安徽工程大学

Anhui Polyte Chnic University

《数据结构》课程设计说明书

课 设 题 目：	文件系统模拟
学 生 姓 名：	武新纪
学　　号：	3190707121
专 业 班 级：	人工智能 191
学　　院：	计算机与信息学院
指 导 老 师：	王陆林

《数据结构》课程设计任务书

一、课程设计题目：文件系统模拟

二、设计任务和内容

本课程设计的内容共有 1 道大题。为应用类题目，涉及线性表、栈、队列、树和图以及查找和排序等主要知识点。

具体任务包括：

1、熟悉课程设计的目的、要求与过程。

2、掌握课程设计题目的设计内容。

3、针对选题进行需求分析，特别是功能分析，划出功能图。

4、算法分析与设计，写出每个模块的设计思想，画出每个算法的流程图或 N-S 图。

5、编写源程序。

6、静态走查程序和上机调试程序，写出调试报告。

7、综合程序。当有多个功能演示时，可以以命令菜单的方式，供用户选择每题实现的功能，并进行调试。即通过菜单的方式实现功能的选择。

8、书写上述文档和撰写课程设计说明书。

三、设计步骤和要求

1、设计步骤

首先进行相关资料查阅和学习、调查研究，了解基本的业务流程和系统功能、数据需求。然后结合软件工程的理论完成设计任务，主要包括需求分析、总体设计以及详细设计、编码与调试部分。

2、课程设计说明书要求与格式

(1) 基本要求：

① 能反映完成了上述设计内容要求。

② 要求撰写不少于 5000 个文字（20 页）的文档。

③ 文档中至少要包括：主要功能分析、系统功能结构图、算法流程图、系统运行与调试截图、总结、参考文献。

④ 课程设计说明书中，用户界面设计、程序运行与调试需要截图。

(2) 文档格式要求

① 封面

② 本课程设计任务书（须有指导教师签名及日期）

③ 摘要（中文和英文）

④ 目录

⑤ 引言

⑥ 正文（分章、层次等，每一章从新一页开始）**（每章内容参考如下：详见模板范文）**

．第 1 章 概述

包括课题的课程设计的目的与要求或者要解决什么问题、选题的内容、设计的过程、开发环境等内容。

．第 2 章 需求分析

描述完成哪些功能或者解决哪些问题，以及每一个功能或者问题所完成的任务。

．第 3 章 系统设计

3.1 总体设计

画出系统功能结构图或者总的思路框架图或者能从总体上描述清楚总体设计的文字内容。

． 3.2 详细设计

画出主要算法的流程图。

确定程序中所用的主要数据的数据类型，选择所用的数据结构，如：结构体、数组、顺序表、链表、顺序栈、链栈、顺序队、链队、树、图等数据结构。

列出程序中用到的所有函数或者算法的名称、功能，调用关系。

界面设计

第 4 章 编码

列出程序中结构体类型的定义。

列出程序中全局变量、程序中通用的符号常量。

列出自己设计的所有算法函数的原型，并说明算法函数的功能、参数的作用和含义。

给出主要功能的代码并有适当的注释。其余代码不要放在正文中 ，可以放在最后附录部分。

第 5 章 运行与调试

运行、调试程序并截图，进行说明调试的目的与结果。初步学会软件测试。用尽可能多的测试用例，对程序进行测试。

第 6 章 小结

对本次课程设计过程中的收获、体会以及不足或存在的问题进 行总结。

参考文献

附录：第 4 章没有列出的代码，附加到附录部分。

(3) 装订要求

课程设计报告必须按要求和顺序装订，封面的指导老师必须和课表中课程设计指导老师一致，不一定是理论课老师。具体有老师安排指定。

教师签名：

日期：2021 年 1 月 20 日

摘要

近些年来，随着计算机技术的发展，我们的生活变得越来越便利，越来越多的人开始接触计算机，去学习计算机，每个人都不想在即将到来的人工智能时代中被淘汰。所以我们要去学习人工智能，而学习人工智能的第一步就是学习计算机的相关知识，在学习了基本的编程语言之后，要想解决更多的问题，我们就需要学习数据结构，去学习如何通过计算机去解决我们身边的问题。

数据结构，作为计算机专业的必修课，是非常的重要的。大一的时候，我们学习了编程语言，我们可以用这些编程语言去做一些简单的程序，比如打印九九乘法表，计算元素的和或阶乘，但是对于更复杂的问题，如迷宫和表达式的求解等问题，我们可能就做不出来。是用编程语言做不出来？还是我们做不出来呢？大一的时候我尝试过参加一些算法比赛，以及在力扣上刷题，但是穷极我的智力，也做不出来几道题，我一直在想为什么。直到这学期学了数据结构，我才明白，我没有学习数据结构。数据结构是计算机多年发展的结晶，总结了几种逻辑结构的形式——表、树和图等。通过这些逻辑结构，结合编程语言本身的特点，我们就可以使用编程语言去实现我们想要的功能。我们学习的数据结构，是使用 C 语言实现的，但是数据结构可以使用各种编程语言实现，它包含两部分，一部分是逻辑结构，另一部分是存储结构。不同的编程语言，提供不同的数据类型，我们使用相关的数据类型去实现存储结构，而逻辑结构，是一种逻辑关系，并不依赖于编程语言，所以使用任何一门编程语言都可以实现。所以学习 C 语言数据结构的时候，我们的重点应该是数据直接的逻辑关系，而且我们要学会举一反三，那样当我们需要使用其他编程语言实现的时候也能够很容易。

作为一名人工智能专业的学生，我已经学习了 C 语言、C++语言和 Python 语言，最近又结束了对数据结构的学习，我非常渴望了解关于计算机的更多知识，更期望能够自己编写软件，甚至编写系统。在编程语言和数据结构的基础上，我选择了这次课程设计的主题——文件系统模拟。选择这个选题一方面可以提升我对编程语言和数据结构知识的掌握能力，让已经掌握的知识更佳精通，没有掌握的知识及时掌握；另一方面，也为我接下来认识和学习操作系统提供良好的铺垫。操作系统应该也是计算机专业的必修课，操作系统中用到了很多的数据结构，如表、栈、队列和树等，要想在学习操作系统是游刃有余的掌握，甚至做出自己的操作系统，熟练的掌握数据结构是很重要的。

这次课程设计的目的并不只在于学习知识和设计出自己的作品，更在于我们了学习解决问题的思维和方法，为我们以后学习打下坚实的基础。

关键词：计算机、数据结构、逻辑结构、操作系统、文件系统模拟

Abstract

In recent years, with the development of computer technology, our life has become more and more convenient, more and more people begin to contact with computers, to learn computer, everyone does not want to be eliminated in the coming era of artificial intelligence.Therefore, we need to learn artificial intelligence, and the first step of learning artificial intelligence is to learn the knowledge related to computers. After learning the basic programming language, we need to learn the data structure to solve more problems, and learn how to solve the problems around us through computers.

Data structure, as a required course for computer science majors, is very important.In the freshman year, we learned programming languages, we can use these programming languages to do some simple programs, such as printing multiplication tables, calculating the sum or factorial of elements, but for more complex problems, such as maze and expression solution, we may not be able to do.Can't you do it in a programming language?Or can't we do it?When I was a freshman, I tried to take part in some algorithm contests and brush the questions on the force button, but I was too poor in intelligence to do several questions. I kept thinking why.It wasn't until I learned about data structures this semester that I realized I wasn't learning about data structures.Data structure is the crystallization of computer development over the years, and summarizes the forms of several logical structures -- tables, trees, graphs, etc.Through these logical structures, combined with the characteristics of the programming language itself, we can use the programming language to achieve the functions we want.The data structure we are studying is implemented in C, but the data structure can be implemented in various programming languages. It consists of two parts, one is logical structure, the other is storage structure.Different programming languages provide different data types. We use relevant data types to implement storage structure. Logical structure is a logical relationship and does not depend on programming language, so it can be implemented in any programming language.Therefore, when learning C language data structure, we should focus on the direct logical relationship of the data, and we should learn to draw inferences, so that we can use other programming languages to implement it easily in the future.

As a student majoring in artificial intelligence, I have learned C language, C++ language and Python language. Recently, I have finished my study on data structure. I am eager to learn more about computers, and I expect to be able to write software or even systems by myself.Based on the programming language and data structure, I chose the theme of this course design -- file system simulation.On the one hand, choosing this topic can improve my ability to master the knowledge of programming language and data structure, so that I can better master the knowledge I have already mastered and timely grasp the knowledge I haven't mastered.On the other hand, it also provides a good foundation for me to understand and learn the operating system in the future.The operating system should also be a required course for computer majors. The operating system uses a lot of data structures, such as tables, stacks, queues and trees, etc. It is very important to master the data structure proficiently if you want to learn the operating system and even make your own operating system.

The purpose of this course design is not only to learn knowledge and design their own works, but also to learn the relevant ideas to solve problems, so as to lay a solid foundation for our future study.

Keywords: computer, data structure, logical structure, operating system, file system simulation

目录

引言

大一的时候，我已经 C 语言去完成很多管理系统，如成绩管理系统、报名管理系统、校运动会项目管理系统，那些管理系统基本上都是使用单链表完成的，比较简单。在学习数据结构之后，我意识到不仅仅有表这种简单的数据结构，还有树和图等复杂的数据结构，这些结构是一种逻辑关系，如树是一对多的关系，而图是多对多的关系，从某种程度上来说，树也是一种图。这些本来是一些逻辑关系，但是和计算机技术相结合，就可以解决很多的问题，如计算机的文件管理系统就是一种树形结构，导航的地图是一种图形结构，这些结构在计算机中存储之后，可以对其中存储的数据进行查找、遍历和修改等操作，从而完成我们对数据的处理。因为每种数据结构有自己的特点，选择适合的数据结构可以更快的获得我们想要的结果。

本次课程设计选择了树这种存储结构，并使用树这种结构去实现文件系统的模拟，使用的是命令行的方式，和 Linux 文件系统类似，可以实现对文件或目录的创建、删除和查看等操作，以模拟简单的文件系统。选择这个题目，一来可以巩固编程语言和数据结构的相关知识，二来为接下来认识和学习操作系统提供良好的铺垫。因为知识所限，并不能够做到对文件系统的真实模拟，即在磁盘中对文件进行操作，但是随着学习的深入，或许我也能够做出一个完善的文件操作系统，甚至是一整套操作系统。

本次课程设计报告总共有六章，第一章是概述，明确选题的目的与要求，以及开发所使用的软硬件环境。第二章和第三章是选择问题后对问题的具体分析与设计；第四章是程序的部分关键代码，包含定义的数据类型以及相关功能实现的函数；第五章是程序运行和调试之后的结果；第六章则是完成这次课程设计任务后的一些总结和感悟。

做这次课程设计，从选择问题到编写代码，再到撰写报告，花费了很多的时间和精力，但是也因此收获了很多，对于数据结构知识有了更进一步的掌握，这对我以后的学习和工作都有很大的好处，所以非常感谢老师给了我们这次做数据结构课程设计的机会。在完成这次的课程设计的过程中，我查阅了很多的资料，所查阅的部分资料写在了本报告的参考文献部分，在此对这些参考文献的作者表示衷心的感谢。程序的所有代码均放在此报告的附录部分，如有错误，欢迎大家指正。

第 1 章 概述

1.1 目的与要求:

1.1.1 目的：

建立相应的数据结构，利用程序实现文件或目录的创建、查看和删除等操作。综合使用所学过的 C 或 C++语言程序设计知识，掌握结构化程序设计的基本思路和方法，开发小型应用程序，培养数据处理的综合能力。

《文件系统模拟》的设计与实现，主要利用 C++语言和数据结构的知识，创建一个文件系统模拟的程序，为以后操作系统的学习打下坚实的基础。结合课程的教学内容循序渐进地进行设计方面的时间训练，以获软件开发经验，了解软件编程的思想，对以后的学习增加更多的兴趣。

1.1.2 要求：

使用高级语言程序实现一个文件系统，模拟文件系统的各种操作，如文件的创建、删除、读取和写入等功能，并可以对文件进行查看和统计。文件操作中所涉及的文件夹和文件，仅在内存中用树结构模拟，而非真正的文件系统。树中结点要包含必要的信息，如：文件或文件夹名称、区分文件或文件夹的标识、文件创建日期等。

1.2 选题的内容:

编写程序，模拟命令行的常用文件系统管理命令，具体要求如下：

1)根路径为 ROOT。首次进入模拟器时，提示符为“ROOT>”，“>”左侧为当前路径，可在“>”后输入下述各命令。

2)切换到当前路径下的某文件夹：cd 文件夹。如“cd music”，若当前路径下存在 music 文件夹，则提示符变为“当前路径\music>”，若不存在，则提示。

3)切换到当前路径的上级文件夹：cd ..。

4)在任意路径下切换回根路径 ROOT：cd \。

5)列出当前路径下的全部文件夹和文件：dir。

6)在当前路径下新建文件夹：md 文件夹名称。

7)在当前路径下新建文件：mf 文件名称。

8)删除当前路径下的某文件或文件夹（及其下所有文件夹及文件）：del 文件或文件夹名称。

1.3 设计的过程:

程序的功能分析，根据综合实践题目的描述和要求，确定程序要实现的功能，然后将这些功能分解和细化，划分成不同的层次并确定各层功能的上下级关系。数据结构设计：确定程序中所用数据的结构类型，数据元素之间的关系，选择所用的数据结构，如数组，链表等。对程序中用到的结构体数据定义其结构体类型。定义程序中使用的全局变量，外部变量等。定义程序中通用的符号常量。进行算法分析与设计。根据功能分析确定使用的主要函数，然后进行函数定义，包括函数名称，功能，参数和函数返回值类

型等。函数间的调用关系要明确，选择合适的调用函数方式实现调用。最后进行编程以及调试运行。

最初使用 C 语言实现，写完了各种函数之后，考虑到学习过了 C++，便使用了 C++面向对象的思想，定义一个结构体，该结构体时文件结点类型，包含了文件的一些信息以及表示文件之间逻辑关系的指针。然后定义一个文件类，用于实现文件操作功能的函数封装为类的成员函数，一些在函数中能够用到的全局变量作为文件类的私有成员，只有文件的成员函数可以访问，保证了数据的安全性。在主函数中，通过文件类创建一个文件对象，以实现文件系统的模拟功能。

1.4 开发环境:

系统环境:Window10 64 位专业版
硬件环境:AMD R7 4800U 16G 内存
软件环境:VSCode+MinGW64

第 2 章　需求分析

2.1 功能需求分析

- 设计并实现一个目录树，根节点为 root，首次进入模拟器时，提示符为“ROOT>”，“>”左侧为当前路径，可在“>”后输入下述各命令。
- 设计并实现一个改变目录的功能，用来把当前目录切换到上一层目录或者当前目录的子目录。该函数使用 cd 命令调用，其中’\’为根目录，’..’为当前目录的上级目录，使用 cd 命令可以完成切换。
- 设计并实现一个目录显示显示功能，使用 dir 命令调用，列出当前路径下的全部文件夹和文件。
- 设计并实现创建目录和文件的功能，其中创建文件时可以对文件写入内容
- 设计并实现删除文件和目录的功能
- 设计并实现对文件读写的功能

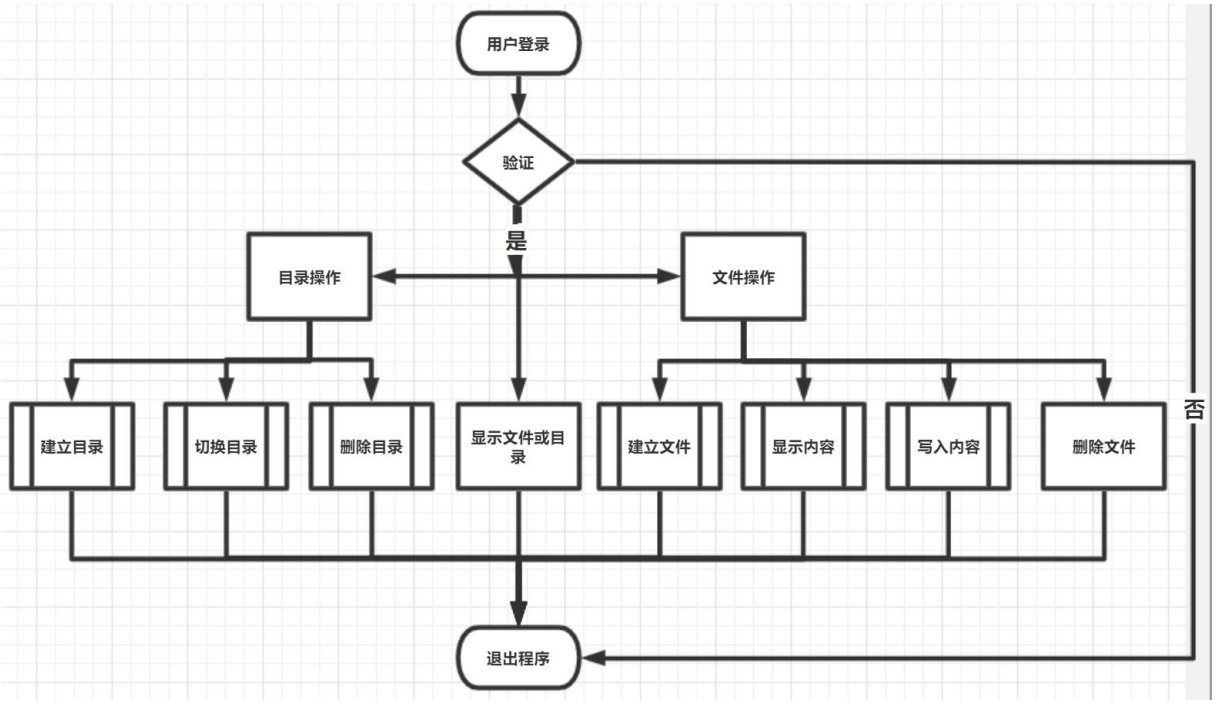
第 3 章　系统设计

3.1　总体设计

用户若想进入系统，需要先输入账号和密码，账号密码设计程序时均设置为 root，当用户输入账号和密码都正确的时候才允许进入系统，执行相应的操作。该系统是文件模拟的系统，主要功能就是对文件和目录的操作，对于目录，有创建目录、切换目录和删除目录三个操作，对于文件，可以建立和删除文件，也可以对文件进行读写，同样，该系统也提供显示文件和目录的功能。该系统主要使用树这种数据结构建立的，使用的存储结构时改良版的兄弟孩子结点存储，每个结点都拥有孩子指针和双亲指针，也有前一个兄弟结点的指针和后一个兄弟节点的指针，因为这种结构，使得从所建立的文件树的任何一个结点出发，都能够访问到该树上的任意一个结点，这也就保证了从用户可以从任意目录出发，访问所有的文件和目录。

该系统使用了面向对象的思想，把主要的函数变量都封装成了一个类，在主函数中通过类生成的对象来调用相关的函数，保证了程序的安全性，保证程序运行中的数据不会被篡改，同时也为程序复用提供了可能。

3.2 流程图



第 4 章　编码

4.1　定义的结构体类型

```
typedef struct file
{
    string filename;     //文件或目录名
    int isdir;           //是否是目录,是的话为 1,不是为 0
    int opened;          //是否打开
    string filecontent; //文件内容
    file *parent;        //双亲节点
    file *child;         //孩子节点
    file *next;          //后一个兄弟节点
    file *pre;           //前一个兄弟节点
} File;
```

4.2　定义的类

```
class FileClass
{
public:
    File *InitFile(string filename, int isdir); // 创建文件与目录结点
    void CreateRoot();                          // 创建文件存储根结点
    void FindPara(string topara);               //寻找当前路径的父路径
和祖先路径
    void Mkdir();                               //创建目录
    void Create();                              //创建文件
    void Read();                                //读取文件内容
    void Write();                               //向文件写入内容
    void Del();                                 //删除文件或目录
    void Cd();                                  //切换目录
    void Dir();                                 //显示文件和目录
    void Help();                                //帮助函数
    int Run();                                  //程序运行函数

private:
    File *root, *recent, *temp, *ttemp;         //几个指针，用于类中函
数的调用
    string para, command, temppara, recentpara; //几个字符串变量，用来
存储路径以及输入的命令
};
```

4.3 主要功能的代码及注释

1） 创建文件

```
void FileClass::Create()
{
    temp = InitFile(" ", 0);
    cin >> temp->filename;
    cin >> temp->filecontent;
    if (recent->child == NULL)
    {
        temp->parent = recent;
        temp->child = NULL;
        recent->child = temp;
        temp->pre = temp->next = NULL;
        cout << " 文件创建成功!" << endl;
    }
    else
    {
        ttemp = recent->child;
        if (ttemp->filename == temp->filename && ttemp->isdir == 0)
        {
            cout << " 文件已经存在!" << endl;
            return;
        }
        else
        {
            while (ttemp->next)
            {
                ttemp = ttemp->next;
                if (ttemp->filename == temp->filename && ttemp->isdir == 0)
                {
                    cout << " 文件已存在!" << endl;
                    return;
                }
            }
        }
        ttemp->next = temp;
        temp->parent = NULL;
        temp->child = NULL;
        temp->pre = ttemp;
        temp->next = NULL;
```

```
        cout << " 文件创建成功!" << endl;
    }
}
```

2) 创建文件夹

```
void FileClass::Mkdir()
{
    temp = InitFile(" ", 1);
    cin >> temp->filename;
    if (recent->child == NULL)
    {
        temp->parent = recent;
        temp->child = NULL;
        recent->child = temp;
        temp->pre = temp->next = NULL;
        cout << " 目录创建成功!" << endl;
    }
    else
    {
        ttemp = recent->child;
        if (ttemp->filename == temp->filename && ttemp->isdir == 1)
        {
            cout << " 目录已经存在!" << endl;
            return;
        }
        else
        {
            while (ttemp->next)
            {
                ttemp = ttemp->next;
                if (ttemp->filename == temp->filename && ttemp->isdi
r == 1)
                {
                    cout << " 目录已经存在!" << endl;
                    return;
                }
            }
        }
        ttemp->next = temp;
        temp->parent = NULL;
        temp->child = NULL;
        temp->pre = ttemp;
        temp->next = NULL;
        cout << " 目录创建成功!" << endl;
```

```
    }
}
```

3) 显示文件或目录

```
void FileClass::Dir()
{
    int i = 0, j = 0;
    temp = recent;
    if (temp != root)
    {
        cout << " <目录> .." << endl;
        i++;
    }
    if (temp->child == NULL)
    {
        cout << "总计:"
             << " 目录:" << i << "\t 文件:" << j << endl;
        return;
    }
    temp = temp->child;
    while (temp != NULL)
    {
        if (temp->isdir)
        {
            cout << " <目录> " << temp->filename << endl;
            i++;
        }
        else
        {
            cout << " <文件> " << temp->filename << endl;
            j++;
        }
        temp = temp->next;
    }
    cout << "总计: "
         << "目录: " << i << "\t 文件:" << j << endl;
}
```

4) 删除文件

```
void FileClass::Del()
{
    string filename;
    cin >> filename;
    if (recent->child != NULL)
```

```
    {
        temp = recent->child;
        while (temp->next && temp->filename != filename)
            temp = temp->next;
        if (temp->filename != filename)
        {
            cout << " 不存在该文件!" << endl;
            return;
        }
    }
    else
    {
        cout << " 不存在该文件! " << endl;
        return;
    }
    if (temp->parent == NULL)
    {
        temp->pre->next = temp->next;
        if (temp->next)
            temp->next->pre = temp->pre;
        temp->pre = temp->next = NULL;
    }
    else
    {
        if (temp->next)
            temp->next->parent = temp->parent;
        temp->parent->child = temp->next;
    }
    delete temp;
    cout << "文件已删除!" << endl;
}
```

5) 程序运行函数

```
int FileClass::Run()
{
    cout << "root" << para << ">";
    cin >> command;
    if (command == "mkdir")
        Mkdir();
    else if (command == "dir")
        Dir();
    else if (command == "cd")
        Cd();
    else if (command == "create")
```

```
        Create();
    else if (command == "read")
        Read();
    else if (command == "write")
        Write();
    else if (command == "del")
        Del();
    else if (command == "help")
        Help();
    else if (command == "exit")
    {
        cout << "即将退出文件系统模拟，谢谢您的使用！" << endl;
        return 0;
    }
    else
        cout << " 请参考 help 提供的命令列表!" << endl;
    return 1;
}
```

第 5 章　运行与调试

5.1　运行结果与调试说明

1） 初始菜单及登录界面

```
(base) PS D:\MyFile\Desktop\C++>  cd "d:\MyFile\Desktop\C++\文件系统模拟\" ; if ($?) { g++ 文件系
统模拟.exe }
                    ====================文件系统模拟====================
                    *                                                  *
                    *         1>. 你只有三次机会来试验账号             *
                    *         2>. 键入help 可以获取帮助                *
                    *                                    欢迎使用本系统!*
                    ====================================================

请输入用户名:root
请输入密码:root
root/>
```

2） 创建文件

```
root/>create test.cpp cout<<"HelloWorld!";
 文件创建成功!
root/>create test2.c printf("HelloWorld!");
 文件创建成功!
root/>dir
 <文件> test.cpp
 <文件> test2.c
总计: 目录: 0   文件:2
root/>
```

3） 创建目录

```
root/>mkdir Test
 目录创建成功!
root/>mkdir Test1
 目录创建成功!
root/>dir
 <文件> test.cpp
 <文件> test2.c
 <目录> Test
 <目录> Test1
总计: 目录: 2   文件:2
root/>
```

4） 读取和写入文件

```
root/>dir
 <文件> test.cpp
 <文件> test2.c
 <目录> Test
 <目录> Test1
总计: 目录: 2   文件:2
root/>read test.cpp
cout<<"HelloWorld!";
root/>read test2.c
printf("HelloWorld!");
root/>write test.cpp #include<iostream>
文件写入成功!
root/>read test.cpp
#include<iostream>
root/>
```

5） 切换目录

```
root/>cd Test
root/Test>dir
 <目录> ..
 <目录> Test
 <文件> test3.cpp
总计: 目录: 2   文件:1
root/Test>cd Test
root/Test/Test>dir
 <目录> ..
总计: 目录:1    文件:0
root/Test/Test>cd /
root/>dir
 <文件> test.cpp
 <文件> test2.c
 <目录> Test
 <目录> Test1
总计: 目录: 2   文件:2
root/>cd /Test
root/Test>cd ..
root/>dir
 <文件> test.cpp
 <文件> test2.c
 <目录> Test
 <目录> Test1
总计: 目录: 2   文件:2
root/>
```

6） 删除文件或目录

```
root/>dir
 <文件> test.cpp
 <文件> test2.c
 <目录> Test
 <目录> Test1
总计: 目录: 2   文件:2
root/>del test.cpp
文件已删除!
root/>del test2.c
文件已删除!
root/>dir
 <目录> Test
 <目录> Test1
总计: 目录: 2   文件:0
root/>del Test
文件已删除!
root/>del Test1
文件已删除!
root/>dir
总计: 目录:0    文件:0
root/>
```

。

7） 帮助界面

```
root/>help
                    ======================帮助命令一览====================
                    *                                                    *
                    *          1>. create: 创建文件                      *
                    *          2>. read: 读取文件                        *
                    *          3>. write: 写入文件                       *
                    *          4>. del : 删除文件                        *
                    *          5>. mkdir: 建立目录                       *
                    *          6>. cd: 切换目录                          *
                    *          7>. exit: 退出登录                        *
                    *                                  欢迎使用本系统!*
                    ======================================================
root/>
```

第 6 章　总结

6.1 收获

大一的时候做过 C 语言的课程设计，那次的课程设计是使用链表做的，比较简单，但是也因此对 C 语言基础知识有了进一步了解，以及对数据结构产生了浓厚的兴趣，发现编程语言和数据结构相结合起来能够做出很多强大又有趣的东西。这学期数据结构学完之后我们进行了数据结构的课程设计，去年使用链表这种数据结构时还是会遇到很多棘手的问题，但是在今年的我看来不是什么问题，本着对链表掌握的自信，以及对接下来学习操作系统的打算，我选择了这次的课程设计题目——文件系统模拟，主要使用树这种数据结构去模拟文件系统，以实现创建文件、删除文件、读取和修改文件等操作，真的是非常的神奇。从网上查阅了相关的知识，我了解到一个完整的操作系统是各种数据结构的综合，操作系统中会用到树、栈、队列和图等数据结构，尽管现在凭借我的能力是不可能做出操作系统的，但是可以从宏观上慢慢地去了解，以丰富自己的知识，提高自己的能力。

这次的课程设计，对于我来说并不是很难，但是也不简单。在选择这道题目之前我有了具体的思路，就是使用树去实现，但是真正去实现的时候还是遇到了很多的问题，最初我是使用 C 语言做的，对于字符的处理不是使用 C 语言中的 char 类型数组，就是使用 char *类型的字符串，后来想了想，自己也学习过 C++，就是用了 C++的类和 string 类型去实现，但是转化的过程中出现了很多的问题，所以查找了很多的资料，虽然很麻烦，很困难，但是还是做了出来，做出来之后真的感觉非常的有成就感。

6.2 不足和存在的问题

本次的课程设计虽然使用了 C++和类进行设计，但是只是涉及了一些简单的面向对象的概念，并没有使用多态、继承等相关概念。另外也没有给类添加构造函数和析构函数，一来是因为自己能力不足，而来觉得自己这是个小程序，没有花费更多时间和精力去设计比较复杂的析构函数，寒假如果有机会，一定给添加上去。另外，程序中有一些代码需要优化，在设计时，只考虑了功能的实现，并没有着重考虑算法的时间和空间复杂度，以及算法的可读性，比如在运行函数 Run()中有多个 if…else if…else 语句，对于 python 语言，我知道可以使用字典去优化，但是对于 C 或者 C++语言，暂时我还没有比较好的方法优化，只有等我学习更多的知识后再来优化吧。另外，对于树这种数据结构我还是不够了解，这次的树存储方式采用的是兄弟孩子存储结构，我能够熟练地进行插入、查找和删除操作，但是对于寻找祖先结点和以及销毁整棵树等问题我还是不能够独立解决。树还有孩子存储和双亲存储两种方式，等以后学会了更多的知识，会考虑使用多种存储结构实现这次的数据结构课程设计。

由于时间有限以及期末复习，对于这次数据结构课程设计的报告写的也比较差，说到底还是自己的能力不足，以后一定会多练习写论文，练习 word 或 latex 排版，争取能够在老师规定的时间内优秀地完成任务。

6.3 对未来的展望

数据结构地课程设计算是完成了，但是对于数据结构的学习可不是到此为止。数据

结构在我们生活中有很多的应用，软件和游戏的开发是离不开数据结构和算法的，同样人工智能系统的建立也是离不开的。我们即将要学习的人工智能算法有很多也是建立在数据结构的基础上的。数据结构学习的也不只是几种基本的数据结构形式——表、树和图，更多的是让我们学习一种利用计算机去设计的思维，可能将来我们还可以创造出更优秀的数据结构和算法，从而给我们的社会带来更高的提升。

我觉的，编程语言和数据结构很多时候教会我们如何去思考，培养我们的是一种逻辑思维能力，我们有了这种能力，在解决问题时就会懂得如何去思考，从而简化我们解决问题的步骤或方法。可能我们掌握了这些思想之后，将来会赋予我们所制造的人工智能系统以同样的逻辑思维能力，从而让人工智能实现向我们人类一样的思考，从而实现真正的人工智能，甚至时超越人类的人工智能。

经过了几十年的发展，计算机的世界时如此的丰富多彩，如今我们的生活中处处都离不开计算机，各种事情都和计算机息息相关，人工智能技术又不断发展，要想在未来不被计算机和人工智能取代，我们就应该学习计算机，去深入了解它，让计算机为人类所用。

参考文献

[1] 周鸣争, 钟志水. C 语言程序设计 .[M] 成都 电子科技大学出版社 2018.
[2] [美] Dennis M. Ritchie 等著. C 程序设计语言[M]. 徐文宝等译. 北京: 机械工业出版 社, 2004.
[3] 刘 涛，叶明全. C 语言程序设计学习指导与实践教程[M]. 上海：上海交通大学出版社，2017.
[4] 谭浩强. C 程序设计（第五版）[M]. 北京：清华大学出版社，2017.
[5] 李春葆 数据结构教程(第五版)[M]. 北京:清华大学出版社,2019.
[6] 严蔚敏，吴伟民. 数据结构（C 语言版）[M]. 北京:清华大学出版社,2020
[7] 马石安、魏文平. 面向对象程序设计教程（C++语言描述）(第 3 版)[M]. 北京:清华大学出版社,2019.
[8].胡凡、曾磊.《算法笔记》.机械工业出版社[M]，2016.7
[9].程杰.《大话数据结构》.清华大学出版社[M]，2011.6
[10].杨路明.《C 语言程序设计教程》北京邮电大学出版社[M]

附　录

源代码：

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

typedef struct file
{
    string filename;    //文件或目录名
    int isdir;          //是否是目录,是的话为1,不是为0
    int opened;         //是否打开
    string filecontent; //文件内容
    file *parent;       //双亲节点
    file *child;        //孩子节点
    file *next;         //后一个兄弟节点
    file *pre;          //前一个兄弟节点
} File;

class FileClass
{
public:
    File *InitFile(string filename, int isdir); // 创建文件与目录结点
    void CreateRoot();                          // 创建文件存储根结点
    void FindPara(string topara);               //寻找当前路径的父路径
和祖先路径
    void Mkdir();                               //创建目录
    void Create();                              //创建文件
    void Read();                                //读取文件内容
    void Write();                               //向文件写入内容
    void Del();                                 //删除文件或目录
    void Cd();                                  //切换目录
    void Dir();                                 //显示文件和目录
    void Help();                                //帮助函数
    int Run();                                  //程序运行函数

private:
    File *root, *recent, *temp, *ttemp;         //几个指针，用于类中函
数的调用
    string para, command, temppara, recentpara; //几个字符串变量，用来
存储路径以及输入的命令
};
```

```
// 创建文件与目录结点
File *FileClass::InitFile(string filename, int isdir)
{
    File *node = new File;
    node->filename = filename;
    node->isdir = isdir;
    node->opened = 0;
    node->parent = NULL;
    node->child = NULL;
    node->next = NULL;
    node->pre = NULL;
    return node;
}

// 创建文件存储根结点
void FileClass::CreateRoot()
{
    recent = root = InitFile("/", 1);
    root->parent = NULL;
    root->child = NULL;
    root->pre = root->next = NULL;
    para = "/";
}

//创建目录
void FileClass::Mkdir()
{
    temp = InitFile(" ", 1);
    cin >> temp->filename;
    if (recent->child == NULL)
    {
        temp->parent = recent;
        temp->child = NULL;
        recent->child = temp;
        temp->pre = temp->next = NULL;
        cout << " 目录创建成功!" << endl;
    }
    else
    {
        ttemp = recent->child;
        if (ttemp->filename == temp->filename && ttemp->isdir == 1)
        {
```

```
            cout << " 目录已经存在!" << endl;
            return;
        }
        else
        {
            while (ttemp->next)
            {
                ttemp = ttemp->next;
                if (ttemp->filename == temp->filename && ttemp->isdi
r == 1)
                {
                    cout << " 目录已经存在!" << endl;
                    return;
                }
            }
        }
        ttemp->next = temp;
        temp->parent = NULL;
        temp->child = NULL;
        temp->pre = ttemp;
        temp->next = NULL;
        cout << " 目录创建成功!" << endl;
    }
}

//创建文件
void FileClass::Create()
{
    temp = InitFile(" ", 0);
    cin >> temp->filename;
    cin >> temp->filecontent;
    if (recent->child == NULL)
    {
        temp->parent = recent;
        temp->child = NULL;
        recent->child = temp;
        temp->pre = temp->next = NULL;
        cout << " 文件创建成功!" << endl;
    }
    else
    {
        ttemp = recent->child;
        if (ttemp->filename == temp->filename && ttemp->isdir == 0)
```

```
        {
            cout << " 文件已经存在!" << endl;
            return;
        }
        else
        {
            while (ttemp->next)
            {
                ttemp = ttemp->next;
                if (ttemp->filename == temp->filename && ttemp->isdi
r == 0)
                {
                    cout << " 文件已存在!" << endl;
                    return;
                }
            }
        }
        ttemp->next = temp;
        temp->parent = NULL;
        temp->child = NULL;
        temp->pre = ttemp;
        temp->next = NULL;
        cout << " 文件创建成功!" << endl;
    }
}

//显示文件和目录
void FileClass::Dir()
{
    int i = 0, j = 0;
    temp = recent;
    if (temp != root)
    {
        cout << " <目录> .." << endl;
        i++;
    }
    if (temp->child == NULL)
    {
        cout << "总计:"
             << " 目录:" << i << "\t 文件:" << j << endl;
        return;
    }
    temp = temp->child;
```

```
    while (temp != NULL)
    {
        if (temp->isdir)
        {
            cout << " <目录> " << temp->filename << endl;
            i++;
        }
        else
        {
            cout << " <文件> " << temp->filename << endl;
            j++;
        }
        temp = temp->next;
    }
    cout << "总计: "
         << "目录: " << i << "\t 文件:" << j << endl;
}

//读取文件内容
void FileClass::Read()
{
    string filename;
    cin >> filename;
    if (recent->child == NULL)
    {
        cout << "文件不存在!" << endl;
        return;
    }
    else
    {
        temp = recent->child;
        while (temp)
        {
            if (temp->filename == filename)
            {
                cout << temp->filecontent << endl;
                return;
            }
            temp = temp->next;
        }
        cout << " 文件不存在!" << endl;
    }
}
```

```
//向文件写入内容
void FileClass::Write()
{
    string filename;
    cin >> filename;
    if (recent->child == NULL)
    {
        cout << "文件不存在!" << endl;
        return;
    }
    else
    {
        temp = recent->child;
        while (temp != NULL)
        {
            if (temp->filename == filename)
            {
                recent->opened = 1; //设置文件标记为打开
                cin >> temp->filecontent;
                recent->opened = 0; //设置文件标记为关闭
                cout << "文件写入成功!" << endl;
                return;
            }
            temp = temp->next;
        }
        cout << "文件不存在!" << endl;
    }
}

//删除文件或目录
void FileClass::Del()
{
    string filename;
    cin >> filename;
    if (recent->child != NULL)
    {
        temp = recent->child;
        while (temp->next && temp->filename != filename)
            temp = temp->next;
        if (temp->filename != filename)
        {
            cout << " 不存在该文件!" << endl;
```

```
            return;
        }
    }
    else
    {
        cout << " 不存在该文件! " << endl;
        return;
    }
    if (temp->parent == NULL)
    {
        temp->pre->next = temp->next;
        if (temp->next)
            temp->next->pre = temp->pre;
        temp->pre = temp->next = NULL;
    }
    else
    {
        if (temp->next)
            temp->next->parent = temp->parent;
        temp->parent->child = temp->next;
    }
    delete temp;
    cout << "文件已删除!" << endl;
}

//切换目录
void FileClass::Cd()
{
    string topara;
    cin >> topara;
    if (topara == "..") //切换上一级目录
    {
        int i;
        while (recent->pre)
            recent = recent->pre;
        if (recent->parent)
            recent = recent->parent;
        i = para.length();
        while (para[i] != '/' && i > 0)
            i--;
        para.replace(i + 1, para.length(), "\0");
    }
    else
```

```
        FindPara(topara);
}

//寻找当前路径的父路径和祖先路径
void FileClass::FindPara(string topara)
{
    int i = 0;
    int sign = 1;
    char str[50];
    if (topara == "/") //切换回根结点
    {
        recent = root;
        para = "/";
        return;
    }
    temp = recent;
    temppara = para;
    if (topara[0] == '/') //切换为根节点的第一级子目录
    {
        recent = root->child;
        i++;
        para = "/";
    }
    else
    {
        if (recent != NULL && recent != root)
            para += "/";
        if (recent && recent->child)
        {
            if (recent->isdir)
                recent = recent->child;
            else
            {
                cout << "路径错误!" << endl;
                return;
            }
        }
    }
    while (i <= signed(topara.length()) && recent) //以下主要对输出路
径进行处理
    {
        int j = 0;
        if (topara[i] == '/' && recent->child)
```

```
        {
            i++;
            if (recent->isdir)
                recent = recent->child;
            else
            {
                cout << "路径错误!" << endl;
                return;
            }
            para += "/";
        }
        while (topara[i] != '/' && i <= signed(topara.length()))
        {
            str[j] = topara[i];
            i++;
            j++;
        }
        str[j] = '\0';
        recentpara = string(str);
        while ((recent->filename != recentpara || (recent->isdir !=
1)) &&
               recent->next != NULL)
        {
            recent = recent->next;
        }
        if (recent->filename == recentpara)
        {
            if (recent->isdir == 0)
            {
                para = temppara;
                recent = temp;
                cout << " 是文件不是目录。\n";
                return;
            }
            para += recent->filename;
        }
        if (recent->filename != recentpara || recent == NULL)
        {
            para = temppara;
            recent = temp;
            cout << " 输入路径错误\n";
            return;
        }
```

```
    }
}

//帮助函数
void FileClass::Help()
{
    cout << "\t\t\t====================帮助命令一览
====================\n";
    cout << "\t\t\t*
   *\n";
    cout << "\t\t\t*          1>. create: 创建文
件                      *\n";
    cout << "\t\t\t*          2>. read: 读取文
件                       *\n";
    cout << "\t\t\t*          3>. write: 写入文
件                      *\n";
    cout << "\t\t\t*          4>. del : 删除文
件                       *\n";
    cout << "\t\t\t*          5>. mkdir: 建立目
录                      *\n";
    cout << "\t\t\t*          6>. cd: 切换目
录                        *\n";
    cout << "\t\t\t*          7>. exit: 退出登
录                       *\n";
    cout << "\t\t\t*                                          欢迎使用本系
统!*\n";
    cout << "\t\t\t================================================
====\n";
}

//程序运行函数
int FileClass::Run()
{
    cout << "root" << para << ">";
    cin >> command;
    if (command == "mkdir")
        Mkdir();
    else if (command == "dir")
        Dir();
    else if (command == "cd")
        Cd();
    else if (command == "create")
        Create();
```

```
    else if (command == "read")
        Read();
    else if (command == "write")
        Write();
    else if (command == "del")
        Del();
    else if (command == "help")
        Help();
    else if (command == "exit")
    {
        cout << "即将退出文件系统模拟，谢谢您的使用！" << endl;
        return 0;
    }
    else
        cout << " 请参考 help 提供的命令列表!" << endl;
    return 1;
}
int main()
{
    int i = 0;
    bool in = false;      //in 为 true 是允许进入系统，负责不允许进入
    FileClass f;          //构建一个 FileClass 对象
    string user, passwd; //用于存储用户名和密码
    cout << "\t\t\t====================文件系统模拟
====================\n";
    cout << "\t\t\t*
   *\n";
    cout << "\t\t\t*         1>. 你只有三次机会来试验账
号              *\n";
    cout << "\t\t\t*         2>. 键入 help 可以获取帮
助                  *\n";
    cout << "\t\t\t*                                          欢迎使用本系
统!*\n";
    cout << "\t\t\t=================================================
====\n";
    cout << endl;
    while (i < 3) //设置输入账号和密码的次数最多为三次
    {
        cout << "请输入用户名:";
        cin >> user;
        cout << "请输入密码:";
        cin >> passwd;
        if (user == "root" && passwd == "root")
```

```
        {
            in = true;
            break;
        }
        else
        {
            if (2 - i != 0)
                cout << "\t 您输入的账号或密码错误，请重新输入,您还有
" << 2 - i << "次机会!" << endl;
            else
            {
                cout << "\t 您的机会已用完！谢谢您的使用" << endl;
                exit(0);
            }
        }
        i++;
    }
    f.CreateRoot(); //创建根节点
    while (in)      //进入消息循环
    {
        if (!f.Run())
            break;
    }
    return 0;
}
```