家谱管理系统

作 者 姓 名： 李翠琪

学 号： 1751022

指 导 教 师： 张颖 \_\_

学院、 专业： 软件工程 \_\_

同济大学

Tongji University

目录

[1 分析 1](#_Toc24987825)

[1.1 项目简介 1](#_Toc24987826)

[1.2 功能要求 1](#_Toc24987827)

[2 设计 1](#_Toc24987828)

[2.1 Node类设计 1](#_Toc24987829)

[2.2 LinkedList类设计 2](#_Toc24987830)

[2.3 主程序设计 2](#_Toc24987831)

[3 实现 2](#_Toc24987832)

[3.1 Node类实现 2](#_Toc24987833)

[3.2 LinkedList类实现 3](#_Toc24987834)

[3.2.1 findFromTree(Node<T> \* x,T familyName) 3](#_Toc24987835)

[3.2.2 add(T familyRoot) 4](#_Toc24987836)

[3.2.3 add(T familyName,T childName) 4](#_Toc24987837)

[3.2.4 deleteFromTree(T fatherName) 4](#_Toc24987838)

[3.2.5 update(T fatherName,T updateName) 5](#_Toc24987839)

[3.2.6 show(T familyName) 5](#_Toc24987840)

[3.2.7 findFromTree(T familyName) 6](#_Toc24987841)

[3.3 主函数的实现 6](#_Toc24987842)

[3.3.1 总体系统描述 6](#_Toc24987843)

[3.3.2 总体系统核心代码 7](#_Toc24987844)

[4 测试 10](#_Toc24987845)

[4.1 功能测试 10](#_Toc24987846)

[4.1.1 建立功能测试 10](#_Toc24987847)

[4.1.2 插入功能测试 11](#_Toc24987848)

[4.1.3 删除功能测试 12](#_Toc24987849)

[4.1.4 查找功能测试 14](#_Toc24987850)

[4.1.5 修改功能测试 14](#_Toc24987851)

[4.2 边界测试 15](#_Toc24987852)

[4.2.1 要建立家庭的人不在家谱中 15](#_Toc24987853)

[4.2.2 要增加子女的人不在家谱中 15](#_Toc24987854)

[4.2.3 要解散家庭的人不在家谱中 16](#_Toc24987855)

[4.2.4 要更改姓名的人不在家谱中 16](#_Toc24987856)

[4.2.5 要查询的人不在家谱中 16](#_Toc24987857)

# 1 分析

## 1.1 项目简介

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 功能要求

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 Node类设计

Node类是一个模板类,存储一个T类型的值和两个指针,分别是代表指向兄弟节点的next指针和指向自己子节点的link指针。

## 2.2 LinkedList类设计

首先，LinkedList依旧是模板类，它存储了两个成员变量，分别是代表家谱祖先的root节点，以及将来用于代表所操作的人物的curFind。主要有六个public函数，分别是：

add(T familyRoot) 传入的是祖先名字，创建家谱中第一个元素。

void add(T familyName,T childName) 给家谱中的某个人添加子女。

void deleteFromTree(T fatherName) 删除家谱中的某个人，连带子女一起从家谱中删除。

void update(T fatherName,T updateName)更改家谱中某个人的姓名

void show(T familyName)展示家谱中某个人的子女

bool findFromTree(T familyName)找出在家谱中这个人是否存在

还有一个private函数：

void findFromTree(Node<T> \* x,T familyName) 从家谱中找到某个人，并将它存在curFind中，供其他函数使用

## 2.3 主程序设计

首先需要打印出提示信息，让用户知道如何建立家谱树。接着，首先让用户输入家谱中祖先的姓名并展示。之后让用户通过指令自由选择对家谱的操作，完成增删改查等对家谱的基本操作，并且操作完能妥善退出。

# 3 实现

## 3.1 Node类实现

其实就是在以前的Node类基础上添加了一个指向自己子类的指针link。

template <class T> class Node  
{  
public:  
 T value;  
 Node<T> \*next;  
 Node<T> \*link;  
  
 Node(T value) : value(value), next(nullptr), link(nullptr) {}  
};

## 3.2 LinkedList类实现

### 3.2.1 findFromTree(Node<T> \* x,T familyName)

因为该私有函数是其他函数的基础，所以先说它的实现。

这是一个递归类型的函数，首先，其他函数调用它的时候，基本都会传入家谱的根节点root和要查找人的姓名familyName。然后，该函数的终止条件有两个，一个是在家谱中找到了那个人，那么令curFind指向这个人，并且返回。另一个是整个家谱树中都找不到这个人，即直接返回。（因为每一次调用这个函数时curFind都重置为nullptr，所以这个函数之后，如果没有找到，相当于就是curFind为nullptr，所以之后有一些也可以用这个作为判断依据）。

而该函数的查找流程是，对于传入的x，先查询它本身是否为这个人，然后再它所有子节点中递归查找，最后再在他之后的兄弟节点中递归查找。

void findFromTree(Node<T> \* x,T familyName)  
{  
 if(x== nullptr)  
 {  
 return;  
 }  
 if(x->value==familyName)  
 {  
 curFind = x;  
 return;  
 }  
 findFromTree(x->link,familyName);  
 for (Node<T> \* temp=x->next;temp!= nullptr;temp=temp->next)  
 {  
 findFromTree(temp,familyName);  
 }  
}

### 3.2.2 add(T familyRoot)

该函数往家谱中添加第一个节点，即祖先。

void add(T familyRoot)  
{  
 root = new Node<T>(familyRoot);  
}

### 3.2.3 add(T familyName,T childName)

先把curFind置为nullptr，然后调用之前所说的findFromTree函数，把curFind节点指向familyName这个人。如果curFind为nullptr，或者值为-1,(删除函数中会说明)说明查无此人，提示并返回。否则，往这人的link节点（即代表他的子女）后面插入一个名为childName的节点，代表他的孩子。（这里要分一下，如果本来curFind没子节点，就直接增加一个。如果本来有，则要走到curFind子节点的末尾，再在末尾加入该子女childName）。

void add(T familyName,T childName)  
{  
 curFind = nullptr;//每次都要重新开始  
 findFromTree(root,familyName);//找到了父节点  
 if(curFind== nullptr || curFind->value=="-1")  
 {  
 return;  
 }  
 Node<T> \*curChild = curFind->link;//父节点的子节点,不一定存在  
 if(curChild== nullptr)  
 {  
 curChild = new Node<T>(childName);  
 curFind->link = curChild;  
 }  
 else  
 {  
 //平行节点  
 Node<T> \*x = curChild;  
 while(x->next!= nullptr)  
 {  
 x = x->next;  
 }  
 x->next = new Node<T>(childName);  
 }  
}

### 3.2.4 deleteFromTree(T fatherName)

该函数作用是将给定人物和他子女都删除。

首先还是将curFind置为找到的那个人，如果没有找到就提示返回。然后将这个人的值置为“-1”（由于没有前驱指针，所以找不到该人的上一个兄弟，所以采用这种延时删除的操作，当查询或者是其他的，看到了“-1”，就代表了这个人不存在），代表该人已经被删除了。并且将指向他子女的节点置为nullptr，并且将他的子女一一删除。

void deleteFromTree(T fatherName)  
{  
 curFind = nullptr;//每次都要重新开始  
 findFromTree(root,fatherName);  
 if(curFind== nullptr || curFind->value=="-1")  
 {  
 return;  
 }  
 curFind->value = "-1";//延时实现删除操作  
 Node<T> \*x=curFind->link;  
 curFind->link = nullptr;//子代没了  
  
}

### 3.2.5 update(T fatherName,T updateName)

该函数主要更新家谱中那个人的姓名。和前面的操作差不多，都是先查找到那个人，找不到就提示并且返回，找到就把找到那个人的名字改了。

void update(T fatherName,T updateName)  
{  
 curFind = nullptr;//每次都要重新开始  
 findFromTree(root,fatherName);  
 if(curFind== nullptr || curFind->value=="-1")  
 {  
 return;  
 }  
 Node<T> \*cur = curFind;  
 cur->value = updateName;  
}

### 3.2.6 show(T familyName)

同样是先找到那个人，找不到就提示并且返回，然后对他的所有子女节点进行遍历，如果他的子女节点值不是-1的话，就打印他的值，这样之前所说的延时删除才有了意义。

同时，用一个变量N记录子女的个数，如果子女个数为0，就打印一条提示信息null，代表该节点没有子女，以免用户在操作时产生困惑。

void show(T familyName)  
 {  
 int N = 0;//子女个数,为了在为空的时候输出一条信息  
 curFind = nullptr;//每次都要重新开始  
 findFromTree(root, familyName);  
 if(curFind== nullptr || curFind->value=="-1")  
 {  
 return;  
 }  
 for (Node<T> \* temp=curFind->link;temp!= nullptr;temp=temp->next)  
 {  
 if(temp->value!="-1")  
 {  
 ++N;  
 cout << temp->value << " ";  
 }  
 }  
 if(N==0)  
 {  
 cout << "null"<<endl;  
 }  
 }

### 3.2.7 findFromTree(T familyName)

该函数是为了查看要查找的人在家谱中是否存在，实现很简单，但也很有用。

通过调用findFromTree函数，如果找到了就返回true，否则返回false。

bool findFromTree(T familyName)  
{  
 curFind = nullptr;  
 findFromTree(root, familyName);  
 if(curFind== nullptr || curFind->value=="-1")  
 return false;  
 else  
 return true;  
}

## 3.3 主函数的实现

### 3.3.1 总体系统描述

先输出一些用户提示信息。然后让用户输入家谱中的祖先的名字。随后进入一个直到用户主动退出才结束的while循环，让用户选择相应的操作。当用户选择相应的操作后，给出相应的提示，让用户进行相应的输入，程序调用LinkedList类的相应增删改查函数操作，并且把结果输出给用户。因为每个操作大体类似，所以用删除操作举例：

当用户选择C时，代表要解散家谱中的某个家庭，这时给出提示信息让用户输入要解散的人的名字，然后通过LinkedList类的findFromTree函数判断输入的名字是否存在在家谱中，不存在就打印提示信息并且进入下一个循环选择。如果存在首先通过show函数展示这个人的第一代子孙，并且用deleteFromTree函数输出该人和他的子女。

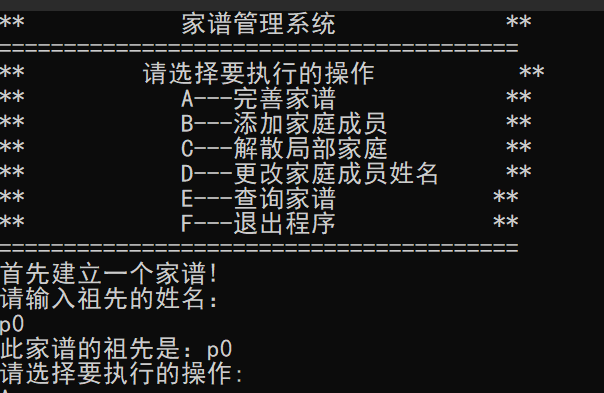
### 3.3.2 总体系统核心代码

cout<<"请选择要执行的操作:"<<endl;  
 string ops;  
 cin >> ops;  
 while(true)  
 {  
 if(ops=="A")  
 {  
 string fatherName;  
 int childCount;  
 cout<<"请输入要建立家庭的人的姓名:";  
 cin >> fatherName;  
if (families.findFromTree(fatherName))//这个人存在了才继续给他加入子女  
{  
 cout << "请输入" << fatherName << "的儿女人数：";  
 cin >> childCount;  
 cout << "请依次输入" << fatherName << "的儿女的姓名";  
 string childName;  
 for (int i = 0; i < childCount; ++i)  
 {  
 cin >> childName;  
 families.add(fatherName, childName);  
 }  
 cout << fatherName << "的第一代子孙是：";  
 families.show(fatherName);  
}  
else  
{  
 cout << "该人不存在!";  
}   
 }  
 else if(ops=="B")  
 {  
 cout << "请输入要添加儿子（或女儿）的人的姓名:";  
 string fatherName;  
 cin >> fatherName;  
if (families.findFromTree(fatherName))  
{  
 cout << " 请输入" << fatherName << "新添加的儿子（或女儿）的姓名：";  
 string childName;  
 cin >> childName;  
 families.add(fatherName, childName);  
 cout << fatherName << "的第一代子孙是：";  
 families.show(fatherName);  
}  
else  
{  
 cout << "该人不存在!";  
}  
 }  
 else if(ops=="C")  
 {  
 cout << "请输入要解散家庭的人的姓名：";  
 string fatherName;  
 cin >> fatherName;  
 //删除操作  
 cout << "要解散家庭的人是"<< fatherName<<endl;  
if (families.findFromTree(fatherName))  
{  
 cout << fatherName << "的第一代子孙是:";  
 families.show(fatherName);  
 families.deleteFromTree(fatherName);  
}  
else  
{  
 cout << "该人不存在!";  
}  
  
 }  
 else if(ops=="D")  
 {  
 cout << "请输入更改姓名的人的目前姓名";  
 string fatherName;  
 cin >> fatherName;  
if (families.findFromTree(fatherName))  
{  
 cout << "请输入更改后的姓名";  
 string updateName;  
 cin >> updateName;  
 cout << fatherName << "已更名为" << updateName;//不成功的更名也会有这个提示,但是不想弄了  
 families.update(fatherName, updateName);  
}  
else  
{  
 cout << "该人不存在!";  
}  
 }  
 else if(ops=="E")  
 {  
 string fatherName;  
 cout << " 请输入要查询人的姓名：";  
 cin >> fatherName;  
 if(families.findFromTree(fatherName))  
 {  
 cout <<fatherName<< "的第一代子孙是：";  
 families.show(fatherName);  
 }  
 else  
 {  
 cout << "该人不存在!";  
 }  
 }  
 else if(ops=="F")  
 {  
 break;  
 }  
 cout<<endl<<"请选择要执行的操作:"<<endl;  
 cin>>ops;  
 }

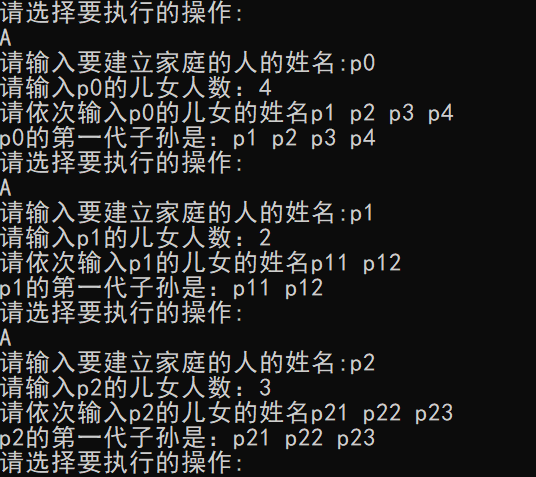
# 4 测试

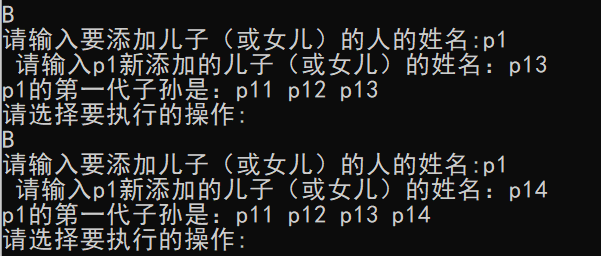
## 4.1 功能测试

### 4.1.1 建立功能测试

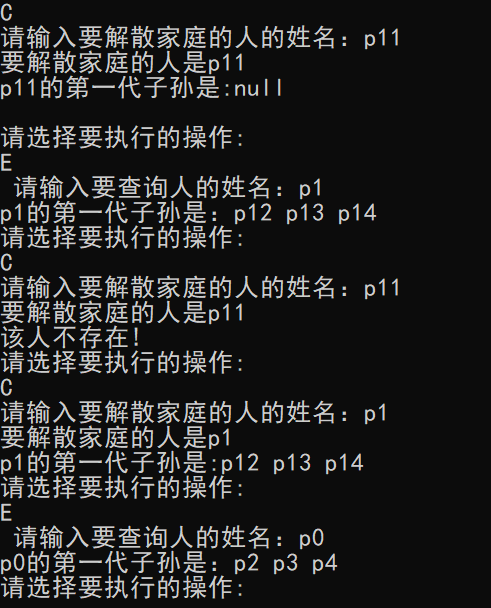


### 4.1.2 插入功能测试

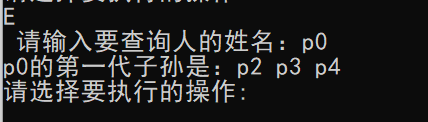




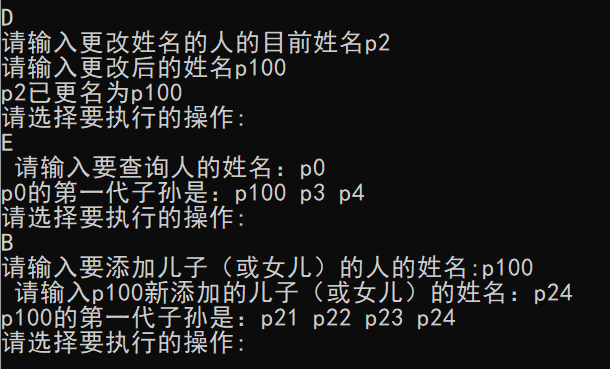
### 4.1.3 删除功能测试



### 4.1.4 查找功能测试

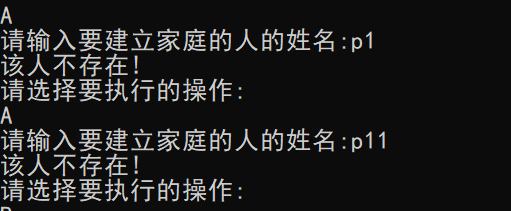


### 4.1.5 修改功能测试

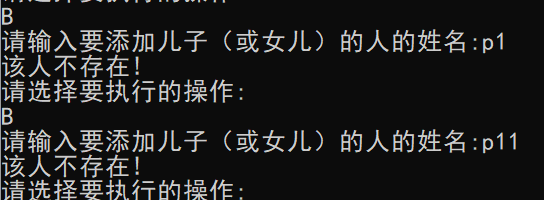


## 4.2 边界测试

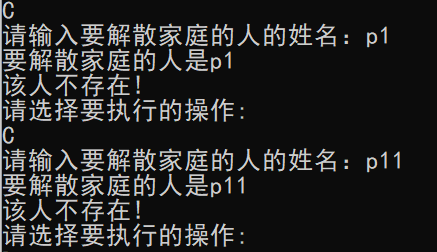
### 4.2.1 要建立家庭的人不在家谱中



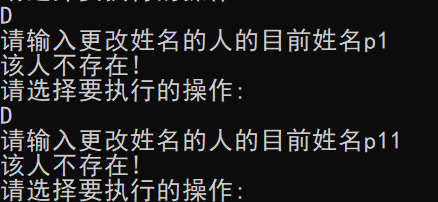
### 4.2.2 要增加子女的人不在家谱中



### 4.2.3 要解散家庭的人不在家谱中



### 4.2.4 要更改姓名的人不在家谱中



### 4.2.5 要查询的人不在家谱中

