项目说明文档

数据结构课程设计

——家谱管理系统

作 者 姓 名： 王加炜

学 号： 2150265

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 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

## 2.1数据结构设计

如上功能分析所述，该系统要求每一个家庭成员下对应众多子女分支，同时要求大量的增加、删除、修改操作，故考虑使用树的形式存储家庭成员信息，每个家庭成员下分支数不固定。

## 2.2类结构设计

本项目中的家谱形式要求家庭树的每一个节点的子节点数量不固定，所以树的节点考虑采用“左长子右兄弟”的含两个指针域的结点。

## 2.3成员与操作设计

treenode结构体：

struct treenode {//树节点存储每个成员的信息

string data;

treenode\* leftchild;

treenode\* nextsibling;

treenode(string name)

{

data = name;

leftchild = NULL;

nextsibling = NULL;

}

};

Tree类：

class Tree {//树包含所有树节点

private:

treenode\* root;

public:

Tree(treenode\* p = NULL)

{

root = p;

}

treenode\* Parent(treenode\* p, treenode\* current);//寻找父亲节点

treenode\* Find(treenode\* p, string value);//寻找当前结点

void input();//初始化输入

void createfamily();//完善家庭

void addfamilymember();//添加家庭成员

void deletefamily();//删除家庭成员

void changename();//改名

};

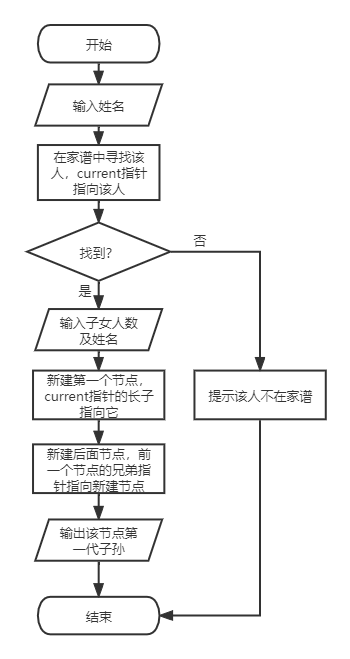
## 2.4系统设计

系统首先在main函数中创建一颗新树，然后调用menu函数生成菜单，执行input函数，用户在该函数中输入操作码，操作码对应跳转到树的相应函数进行执行。

# 三．实现

## 3.1创建家庭

### 3.1.1流程图



### 3.1.2文字说明

该函数作用是创建一个家庭。

用户输入想要创建家庭的人的姓名，通过Find函数可以找到存储这个姓名的结点。之后用户依次输入想要加入家庭的孩子的姓名。该树采用的是左孩子右兄弟的存储方式，而这种方式对于存储孩子的效率是非常高效的。只需要调用next->sibling指针就可以轻松地为下一个兄弟进行赋值。最后通过打印的方式，输出该结点的第一代子孙的相关信息。

### 3.1.3代码实现

void Tree::createfamily()

{

string name;

//包含错误输入，若输入一个不在该家庭中的名字，输出错误信息

while (1)

{

cout << "请输入要建立家庭的人的姓名：";

cin >> name;

if (Find(root, name) == NULL)

{

cout << "该家庭不包含该成员，请重新输入！" << endl;

}

else if (Find(root, name)->leftchild != NULL)

{

cout << "此人已有家庭，无法新建家庭！" << endl;

}

else

{

break;

}

}

cout << "请输入" << name << "的儿女人数：";

int num;

cin >> num;

cout << "请依次输入" << name << "的儿女的姓名：";

treenode\* p = Find(root, name);

treenode\* current = p;

string child;

for (int i = 0; i < num; i++)

{

cin >> child;

if (i == 0)

{

current->leftchild = new treenode(child);

current = current->leftchild;

}

else

{

current->nextsibling = new treenode(child);

current = current->nextsibling;

}

}

cout << name << "的第一代子孙是：";

for (current = p->leftchild; current != NULL; current = current->nextsibling)

{

cout << current->data << " ";

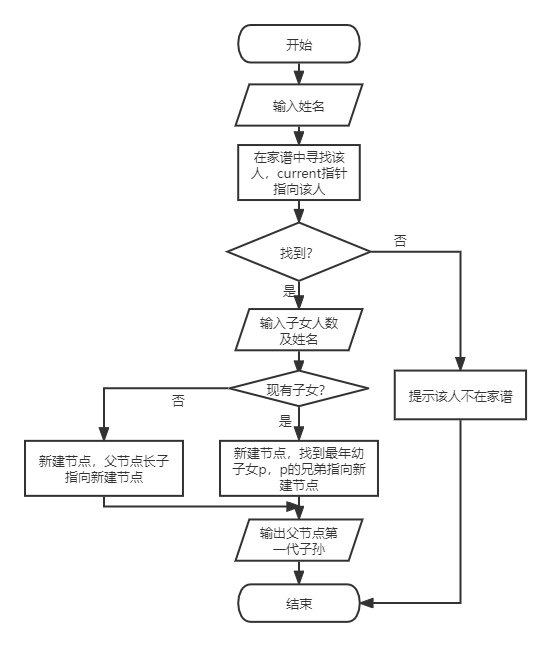
}

cout << endl;

}

## 3.2添加家庭成员

### 3.2.1流程图



### 3.2.2文字说明

该函数作用是添加一个家庭成员到家谱系统中。

用户输入想要添加家庭成员的人的姓名，通过Find函数可以找到存储这个姓名的结点。如果该结点无左右子树，说明该家庭成员无孩子，所以添加的孩子应该存放在当前节点的左孩子上；如果该结点有左孩子，则一直遍历该左孩子的右兄弟，直到遍历到最后一个兄弟为止，将当前结点的指针指向它，然后将新添加的孩子存放在当前节点的右兄弟上。最后通过打印的方式，输出想要添加家庭成员的人的第一代子孙的相关信息。

### 3.2.3代码实现

void Tree::addfamilymember()

{

treenode\* p;

string name;

treenode\* current\_;

while (1)

{

cout << "请输入要添加儿子(或女儿)的人的姓名：";

cin >> name;

if ((p = Find(root, name)) == NULL)

{

cout << "该家庭不包含该成员，请重新输入！" << endl;

}

else

{

break;

}

}

treenode\* current = p->leftchild;

string child;

while(1)

{

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

cin >> child;

if ((current\_ = Find(root, child)) != NULL)

{

cout << "新添加的姓名已经被家庭成员所占用，请重新输入！" << endl;

}

else

{

break;

}

}

if (p->leftchild == NULL)

{

p->leftchild = new treenode(child);

}

else

{

while (current->nextsibling != NULL)

{

current = current->nextsibling;

}

current->nextsibling = new treenode(child);

}

cout << name << "的第一代子孙是：";

for (current = p->leftchild; current != NULL; current = current->nextsibling)

{

cout << current->data << " ";

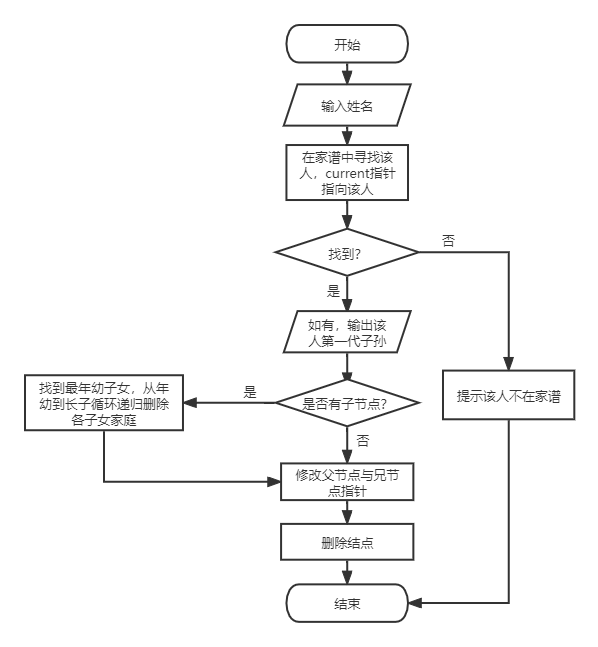
}

cout << endl;

}

## **3.3删除家庭**

### 3.3.1流程图



### 3.3.2文字说明

该函数作用是在家谱系统中删除一个家庭。用户输入想要删除家庭的人的姓名，通过Parent函数找到该结点的父节点。如果该结点是父节点的左孩子，则直接令父节点的左孩子指向第一个孩子的右兄弟；如果该结点是父节点左孩子的兄弟结点，则找到该结点的前一个兄弟结点，将该兄弟结点的右孩子指针指向当前结点的右孩子。最后输出要解散的人的子孙信息。

### 3.3.3代码实现

void Tree::deletefamily()

{

treenode\* p;

string name;

while (1)

{

cout << "请输入要解散家庭的人的姓名：";

cin >> name;

if ((p = Find(root, name)) == NULL)

{

cout << "该家庭不包含该成员，请重新输入！" << endl;

}

else

{

break;

}

}

treenode\* t = Parent(root, p);

treenode\* current;

if (t->leftchild->data == p->data)

{

t->leftchild = p->nextsibling;

}

else

{

for (current = t->leftchild; current != NULL; current = current->nextsibling)

{

if (current->nextsibling->data == p->data)

{

current->nextsibling = p->nextsibling;

break;

}

}

}

cout << "要解散家庭的人是：" << name << endl;

cout << name << "的第一代子孙是：";

for (current = p->leftchild; current != NULL; current = current->nextsibling)

{

cout << current->data << " ";

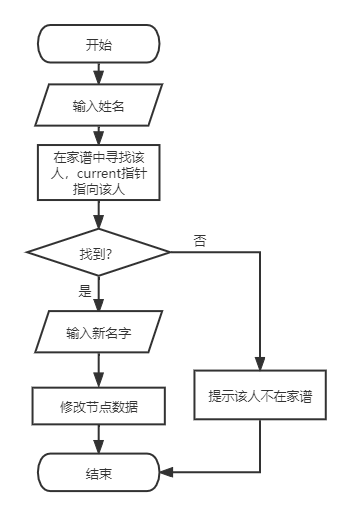
}

cout << endl;

}

## **3.4更改家庭成员名字**

### 3.4.1流程图



### 3.4.2文字说明

该函数作用是更改家庭成员的姓名。用户输入想要更改名字的人的姓名，通过Find函数可以找到存储该姓名的结点，之后将该结点的名字修改为新的名字即可。最后以打印的方式输出更改之后成员的姓名。

### 3.4.3代码实现

void Tree::changename()

{

string name;

treenode\* current;

treenode\* current\_;

while (1)

{

cout << "请输入要更改名字的人的目前姓名：";

cin >> name;

if ((current = Find(root, name)) == NULL)

{

cout << "该家庭不包含该成员，请重新输入！" << endl;

}

else

{

break;

}

}

string newname;

while (1)

{

cout << "请输入更改后的姓名：";

cin >> newname;

if ((current\_ = Find(root, newname)) != NULL)

{

cout << "该新名字已经被家庭成员所占用，请重新输入！" << endl;

}

else

{

current->data = newname;

cout << name << "已更名为" << newname;

cout << endl;

break;

}

}

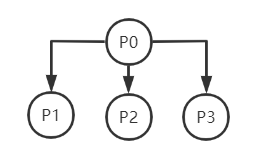
}

# 四．测试

## 4.1功能测试及边界测试

### 4.1.1完善家庭成员功能：

**原有家谱：**



**测试用例**：P2 3 P21 P22 P23

**预期结果**：P2的第一代子孙是：P21 P22 P23

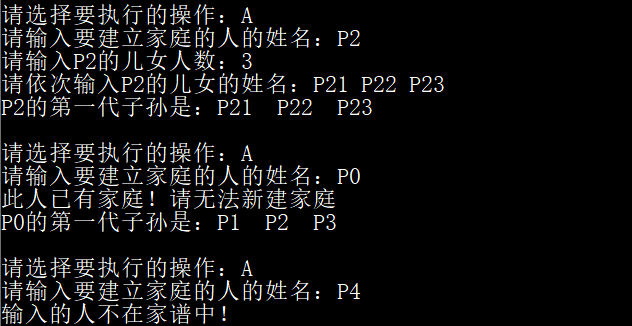
**测试用例**：P0

**预期结果**：提示此人已有家庭，无法新建家庭

**测试用例：**P4

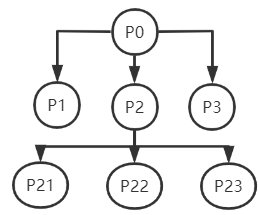
**预期结果：**提示输入的人不在家谱中

**实验结果**



### 4.1.2增加家庭成员功能：

**原有家谱：**



**测试用例**：P3添加P31（原无子女）

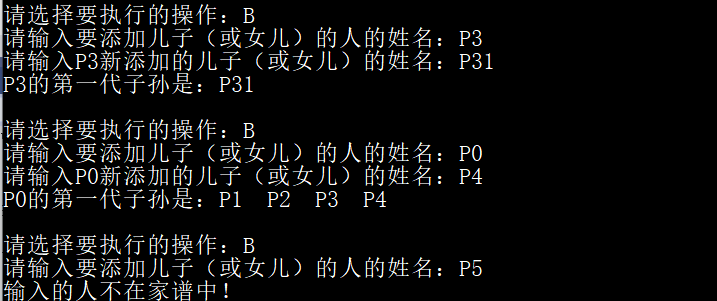
**预期结果**：P3的第一代子孙是：P31

**测试用例**：P0添加P4（原有子女）

**预期结果**：P0的第一代子孙是：P1 P2 P3 P4

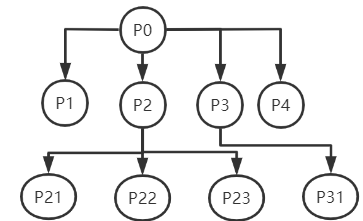
**测试用例：**P5

**预期结果：**提示输入的人不在家谱中



### 4.1.3解散家庭功能测试：

**原有家谱：**

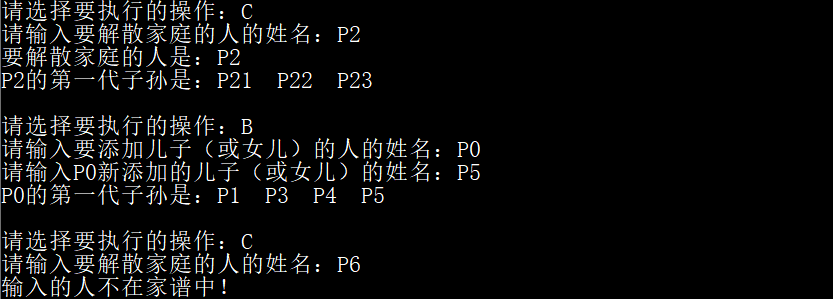


**测试用例**：P2

**预期结果**：通过添加P0子女时输出的结果证明P2已经被删掉

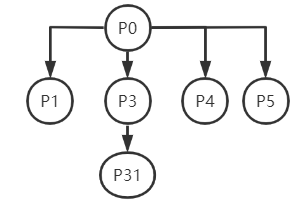
**测试用例**：P6

**预期结果**：提示输入的人不在家谱中



### 4.1.4 更改姓名功能测试

**原有家谱：**

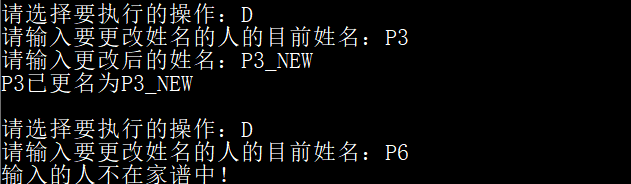
****

**测试用例**：P3->P3\_NEW

**预期结果**：P3已更名为P3\_NEW

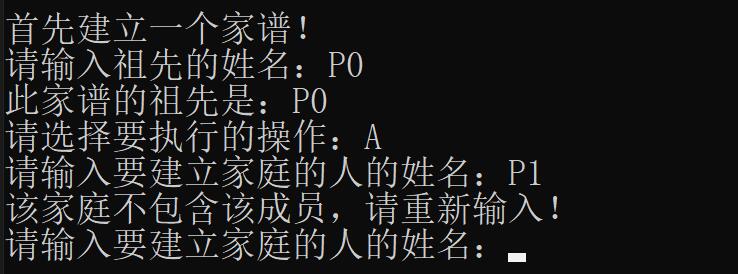
**测试用例**：P6

**预期结果**：提示输入的人不在家谱中

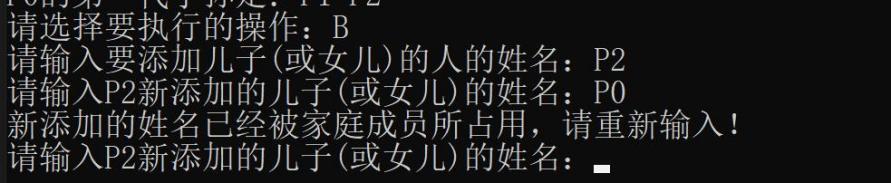


## 4.2错误测试

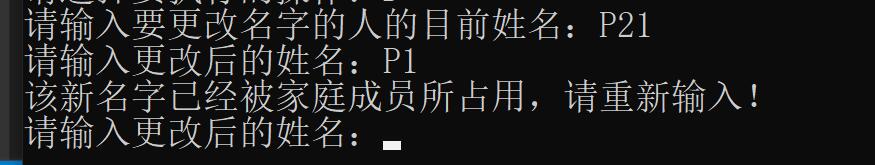
### 4.2.1输入一个未在家谱中的姓名:



### 4.2.2新添加成员名字与原家谱中姓名重合：



### 4.2.3更改的成员姓名与原家谱中姓名重合：



### 4.2.4操作码不符合要求：

