****

**Tongji University**

**数据结构课程设计实验报告**

**006-家谱管理系统**

专 业： 软 件 工 程

指导教师： 张 颖

学 号： 2 1 5 3 0 6 1

姓 名： 谢 嘉 麒

1. **项目概览**
   1. **项目背景**

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。

* 1. **项目内容**

项目实现程序以完成家谱的建立，并且实现输入祖先，建立家庭，解散家庭，更改家庭成员等功能。满足客户程序化建立家谱并且查看家谱成员，修改家庭结构的需要。

* 1. **项目要求**
     1. **项目功能要求**

本项目要求完成家谱成员信息的建立，查找，插入，修改，删除等功能。

* + 1. **项目输入要求**

项目要求实现菜单界面，用户通过在菜单界面选择功能，进一步跳转并实现用户需要。

* + 1. **项目输出要求**

项目要求按功能不同进行不同的输出。

* 1. **需求分析**

考虑到家谱系统需要满足用户随时添加并且删除、修改家庭成员的需求，因此需要全面考虑用户可能需要的功能，保证程序存储严谨，功能无误。

* + 1. **系统功能的完善性**

家谱管理系统需要实现完善的添加成员，组建家庭，删除家庭和修改成员等一系列功能，并且为了方便用户直观了解家庭结构考虑，本次项目新设定打印家谱树的功能。

* + 1. **程序的正确性**

由于家谱性质重要，程序应该以严谨无误为要求，保证数据存储的正确性，避免数据丢失等错误。

* + 1. **程序的健壮性**

程序需要在用户输入错误数据时及时提示，防止客户在建立家谱时出现错误数据。

1. **项目设计思路**
   1. **项目数据结构设计**

考虑到家谱呈现树的分支结构，因此选择使用树作为数据结构进行项目设计。而为了方便进行家谱的增删查改，故选用向量类和链表类作为项目底层数据结构，对家庭成员进行存储。

* + 1. **链表类的设计**

本次实验承用之前完成的链表类作为向量类和树类的底层数据结构。且使用双向链表在牺牲空间复杂度的条件下降低时间复杂度，提高链表增删查改的速度，减少不必要的遍历和操作，优化用户体验。

因链表类与之前勇闯迷宫项目设计相同，因此此处不再赘述细节。

* + 1. **向量类的设计**

考虑到同一个祖先下的孩子总数不确定，并且存在增删查改的功能需要。因此如果设计孩子指针的话会造成空间浪费和程序的安全隐患。故选择使用向量类作为各个祖先的孩子存储空间，添加孩子时只需要push\_back即可，同理删除和修改成员的空间复杂度和时间开销也大大降低。

因向量类与之前勇闯迷宫项目设计相同，因此此处不再赘述细节。

* + 1. **成对类的设计**

成对类主要用于存储具有一定关系的数据组，本次项目中作为辅助层次遍历家谱并打印家谱而使用。

因成对类与之前勇闯迷宫项目设计相同，因此此处不再赘述细节。

* + 1. **树类的设计**

树形结构是以分支关系定义的层次结构，是一类重要的非线性数据结构，在计算机领域有广泛应用。

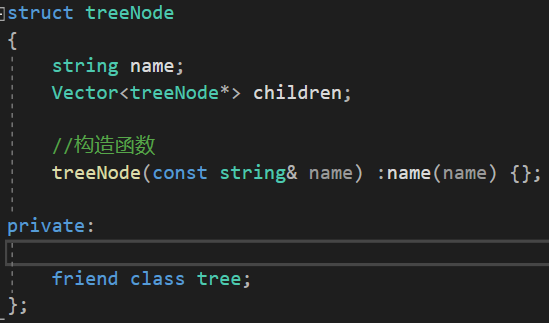
本次项目进行的家谱设计是树重要应用场景，利用树的跟表示祖先，树的分支分别表示此祖先的各个孩子，进行递归性质的建立，即可得到完整家谱。

1. 设定树结点结构体：

建立树则需要对树的结点进行设计，即对每一个家谱个体成员进行存储。考虑到家谱的泛用性，如不同国家人名习惯不同，存在数据差异等，因此采用string结构定义name变量，并递归设定孩子向量体。

此结构体将树类声明为友元类，方便进行数据读写。

1. string name;
2. Vector<treeNode\*> children;



1. 设定树类

在树类中定义treeNode\*类型的root变量，作为整棵树的根结点。

将家谱树需要的初始化，添加祖先，添加家庭，删除家庭，修改成员以及打印家谱等功能函数作为类成员函数封装在树类中。

1. //默认构造函数

tree() {};

1. //建立树

void buildTree();

1. //建立家庭

void buildFamily();

1. //加入家庭成员

void addFamilyMem();

1. //解散局部家庭家庭

void delFamily();

1. //更改家庭成员姓名

void renameFamilyMem();

1. //竖向打印家谱

void showFamily(treeNode\* ptr, Vector<Pair<int, bool>>& memberPlace, bool lastMember)const;

1. //用户选择打印家谱

void showTree();

1. //寻找家谱中成员，有则返回结点

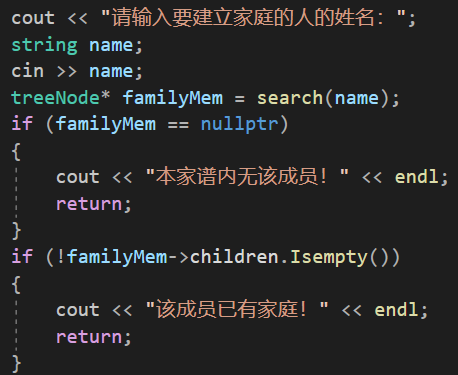
treeNode\* search(const string& name)const;

inline int count(const string& name)const;

treeNode\* searchAt(treeNode\* root, const string& name)const;

* 1. **算法及主要功能实现**
     1. **建立家庭函数buildFamily()的实现**

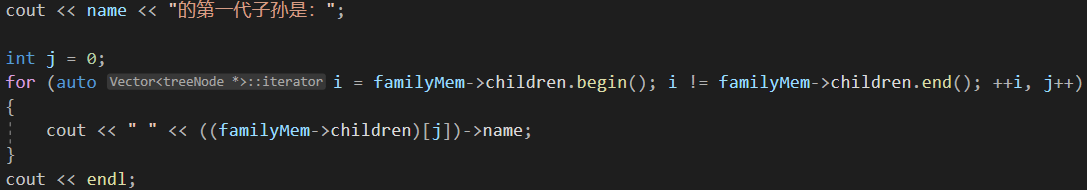
建立家庭首先需要用户输入建立对象。设计string类型的name变量作为被建立家庭者。用户输入名字以后，将它作为形参传入search()函数中，函数结果返回给treeNode\*型变量familyMem中。若familyMem为NULL，则说明此人不在已有家谱中，程序报错提示；若此人在家谱中，则判断此人孩子结点是否为空，若不空，则说明此人已有家庭，程序报错提示。



若可以建立家庭，则用int型变量存储子女数量，利用for语句循环子女数次，依次输入子女名字，并利用push\_back函数将子女加入父母的子女向量体中，完成家庭的建立。



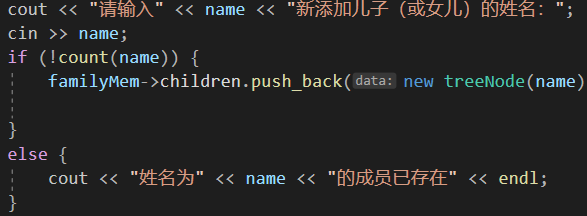
函数完成后打印此家庭。



* + 1. **添加成员函数addFamilyMen()的实现**

需要用户将被添加孩子的成员名输入到string类变量中，交由search函数进行判断。若不在已有家谱中，则报错提示。

若有，则提示用户输入要添加的成员名。用户输入正确后将其交由count函数进行判断，若此姓名的成员已存在，则count函数会返回1，则系统提示成员已存在；若count返回0，则说明可以建立，调用push\_back函数将新成员加入父母的孩子向量中即可。

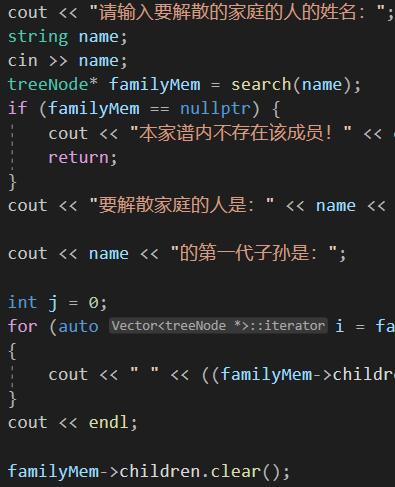


函数完成后打印此人的第一代子孙。

* + 1. **解散家庭函数delFamily()的实现**

需要用户将解散家庭的父辈成员名输入到string类变量中，交由search函数进行判断。若不在已有家谱中，则报错提示。

若有，函数打印其第一代子孙后，调用clear()函数清空此分支即可。



* + 1. **重命名函数renameFamilyMem()的实现**

需要用户将重命名的成员原名输入到string类变量中，交由search函数进行判断。若不在已有家谱中，则报错提示。

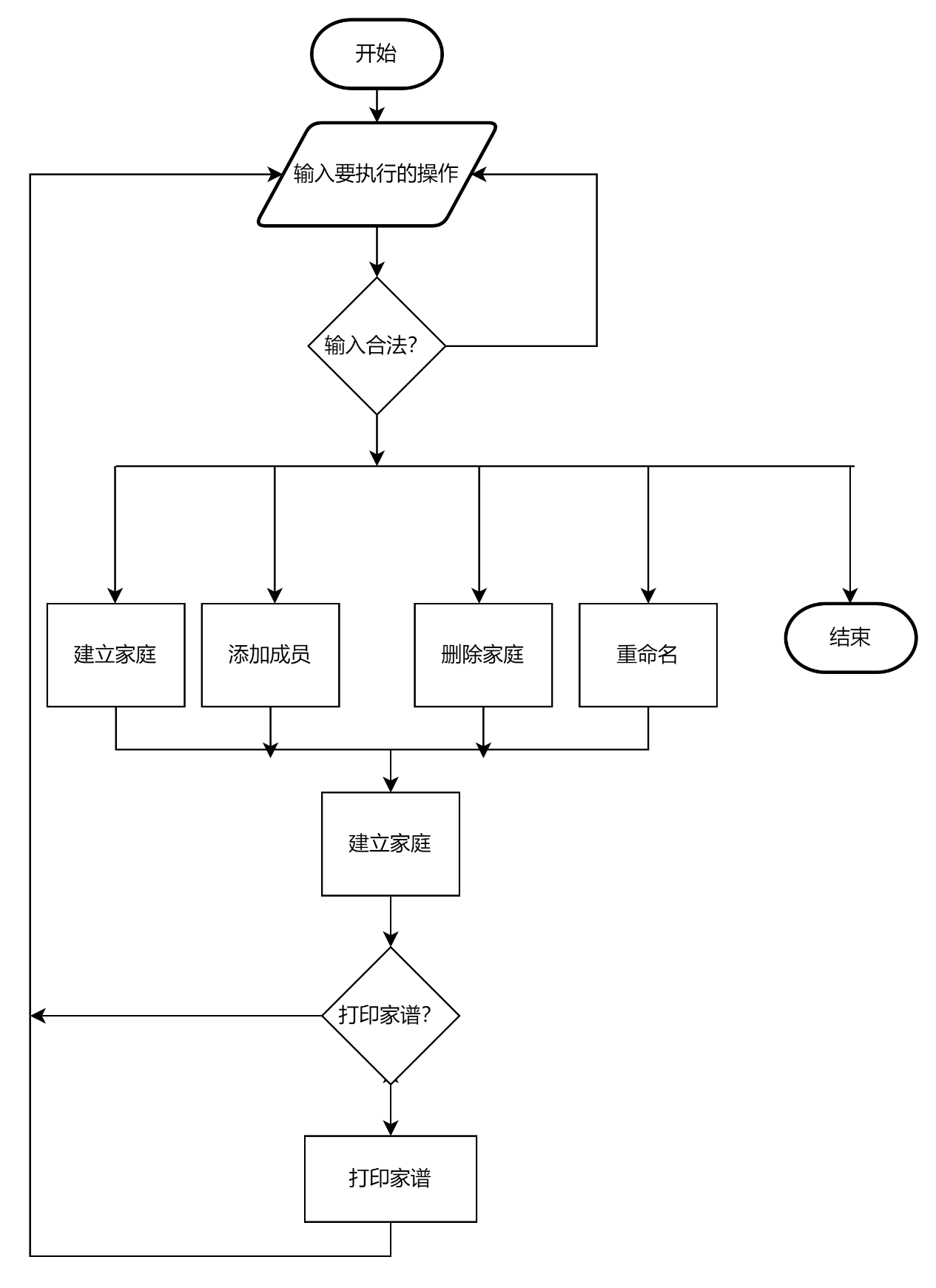
若有，则提示用户输入要更改后的名字。将新名字交由count函数进行判断，若家谱中已有此名字，则提示报错；若没有，则将familyMem->name更新为name。



* + 1. **绘制家谱树函数showFamily()的实现**

本函数主要使用递归的方法，从家谱树的根结点层次遍历，结点之间添加竖线或折线指示成员关系。

* 1. **程序流程图解**

****

否

是

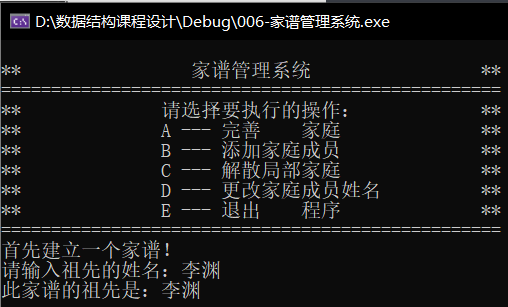
否

是

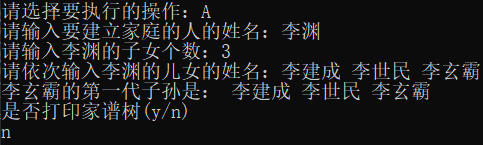
1. **项目测试**

本次项目测试使用唐高宗李渊的家谱作为测试用例。方便起见只输入适量成员

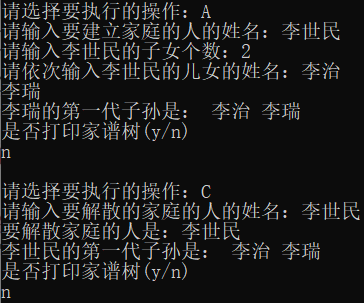
* 1. **输入祖先**

****

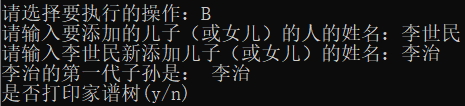
* 1. **建立家庭**



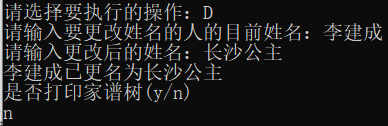
* 1. **删除家庭**

****

* 1. **添加家庭成员**

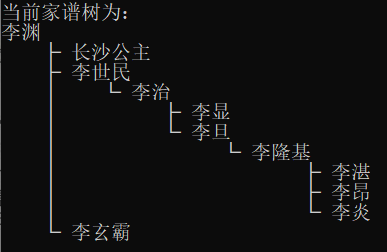
****

* 1. **更改家庭成员姓名**

****

* 1. **完整家谱**

为了方便测试，测试数据仅挑选主要成员进行处理，显示祖先关系。

****