项目说明文档

数据结构课程设计

——家谱管理系统

作 者 姓 名： 李圣

学 号： 1952723

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji Universit

目录

[1 分析 1](#_Toc12611)

[1.1 背景分析 1](#_Toc2074)

[1.2 功能分析 1](#_Toc18370)

[2 设计 1](#_Toc5037)

[2.1 文件结构设计 1](#_Toc18678)

[2.2 数据结构设计 1](#_Toc28914)

[2.3 类结构设计 2](#_Toc10404)

[2.4 成员与函数设计 2](#_Toc7431)

[3 实现 7](#_Toc11813)

[3.1 查找（Find）功能的实现 7](#_Toc11752)

[3.1.1 查找（Find）功能流程图 7](#_Toc23420)

[3.1.2 查找（Find）功能核心代码 8](#_Toc14747)

[3.2 插入功能的实现 9](#_Toc25183)

[3.2.1 插入功能流程图 9](#_Toc30621)

[3.2.2 插入功能核心代码 10](#_Toc15984)

[3.2.3 插入功能截屏示例 10](#_Toc30399)

[3.3 修改功能的实现 11](#_Toc1578)

[3.3.1 修改功能流程图 11](#_Toc21203)

[3.3.2 修改功能核心代码 11](#_Toc20562)

[3.3.3 修改功能截屏示例 12](#_Toc28886)

[3.4 删除功能的实现 12](#_Toc14836)

[3.4.1 删除功能流程图 12](#_Toc15753)

[3.4.2 删除功能核心代码 13](#_Toc14637)

[3.4.3 删除功能截屏示例 13](#_Toc2442)

[4 测试 13](#_Toc2892)

[4.1 鲁棒性测试 13](#_Toc14183)

[4.1.1 正常案例测试 13](#_Toc16669)

[4.1.2 错误指令测试 16](#_Toc21333)

[4.1.3 输入错误测试 16](#_Toc16978)

[4.1.4 成员不存在测试 17](#_Toc10589)

# 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。因此，实现一个家谱管理系统是及其有必要的。本项目拟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，删除家族成员等功能。

## 功能分析

家谱管理系统至少应该具有的基本功能：

1. 家谱成员信息的建立
2. 家谱成员信息的查找
3. 家谱成员信息的插入
4. 家谱成员信息的修改
5. 家谱成员信息的删除

# 设计

## 文件结构设计

表达式求解的过程中需要使用Vector，因此该项目包含两个文件，如下：

main.cpp

MyVector.h

其中main.cpp是主程序，且main.cpp调用了MyVector.h

## 数据结构设计

考虑到家谱的要求，对家谱数据结构的实现采用树的结构。

## 类结构设计

考虑到功能需要，除了Vector相关的类以外，拟定设计以下类：

class MemberNode

class FamilyTree

class Guide

其中MemberNode是数据节点类，FamilyTree是家谱树类，Guide是输入输出指引类

## 成员与函数设计

**MyVectorIterator类：**

template<typename val\_type>

using iterator = MyVectorIterator;

using pointer = val\_type\*;

using reference = val\_type&;

**私有成员：**

pointer \_ptr;

int \_pos;

**公有成员：**

MyVectorIterator(pointer ptr = nullptr, int pos = -1)；

~MyVectorIterator()；

int getPos()；

bool operator==(const iterator& it)const；

bool operator!=(const iterator& it)const；

iterator& operator++()；

iterator& operator++(int)；

iterator& operator--()；

iterator& operator--(int)；

reference operator\*()；

pointer operator->()；

**MyVector类：**

template<typename val\_type>

using vector = MyVector<val\_type>;

using reference = val\_type&;

using iterator = MyVectorIterator<val\_type>;

**私有成员：**

int \_size;

int \_capacity;

val\_type\* \_data;

**公有成员：**

MyVector(int capacity = DEFAULT\_CAPACITY)；

MyVector(const vector& vec)；

~MyVector()；

bool empty()；

int size()；

int length()；

int capacity()；

void resize(int newcapacity)；

void insert(iterator pos, const val\_type& val)；

val\_type erase(iterator pos)；

void push\_back(val\_type val)；

val\_type pop\_back()；

vector& operator=(vector& vec)；

vector& operator=(vector&& vec)；

iterator find(val\_type val)；

reference at(int pos)；

reference operator[](int pos)；

reference front()；

reference back()；

iterator begin()；

iterator end()；

void clear()；

**MemberNode类：**

**私有成员：**

string \_name;

MyVector<MemberNode\*> \_children;

**公有成员：**

MemberNode(string name = string());

string name();

void setName(string name);

MyVector<MemberNode\*>getChildren();

MemberNode\* getChild(string name);

void addChild(string name);

void show(char gap = ' ');

void showChildren();

void showAll();

void deleteAll();

void deleteChildren();

MemberNode\* find(string name, int depth = 0);

**FamilyTree类：**

**私有成员：**

MemberNode\* \_root;

**公有成员：**

FamilyTree();

~FamilyTree();

void create(string name);

MemberNode\* getRoot();

void show();

void clear();

MemberNode\* find(string name);

**Guide类：**

**私有成员：**

FamilyTree \_tree;

**公有成员：**

Guide();

MemberNode\* find();

void setAncestor();

void setAncestor();

void addChildren();

void addChild();

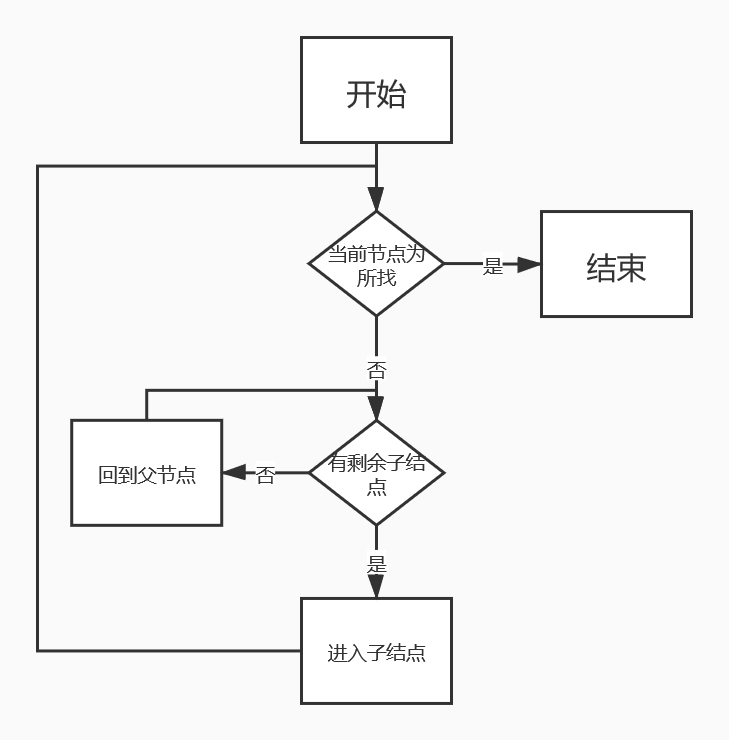
void deleteChildren();

void changeName();

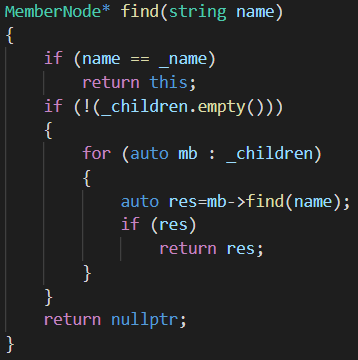
# 实现

## 查找（Find）功能的实现

### 查找（Find）功能流程图

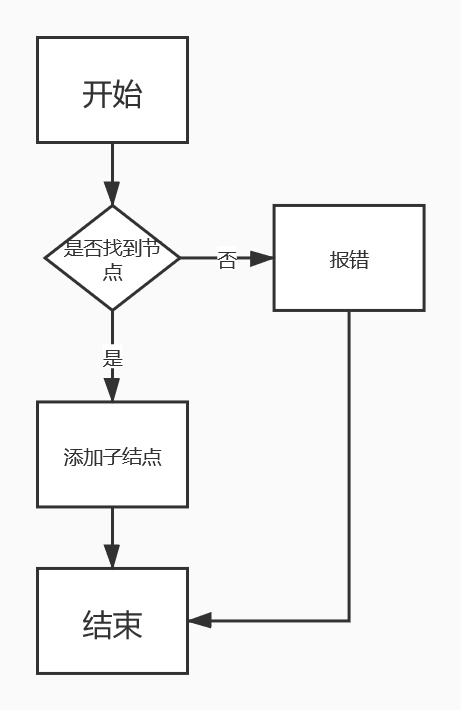


### 查找（Find）功能核心代码



## 插入功能的实现

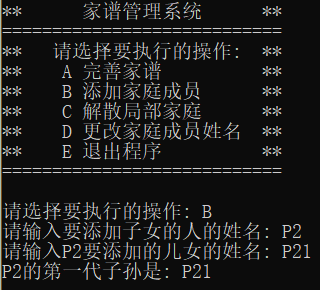
### 插入功能流程图



### 插入功能核心代码

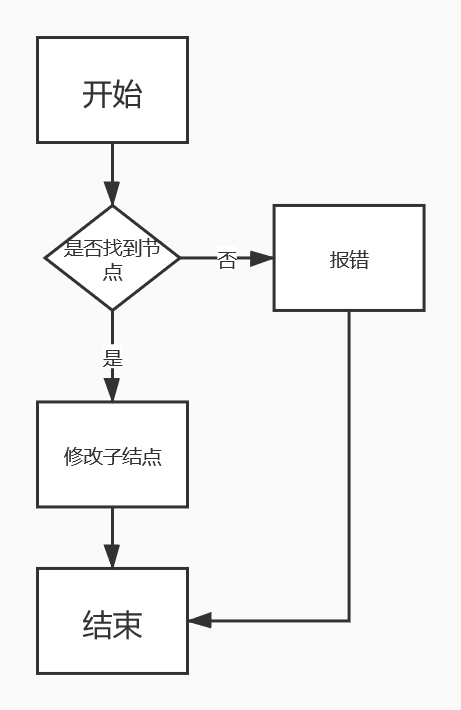


### 插入功能截屏示例

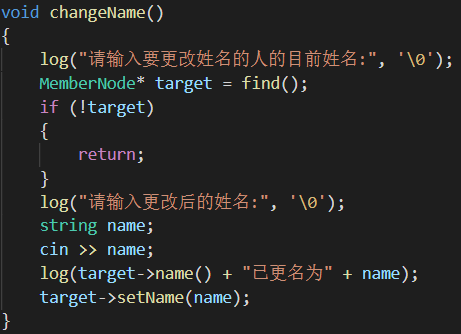


## 修改功能的实现

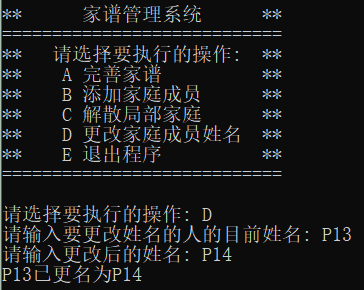
### 修改功能流程图



### 修改功能核心代码

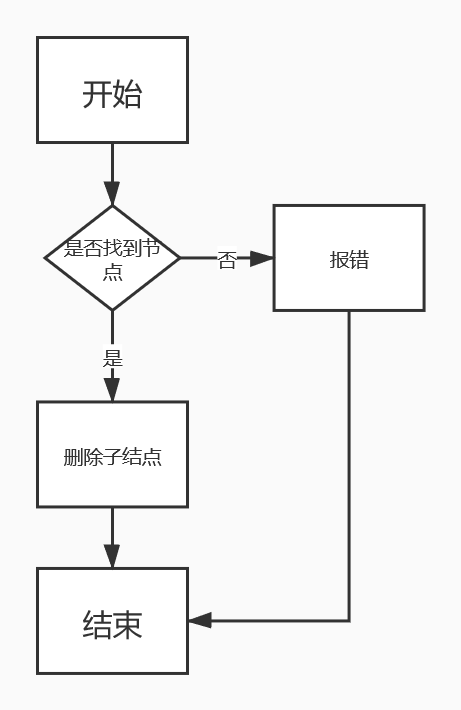


### 修改功能截屏示例

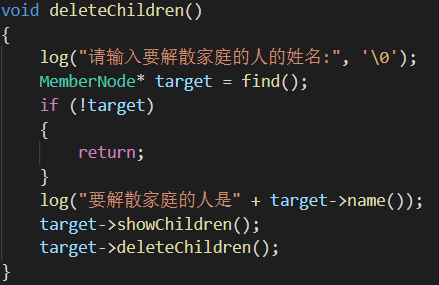


## 删除功能的实现

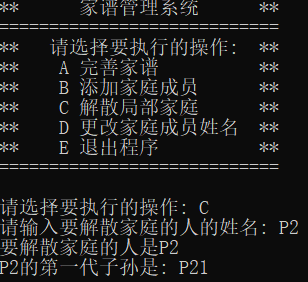
### 删除功能流程图



### 删除功能核心代码



### 删除功能截屏示例



# 测试

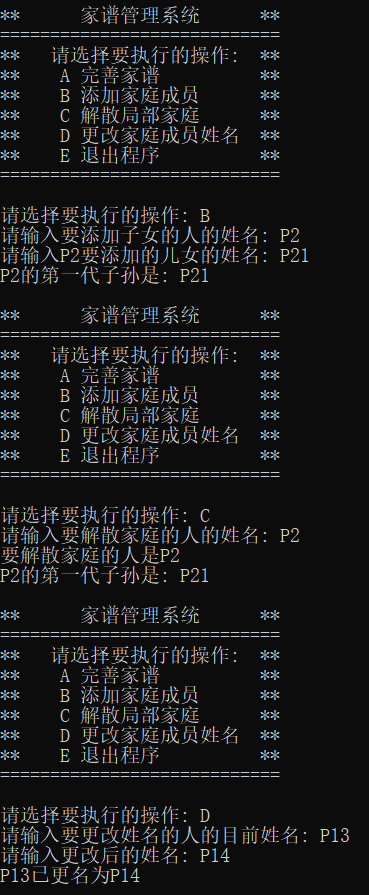
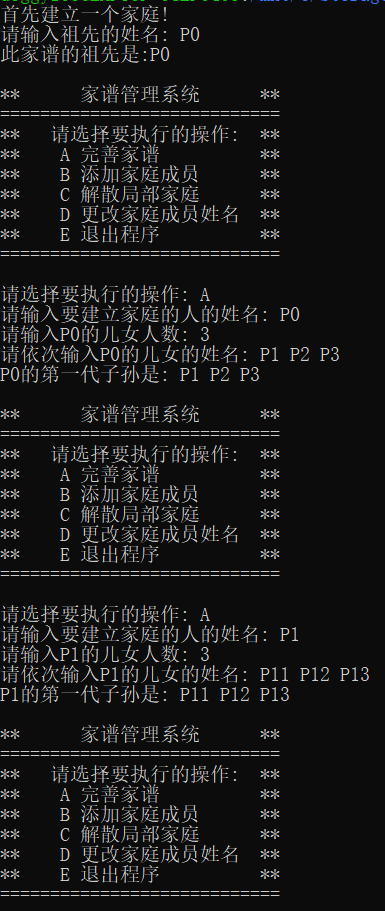
## 鲁棒性测试

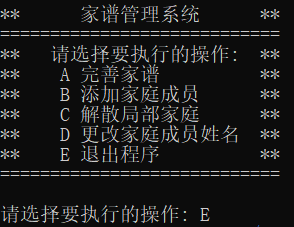
### 正常案例测试

**测试用例：**一般情况

**预期结果：**对应的正确结果。

**实验结果：**



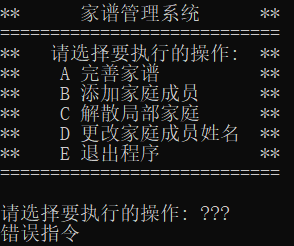


### 错误指令测试

**测试用例：**错误指令

**预期结果：**进行提示，程序继续运行

**实验结果：**

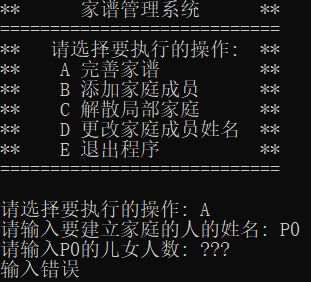


### 输入错误测试

**测试用例：**输入错误

**预期结果：**进行提示，程序继续运行

**实验结果：**



### 成员不存在测试

**测试用例：**成员不存在

**预期结果：**进行提示，程序继续运行

**实验结果：**

