项目说明文档

数据结构课程设计

——家谱管理系统

作 者 姓 名： 谢宇翔

学 号： 1951708

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](#_Toc25242)

[1.1 背景分析 1](#_Toc18281)

[1.2 功能分析 1](#_Toc8504)

[2 设计 1](#_Toc3336)

[2.1 数据结构设计 1](#_Toc26640)

[2.2 类结构设计 1](#_Toc7421)

[2.3 成员与操作设计 2](#_Toc2013)

[2.4 系统设计 5](#_Toc22330)

[3 实现 5](#_Toc29277)

[3.1 建立族谱 5](#_Toc6579)

[3.1.1 功能流程图 5](#_Toc15389)

[3.1.2 核心代码 5](#_Toc17288)

[3.2 完善家庭功能 7](#_Toc31105)

[3.2.1 功能流程图 7](#_Toc27232)

[3.2.2 完善家庭功能核心代码 7](#_Toc14609)

[3.2.3 完善家庭功能截屏示例 9](#_Toc7167)

[3.3 添加家庭成员功能的实现 10](#_Toc11622)

[3.3.1 功能流程图 10](#_Toc5394)

[3.3.2 添加家庭成员功能核心代码 10](#_Toc27420)

[3.3.3 添加家庭成员功能截图示例 12](#_Toc24513)

[3.4 解散局部家庭成员功能的实现 13](#_Toc951)

[3.4.1 解散局部家庭成员功能流程图 13](#_Toc8085)

[3.4.2 解散局部家庭成员功能核心代码 13](#_Toc19558)

[3.4.3 解散局部家庭成员功能截屏示例 15](#_Toc15784)

[3.5 更改家庭成员名字功能的实现 16](#_Toc22518)

[3.5.1 更改家庭成员功能流程图 16](#_Toc32572)

[3.5.2 更改家庭成员功能核心代码 16](#_Toc9611)

[3.5.3 更改家庭成员名字功能截屏示例 17](#_Toc8749)

[3.6 总体系统的实现 18](#_Toc20937)

[3.6.1 总体系统流程图 18](#_Toc13789)

[3.6.2 总体系统核心代码 18](#_Toc18068)

[3.6.3 总体系统截屏示例 20](#_Toc23688)

[4 测试 21](#_Toc1429)

[4.1 功能测试 21](#_Toc16414)

[4.1.1 完善家庭功能测试 21](#_Toc19494)

[4.1.2 添加家庭成员功能测试 22](#_Toc7961)

[4.1.3 解散局部家庭功能测试 22](#_Toc846)

[4.1.4 修改成员信息功能测试 22](#_Toc14575)

[4.2 出错测试 23](#_Toc17657)

[4.2.1 关键码错误 23](#_Toc24327)

[4.2.2 对不在族谱的进行操作 24](#_Toc18318)

[4.2.3 完善家庭时人数错误 24](#_Toc22104)

# 1 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。比起纸质版的家谱，电子版的家谱更容易保存和阅读读取。

## 1.2 功能分析

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。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 数据结构设计

本程序主要用了个树类存储tree族谱，节点结构体node存储每个成员的信息，用左孩子右兄弟的存储方式存储多叉树的家谱结构

## 2.2 类结构设计

树类利用左孩子右兄弟的方式存储节点，并在树内进行增删改查，便可完成族谱的所有操作。

## 2.3 成员与操作设计

**结点类（node）**

struct node

{

string data;

node\* firstchild=NULL;

node\* bro=NULL;

};

**树类（tree）**

**私有成员：**

node\* root, \* current;

**私有函数：**

void funcA();//功能A完善家谱

void funcB();//功能B添加家庭成员

void funcC();//功能C解散家庭

void funcD();//功能D改名字

void freefamily(node \*p);//递归解散家庭

bool find(string want);//在树中找data为want的节点并将current指向它。

bool parent();//在树中寻找current节点的双亲节点，并将current指向它。

bool findparent(node\* root, node\* p);//在以root为根的树中找节点p的双亲节点，并将current指向它

bool isempty();//检查树非空

bool find(node\* root, string want);//在以root为根的树中找data为want的节点，并将current指向它

**共有函数:**

void menu();//菜单

bool tree::isempty()//检查树非空

{

if (root == NULL)

{

return 1;

}

return 0;

}

bool tree::find(node\* root, string want)//在树中找data为want的节点并将current指向它。

{

bool result = false;

if (root->data == want)//搜索成功

{

result = true;

current = root;

}

else//搜索失败

{

node\* q = root->firstchild;

while (q != NULL && !(result = find(q, want)))//去长子链遍历

q = q->bro;//去兄弟链

}

return result;

}

bool tree::findparent(node\* root, node\* p)//在以root为根的树中找节点p的双亲节点，并将current指向它

{

node\* q = root->firstchild;

bool succ;

while (q != NULL && q != p)

{

if (succ = findparent(q, p) == 1)//循环兄弟链

return succ;

q = q->bro;

}

if (q != NULL && q == p)//找到了

{

current = root;

return 1;

}

else//没找到

{

current = NULL;

return 0;

}

}

bool tree::parent()//在树中寻找current节点的双亲节点，并将current指向它。

{

node\* p = current;

if (current == NULL || current == root)//空树或者根为当前节点，返回0

{

current = NULL;

return 0;

}

return findparent(root, p);

}

bool tree::find(string want)//在树中找data为want的节点并将current指向它。

{

if (isempty())

return false;

return find(root, want);

}

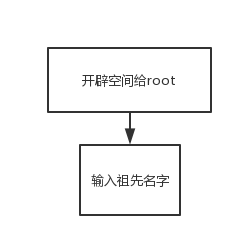
## 2.4 系统设计

首先创建树类T，调用menu函数，先建立族谱，然后根据选择进入不同的函数进行操作。

# 3 实现

## 3.1 建立族谱

### 3.1.1 功能流程图



### 3.1.2 核心代码

cout << "首先建立一个家谱" << endl;

cout << "请输入祖先的名字:" << endl;

root = new node;

if (root == NULL)

{

cout << "开辟空间失败！" << endl;

return;

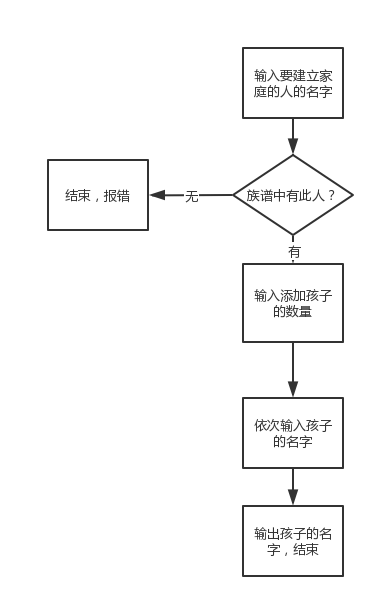
}

cin >> root->data;

cout << "此家族的祖先为:" << root->data << endl;

## 3.2 完善家庭功能

### 3.2.1 功能流程图



### 3.2.2 完善家庭功能核心代码

void tree::funcA()

{

cout << "请输入要建立家庭的成员名字：";

string want;

cin >> want;

bool isfind=find(want);

if (isfind == 0)//找data为want的节点是否存在

{

cout << "族谱无此人!" << endl;

return;

}

int num = 0;

while (1)

{

cout << "请输入" << current->data << "的子女个数:";

cin >> num;

if (!cin.good())

{

cout << "输入错误请重新输入！" << endl;

cin.clear();

cin.ignore(100, '\n');

}

else if (num <= 0)

{

cout << "请输入一个正整数！" << endl;

cin.ignore(1000, '\n');

}

else

break;

}

cout << "请依次输入" << current->data << "的子女的名字:";

for (int i = 0; i < num; i++)//依次输入子女的名字

{

if (i == 0)//长子

{

current->firstchild = new node;

if (current->firstchild == NULL)

{

cout << "开辟空间失败！" << endl;

return;

}

current = current->firstchild;

cin >> current->data;

}

else//长子的兄弟

{

current->bro = new node;

if (current->bro == NULL)

{

cout << "开辟空间失败！" << endl;

return;

}

current = current->bro;

cin >> current->data;

}

}

//后面的代码是起输出子女名字的功能

parent();

cout << current->data << "的第一代子女为：";

current = current->firstchild;

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

while (current->bro != NULL)

{

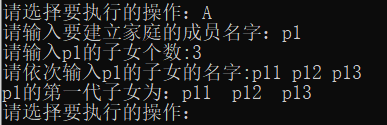
current = current->bro;

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

}

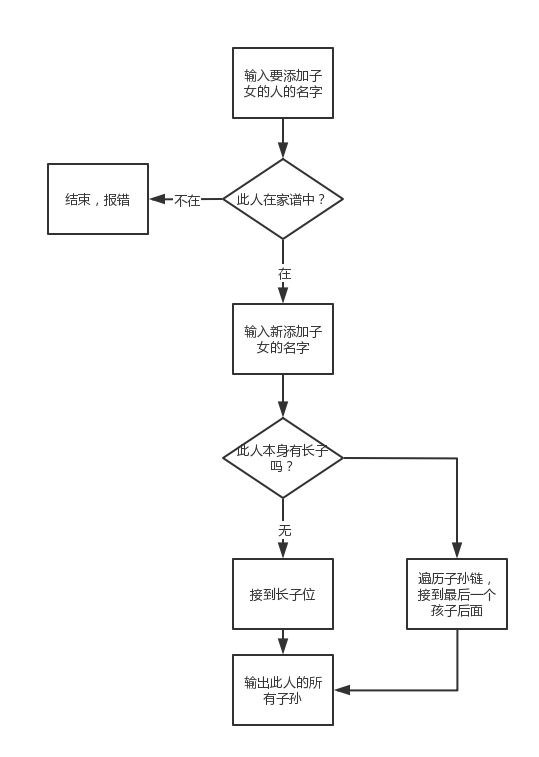
}

### 3.2.3 完善家庭功能截屏示例



## 3.3 添加家庭成员功能的实现

### 3.3.1 功能流程图



### 3.3.2 添加家庭成员功能核心代码

void tree::funcB()

{

cout << "请输入要添加子女的人的名字:";

string want;

cin >> want;

bool isfind = find(want);

if (isfind == 0)

{

cout << "族谱无此人!" << endl;

return;

}

cout << "请输入" << want << "新添加的子女的名字:";

string add;

cin >> add;

if (current->firstchild == NULL)//他没有孩子，就把孩子接到长子的位置上

{

current->firstchild = new node;

if (current->firstchild == NULL)

{

cout << "开辟空间失败！" << endl;

return;

}

current = current->firstchild;

current->data = add;

}

else//有孩子，就把孩子接在长子的兄弟链上

{

current = current->firstchild;

while (current->bro != NULL)//循环兄弟链

{

current = current->bro;

}

current->bro = new node;

if (current->bro == NULL)

{

cout << "开辟空间失败！" << endl;

return;

}

current = current->bro;

current->data = add;

}

//输出

parent();

current = current->firstchild;

cout << want << "的第一代子孙是:";

while (current != NULL)

{

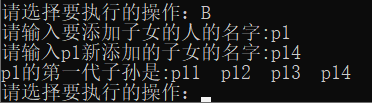
cout << current->data << " ";

current = current->bro;

}

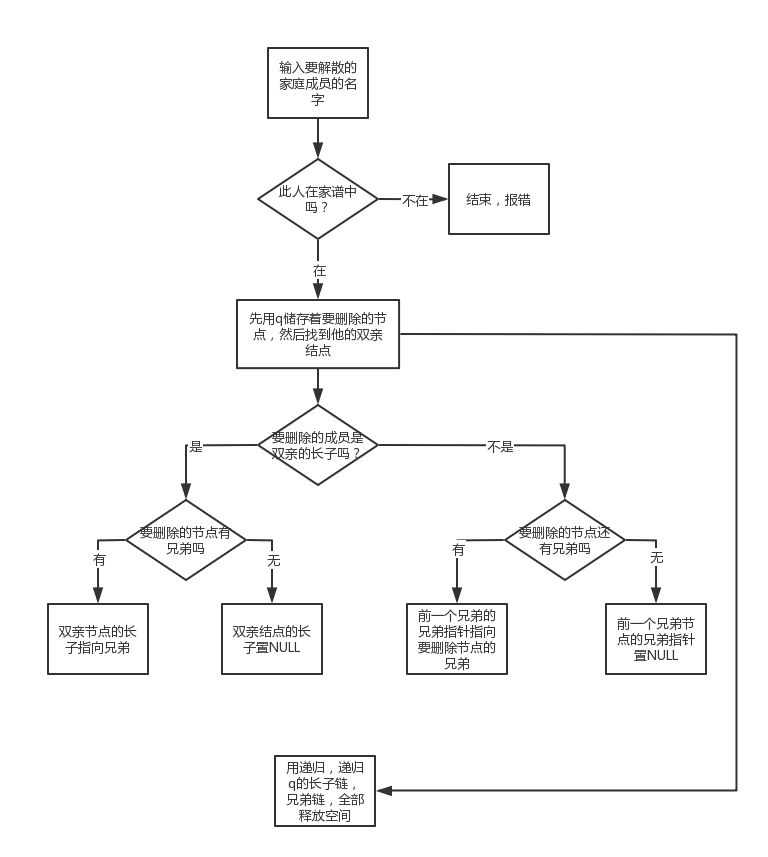
}

### 3.3.3 添加家庭成员功能截图示例



## 3.4 解散局部家庭成员功能的实现

### 3.4.1 解散局部家庭成员功能流程图



### 3.4.2 解散局部家庭成员功能核心代码

void tree::funcC()

{

cout << "请输入要解散家庭的人的姓名:";

string want;

cin >> want;

bool isfind = find(want);

if (isfind == 0)

{

cout << "族谱无此人!" << endl;

return;

}

cout << "要解散家庭的人是:" << want;

parent();

node\* p = current->firstchild;

node\* q;//q储存要删除的节点

if (p->data == want)//删除的人是长子

{

if (current->firstchild->bro == NULL)//没有其他孩子了，就把双亲节点的长子节点置NULL

{

current->firstchild = NULL;

}

else//还有孩子，长子的兄弟接替

current->firstchild = current->firstchild->bro;

q = p;

}

else//删除的人不是长子

{

while (p->bro->data != want)//循环兄弟链

{

p = p->bro;

}

if (p->bro->bro == NULL)//要删除的节点无兄弟了

{

q = p->bro;

p->bro = NULL;

}

else

{

q = p->bro;

p->bro = p->bro->bro;

}

}

freefamily(q->firstchild);

delete q;

}

void tree::freefamily(node \*p)

{

if (p->firstchild != NULL)//去长子链

{

freefamily(p->firstchild);

}

if (p->bro != NULL)//去兄弟链

{

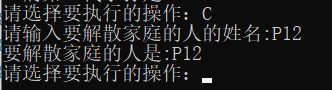
freefamily(p->bro);

}

delete p;

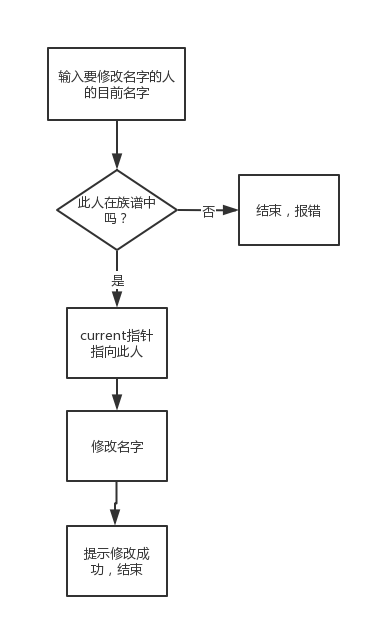
}

### 3.4.3 解散局部家庭成员功能截屏示例



## 3.5 更改家庭成员名字功能的实现

### 3.5.1 更改家庭成员功能流程图



### 3.5.2 更改家庭成员功能核心代码

void tree::funcD()//改名很简单

{

cout << "请输入要更改姓名的人的目前名字:";

string want;

cin >> want;

bool isfind = find(want);

if (isfind == 0)

{

cout << "族谱无此人!" << endl;

return;

}

cout << "请输入更改后的名字:";

string newname;

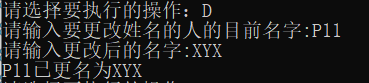
cin >> newname;

current->data = newname;

cout << want << "已更名为" << newname;

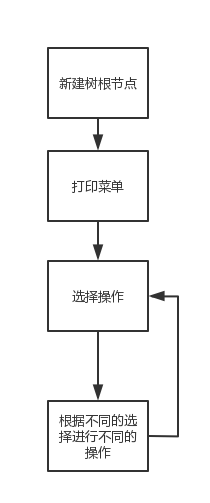
}

### 3.5.3 更改家庭成员名字功能截屏示例



## 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

void tree::menu()

{

cout << "\*\* 家谱管理系统 \*\*" << endl;

cout << "=============================================" << endl;

cout << "\*\* 请选择以下的操作 \*\*" << endl;

cout << "\*\* A---完善家庭 \*\*" << endl;

cout << "\*\* B---添加家庭成员 \*\*" << endl;

cout << "\*\* C---解散局部成员 \*\*" << endl;

cout << "\*\* D---更改家庭成员名字 \*\*" << endl;

cout << "\*\* E---退出程序 \*\*" << endl;

cout << "=============================================" << endl;

cout << "首先建立一个家谱" << endl;

cout << "请输入祖先的名字:" << endl;

root = new node;

if (root == NULL)

{

cout << "开辟空间失败！" << endl;

return;

}

cin >> root->data;

cout << "此家族的祖先为:" << root->data << endl;

while (1)

{

char choice;

while (1)

{

cout << endl << "请选择要执行的操作：";

cin >> choice;

if (!cin.good() || choice < 'A' || choice>'E')

{

cout << "输入错误！请重新输入" << endl;

cin.clear();//重置cin流

cin.ignore(100, '\n');//清空输入缓冲区

}

else

break;

}

switch (choice)

{

case 'A':

case 'a':

funcA();

break;

case 'B':

case 'b':

funcB();

break;

case 'C':

case 'c':

funcC();

break;

case'd':

case'D':

funcD();

break;

case 'e':

case'E':

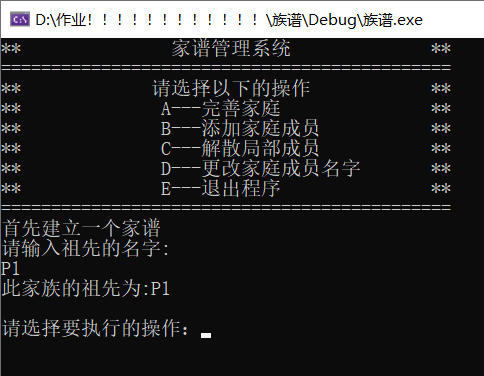
return;

}

}

}

### 3.6.3 总体系统截屏示例



# 4 测试

## 4.1 功能测试

祖先已经设为p1

### 4.1.1 完善家庭功能测试

**测试用例**：A

p1

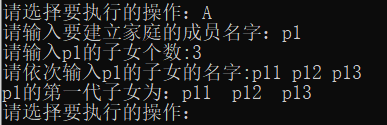
3

p11 p12 p13

**预期结果**：

p1的第一代子女为p11 p12 p13

**实验结果**



### 4.1.2 添加家庭成员功能测试

**测试用例：**

B

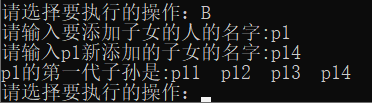
p1

p14

**预期结果：**

p1的第一代子孙为p11 p12 p13 p14

**实验结果：**



### 4.1.3 解散局部家庭功能测试

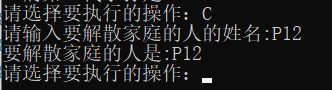
**测试用例：**

**B**

**P12**

**预期结果：**

要解散的家庭的人是:p12

**实验结果：**

### 4.1.4 修改成员信息功能测试

**测试用例：**

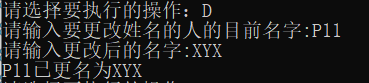
**D**

**P11**

**XYX**

**预期结果：**

P11已更名为XYX

**实验结果：**

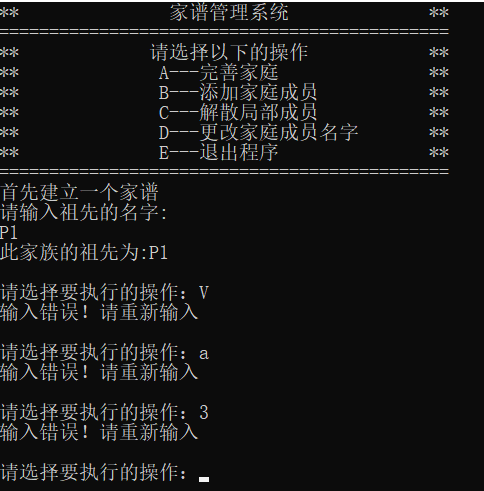
## 4.2 出错测试

### 4.2.1 关键码错误

**测试用例：**关键码错误输入

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**

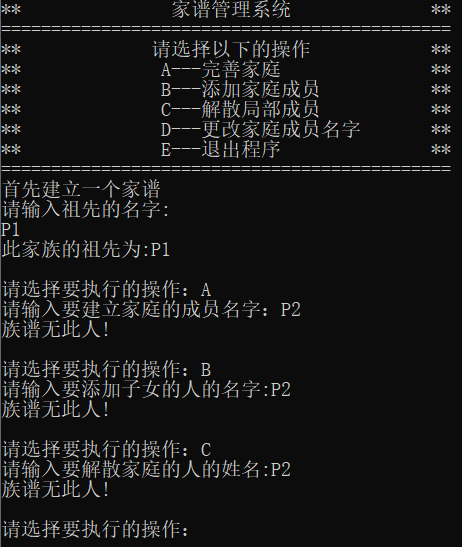


### 4.2.2 对不在族谱的进行操作

**测试用例：输入不在族谱中的人的姓名**

**预期结果：**程序正常运行，不崩溃。

**实验结果：**



### 4.2.3 完善家庭时人数错误

**测试用例：完善家庭时人数错误**

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

