数据结构课程设计

项目说明文档

**家谱管理系统**



同济大学

Tongji University

姓名： 林觉凯

学号： 2253744

指导老师： 张颖

学院专业： 软件学院 软件工程

**目录**

**1.项目分析-------------------------------------------------------------------------3**

**1.1 项目背景分析-------------------------------------------------------------------------3**

**1.2项目功能分析--------------------------------------------------------------------------3**

**1.2.1项目功能要求--------------------------------------------------------------------------------3**

**1.2.2项目输入要求--------------------------------------------------------------------------------3**

**1.2.3项目输出要求--------------------------------------------------------------------------------3**

**1.2.4项目示例--------------------------------------------------------------------------------------3**

**2.项目设计-------------------------------------------------------------------------4**

**2.1 数据结构设计-------------------------------------------------------------------------4**

**2.2类设计-----------------------------------------------------------------------------------4**

**2.2.1家谱树结点类(Member)-------------------------------------------------------------------4**

**2.2.2家谱树类(FamilyTree)---------------------------------------------------------------------4**

**2.3成员与操作设计-----------------------------------------------------------------------5**

**2.4系统流程设计--------------------------------------------------------------------------5**

**3.项目实现-------------------------------------------------------------------------6**

**3.1 建立家谱的实现----------------------------------------------------------------------6**

**3.2 完善家谱的实现----------------------------------------------------------------------7**

**3.3 添加成员的实现----------------------------------------------------------------------7**

**3.4 解散局部家庭 ------------------------------------------------------------------------8**

**3.5 更改家庭成员姓名的实现----------------------------------------------------------9**

**4.项目测试------------------------------------------------------------------------10**

**4.1 项目功能测试------------------------------------------------------------------------10**

**4.2 代码的健壮性测试------------------------------------------------------------------11**

**4.2.1关于输入错误数据类型的健壮性测试------------------------------------------------11**

**4.2.2 关于输入正确但是无法满足操作的数据的健壮性测试--------------------------13**

**5.项目的心得与体会------------------------------------------------------------14**

**1.项目分析**

**1.1 项目背景分析**

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 项目功能分析**

**1.2.1项目功能要求**

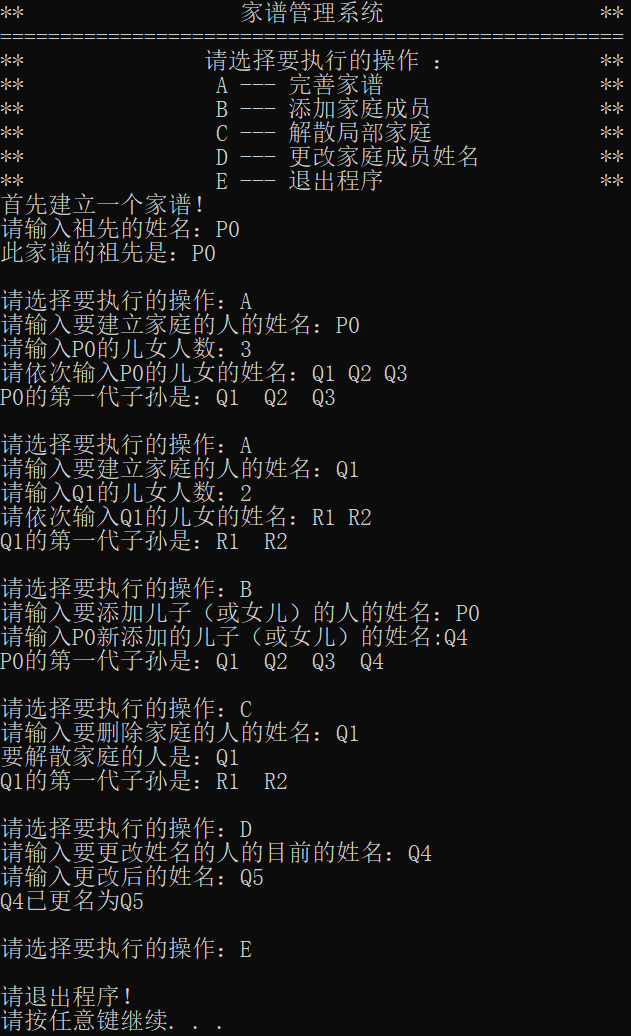
本项目的实质是完成对家谱成员信息的建立，查找，插入，修改，删除等功能，可以首先定义家族成员数据结构，然后将每个功能作为一个成员函数来完成对数据的操作，最后完成主函数以验证各个函数功能并得到运行结果。

**1.2.2项目输入要求**

用户输入所要选择的操作选项，同时根据选项给出的提示输入相应的信息。

**1.2.3项目输出要求**

按照用户的要求输出家谱建立，查找，插入，修改，删除后的信息内容。

**1.2.4项目示例**

**2.项目设计**

**2.1 数据结构设计**

此次项目需要实现一个家谱树的存储和表示，也即一个多叉树的存储和表示。在本程序中，我使用左孩子、右兄弟表示法来存储家谱树：即采用类似于二叉树的存储结构存储多叉树。在这其中，每一个节点的左节点都是它的长子，右节点则是它的兄弟。以此来达到多叉树的存储。

**2.2类设计**

为了实现此次家谱树，我设计了两个类：家谱树结点类和家谱树类，而两个类之间的耦合关系可以采用嵌套、继承等多种关系。本程序也构造了这两个基本的家谱树结点和家谱树类来进行相应的操作。

**2.2.1家谱树结点类**(Member)

\*每一个家谱树结点储存着这个结点的姓名和它的左孩子指针和右兄弟指针。

class Member

{

public:

string name;

Member\* firstChild;

Member\* nextSibling;

Member(string str = "", Member\* point = NULL)

{

name = str;

firstChild = nextSibling = point;

}

};

**2.2.2家谱树类**(FamilyTree)

class FamilyTree

{

public:

FamilyTree() { Ancestor = NULL; };

void Establish();

Member\* Search(Member\* current, string &name);

void Show\_Child(Member\* current);

void Perfect\_Operation();

void Add\_Operation();

void Dissolve\_Operation();

void Change\_Operation();

private:

Member\* Ancestor;

};

**2.3成员与操作设计**

家谱树结点类的公有成员为姓名、左孩子指针和右兄弟指针；家谱树类的私有成员为祖先指针，公有操作为建立树、搜索操作、展示操作、完善树选项、添加选项、解散选项和更改选项。

**2.4系统流程设计**

系统操作的整体流程大致如下：

**程序开始运行**

**建立家谱**

**（输入祖先的信息）**

**选择操作**

**（完善、添加、解散、更改和退出）**

**退出程序**

**执行操作**

**显示操作后数据**

**3.项目实现**

**3.1 建立家谱的实现**

void FamilyTree::Establish()

{

cout << "首先建立一个家谱！" << endl;

cout << "请输入祖先的姓名：";

string Ancestorname;

Ancestorname = Get\_String(Max\_namelength, Min\_namelength, "输入错误，请重新输入！");

Ancestor = new Member;

if (Ancestor == NULL)

{

cout << "家谱创建失败！" << endl;

system("pause");

return;

}

Ancestor->name = Ancestorname;

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

}

建立家谱通过的是初始化根结点的函数。首先通过Get\_String函数获得一个合理的祖先名字，再将其赋值给根结点，从而完成家谱的创建的操作。

**3.2 完善家谱的实现**

void FamilyTree::Perfect\_Operation()

{

cout << "请输入要建立家庭的人的姓名：";

string name = Get\_String(Max\_namelength, Min\_namelength, "输入错误，请重新输入！");

int Child\_number;

Member\* current = Search(Ancestor, name);

Member\* Save\_current = current;

if (current == NULL) //如果current找不到，则说明不存在此人

{

cout << "查无此人！请重新操作！" << endl;

return;

}

cout << "请输入" << name << "的儿女人数：";

Child\_number = Get\_Int("输入错误，请重新输入！");

cout << "请依次输入" << name << "的儿女的姓名：";

string tempname;

for (int i = 0; i < Child\_number; i++)

{

if (i == 0) //第一个输入的是左孩子（长子）

{

Member\* Newchild = new Member;

tempname = Get\_String(Max\_namelength, Min\_namelength, "输入错误，请重新输入！");

Newchild->name = tempname;

current->firstChild = Newchild;

current = Newchild;

}

else

{

Member\* Newchild = new Member; //之后输入的是右兄弟

tempname = Get\_String(Max\_namelength, Min\_namelength, "输入错误，请重新输入！");

Newchild->name = tempname;

current->nextSibling = Newchild;

current = Newchild;

}

}

cout << name << "的第一代子孙是：";

Show\_Child(Save\_current);

cout << endl;

}

在完善家谱的操作过程中，我们首先要通过Get\_String函数和Search函数获得所需建立家庭的人的相应的结点位置，在这过程中做了代码的健壮性判断；之后要通过输入的顺序来选择将孩子插入到左孩子还是右兄弟。最后通过Show\_Child函数将此人第一代的子孙显示出来。

**3.3 添加成员的实现**

void FamilyTree::Add\_Operation()

{

cout << "请输入要添加儿子（或女儿）的人的姓名：";

string name = Get\_String(Max\_namelength, Min\_namelength, "输入错误，请重新输入！");

Member\* current = Search(Ancestor, name);

Member\* Save\_current = current;

if (current == NULL) //如果current找不到，则说明不存在此人

{

cout << "查无此人！请重新操作！" << endl;

return;

}

cout << "请输入" << name << "新添加的儿子（或女儿）的姓名:";

string Add\_childname = Get\_String(Max\_namelength, Min\_namelength, "输入错误，请重新输入！");

if (current->firstChild == NULL) //如果不存在长子，则添加的孩子为左孩子

{

Member\* Newchild = new Member;

Newchild->name = Add\_childname;

current->firstChild = Newchild;

}

else

{ //如果存在长子，则添加的孩子为右兄弟

current = current->firstChild;

while (current->nextSibling != NULL)

current = current->nextSibling;

Member\* Newchild = new Member;

Newchild->name = Add\_childname;

current->nextSibling = Newchild;

}

cout << name << "的第一代子孙是：";

Show\_Child(Save\_current);

cout << endl;

}

在添加成员的操作过程中，我们首先要通过Get\_String函数和Search函数获得所需添加孩子的人的相应的结点位置，在这过程中做了代码的健壮性判断；之后需要判断该结点是否具有左孩子，如果有的话，将添加的结点信息插入至它的左孩子的最右边的右兄弟上；如果没有的话，直接将添加的结点信息插入至它的左孩子即可完成添加操作。

**3.4 解散局部家庭的实现**

void FamilyTree::Dissolve\_Operation()

{

cout << "请输入要删除家庭的人的姓名：";

string name = Get\_String(Max\_namelength, Min\_namelength, "输入错误，请重新输入！");

Member\* current = Search(Ancestor, name);

if (current == NULL) //如果current找不到，则说明不存在此人

{

cout << "查无此人！请重新操作！" << endl;

return;

}

if (current->firstChild == NULL) //如果此人无左孩子，则说明还没有家庭

{

cout << "要解散家庭的人" << name << "没有家庭，请重新操作！" << endl;

return;

}

cout << "要解散家庭的人是：" << name << endl;

cout << name << "的第一代子孙是：";

Show\_Child(current);

current->firstChild=NULL;

cout << endl;

}

在解散局部家庭的操作过程中，我们首先要通过Get\_String函数和Search函数获得所需解散局部家庭的人的相应的结点位置，在这过程中做了代码的健壮性判断；之后通过Show\_Child函数将所解散家庭的信息输出，再将这个结点的孩子家庭全部解散。

**3.5 更改成员姓名的实现**

void FamilyTree::Change\_Operation()

{

cout << "请输入要更改姓名的人的目前的姓名：";

string Origin\_name = Get\_String(Max\_namelength, Min\_namelength, "输入错误，请重新输入！");

Member\* current = Search(Ancestor, Origin\_name);

if (current == NULL) //如果current找不到，则说明不存在此人

{

cout << "查无此人！请重新操作！" << endl;

return;

}

else

{

cout << "请输入更改后的姓名：";

cin >> current->name;

cout << Origin\_name << "已更名为" << current->name << endl;

}

}

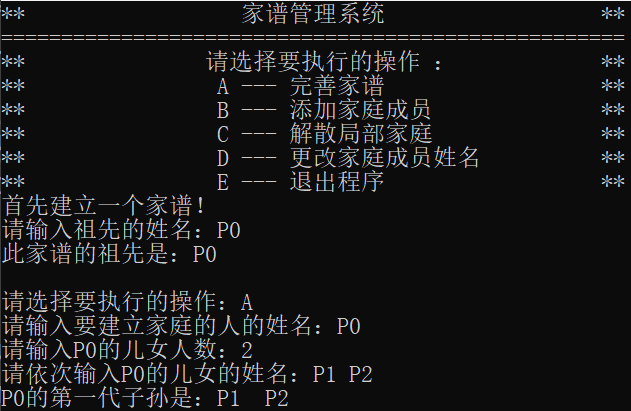
在更改成员姓名的操作过程中，我们首先要通过Get\_String函数和Search函数获得所需更改姓名的人的相应的结点位置，在这过程中做了代码的健壮性判断；之后输入要更改的名字将其输入，程序运行并输出相应结果。

**4.项目测试**

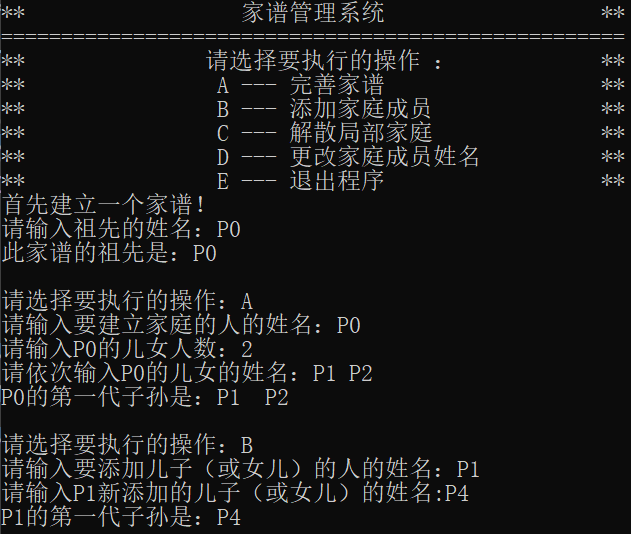
**4.1 项目功能测试**

以下是四个菜单功能的测试：

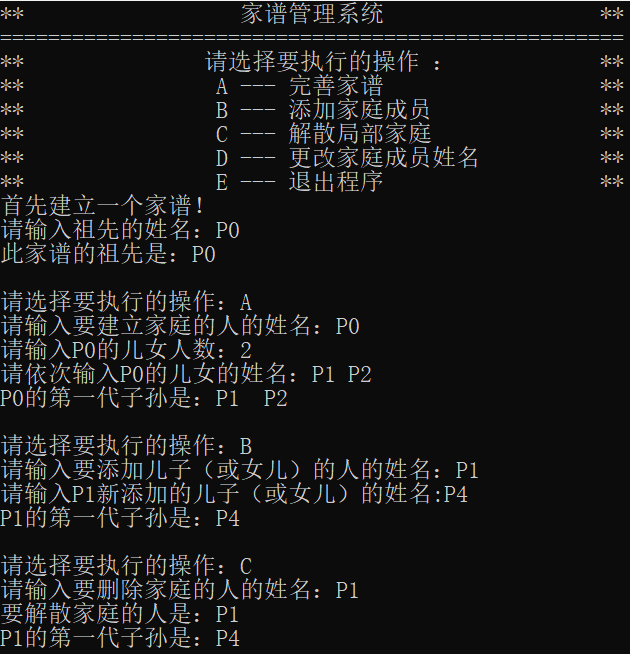
A---完善家谱



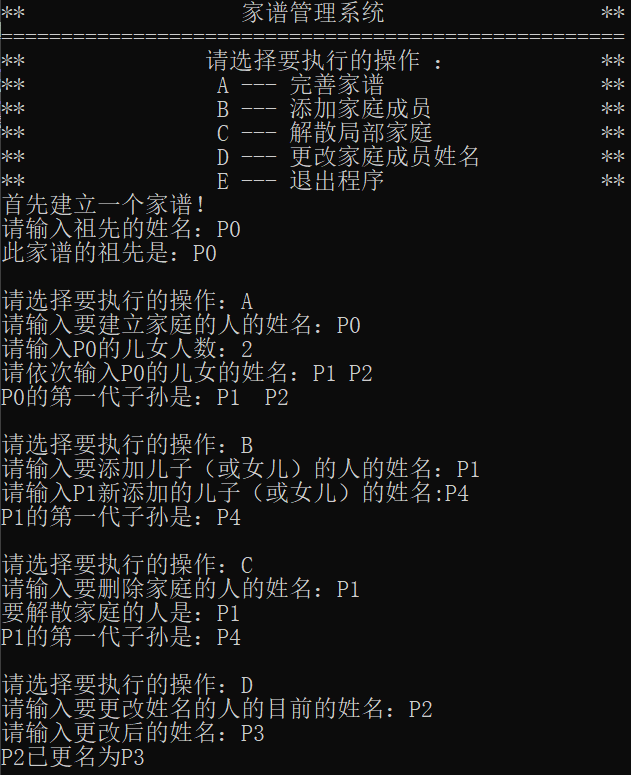
B---添加家庭成员



C--解散局部家庭



1. --更改家庭成员姓名



**4.2 代码健壮性测试**

**4.2.1 关于输入错误数据类型的健壮性测试**

在用户输入操作选择的字符时，会进行数据类型的判断以及输入的数据是否在合法范围的判断，还有在输入姓名时不符合实际（姓名过长、过短等等），都会进行一定的判断。这由三个函数实现该操作：

//获得一个在合理范围的整数函数，并进行错误处理

int Get\_Int(string errortips)

{

int retint;

while (1)

{

cin >> retint;

if (cin.good() == 0 || retint < 1)

{

cin.clear();

cin.ignore(1024, '\n');

cout << errortips;

continue;

}

break;

}

return retint;

}

//获得一个在规定范围的字符函数，并进行错误处理

char Get\_Char(string errortips)

{

char retchar;

cout << endl << "请选择要执行的操作：";

while (1)

{

cin >> retchar;

if (cin.good() == 0 || retchar < 'A' || retchar>'E')

{

cin.clear();

cin.ignore(1024, '\n');

cout << errortips;

continue;

}

break;

}

return retchar;

}

//获得一个在规定范围内的string函数，并进行错误处理

string Get\_String(unsigned int lengthmax, unsigned int lengthmin, string errortips)

{

string retstring;

while (1)

{

cin >> retstring;

if (cin.good() == 0 || retstring.size() < lengthmin || retstring.size() > lengthmax)

{

cin.clear();

cin.ignore(1024, '\n');

cout << errortips;

continue;

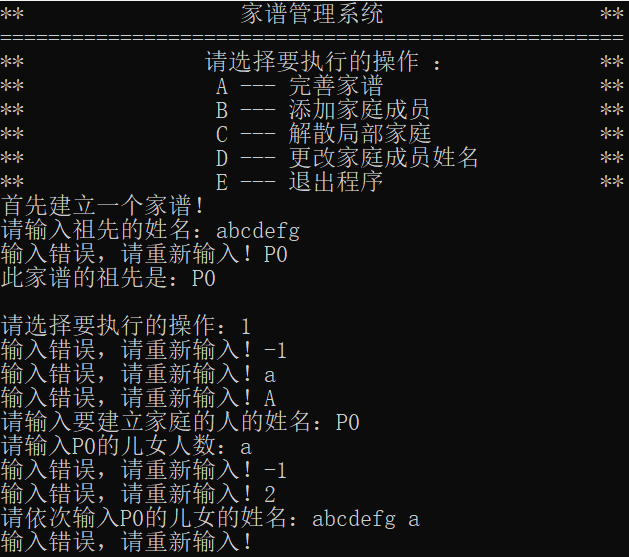
}

break;

}

return retstring;

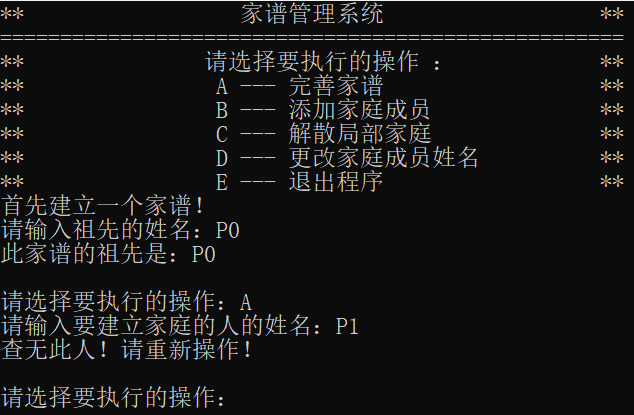
}

****

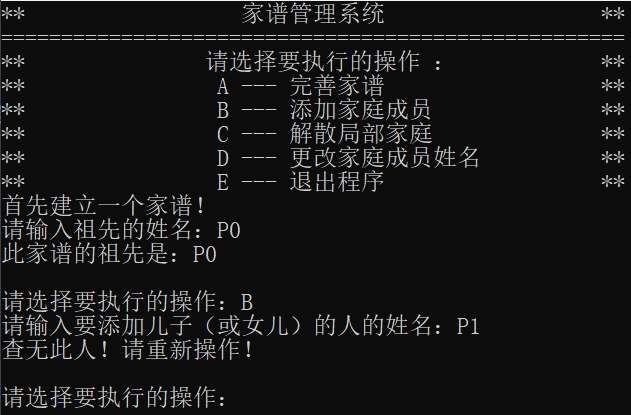
**4.2.2 关于输入正确但是无法满足操作的数据的健壮性测试**

在进行相应操作的时候会有此类代码的健壮性测试。

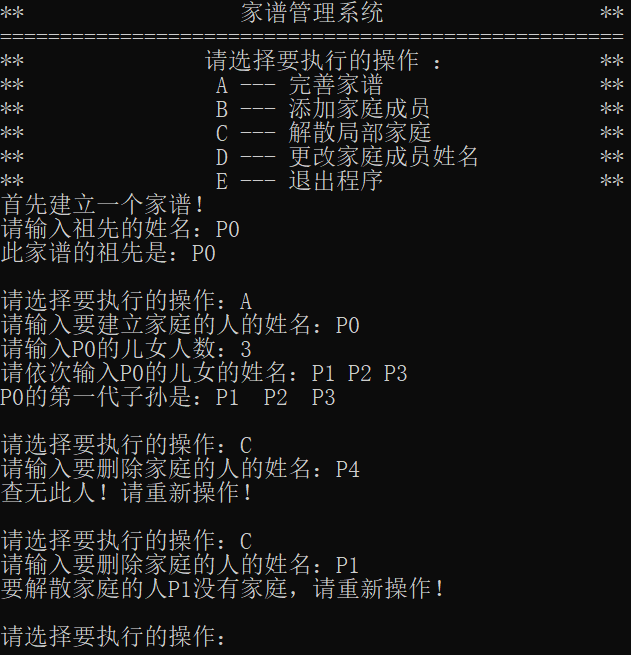
比如在完善家谱时，输入建立家庭的人，但是此人不存在，会给出相应提示；



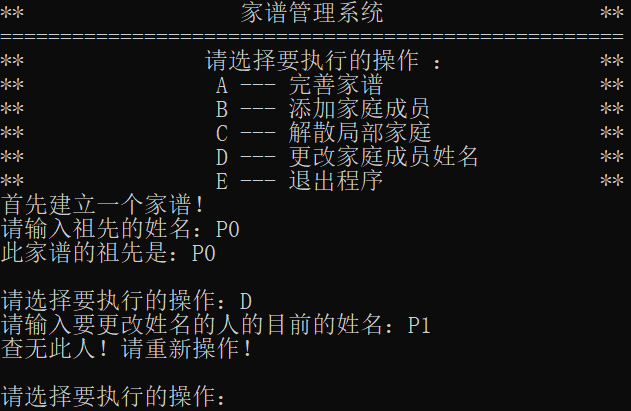
在添加家庭成员时，输入要添加子女的人，但是此人并不存在，则会给出相应的提示；



在解散局部家庭时，输入了要解散家庭的人，如果此人不存在会给出提示；如果这个人没有家庭，也会给出提示；



在更改家庭成员姓名时，输入了要更改家庭成员姓名的人，但是此人不存在会给出相应的提示。



**5.项目心得与体会**

在我看来，树是一种比较复杂但是非常有趣的数据结构。本次项目的家谱需要实现多叉树的储存信息，这里将其类似二叉树转换为“左孩子右兄弟”的树来实现其存储结构。通过这次项目的练习，首先我加深了对C++类的编写程序，更加熟练地运用面向对象的编程方法实现问题；其次，我对树（尤其是二叉树和类似于二叉树的结构）有了更加深入的理解；最重要的是，在树的运用中，经常会用到递归的方法来实现相应操作（找结点、找位置、查找是否存在等等），通过此次练习，我也对树中的递归操作更加熟练地运用。在之后的学习生活中，在此项目的基础上，通过继续练习把对树的理解更加深。