项目说明文档

数据结构课程设计

——家谱管理系统

作 者 姓 名： 李佳诺

学 号： 1751188

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 3

1.1 背景分析 3

1.2 功能分析 3

2 设计 4

2.1 数据结构设计 4

2.2类结构设计 4

2.2 成员与操作设计 5

树结点类 5

树类 5

3 实现 6

3.1 构造函数familyTree() 6

3.2 查找家庭成员函数findFamilyMember() 7

查找功能的流程图 7

3.3 获取家庭成员函数get() 9

获取家庭成员功能的流程图 9

3.4 增加家庭成员函数addFamilyMember() 11

增加成员功能流程图 11

3.5 完善家庭函数completeFamilyMember() 13

完善家庭功能流程图 13

3.6 删除家庭成员函数deleteFamilyMember() 16

删除家庭成员功能流程图 16

3.7 更改家庭成员姓名函数changeName() 18

更改家庭成员姓名功能流程图 18

3.8 释放空间函数destroy() 19

3.9 总体功能的实现 20

总体功能流程图 20

4 测试 22

4.1功能性测试 22

4.1.1 整个系统的初始化功能的测试 22

4.1.2 添加家庭成员的功能测试 22

4.1.3 完善家庭的功能测试 23

4.1.4 更改家庭成员姓名的功能测试 23

4.1.5 查看成员信息的功能测试 23

4.1.6 解散局部成员的功能测试 24

4.2 出错性测试 24

4.2.1 完善家庭成员的出错测试 24

4.2.2 添加家庭成员的出错测试 25

4.2.3 解散家庭成员的出错测试 25

4.2.4 更改家庭成员姓名的出错测试 26

4.2.5 查看家庭成员信息的出错测试 26

# 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。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。

## 功能分析

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。

# 设计

## 数据结构设计

如上功能所述，该系统的各个数据成员之间有着明显的层次结构，即数据之间可能有祖先—后代、兄—弟等分支关系，所以采用了树这个数据结构。然后又考虑到，该系统要进行大量的增加、删除、修改操作，如果使用数组来存储则性能太差。为了尽最大可能的提升性能，帮助人们找到家族的归属感，本系统采用了链表来作为存储结构。

## 2.2类结构设计

树的链表存储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。为方便处理，本系统采用struct描述树结点类（familyMember），这样使得树类类（familyTree）可以访问树结点。

## 成员与操作设计

### 树结点类

/\*----成员结构体----\*/

struct familyMember

{

string m\_name; //成员姓名

familyMember \* father; //父指针

familyMember \* child; //子指针

familyMember \* brotherPre; //兄弟指针

familyMember \* brotherNext; //兄弟指针

int sonNum; //直系后代个数

};

### 树类

/\*----家谱树类定义----\*/

class familyTree

{

private:

familyMember \* ancestor; //树根

public:

familyTree(); //构造函数

~familyTree(); //析构函数

void addFamilyMember(); //添加家庭成员

void completeFamilyMember(); //完善家庭

void deleteFamilyMember(); //解除局部家庭

void changeName(); //更改家庭成员姓名

void get(); //查看成员信息

familyMember \* findFamilyMember(string name, familyMember \* key); //查找家庭成员

void showSons(familyMember \* p); //显示子代

void destroy(familyMember \* p); //递归删除

familyMember \* familyMemberCreate(); //成员生成函数

void ancestorCreate(); //根节点生成函数

};

# 3 实现

## 3.1 构造函数familyTree()

建立新家庭，创建该家庭祖先结点ancestor,并将该结点的各个成员变量初始化如下：

ancestor->m\_name = "";

ancestor->sonNum = 0;

ancestor->father = NULL;

ancestor->child = NULL;

ancestor->brotherPre = NULL;

ancestor->brotherNext = NULL;

## 3.2 查找家庭成员函数findFamilyMember()

#### 查找功能的流程图

开 始

查 找

是否找到

输入成员名字

结 束

输出报错

删 除

是

否

该函数的功能是根据索要查找的家庭成员的姓名在整个家谱管理系统中搜索。本函数并不直接与用户交互，而是为了更方便地执行一些更复杂的操作。本函数实现的基本思想是递归，即在每一个子树中进行相同的遍历操作，直到找到该家庭成员，返回该成员结点，倘若未找到，则返回NULL。其中遍历整个家谱的核心代码如下：

//如果找到则返回

if (p->m\_name == name)

return p;

//在兄弟树中递归调用查找函数

while (temp->brotherNext)

{

ans = findFamilyMember(name, temp->brotherNext);

if (ans)

return ans;

temp = temp->brotherNext;

}

temp = p;

//在子树中递归调用查找函数

if (temp->child)

ans = findFamilyMember(name, temp->child);

return ans;

## 3.3 获取家庭成员函数get()

#### 获取家庭成员功能的流程图

开 始

查 找

是否找到

输入成员姓名

结 束

输出报错

是

否

输出成员家庭信息

该函数的功能是通过与用户交互，获得用户想要查询的人的家庭信息。倘若找到，则打印出该成员的父亲和兄弟（如果有的话）。其中的查找操作是调用的findFamily()成员函数，展示操作是调用的showSons()成员函数。核心代码如下：

//调用查找函数，并将查找结果赋给p

familyMember \* p = findFamilyMember(name, ancestor);

//没有找到该家庭成员

if (p == NULL)

{

cout << "查无此人" << endl;

return;

}

//找到后开始打印该成员的家庭信息

cout << "姓名：" << p->m\_name << endl;

if (p->father)

{

cout << "父亲：" << p->father->m\_name << endl;

if (p->father->sonNum > 0)

{

familyMember \*temp = p->father->child;

cout << "兄弟：";

for (int i = 1; i < p->father->sonNum;)

{

if (temp->m\_name != name)

{

cout << temp->m\_name << " ";

i++;

}

if (temp->brotherNext)

temp = temp->brotherNext;

}

cout << endl;

}

}

showSons(p);

## 3.4 增加家庭成员函数addFamilyMember()

#### 增加成员功能流程图

开 始

查 找

是否找到

输入增加成员父亲姓名

结 束

输出报错

是

否

输出成员第一代子女

输入子女

该函数的主要功能是通过与用户交互，实现新的家庭成员的添加。先执行搜索函数查找某个人，倘若用户输入的要添加子女的人的名字不存在，则返回报错信息。找到之后将需要添加的链表加在该节点下方。兄弟节点连在brotherNext指针后，孩子则连在child后，添加孩子时需要检测孩子指针是否为空，如果不是空就将操作指针移动到最后一个孩子，然后对brotherNext进行操作，如果没有孩子，直接将该节点接在child节点上。在添加完毕后，展示该成员新的第一代子女。其中涉及到插入的核心代码如下：

familyMember \* temp = p;

temp = temp->child;

while (temp->brotherNext)

{

temp = temp->brotherNext;

}

temp->brotherNext = newChild;

## 3.5 完善家庭函数completeFamilyMember()

#### 完善家庭功能流程图

开 始

查 找

是否找到

输入待完善家庭成员姓名

结 束

输出报错

是

否

输入成员家庭信息

输出第一代

该函数的主要功能是通过与用户交互，在家谱系统中为一个人建立家庭，如果用户输入的人在整个家谱系统中未找到或者此人已经建立过家庭，则返回报错信息，否则依次给此人添加用户所输入的子女，其核心代码如下：

cout << "请依次输入" << p->m\_name << "的儿女的姓名：";

p->child = familyMemberCreate();

familyMember \*temp = p->child;

cin >> temp->m\_name;

temp->father = p;

//依次添加sonNum个子女

for (int i = 1; i < p->sonNum; i++)

{

familyMember \*brother = familyMemberCreate();

cin >> brother->m\_name;

brother->father = p;

brother->brotherPre = temp;

temp->brotherNext = brother;

temp = temp->brotherNext;

}

## 3.6 删除家庭成员函数deleteFamilyMember()

#### 删除家庭成员功能流程图

开 始

查 找

是否找到

输入删除成员姓名

结 束

输出报错

删除本身及其子树

是

否

该函数的基本功能是通过与用户交互，解散用户输入的人的家庭。删除节点比较麻烦，其基本的思想是使用后缀遍历的方法遍历该节点的子树，然后将所有的子树全部都删除掉并将空间释放掉。其核心代码如下：

if (p->brotherPre && p->brotherNext)

{

p->brotherPre->brotherNext = p->brotherNext;

p->brotherNext->brotherPre = p->brotherPre;

p->father->sonNum--;

}

if (p->brotherPre && !p->brotherNext)

{

p->brotherPre->brotherNext = NULL;

p->father->sonNum--;

}

if (!p->brotherPre && p->brotherNext)

{

p->father->child = p->brotherNext;

p->brotherNext->brotherPre = NULL;

p->father->sonNum--;

}

if (!p->brotherPre && !p->brotherNext)

{

p->father->child = NULL;

p->father->sonNum--;

}

if (p->child)

destroy(p->child);

delete p;

## 3.7 更改家庭成员姓名函数changeName()

#### 更改家庭成员姓名功能流程图

开 始

查 找

是否找到

输入更改成员原名

结 束

输出报错

改名

是

否

输出新名字

该函数的基本功能是通过与用户交互进行家庭成员姓名的更改。主要的思路是先在家谱中用findFamily()函数查找该成员，若未找到，返回报错信息，若找到，则将其名字改为用户输入的名字，修改成功后打印出修改后的姓名。其核心代码如下：

string name;

cout << "请输入要更改姓名的人的姓名:";

cin >> name;

familyMember \*p = findFamilyMember(name, ancestor);

if (p == NULL)

{

cout << "查无此人" << endl;

return;

}

cout << "请输入更改后的姓名:";

cin >> p->m\_name;

cout << name << "已经更名为" << p->m\_name << endl;

## 3.8 释放空间函数destroy()

该函数不直接参与用户交互，但起到回收内存的作用，这为该家谱系统能够稳定运行奠定了重要基础，也是家族人丁兴旺的重要保障。基本思路是用递归，回收其兄弟和子节点。核心代码如下：

familyMember \*temp = p;

while (temp->brotherNext)

{

temp = temp->brotherNext;

}

while (temp->brotherPre)

{

temp = temp->brotherPre;

temp->brotherNext->brotherPre = NULL;

temp->brotherNext->brotherNext = NULL;

destroy(temp->brotherNext);

}

if (temp->child)

destroy(temp->child);

delete p;

## 3.9 总体功能的实现

#### 总体功能流程图

开 始

建 立

增加成员

查 找

删 除

退 出

改 名

要想实现整个家谱系统，仅仅设计家族树类是不够的，还通过一定的逻辑，使得整个系统与用户交互贴合得顺滑如丝。这里的基本操作是，我们先初始化整个家谱系统，并构造家族树类的一个实例。接下来，通过switch语句，判定用户选择执行的操作，并调用相应的成员函数实现并反馈。其核心代码如下：

while (order != 'F')

{

switch (order)

{

case 'A':

Family.completeFamilyMember();

break;

case 'B':

Family.addFamilyMember();

break;

case 'C':

Family.deleteFamilyMember();

break;

case 'D':

Family.changeName();

break;

case 'E':

Family.get();

break;

default:

cout << "输入有误";

break;

}

cout << endl << endl;

cout << "请选择要执行的操作:";

cin >> order;

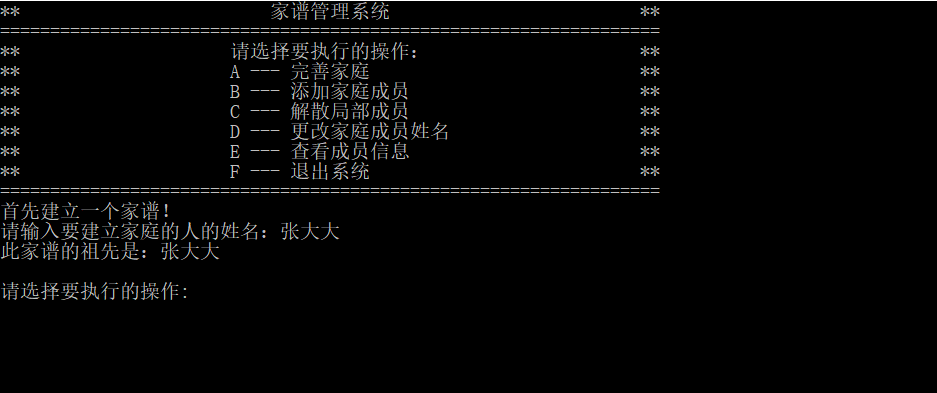
}

# 测试

## 4.1功能性测试

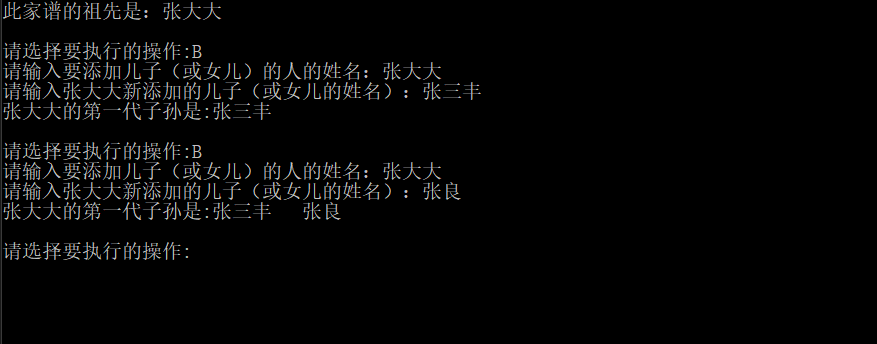
#### 4.1.1 整个系统的初始化功能的测试

本测试主要是查看是否能成功初始化整个家谱系统。张大大的家族树开张啦。测试结果如下：



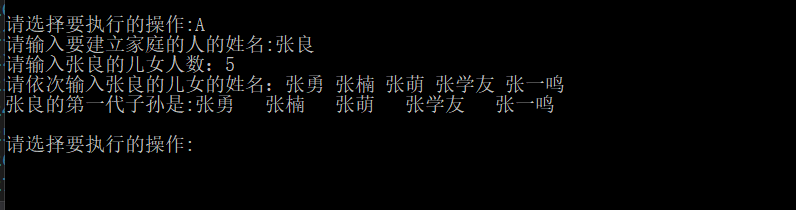
#### 4.1.2 添加家庭成员的功能测试

该测试主要是验证是否可以往指定成员结点添加直系子女。如我们伟大的张大大有俩优秀的子女，于是我们往其中添加了两位子女。测试结果如下：



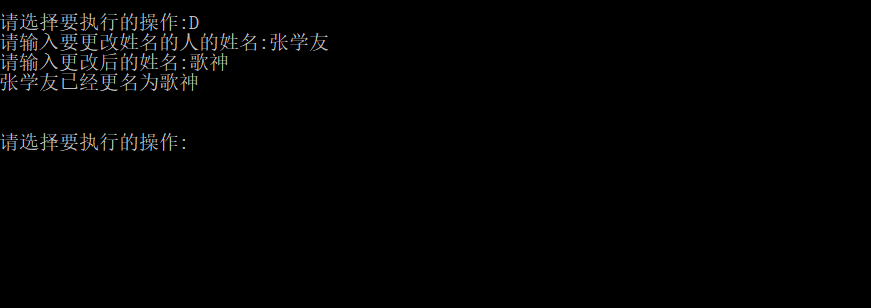
#### 4.1.3 完善家庭的功能测试

该测试主要是验证是否可以往家谱系统中任意成员添加一定个数的子女。如张良家人丁兴旺，随着国家全面开放二胎的落实，他们家又生了5个娃，所以我们要完善其家谱。测试结果如下：



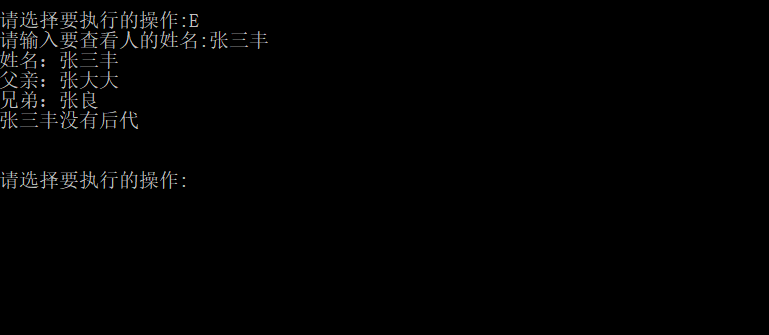
#### 4.1.4 更改家庭成员姓名的功能测试

该测试主要验证是否可以更爱家谱系统中某成员的姓名。如张良家的二儿子张学友，很有个性，他觉得自己唱歌那么好听，应该叫歌神，于是改名了。测试结果如下：



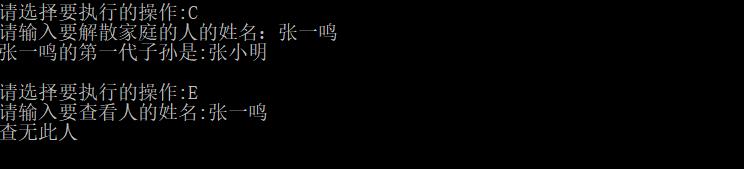
#### 4.1.5 查看成员信息的功能测试

该测试主要验证用户是否可以随时随地查看家谱信息。如张三丰的记性不太好，老不记得自己的弟弟不说，连最疼爱他的父亲，张大大也认错，此功能可以帮助他快速及时地认清自己的亲人。测试结果如下：



#### 4.1.6 解散局部成员的功能测试

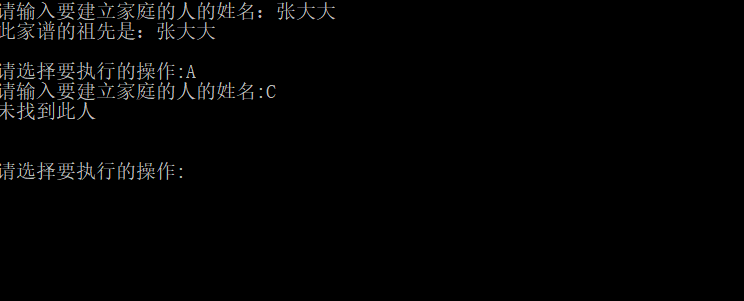
该测试主要验证用户是否可以准确无误的解散家庭，通过查看成员信息功能来查看结果是否成功,如果返回查无此人则解散成功。如张良养了个叛逆的儿子张一鸣，想要和他解除父子关系，于是家族长老决定将张一鸣从家谱中除名。此时我们需要在家谱系统中，解散他俩的关系。测试结果如下：



## 4.2 出错性测试

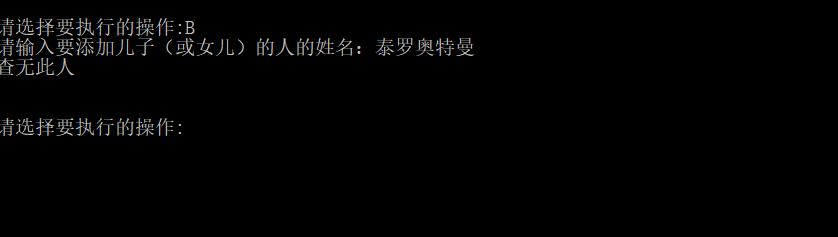
#### 4.2.1 完善家庭成员的出错测试

该测试主要时验证当用户输入的人不在家谱系统中时，系统不崩溃，并且可以返回报错信息。测试结果如下：



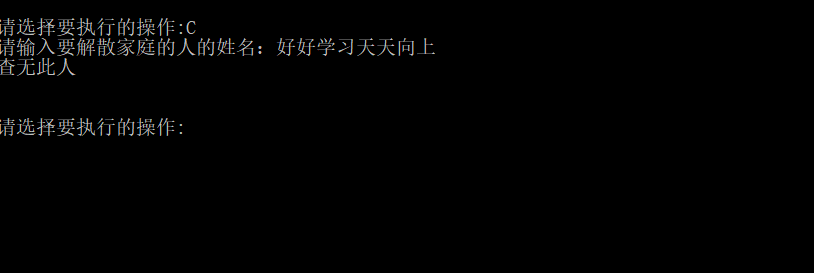
#### 4.2.2 添加家庭成员的出错测试

该测试主要时验证当用户输入的人不在家谱系统中时，系统不崩溃，并且可以返回报错信息。测试结果如下：



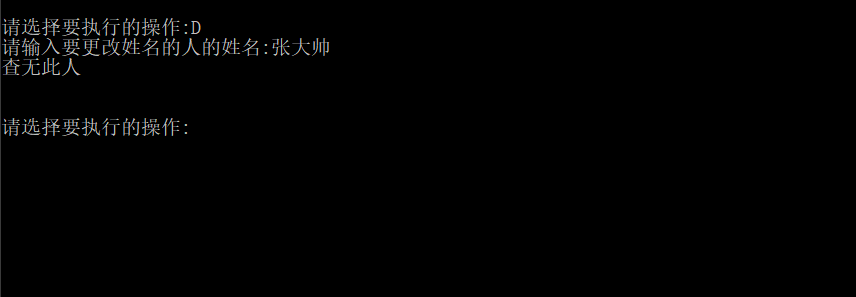
#### 4.2.3 解散家庭成员的出错测试

该测试主要时验证当用户想解散家庭的人不在家谱系统中时，系统不崩溃，并且可以返回报错信息。测试结果如下：



#### 4.2.4 更改家庭成员姓名的出错测试

该测试主要时验证当用户想要修改其名字的人不在家谱系统中时，系统不崩溃，并且可以返回报错信息。测试结果如下：



#### 4.2.5 查看家庭成员信息的出错测试

该测试主要时验证当用户想要查询的人不在家谱系统中时，系统不崩溃，并且可以返回报错信息。测试结果如下：

