组号: 1

图片包含 游戏机, 画

描述已自动生成

上海大学计算机工程与科学学院

**实 验 报 告**

（数据结构1）

学 期：

组 长：

学 号：

指导教师：

成绩评定： （教师填写）

二〇二三年四月六日

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **小组信息** | | | | |
| 登记序号 | 姓名 | 学号 | 分工 | 签名 |
| 7 | 余 | 2112 | 代码撰写 |  |
| 7 | 周 | 2112 | 报告撰写 |  |
| 7 | 付 | 2112 | 代码撰写 |  |
| 7 | 茅 | 2112 | PPT制作 |  |
| 7 | 张 | 2112 | 代码的测试 |  |

|  |  |
| --- | --- |
| **实验概述** | |
| 实验零 | （熟悉上机环境、进度安排、评分制度；确定小组成员） |
| 实验一 | 家谱管理系统设计 |
| 实验二 |  |
| 实验三 |  |
| 实验四 |  |

# 实验一

**一、实验题目**

家谱管理系统设计

**二、实验内容**

**1、任务目标**

   基于树的相关知识和代码实现，开发一个简单的家谱管理系统

**2、任务描述**

  家谱管理系统可以清楚地查询家族成员的详细信息，典型家谱图如下图所示。成员的信息一般包含这些内容：姓名、出生日期、婚姻状况（未婚、已婚、离异、丧偶等）、住址、目前状况（健在或身故）、死亡日期（若已身故）等，也可以附加其他信息。要求：

   1）设计合理的数据结构存储家谱树中各成员信息（一般定义结构体），以及成员之间的关系；

   2）系统提供菜单，可以实现插入、查询、修改、删除等功能，详见下方功能列表；

   3）自行定义家谱信息的文件结构，用以家谱信息的加载和存储；

   4）输入数据：初始数据由文件输入；

   5）输出数据：系统运行结束时，保存更新后的家谱信息到输出文件。

**3、功能列表**

   1）从文件输入信息，建立初始家谱树；

   2）自行设计如何显示家谱树；

   3）根据代号n，显示第n代所有人的姓名和人数；

    ​4）按姓名查询，输出成员信息（包括其本人、上一代和孩子的信息，以及在家谱中的代数）；

   5）按出生日期查询成员名单；

   6）输入两人姓名，确定其关系；

   7）给某成员添加孩子；

   8）删除某成员；

   9）修改某成员信息。

**三、解决方案**

**1、算法设计**

在本实验中使用C++语言进行编程，存储家谱信息采用了“树”的数据结构，根据实验要求，我们设计了一个FamilyTree类，用于管理家谱信息，并使用结构体TreeNode和MemberNode分别用于表示家谱树中的成员节点和各个成员节点所包含的个人信息（姓名（Name），出生日期（Birthday），婚姻状况（Marital），当前状况（isAlive），死亡日期（Death\_date），性别（isFemale），上一代姓名（Parent））。

|  |
| --- |
| 类和结构体 的数据成员设计 |
| //class FamilyTree  public:  TreeNode\* root; // 根节点  //struct TreeNode  // 结构体，表示家谱树中的一个成员节点  MemberNode member; // 成员信息  vector<TreeNode\*> childs; // 用容器储存孩子节点  int depth; // 当前成员的代号  //struct MemberNode  // 结构体，用于储存成员信息  string Name; // 姓名  int Birthday; // 出生日期  Status Marital; // 婚姻状况  bool isAlive; // 当前状况，由于只有“健在”与“身故”两种状态，选用布尔值  int Death\_date; // 死亡日期  bool isFemale; // 性别，由于只有“男”和“女”两种性别，同样选用布尔值  string Parent; // 上一代姓名 |

主要采用了面向对象的编程思想，将家谱管理系统的各个功能封装成了不同的类和结构体，并通过这些类和结构体进行数据的存储和操作。下面对其算法思想进行详细展开叙述。

1. 家谱树的构建

家谱树的构建过程主要通过读取文件信息，然后创建成员节点，并将其插入到家谱树中。具体实现过程中，首先读取文件信息，然后创建家谱树的根节点。接着，一条条读取家谱信息，根据成员信息创建成员节点，并将其插入到家谱树中。在插入节点时，需要根据节点的父节点找到其在家谱树中的位置，并将其作为父节点的子节点插入到家谱树中。这个过程一直持续到文件信息全部读取完成，家谱树构建完成。

1. 家谱树的遍历

家谱树的遍历过程主要通过递归实现，从根节点开始遍历整棵家谱树，每访问一个节点就进行一定的操作。具体实现过程中，遍历过程根据遍历方式的不同可以分为前序遍历、中序遍历和后序遍历。在遍历过程中，我们可以通过递归方式依次访问家谱树的每个节点，对每个节点进行一定的操作，例如输出节点信息、统计节点信息等。

1. 家谱树的查找

家谱树的查找过程主要通过递归实现，从根节点开始遍历整棵家谱树，每访问一个节点就进行一定的判断。具体实现过程中，查找过程可以根据不同的查找方式进行分类，例如按姓名查找、按出生日期查找、查询两人关系等。在查找过程中，我们可以通过递归方式依次访问家谱树的每个节点，对每个节点进行一定的判断，例如比较节点的姓名或出生日期，找到符合条件的节点后进行一定的操作。

1. 文件读写

文件读写过程主要通过文件流实现，将家谱信息存储到文件中或从文件中读取家谱信息。具体实现过程中，我们可以通过文件流打开文件，并使用文件流的读取和写入操作进行文件的读取和写入。在读取文件信息时，我们可以通过文件流每次读取一行信息，并将其解析为成员信息，然后创建成员节点并将其插入到家谱树中。在写入文件信息时，我们可以通过文件流将家谱树中的所有成员信息依次写入到文件中。

1. 其他算法

除了上述，家谱管理系统中还涉及到了其他一些算法，例如容器的使用、字符串比较和日期比较等。容器的使用主要体现在使用vector和map来存储和操作家谱树中的成员信息和节点信息。字符串比较和日期比较主要涉及到按姓名查找和按出生日期查找等功能，我们可以通过字符串比较和日期比较来确定符合条件的节点，然后进行一定的操作。

**3、实验结果**

**4、算法分析**

在本实验的家谱管理系统设计中涉及到了家谱树的构建、遍历、查找等操作，以及文件读写等操作

1. 家谱树的构建

在家谱树的构建过程中，我们需要读取文件中的家谱信息，并根据这些信息构建家谱树。这个过程可以看作是一个插入节点的过程，每读取一条家谱信息就需要创建一个成员节点，并将其插入到家谱树中。因此，这个过程的时间复杂度为O(nlogn)，其中n是家谱信息的数量，logn是家谱树的高度。

1. 家谱树的遍历

家谱树的遍历是家谱管理系统中的核心操作之一，它涉及到了家谱树的遍历、统计等操作。在遍历家谱树时，我们一般采用递归的方式，从根节点开始遍历整棵树，每访问一个节点就进行一定的操作。因此，家谱树的遍历时间复杂度为O(n)，其中n是家谱树的节点数量。

1. 家谱树的查找

家谱树的查找是家谱管理系统中的另一个核心操作，它涉及到了按姓名查找、按出生日期查找、查询两人关系等操作。在家谱树的查找过程中，我们一般采用递归的方式，从根节点开始遍历整棵树，每访问一个节点就进行一定的判断。因此，家谱树的查找时间复杂度为O(n)，其中n是家谱树的节点数量。

1. 文件读写

家谱管理系统中还涉及到了文件读写操作，用于将家谱信息存储到文件中或从文件中读取家谱信息。在文件读写操作中，我们一般采用文件流的方式进行操作，时间复杂度主要取决于文件的大小和硬盘的读写速度。

综上所述，家谱管理系统的时间复杂度主要取决于家谱信息的数量、家谱树的高度和文件的大小。在实际应用中，我们需要根据具体情况进行优化，提高系统的性能和效率。

**5、总结与心得**

通过本次实验，我们学习了如何使用面向对象的思想设计并实现一个家谱管理系统。在实现过程中，我们需要对家谱管理系统进行功能划分，并将其封装成不同的类和结构体，以实现数据的存储和操作。同时，我们还需要进行算法设计和优化，以提高系统的性能和效率，此过程中我们掌握了递归、容器、文件流等技术的使用，提高了我们的编程能力和实践能力。同时，我们也深入理解了面向对象的思想，更好地掌握了C++语言的使用。在实现过程中，我们还遇到了一些难点，例如如何将文件中的家谱信息插入到家谱树中，如何进行家谱树的遍历和查找等。通过不断地思考和实践，我们最终成功地解决了这些难点，并实现了一个完整的家谱管理系统。此外，我们还深刻地认识到了面向对象编程的优势。通过将家谱管理系统的各个功能封装成不同的类和结构体，我们可以更好地组织和管理代码，提高代码的可读性和可维护性。