项目说明文档

数据结构课程设计

——家谱管理系统

作 者 姓 名： 伊啸

学 号： 1951220

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 1](#_Toc60133020)

[1.1 背景分析 1](#_Toc60133021)

[1.2 功能分析 1](#_Toc60133022)

[2 设计 1](#_Toc60133023)

[2.1 数据结构设计 1](#_Toc60133024)

[2.2 类结构设计 1](#_Toc60133025)

[2.3 成员与操作设计 1](#_Toc60133026)

[2.4 系统设计 3](#_Toc60133027)

[3 实现 4](#_Toc60133028)

[3.1 完善家谱功能的实现 4](#_Toc60133029)

[3.1.1 完善家谱功能流程图 4](#_Toc60133030)

[3.1.2 完善家谱功能核心代码 6](#_Toc60133031)

[3.1.3 插入功能截屏示例 7](#_Toc60133032)

[3.2 添加家庭成员功能的实现 8](#_Toc60133033)

[3.2.1 添加家庭成员功能流程图 8](#_Toc60133034)

[3.2.2 添加家庭成员功能核心代码 10](#_Toc60133035)

[3.2.3 删除功能截屏示例 10](#_Toc60133036)

[3.3 解散局部家庭功能的实现 11](#_Toc60133037)

[3.3.1 解散局部家庭功能流程图 11](#_Toc60133038)

[3.3.2 解散局部家庭功能核心代码 13](#_Toc60133039)

[3.3.3 解散局部家庭功能截图示例 14](#_Toc60133040)

[3.4 更改家庭成员姓名功能的实现 15](#_Toc60133041)

[3.4.1 更改家庭成员姓名功能流程图 15](#_Toc60133042)

[3.4.2 更改家庭成员姓名功能核心代码 17](#_Toc60133043)

[3.4.3 更改家庭成员姓名功能截屏示例 17](#_Toc60133044)

[3.5 查找功能的实现 18](#_Toc60133045)

[3.5.1 查找功能流程图 18](#_Toc60133046)

[3.5.2 查找功能核心代码 18](#_Toc60133047)

[3.5.3 查找功能截屏示例 20](#_Toc60133048)

[3.6 总体系统的实现 21](#_Toc60133049)

[3.6.1 总体系统流程图 21](#_Toc60133050)

[3.6.2 总体系统核心代码 23](#_Toc60133051)

[3.6.3 总体系统截屏示例 24](#_Toc60133052)

[4 测试 24](#_Toc60133053)

[4.1 功能测试 24](#_Toc60133054)

[4.1.1 完善家谱功能测试 24](#_Toc60133055)

[4.1.2 添加家庭成员功能测试 25](#_Toc60133056)

[4.1.3 解散局部家庭功能测试 26](#_Toc60133057)

[4.1.4 更改家庭成员姓名功能测试 26](#_Toc60133058)

[4.1.5 查找功能测试 26](#_Toc60133059)

[4.2 边界测试 27](#_Toc60133060)

[4.2.1 删除根结点 27](#_Toc60133061)

[4.2.2 删除后子女节点为空 27](#_Toc60133062)

[4.3 出错测试 28](#_Toc60133063)

[4.3.1 操作码错误 28](#_Toc60133064)

[4.3.2 完善家谱、添加家庭成员、解散局部家庭、更改姓名时成员不存在 28](#_Toc60133065)

# 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。

## 1.2 功能分析

本项目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。

综上所述，一个家谱管理系统至少应该具有输入、输出、插入、删除、修改、退出的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求大量的增加、删除、修改操作，而链表进行增加、删除等操作十分简便，因此考虑使用链表数据结构。同时一个父亲节点可以指向多个子女节点，因此考虑用树实现。为便于操作，采用子女——兄弟树实现。

## 2.2 类结构设计

采用树节点（TreeNode）和树（class Tree），树节点为struct结构，便于树直接读取节点中的数据成员。另外，还设计了一个栈，便于查找时实现树的前序遍历。

## 2.3 成员与操作设计

**树结点类（TreeNode）**

**公有成员：**

    string name;//家族成员的名字

    TreeNode \*firstChild, \*nextSibling, \*parent;

//指向第一个孩子，下一个兄弟节点，指向父亲节点的指针

**公有操作：**

 TreeNode(string value = "NULL", TreeNode\* fc = NULL, TreeNode\* ns = NULL, TreeNode\* pa = NULL)

        : name(value)

        , firstChild(fc)

        , nextSibling(ns)

        , parent(pa)

    {

    }*//构造函数*

**链表类（LinkList）**

**私有成员：**

 TreeNode \*root, \*current;

*//根节点和当前节点，设置当前节点以便于进行各种操作*

**公有操作：**

Tree() { root = current = NULL; }*//构造函数*

    ~Tree() { }*//析构函数*

    void Initial();*//初始化界面以及根节点*

    bool Root();*//使当前节点为根节点*

    bool IsEmpty() { *return* root == NULL; }*//判断当前节点是否为空*

    bool FirstChild();*//使当前节点为当前节点的第一个子女节点*

    bool NextSibling();*//使当前节点为当前节点的下一个兄弟节点*

    bool Parent();*//使当前节点为当前节点的双亲节点*

    bool Find(string Target);*//前序遍历查找名字为Target的节点*

    bool RemovesubTree();*//移除某个节点以及其所有的子女节点*

    bool Add();*//给某个节点添加若干个子女*

    bool Insert();*//给某个节点添加一个子女*

    bool Change\_Name();*//改变某个节点的名字*

 bool Search();*//查找某个节点的双亲节点以及所有的子节点*

## 2.4 系统设计

系统首先实现对屏幕的初始化，完成对树的创建和输入数据工作，然后根据用户所输入的操作码（operation）执行树对应的成员函数。

# 3 实现

## 3.1 完善家谱功能的实现

### 3.1.1 完善家谱功能流程图



### 3.1.2 完善家谱功能核心代码

cin >> child[0];

    TreeNode\* first\_child = new TreeNode(child[0]);*//先插入长子节点*

current->firstChild = first\_child;

//current是指当前节点，此时为父节点

    first\_child->parent = current;

    TreeNode\* p = current;

    current = first\_child;

*for* (int i = 1; i < numbers; i++) {*//插入兄弟节点*

        cin >> child[i];

        TreeNode\* newchild = new TreeNode(child[i]);

        newchild->parent = p;

        current->nextSibling = newchild;

        current = newchild;

    }

    cout << Parent << "的第一代子孙是：";

    p = p->firstChild;

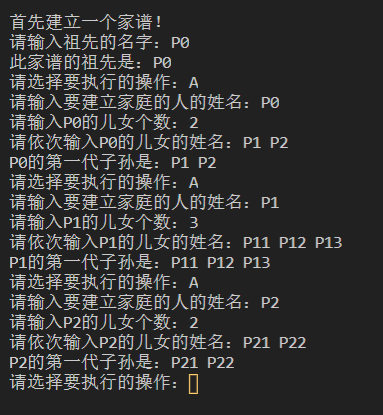
*while* (p) {

        cout << p->name << " ";

        p = p->nextSibling;

 }

### 3.1.3 插入功能截屏示例



## 3.2 添加家庭成员功能的实现

### 3.2.1 添加家庭成员功能流程图



### 3.2.2 添加家庭成员功能核心代码

cin >> child;

*if* (current->firstChild == NULL) {

*//无长子就变成长子，否则插入最后一个兄弟节点的末尾*

        TreeNode\* first\_child = new TreeNode(child);

        current->firstChild = first\_child;

        first\_child->parent = current;

        current = first\_child;

    }

*else* {

        current = current->firstChild;

*while* (current->nextSibling != NULL)

            current = current->nextSibling;

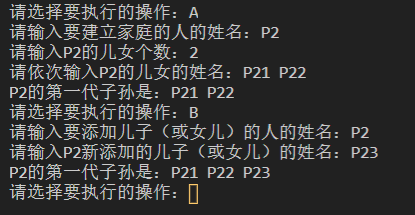
        TreeNode\* next\_sibling = new TreeNode(child);

        current->nextSibling = next\_sibling;

        next\_sibling->parent = p;

    }

### 3.2.3 删除功能截屏示例



## 3.3 解散局部家庭功能的实现

### 3.3.1 解散局部家庭功能流程图



### 3.3.2 解散局部家庭功能核心代码

*if* (Find(par)) {*//令current指针再次指向名字为par的节点*

*if* (current == root) {

            root->firstChild = NULL;

            root = NULL;

            current = NULL;

            delete root;

*return* true;

        }

*if* (Parent()) {*//current指针指向目标节点的双亲节点*

            TreeNode\* q = current->firstChild;

*if* (q->name == par) {*//删除长子节点*

                current->firstChild = q->nextSibling;

                q->parent = NULL;

                q->nextSibling = NULL;

                delete q;

*return* true;

            }

            TreeNode\* r = q->nextSibling;

*while* (r) {*//在兄弟链表中删除该节点*

*if* (r->name == par) {

                    q->nextSibling = r->nextSibling;

                    r->nextSibling = NULL;

                    r->parent = NULL;

                    delete r;

*return* true;

                }

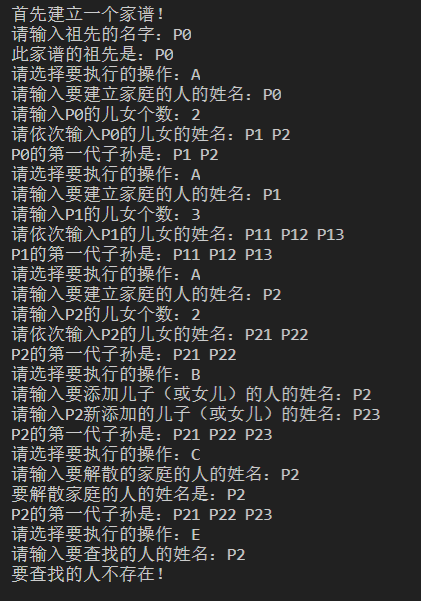
                q = r;

                r = r->nextSibling;

            }

        }

### 3.3.3 解散局部家庭功能截图示例



## 3.4 更改家庭成员姓名功能的实现

### 3.4.1 更改家庭成员姓名功能流程图



### 3.4.2 更改家庭成员姓名功能核心代码

string member, new\_name;

    cout << "请输入要更改姓名的人的目前姓名：";

    cin >> member;

*if* (!Find(member)) {//current指向要更改姓名的成员

        cout << "要更改姓名的人不存在！" << endl;

*return* false;

    }

    cout << "请输入更改后的姓名：";

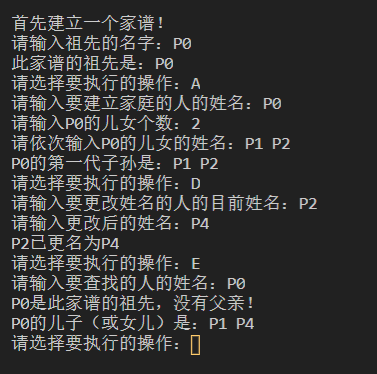
    cin >> new\_name;

    current->name = new\_name;

    cout << member << "已更名为" << new\_name;

*return* true;

### 3.4.3 更改家庭成员姓名功能截屏示例



## 3.5 查找功能的实现

### 3.5.1 查找功能流程图



### 3.5.2 查找功能核心代码

*if* (root->name == Target) {//Target为要查找的名字

        current = root;

*return* true;

    }

    current = root;

*if* (FirstChild()) {//从长子节点开始

        p = current;

        Stack S;

        S.Push(q);

*while* (p != NULL) {

*if* (p->name == Target) {

                current = p;

*return* true;

            }

*if* (p->nextSibling != NULL)

                S.Push(p->nextSibling);//兄弟节点入栈

*if* (p->firstChild != NULL)

                p = p->firstChild;

*else*

                S.Pop(p);

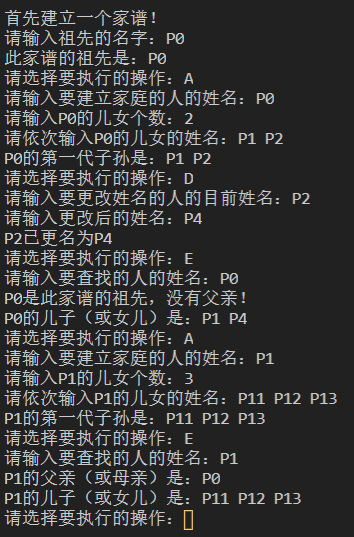
        }

*return* false;

    } *else*

*return* false;

### 3.5.3 查找功能截屏示例



## 3.6 总体系统的实现

### 3.6.1 总体系统流程图



### 3.6.2 总体系统核心代码

    FamilyTree.Initial();

*while* (1) {

        cout << endl;

        cout << "请选择要执行的操作：";

        cin >> option[0];

*switch* (option[0]) {

*case* 'A':

            FamilyTree.Add();

*break*;

*case* 'B':

            FamilyTree.Insert();

*break*;

*case* 'C':

            FamilyTree.RemovesubTree();

*break*;

*case* 'D':

            FamilyTree.Change\_Name();

*break*;

*case* 'E':

            FamilyTree.Search();

*break*;

*case* 'F':

            system("pause");

*return* 0;

*default*:

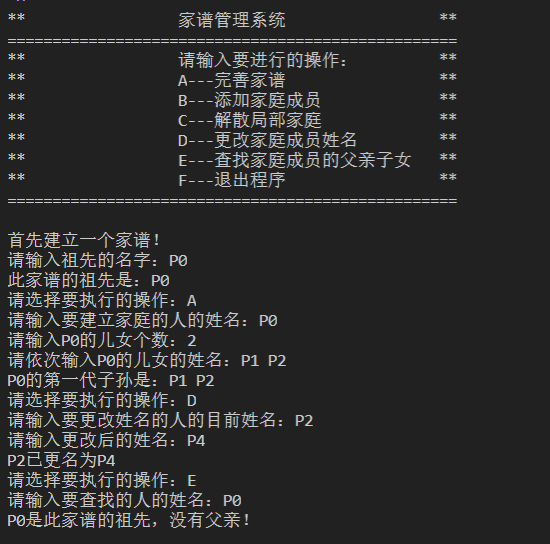
            cout << "输入非法！请输入合适的选项！" << endl;

*break*;

        }

    }

### 3.6.3 总体系统截屏示例



# 4 测试

## 4.1 功能测试

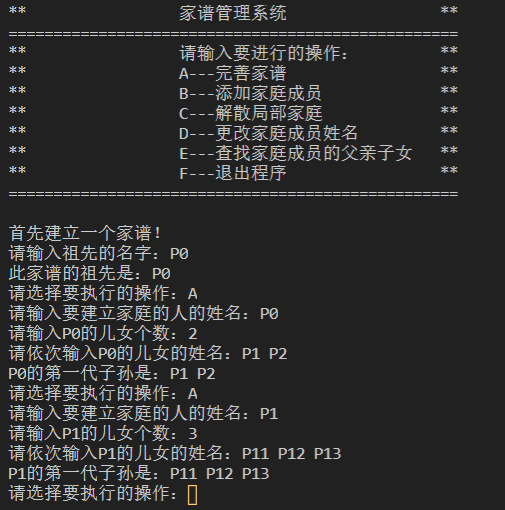
### 4.1.1 完善家谱功能测试

**测试用例**：P0子女P1 P2 P1子女P11 P12 P13

**预期结果**：

P1 P2 P11 P12 P13

**实验结果**



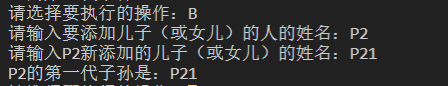
### 4.1.2 添加家庭成员功能测试

**测试用例：**添加P21作为P2的子女

**预期结果：**

输出P2的子女为P21

**实验结果：**



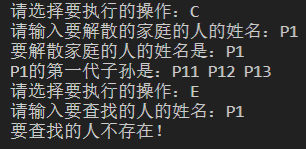
### 4.1.3 解散局部家庭功能测试

**测试用例：**解散P1家庭

**预期结果：**

P1家庭被解散

**实验结果：**



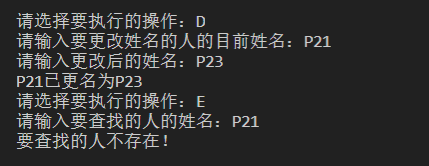
### 4.1.4 更改家庭成员姓名功能测试

**测试用例：P21->P23**

**预期结果：**

P21->P23

**实验结果：**



### 4.1.5 查找功能测试

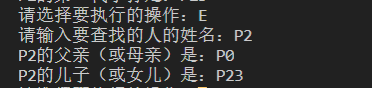
**测试用例：**查找名字为P2的家庭成员

**预期结果：**

P2父亲为P0

P2孩子为P23

**实验结果：**



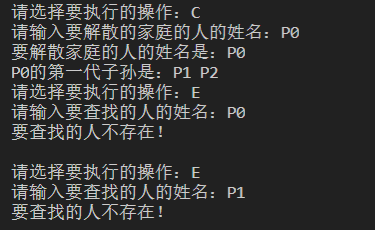
## 4.2 边界测试

### 4.2.1 删除根结点

**测试用例：**删除根结点

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

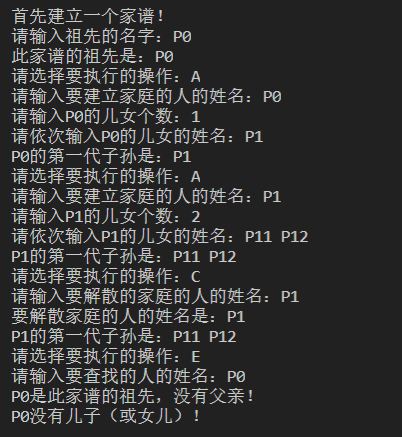


### 4.2.2 删除后子女节点为空

**测试用例：**删除前根节点只有一个家庭，删除后只剩下根节点

**预期结果：**程序正常运行，不崩溃。

**实验结果：**



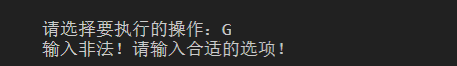
## 4.3 出错测试

### 4.3.1 操作码错误

**测试用例：**输入操作码错误

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**



### 4.3.2 完善家谱、添加家庭成员、解散局部家庭、更改姓名时成员不存在

**测试用例：输入不存在的家庭成员**

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

