**项目说明文档**

**数据结构课程设计**

**——家谱管理系统**

作者姓名：香宁雨

学号：1954098

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.题目要求及功能分析 3](#_Toc58843523)

[2.设计 4](#_Toc58843524)

[2.1数据结构设计 4](#_Toc58843525)

[2.2类结构设计 4](#_Toc58843526)

[2.3类的成员与方法 4](#_Toc58843527)

[2.4测试函数 6](#_Toc58843528)

[3.实现 9](#_Toc58843529)

[3.1insert(Tree\* node, string name)函数 9](#_Toc58843530)

[3.2modify(Tree\* root, string new\_name)函数 9](#_Toc58843531)

[3.3find(Tree\* node, string name) 10](#_Toc58843532)

[3.4find\_parent(Tree\* node, string name)函数 10](#_Toc58843533)

[3.5del(Tree\* node)函数 11](#_Toc58843534)

[3.6del\_child\_tree(Tree\* node\_parent, Tree\* node)函数 11](#_Toc58843535)

[3.7print()函数 12](#_Toc58843536)

[3.8完善家谱树 13](#_Toc58843537)

[3.8.1流程图 13](#_Toc58843538)

[3.8.2代码实现 14](#_Toc58843539)

[3.9添加子女 15](#_Toc58843540)

[3.9.1流程图 15](#_Toc58843541)

[3.9.2代码实现 16](#_Toc58843542)

[3.10摧毁某人家谱树 17](#_Toc58843543)

[3.10.1流程图 17](#_Toc58843544)

[3.10.2代码实现 18](#_Toc58843545)

[3.11修改姓名 19](#_Toc58843546)

[3.11.1流程图 19](#_Toc58843547)

[3.11.2代码实现 20](#_Toc58843548)

[3.12退出 20](#_Toc58843549)

[3.13错误检测 20](#_Toc58843550)

[4.输出测试 21](#_Toc58843551)

[4.1样例输入输出测试 21](#_Toc58843552)

[4.2错误输入输出 22](#_Toc58843553)

# 1.题目要求及功能分析

项目简介：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。

功能要求：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。

功能分析：本项目不仅需要完成对家谱成员信息的建立，查找，插入，修改，删除等功能，而且要体现家谱成员之间的联系，在生活中对家谱成员的描述主要是通过“树”这个结构，也就是常说的家谱树，因此本项目也采用树这个数据结构进行对家谱成员信息的管理。

# 2.设计

## 2.1数据结构设计

根据题目描述选择最合适的数据结构进行数据的分析与处理，题目中表达为家谱树，故此项目采用树这种数据结构进行数据处理。

## 2.2类结构设计

树是一种数据结构，它是由n(n>=1)个有限结点组成一个具有层次关系的集合。把它叫做“树”是因为它看起来像一棵倒挂的树，也就是说它是根朝上，而叶朝下的。它具有以下的特点：

每个结点有零个或多个子结点；没有父结点的结点称为根结点；每一个非根结点有且只有一个父结点；除了根结点外，每个子结点可以分为多个不相交的子树。

本项目中需体现父子关系及兄弟关系，所以采用左子女右兄弟的方式进行数据储存，即左子树储存此结点的儿女，右子树储存此节点的兄弟，实现对家谱树的存储。

## 2.3类的成员与方法

Tree类

## 2.4测试函数







# 3.实现

## 3.1insert(Tree\* node, string name)函数

进行孩子的插入，第一个参数为当前结点，第二个参数为孩子的名字，插入的结点处于当前节点的左子树上。

## 3.2modify(Tree\* root, string new\_name)函数

进行当前节点名字的修改。

## 3.3find(Tree\* node, string name)

以当前节点的子树为搜索范围，对姓名为name的结点进行搜索。



## 3.4find\_parent(Tree\* node, string name)函数

以当前结点的字数为搜索范围，对姓名为name的结点搜索其父节点。



## 3.5del(Tree\* node)函数

因为动态分配了内存，并且无析构函数，所以通过递归调用del函数进行内存的释放。同时，本函数也可以进行子树的删除。



## 3.6del\_child\_tree(Tree\* node\_parent, Tree\* node)函数

一个节点的左右子树分别为子女和兄弟，如果只删除一个孩子则需要对孩子的兄弟进行合并处理，本函数实现只删除一个孩子并对其他孩子进行合并处理的方法。

## 3.7print()函数

对当前节点的孩子进行输出。



## 3.8完善家谱树

### 3.8.1流程图



### 3.8.2代码实现



## 3.9添加子女

### 3.9.1流程图



### 3.9.2代码实现



## 3.10摧毁某人家谱树

### 3.10.1流程图



### 3.10.2代码实现



## 3.11修改姓名

### 3.11.1流程图



### 3.11.2代码实现



## 3.12退出

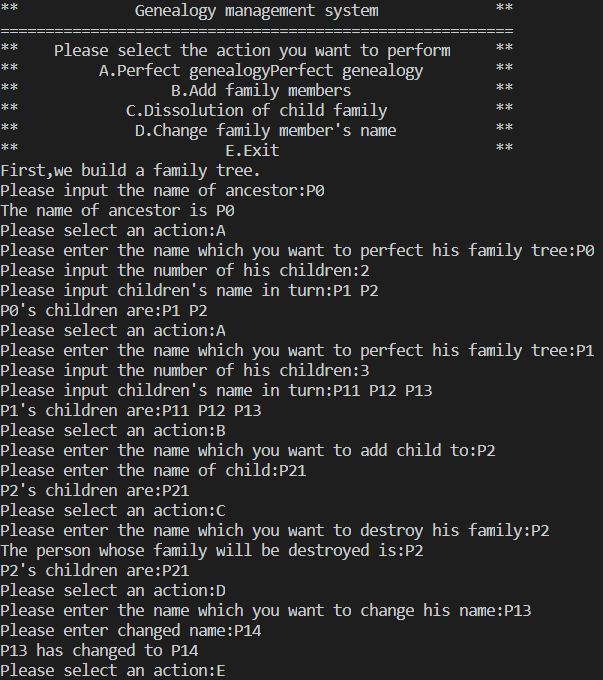


## 3.13错误检测



# 4.输出测试

## 4.1样例输入输出测试



## 4.2错误输入输出

