
    cout<<"**              D --- 更改家庭成员姓名            **"<<endl;
    cout<<"**                  E --- 退出程序                **"<<endl;
    cout<<"----------------------------------------"<<endl;
    cout<<"首先建立一个家谱！"<<endl;
}//输出提示信息
```

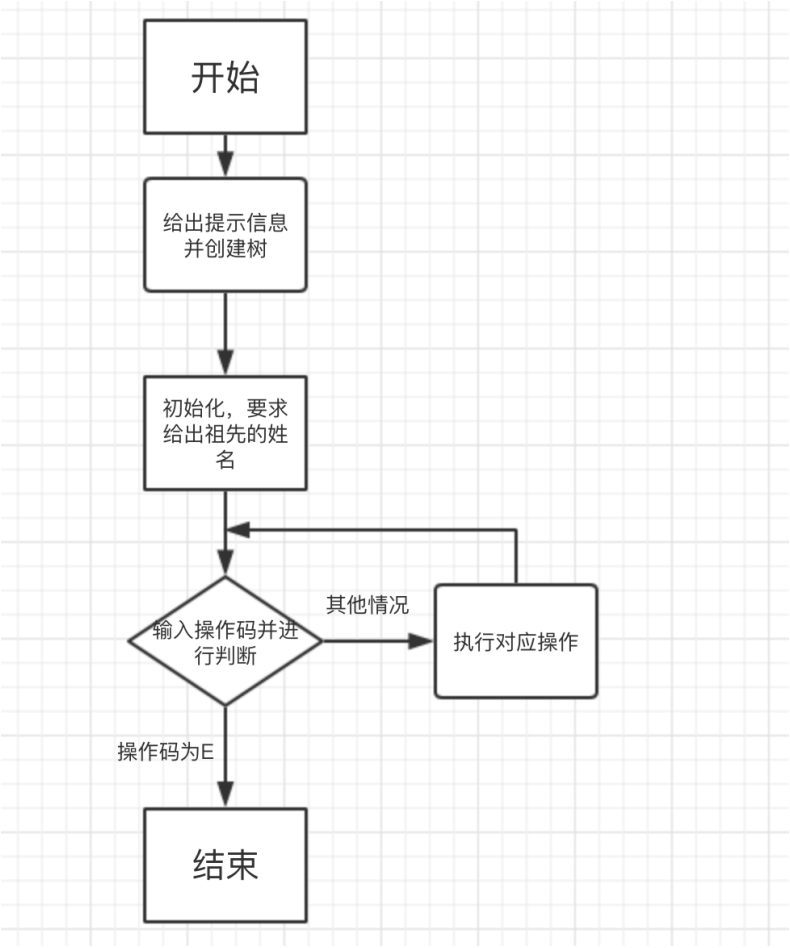
2.4 系统设计

程序运行之后，系统会首先调用初始化函数给出提示信息，之后会立刻创建一个家谱树类对象T1，并调用T1的初始化函数来建立它的祖先；之后，程序会对用户的输入进行选择，并根据用户的输入执行不同的操作。

3.实现

3.1 总体功能的实现

3.1.1 总体功能流程图

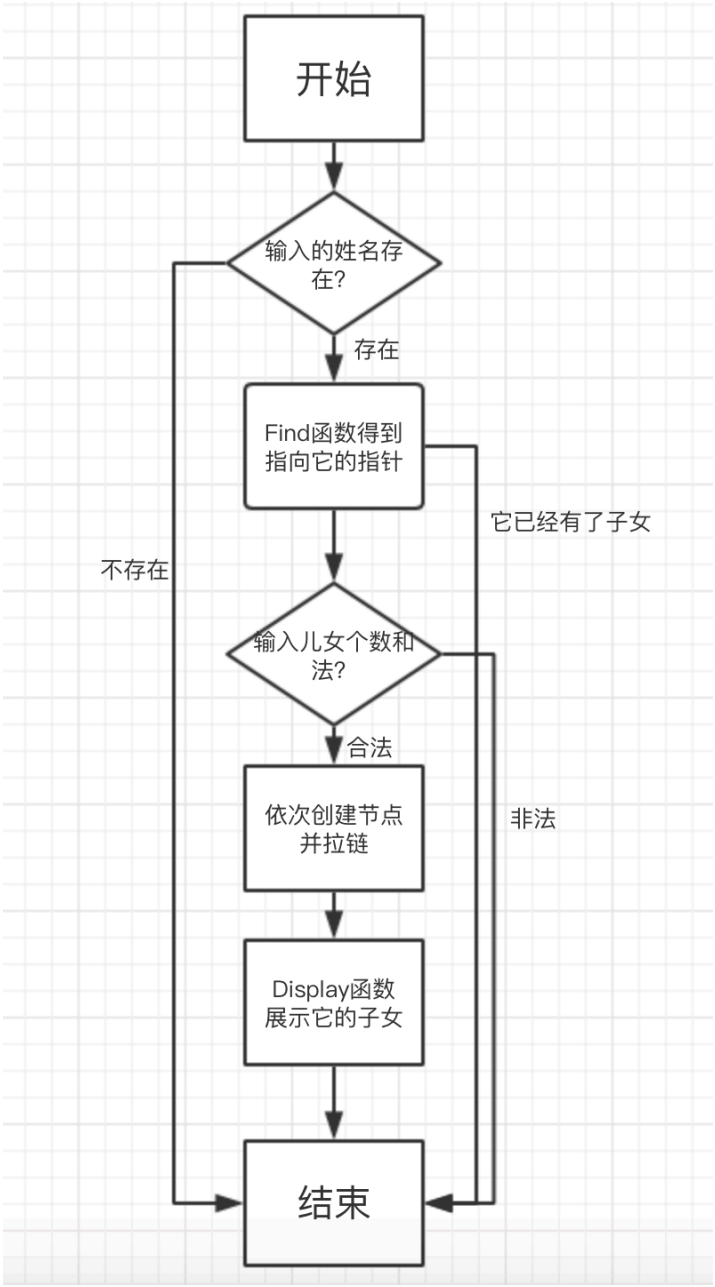


3.1.2 总体功能核心代码

```
Init();
FamilyTree T1;
T1.InitTree();

char Op;
cout<<endl<<"请选择要执行的操作: ";cin>>Op;
while (Op!='E'){


switch(Op){
    case 'A':T1.PerfectTree();break;
    case 'B':T1.AddMember();break;
    case 'C':T1.DeleteMember();break;
    case 'D':T1.ChangeName();break;
    default:cout<<"您输入的操作码不存在，请重新输入!";
}
    cout<<endl<<"请选择要执行的操作: ";cin>>Op;
}
```

3.2 完善功能的实现

3.2.1 完善功能流程图

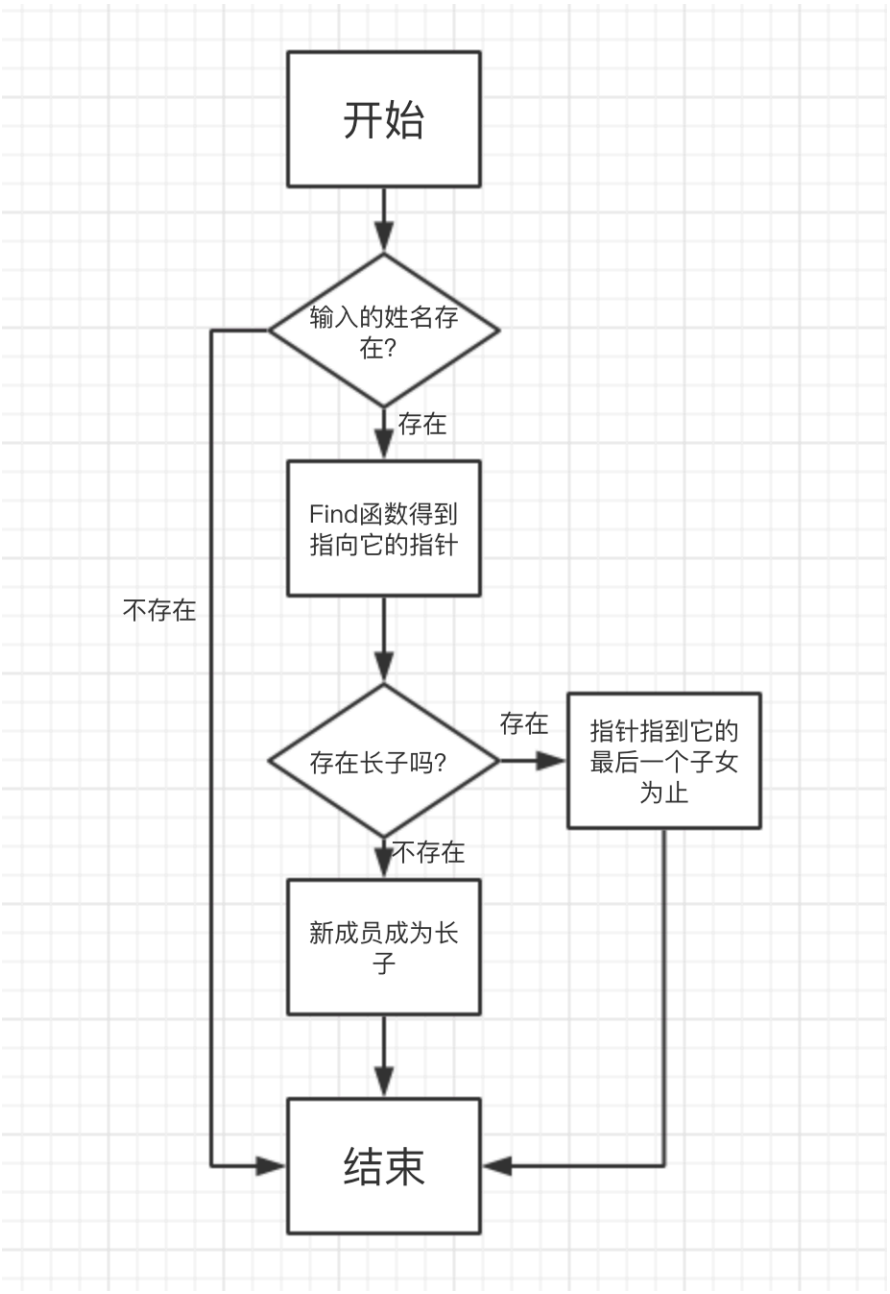


3.2.2 完善功能核心代码

```
void FamilyTree::PerfectTree(){
    string parent_name,child_name;FamilyMember *current=new FamilyMember,*current1;
    cout<<"请输入要建立家庭的人的姓名：";cin>>parent_name;
    if(Find(parent_name,&current,first)&&!current->HasChild()){
        cout<<"请输入"<<parent_name<<"的儿女人数:";
        int number;cin>>number;
        while(number<=0){cout<<"输入的人数非法，请重新输入：";cin>>number;}

        cout<<"请依次输入"<<parent_name<<"的儿女姓名：";
        current1=current;cin>>child_name;
        current1->first_child=new FamilyMember(NULL,NULL,child_name);
        current1=current1->first_child;
        for(int i=0;i<number-1;i++){cin>>child_name;
            current1->next_sibling=new FamilyMember(NULL,NULL,child_name);
            current1=current1->next_sibling;
        }
        current->DisplayChild();
    }

    else {
        if(current->HasChild()){cout<<"当前成员已经有了孩子，无法对他进行完善"<<endl;
            cout<<"如果想对当前成员的子女进行操作，请尝试其他操作";
        }
        else
        cout<<"想要建立家庭的人员不存在"<<endl;}


}
```

3.3 添加成员功能的实现

3.3.1 添加成员功能流程图

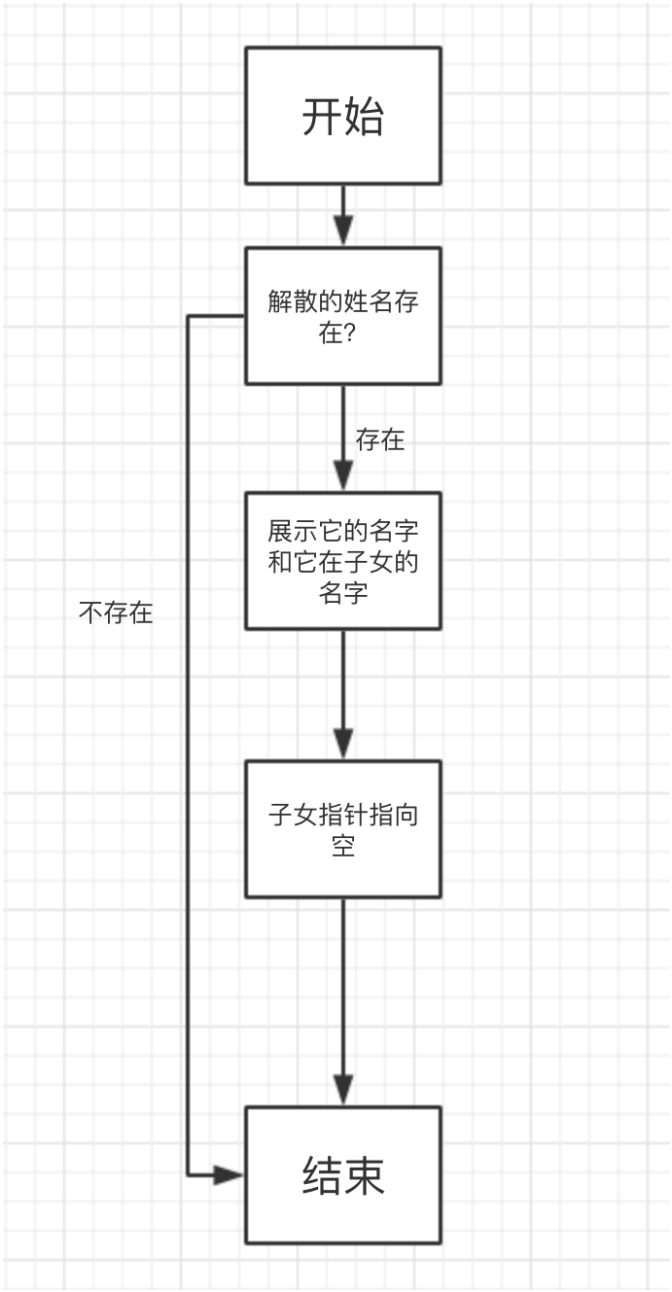


3.3.2 添加成员功能核心代码

```
void FamilyTree::AddMember(){
    cout<<endl<<"请输入要添加儿子（或女儿）的人的姓名：";
    string parent_name,child_name;cin>>parent_name;

    FamilyMember *current=new FamilyMember,*current1;
    if(Find(parent_name,&current,first)){
        cout<<endl<<"请输入"<<parent_name<<"新添加的儿子（或女儿)的姓名：";
        cin>>child_name;
        current1=current;
        if(!current->HasChild())current->first_child=new FamilyMember(NULL,NULL,child_name);

        else {current=current->first_child;
            while(current->HasSibling())current=current->next_sibling;
            current->next_sibling=new FamilyMember(NULL,NULL,child_name);

        }
        current1->DisplayChild();
    }
    else cout<<"没有找到所给人员"<<endl;
}
```

3.4 解散家庭功能的实现

3.4.1 解散家庭功能流程图

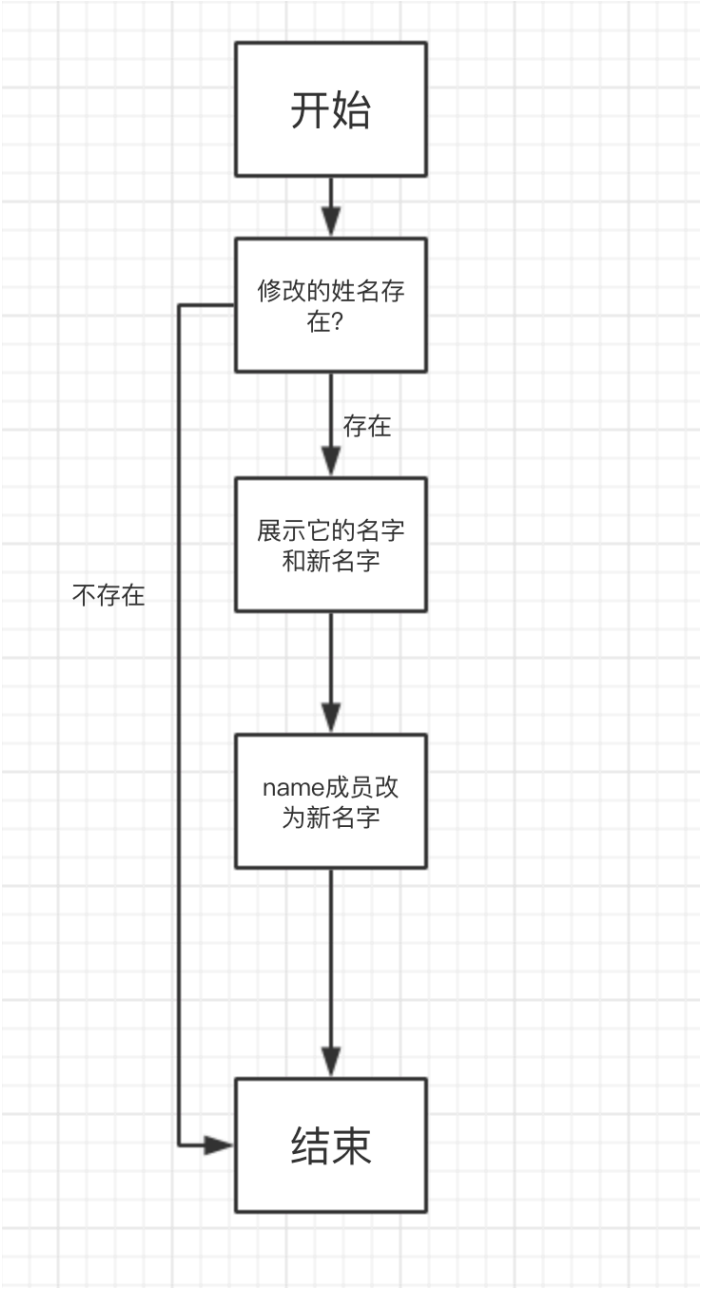


3.4.2 解散家庭功能核心代码

```
void FamilyTree::DeleteMember(){
    cout<<endl<<"请输入要解散家庭的人的姓名:";
    string del_name;cin>>del_name;
    FamilyMember *current=new FamilyMember;
    if(Find(del_name,&current,first)&&current->HasChild()){
        cout<<"要解散家庭的人是: "<<del_name<<endl;
        current->DisplayChild();
        current->first_child=NULL;


    }
    else cout<<"要解散者的姓名不存在或他没有子女"<<endl;

}
```

3.5 改名功能的实现

3.5.1 改名功能流程图



3.5.2 改名功能核心代码

```
void FamilyTree::ChangeName(){
    cout<<"请输入要更改姓名的人的目前姓名：";
    string current_name,new_name;cin>>current_name;
    FamilyMember *current=new FamilyMember;
    if(Find(current_name,&current,first)){
        cout<<"请输入更改后的姓名：";cin>>new_name;
        current->name=new_name;
        cout<<current_name<<"已更名为"<<new_name<<endl;
    }
    else cout<<"输入的当前姓名不存在"<<endl;
}
```

4.测试

4.1 常规测试

4.1.1 初始化家谱

测试用例：P0

实验结果：

```
**                  家谱管理系统                  **
----------------------------------------
**                  请选择要执行的操作            **
**                    A --- 完善家谱              **
**                  B --- 添加家庭成员            **
**                  C --- 解散局部家庭            **
**                D --- 更改家庭成员姓名          **
**                    E --- 退出程序              **
----------------------------------------
首先建立一个家谱!
请输入祖先的姓名：P0
此家谱的祖先是：P0
```

4.1.2 完善家庭

测试用例：P1 P2

实验结果：

请选择要执行的操作： A
请输入要建立家庭的人的姓名： P0
请输入P0的儿女人数：2
请依次输入P0的儿女姓名： P1 P2
P0的第一代子孙是：P1 P2

4.1.3 完善下一代家庭

测试用例：P11 P12 P13

实验结果：

请选择要执行的操作： A
请输入要建立家庭的人的姓名： P1
请输入P1的儿女人数：3
请依次输入P1的儿女姓名： P11 P12 P13
P1的第一代子孙是：P11 P12 P13

4.1.4 添加子女

测试用例：p21

实验结果：

请选择要执行的操作： B

请输入要添加儿子（或女儿）的人的姓名： P2

请输入P2新添加的儿子（或女儿）的姓名： P21
P2的第一代子孙是：P21

4.1.5 解散家庭

测试用例：P2

实验结果：

请输入要解散家庭的人的姓名：P2
要解散家庭的人是：P2
P2的第一代子孙是：P21

由于解散成功，可以继续对P2添加子女

请选择要执行的操作： A
请输入要建立家庭的人的姓名： P2
请输入P2的儿女人数：2
请依次输入P2的儿女姓名： P22 P23
P2的第一代子孙是：P22 P23

4.1.6 改名

测试用例：P13->P14

实验结果：

请选择要执行的操作： D
请输入要更改姓名的人的目前姓名： P13
请输入更改后的姓名： P14
P13已更名为P14

4.2 错误测试

4.2.1 不存在的家庭成员

实验结果：

```
请选择要执行的操作：A
请输入要建立家庭的人的姓名：P3
想要建立家庭的人员不存在
```

4.2.2 不能完善的成员

```
请选择要执行的操作：A
请输入要建立家庭的人的姓名：P2
当前成员已经有了孩子，无法对他进行完善
如果想对当前成员的子女进行操作，请尝试其他操作
请选择要执行的操作：
```

4.2.3 错误的人数输入

```
请输入P0新添加的儿子（或女儿）的姓名：P3
P0的第一代子孙是：P1 P2 P3

请选择要执行的操作：A
请输入要建立家庭的人的姓名：P3
请输入P3的儿女人数：-1
输入的人数非法，请重新输入：
```