**江苏科技大学**

**课程设计文档**

课 程： 操作系统课程设计

课 题： 文件管理系统

学 院： 计算机学院

组 员： 182210710119 陈四贵

182210710141 仲 响

182210710142 周泽江

班 级： 1822107101

指导老师： 周 扬

目 录

[一、 需求分析 1](#_Toc58264577)

[二、 小组分工 1](#_Toc58264578)

[三、 系统设计 2](#_Toc58264579)

[1、设计思路 2](#_Toc58264580)

[2、各部分算法设计 3](#_Toc58264581)

[3、约定规范 4](#_Toc58264582)

[四、 详细设计与系统实现 4](#_Toc58264583)

[◎初始化文件管理系统(Catalog()函数) 4](#_Toc58264584)

[◎流程控制(menu\_Control()函数) 5](#_Toc58264585)

[◎建立文件功能(file\_Build()函数) 6](#_Toc58264586)

[◎打开文件功能(file\_Open()函数) 7](#_Toc58264587)

[◎删除文件功能(file\_Delete()函数) 9](#_Toc58264588)

[◎关闭文件功能(file\_Close()函数) 10](#_Toc58264589)

[◎读文件功能(file\_Read()函数) 12](#_Toc58264590)

[◎写文件功能(file\_Write()函数) 12](#_Toc58264591)

[◎文件查重功能(file\_Check()函数) 14](#_Toc58264592)

[五、 调试分析 14](#_Toc58264593)

[1.测试结果 15](#_Toc58264594)

[2.算法时间复杂度分析 16](#_Toc58264595)

[六、 结论与展望 16](#_Toc58264596)

[附件 18](#_Toc58264597)

# 需求分析

在本设计项目中，需要使用C++语言模拟设计一个简单的多用户文件管理系统，以此深入了解操作系统中文件系统的理论，加深对教材中重要算法的理解，进而提高综合运用各专业课知识的能力。

该系统采用两级目录进行文件管理（最多不超过5个用户），两级目录分别是采用顺序表存储结构的主目录（MFD）与采用链式存储结构的用户文件目录（UFD）；且该系统需采用菜单方式与用户交互，实现文件操作。需设立的功能有：建立文件、打开文件、删除文件、关闭文件、读文件、写文件；此外，该系统需要支持反复选择不同用户，实施有关文件操作。

由此，我们初步得出编程思路：将文件抽象为File结构体，内含文件所必须的属性；将二级目录文件管理系统抽象为由File结构体以某种数据结构（邻接表）连结而成的实体，将其定义为Catalog类；由于用户文件目录采用链式存储，菜单项中的六大操作实际上是对Catalog类的实体（对象）的各种操作，实质上是对数组与单链表的处理。

# 小组分工

◎182210710119 陈四贵：

1)带领小组成员梳理编程思维，制定规范，分工任务，并积极协调各成员进度；

2)搭建系统基本框架，并实现建立文件、文件查重等功能，设计程序流程；

3)课程设计文档编写。

◎182210710141 仲 响：

1)实现打开文件、删除文件与关闭文件等功能；

2)课程设计文档完善；

3)答辩ppt的制作。

◎182210710142 周泽江：

1)实现读文件与写文件等功能；

2)整合小组各成员代码，测试代码并将其优化；

3)课程设计文档与答辩ppt的完善。

共同任务：需求分析、约定规范。

# 系统设计

## 1、设计思路

将文件类型抽象为File结构体，内部封装文件名name[20]、内容content[100]，以及由于链式存储需要封装的指向File对象的指针next等基本属性，进而通过邻接矩阵的形式将File结构体的对象连结在一起成为封装有属性和文件管理系统相关操作的Catalog类，由此来模拟实现主目录采取顺序表存储结构、用户文件目录采用链式存储结构的文件管理系统。

name：抽象文件名

content：抽象文件内容

opened：指针标志，指示文件是否打开。1为被打开，0为未打开

\*next：指向File对象的指针，以实现链式存储

图3-1-1 文件抽象File结构体

图3-1-2 class结构

…

…

…

…

图3-1-3 文件管理系统结构示意图

采用顺序表存储结构的主目录

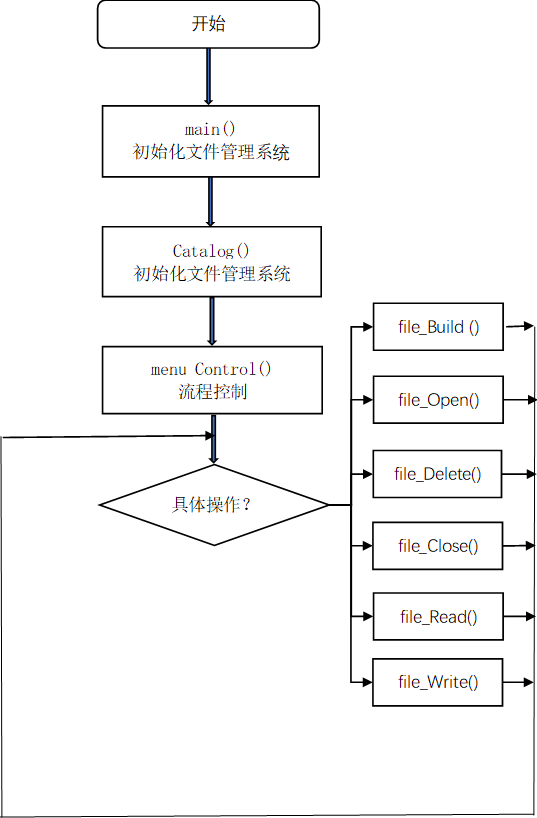
采用链式存储结构的用户文件目录

文件管理系统

## 2、各部分算法设计

我们将文件管理系统抽象为一种复合数据类型（即类）Catalog。其中封装了该文件管理系统应有的属性与相关操作，相关操作（函数）均是对Catalog结构中File对象的处理，涉及单链表的查找、增添、删除等操作。对于大部分功能来说，都使用了单链表的查找技术。

因而，各部分算法中，我们都使用了choice与name[20]作为索引，寻找合适的位置进行各种常规操作，以此来模拟文件管理系统。各算法流程图、具体实现代码见“详细设计”篇。

图3-2-1 程序流程示意图

## 3、约定规范

本设计严格遵守国际命名规范，采用驼峰式命名规则。具体命名规则如下：

表1-1 命名规范表

|  |  |  |
| --- | --- | --- |
|  | 命名规则 | 示例 |
| 类、结构体 | 依据用途命名，首字母大写 | File、Catalog |
| 函数 | 依据用途命名，第一个单词全小写，第二个单词首字母大写 | file\_Open()  file\_Close()  file\_Delete() |
| 变量 | 依据用途命名，全字母小写 | choice、name |

# 详细设计与系统实现

## ◎初始化文件管理系统(Catalog()函数)

设计思想：在非法检查的基础上，依据用户输入的数值num决定该文件管理系统中的用户数量，完成邻接矩阵（系统）的初始化。

算法流程图：



图4-1-1 Catalog()函数算法流程图

实现代码：

1. Catalog() {//构造函数，初始化“邻接表”,用户目录为空（此时没有文件）
2. cout << "请输入用户个数：";
3. cin >> num;
4. **if** (num <= 0) {
5. cout << "输入非法，请重新输入：";
6. cin >> num;
7. }
8. **while** (num > 5) {
9. cout << "用户太多，超出所能承受范围，请重新输入：";
10. cin >> num;
11. }
12. Head\_MFD = **new** File[num];
13. **for** (**int** i = 0; i < num; i++) {
14. Head\_MFD[i].name[0] = (**char**)(i + 1);
15. Head\_MFD[i].next = NULL;
16. }
17. cout << "主目录构建成功！" << endl;
18. }

## ◎流程控制(menu\_Control()函数)

设计思想：依据用户输入来控制程序流程，进而达到实现某种特定功能的目的，保证程序正常运转。

算法流程图：

图4-1-2 menu\_Control()函数算法流程图

实现代码：

1. **void** menu\_Control() {//正常工作状态，循环菜单栏
2. **do** {
3. cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n1.建立文件\n2.打开文件\n3.删除文件\n4.关闭文件\n5.读文件\n6.写文件" << endl;
4. cout << "请输入您想选择的功能：";
5. **int** direction;
6. cin >> direction;
7. **switch** (direction) {
8. **case** 1:file\_Build(); **break**;
9. **case** 2:file\_Open(); **break**;
10. **case** 3:file\_Delete(); **break**;
11. **case** 4:file\_Close(); **break**;
12. **case** 5:file\_Read(); **break**;
13. **case** 6:file\_Write(); **break**;
14. }
15. } **while** (1 > 0);
16. }

## ◎建立文件功能(file\_Build()函数)

设计思想：在文件名重复检查与输入合法的基础上通过用户输入决定在哪个主目录里建立文件，以及建立什么文件。若二级目录为空，则直接插；若二级目录不为空，则在链尾插。

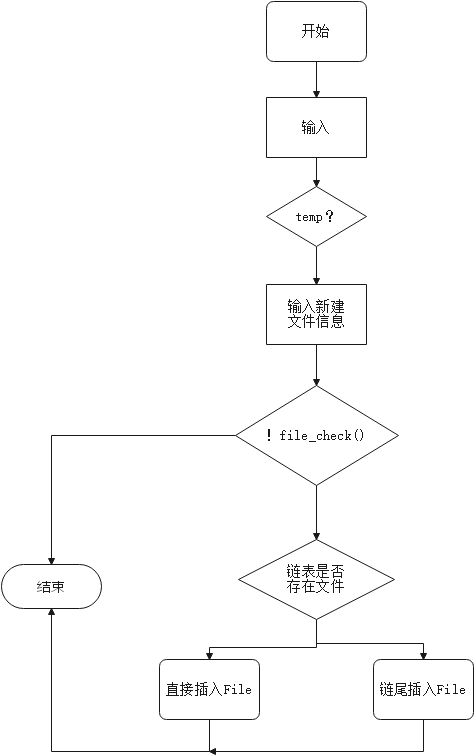
算法流程图：

图4-1-3 file\_Build()函数算法流程图

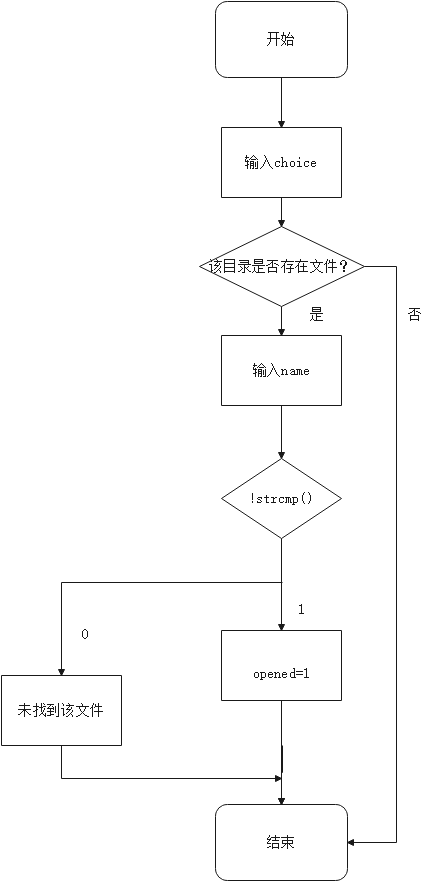
实现代码：

1. **void** file\_Build() {//函数·建立文件
2. **int** temp;
3. cout << "\n目前有" << num << "个用户，分别是：";
4. **for** (**int** i = 0; i <= num - 1; i++) {
5. cout << "用户" << i + 1 << "\t";
6. }
7. cout << "\n请选择在哪个用户文件夹里建立文件：";
8. cin >> temp;
9. **while** (temp > num || temp <= 0) {
10. cout << "输入非法，请重新输入：";
11. cin >> temp;
12. }
13. p = Head\_MFD[temp - 1].next;
14. q = **new** File;
15. cout << "请输入新建文件名：";
16. cin >> q->name;
17. cout << "请输入文件内容：";
18. cin >> q->content;
19. q->next = NULL;
20. q->opened = 1;
21. **if** (p == NULL) {//直接插文件
22. Head\_MFD[temp - 1].next = q;
23. }
24. **else** {//文件夹中存在文件的情况，在链尾插
25. **if** (!file\_Check(Head\_MFD[temp - 1].next, q->name)) {
26. cout << "该目录中已存在该文件。\n";
27. **return**;
28. }
29. **while** (p->next != NULL) {
30. p = p->next;
31. }
32. p->next = q;
33. }
34. cout << "文件创建完毕！位于目录" << temp << "中\n";
35. }

## ◎打开文件功能(file\_Open()函数)

设计思想：打开文件即显示文件的内容，因而该算法核心思想是查找与对比。通过用户输入定位文件夹（目录），再依次查询该目录是否存在要打开的文件，若存在，输出文件信息；若不存在，反馈错误信息。

算法流程图：

图4-1-4 file\_Open()函数算法流程图

实现代码：

1. **void** file\_Open() {//函数·打开文件
2. **int** choice;
3. **char** name[20];
4. cout << "请问你想打开哪个用户文件夹中的文件（输入数字)：";
5. cin >> choice;
6. **if** (Head\_MFD[choice - 1].next == NULL) {
7. cout << "该文件夹中不存在文件。\n";
8. **return**;
9. }
10. cout << "请问你想打开用户" << choice << "中哪个文件？（输入名字）：";
11. cin >> name;
12. p = Head\_MFD[choice - 1].next;
13. **while** (p != NULL) {
14. **if** (!strcmp(p->name, name)) {
15. p->opened = 1;
16. **break**;
17. }
18. p = p->next;
19. }
20. **if** (p == NULL) {
21. cout << "未找到该文件，无法打开。\n";
22. **return**;
23. }
24. cout << "已打开" << p->name << "文件，内容是" << p->content << "！\n";
25. }

## ◎删除文件功能(file\_Delete()函数)

设计思想：删除文件实质上涉及单链表的查找与删除两个技术核心，由于二级目录采用链式存储，我们需要保存要删除的文件的前一个节点，使该节点指向要删除的节点（文件）的下一个节点（文件），以此达到删除文件的目的。

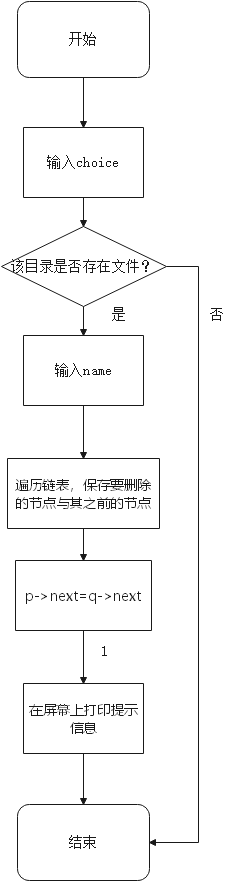
算法流程图：

图4-1-5 file\_Delete()函数算法流程图

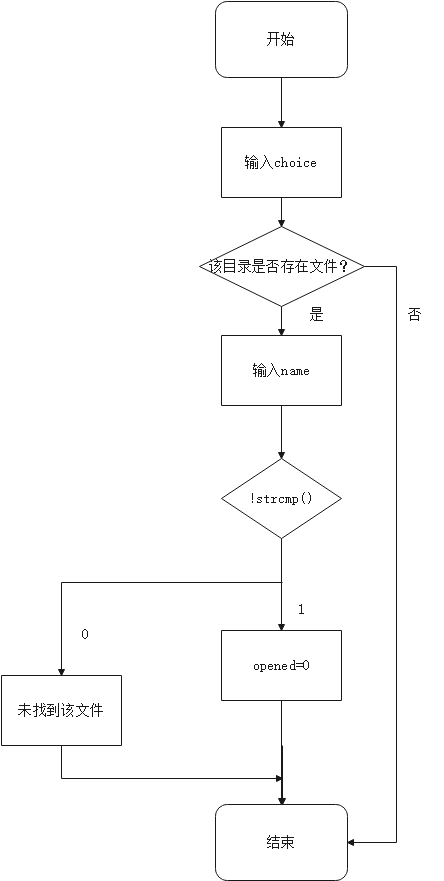
实现代码：

1. **void** file\_Delete() {//函数·删除文件
2. **int** choice;
3. **char** name[20];
4. cout << "请问你想删除哪个用户文件夹中的文件（输入数字)：";
5. cin >> choice;
6. **if** (Head\_MFD[choice - 1].next == NULL) {
7. cout << "该文件夹中不存在文件。\n";
8. **return**;
9. }
10. cout << "请问你想删除用户" << choice << "中哪个文件？（输入名字）：";
11. cin >> name;
12. p = Head\_MFD[choice - 1].next;
13. **if** (!strcmp(p->name, name)) {//第一个位置就是要找的文件
14. Head\_MFD[choice - 1].next = Head\_MFD[choice - 1].next->next;
15. }
16. **else** {//要找的文件在后面，或者不存在
17. q = p->next;
18. **while** (q != NULL) {
19. **if** (!strcmp(q->name, name)) {
20. p->next = q->next;
21. **break**;
22. }
23. }
24. **if** (q == NULL) {
25. cout << "未找到该文件，无法删除。\n";
26. **return**;
27. }
28. }
29. cout << "已删除" << q->name << "文件！\n";
30. }

## ◎关闭文件功能(file\_Close()函数)

设计思想：由于我们是模拟的一个文件系统，本身并不是真正创建一个文件。因而不能通过调用函数实现关闭文件。我们能做的仅仅只是将“opened（1表示文件打开，0表示文件关闭）”封装进File节点，查找到相关文件后通过opened标志来表示文件状态。

算法流程图：

图4-1-6 file\_Close()函数算法流程图

实现代码：

1. **void** file\_Close() {//函数·关闭文件
2. **int** choice;
3. **char** name[20];
4. cout << "请问你想关闭哪个用户文件夹中的文件（输入数字)：";
5. cin >> choice;
6. **if** (Head\_MFD[choice - 1].next == NULL) {
7. cout << "该文件夹中不存在文件。\n";
8. **return**;
9. }
10. cout << "请问你想关闭用户" << choice << "中哪个文件？（输入名字）：";
11. cin >> name;
12. p = Head\_MFD[choice - 1].next;
13. **while** (p != NULL) {
14. **if** (!strcmp(p->name, name)) {
15. p->opened = 0;
16. **break**;
17. }
18. p = p->next;
19. }
20. **if** (p == NULL) {
21. cout << "未找到该文件，无法关闭。\n";
22. **return**;
23. }
24. cout << "已关闭" << p->name << "文件"  << "！\n";
25. }

## ◎读文件功能(file\_Read()函数)

设计思想：打开文件不修改即为读文件，因而该算法核心思想同“打开文件”，通过用户输入确定在哪个文件夹（主目录）查找，再对单链表进行遍历查找。

算法流程图：见图4-1-4

实现代码：

1. **void** file\_Read() {//函数·读文件
2. file\_Open();
3. }

## ◎写文件功能(file\_Write()函数)

设计思想：写文件即对文件进行修改。在我们的抽象中，能修改的仅仅只是文件的名字(name[20])、内容(content[100])属性。因而，我们的写文件操作实质是在二级目录（单链表）中找到要修改的文件，并按照用户意愿相关修改。

算法流程图：见图4-1-6

实现代码：

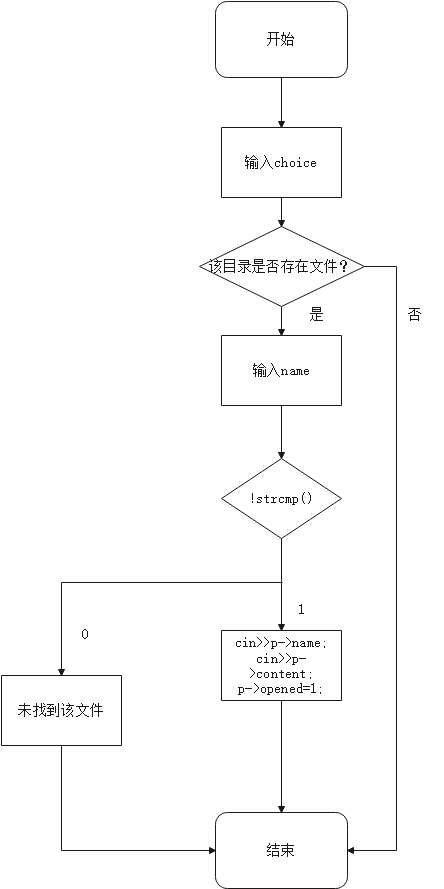
1. **void** file\_Write() {//函数·写文件
2. **int** choice;
3. **char** name[20];
4. cout << "请问你想修改哪个用户文件夹中的文件（输入数字)：";
5. cin >> choice;
6. **if** (Head\_MFD[choice - 1].next == NULL) {
7. cout << "该文件夹中不存在文件。\n";
8. **return**;
9. }
10. cout << "请问你想修改用户" << choice << "中哪个文件？（输入名字）：";
11. cin >> name;
12. p = Head\_MFD[choice - 1].next;
13. **if** (!strcmp(p->name, name)) {//第一个就找到文件
14. cout << "请您输入修改后的文件名：";
15. cin >> p->name;
16. cout << "请您输入修改后的文件内容：";
17. cin >> p->content;
18. p->opened = 1;
19. }
20. **else** {//要找的文件不在第一个（在后面的位置或不存在）
21. **while** (p != NULL) {
22. **if** (!strcmp(p->name, name)) {
23. cout << "请您输入修改后的文件名：";
24. cin >> p->name;
25. cout << "请您输入修改后的文件内容：";
26. cin >> p->content;
27. p->opened = 1;
28. **break**;
29. }
30. p = p->next;
31. }
32. **if** (p == NULL) {
33. cout << "未找到该文件，无法打开。\n";
34. **return**;
35. }
36. }
37. cout << "已修改" << p->name << "文件，内容是" << p->content << "！\n";
38. }

图4-1-6 file\_Write()函数算法流程图

## ◎文件查重功能(file\_Check()函数)

设计思想：使用strcmp()函数直接对两个字符数组进行比较，若返回0则两字符数组内容完全相同，若返回1则字符数组内容不完全相同。

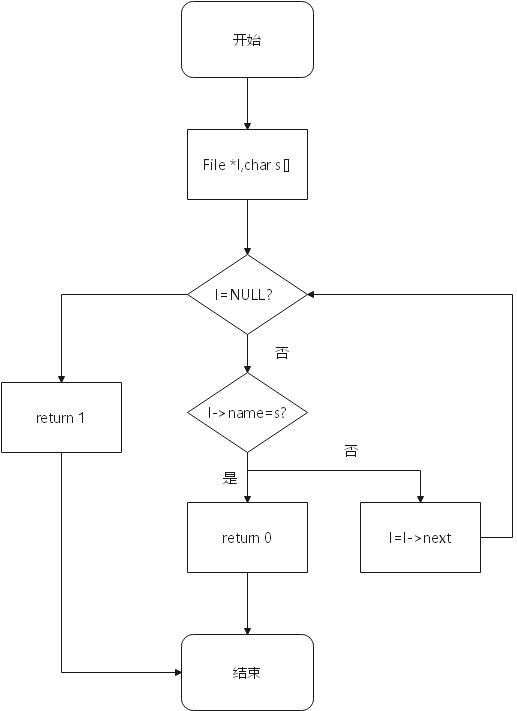
算法流程图：

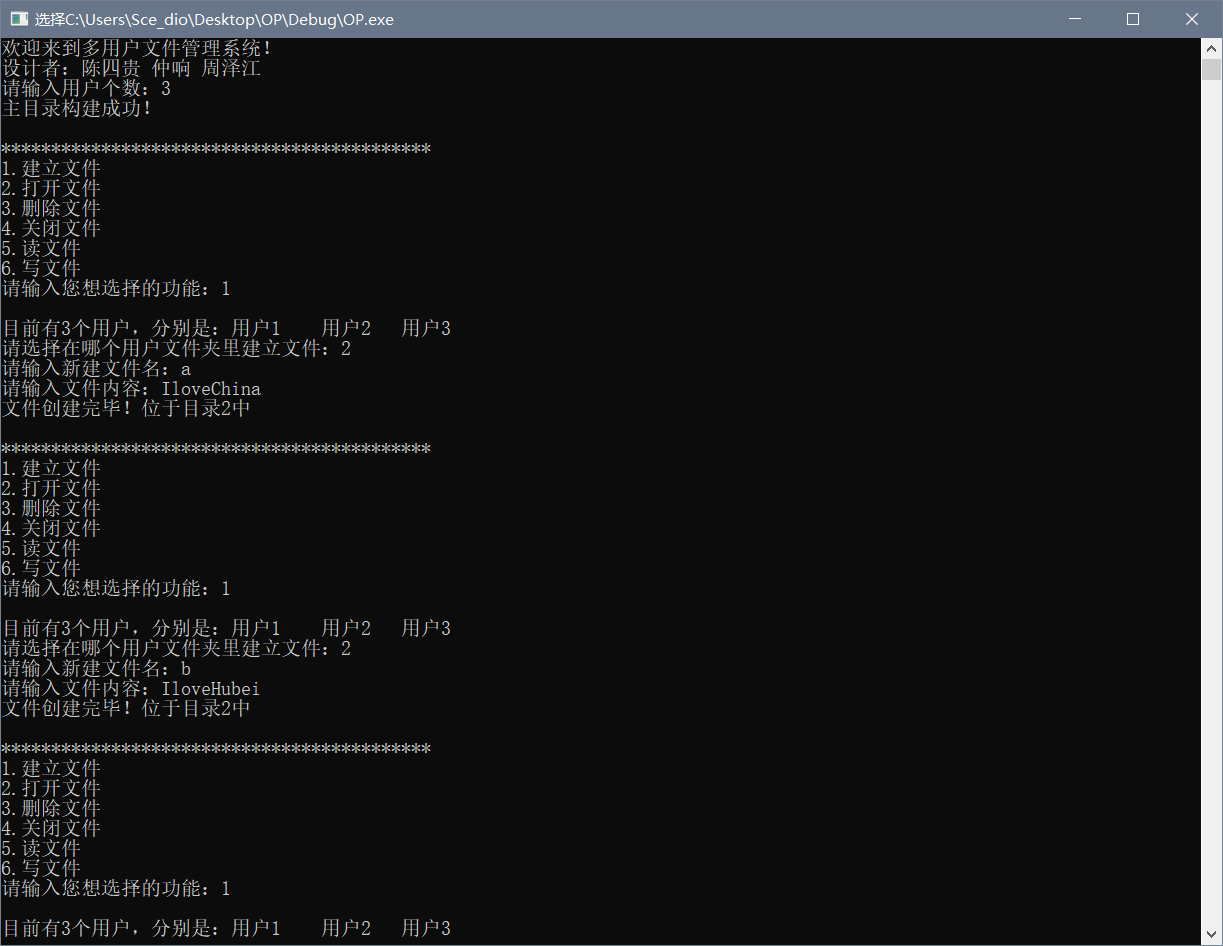
图4-1-7 file\_Check()函数算法流程图

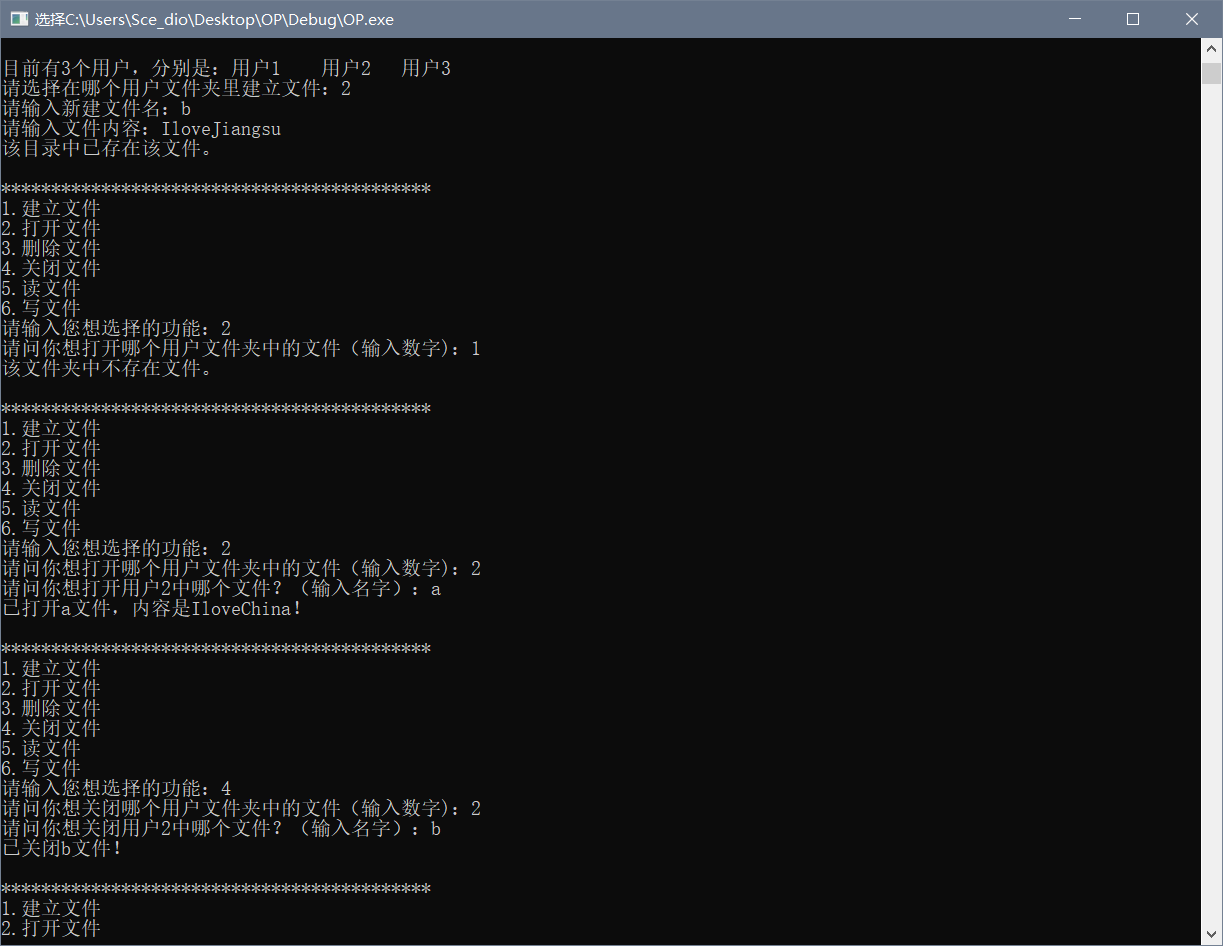
实现代码：

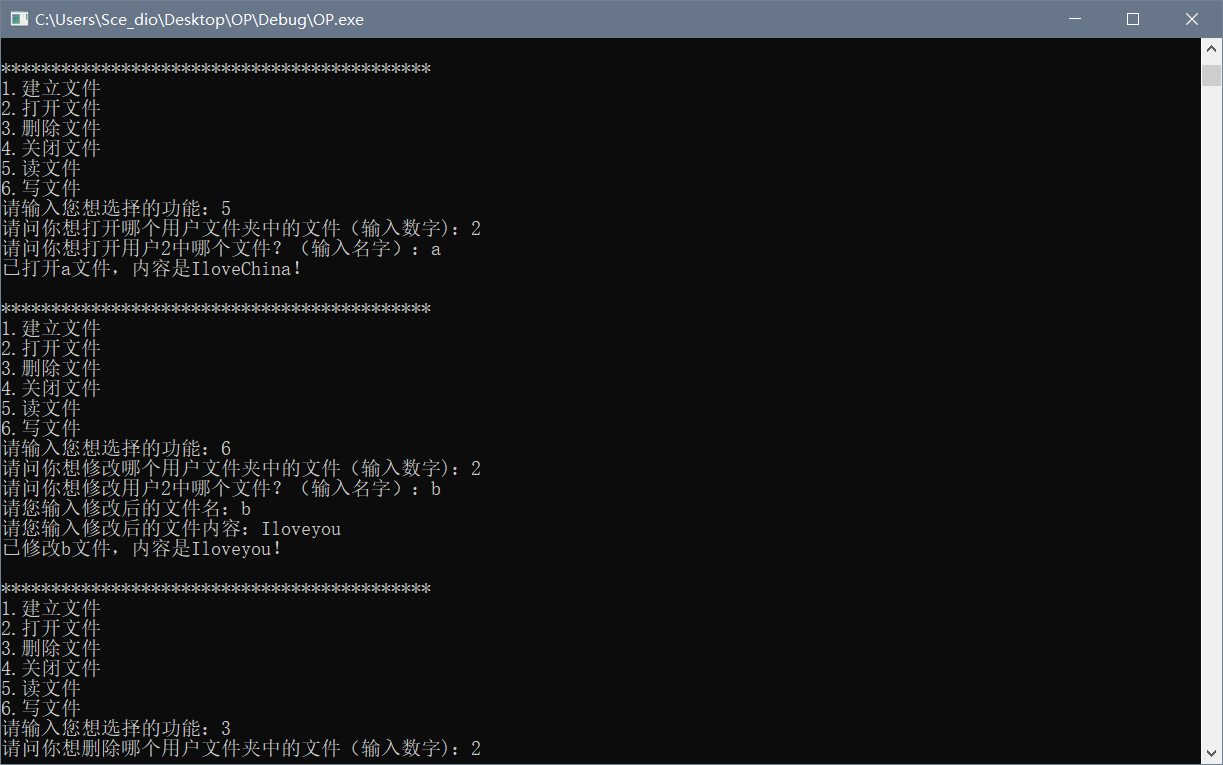
1. **bool** file\_Check(File \*l,**char** s[]) {//函数·创建文件时查重
2. **while** (l != NULL) {
3. **if** (!strcmp(l->name, s)) {
4. **return** **false**;
5. }
6. l = l->next;
7. }
8. **return** **true**;
9. }

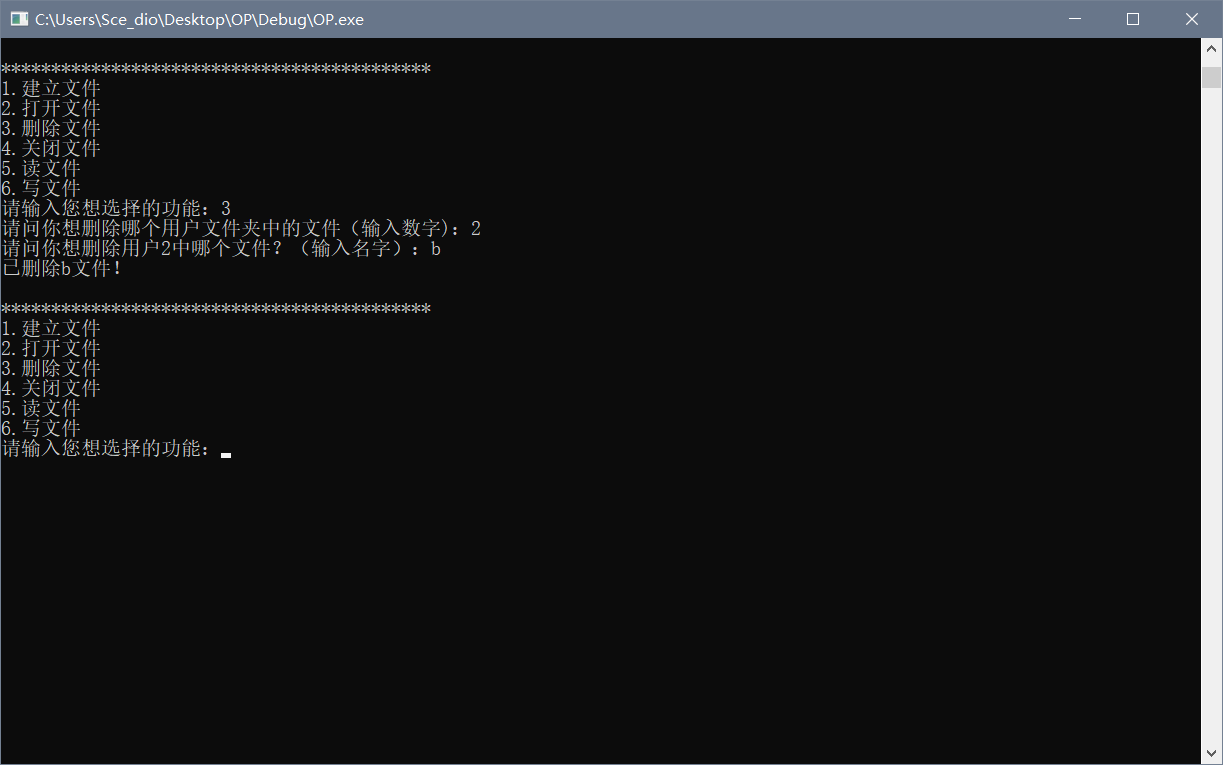
# 调试分析

## 1.测试结果









## 2.算法时间复杂度分析

由于整个程序存在多个基本语句，时间复杂度很难定量说明。定性分析，各算法都算是对线性表的操作，又未涉及多重循环，因而可以近似地看作时间复杂度一般为O（n）.

# 结论与展望

1.编程抽象是最首要的问题，这直接关系程序的定位和实现。于此，看到二级文件管理系统的顺序存储和链式存储后，我们想到了邻接表与邻接矩阵，因而我们将文件抽象为一个节点，封装其应有的属性，再将文件管理系统抽象为一个由无数个节点以某种数据结构组成的对象，封装其应有属性与操作。打开了此题的门径。

2.数据类型的处理也是一大重要难题。身为设计者，我们本身希望该文件管理系统中能够随意填充文件内容，无论长度。基于自身的认知，我们首先想到的是将内容content设为字符指针类型，直接赋值，但可惜的是其值不能修改；随后我们想到了”string.h”库中的string数据类型，但其不能直接被赋值号赋值。因而，我们退而求其次，采用了固定长度限制的字符数组作为文件名和文件内容属性的实质。

3.操作系统对计算机来说是很重要的软件，它的设计直接关系计算机相关基本操作，影响计算机的计算性能。

# 附件

1. #include <iostream>
2. #include <string.h>
3. **using** **namespace** std;
5. **struct** File {//抽象文件
6. **char** name[20];//文件名
7. **char** content[100];//文件内容，供写文件用
8. **bool** opened;//指针标志，指示文件是否打开。1为被打开，0为未打开。
9. File \*next;
10. };
12. **class** Catalog {//二级文件管理系统
13. **private**:
14. **int** num;//用户的个数（主目录个数）
15. File \*p, \*q;//工作指针
16. File \*Head\_MFD;//指向目录的指针
18. **public**:
19. Catalog() {//构造函数，初始化“邻接表”,用户目录为空（此时没有文件）
20. cout << "请输入用户个数：";
21. cin >> num;
22. **if** (num <= 0) {
23. cout << "输入非法，请重新输入：";
24. cin >> num;
25. }
26. **while** (num > 5) {
27. cout << "用户太多，超出所能承受范围，请重新输入：";
28. cin >> num;
29. }
30. Head\_MFD = **new** File[num];
31. **for** (**int** i = 0; i < num; i++) {
32. Head\_MFD[i].name[0] = (**char**)(i + 1);
33. Head\_MFD[i].next = NULL;
34. }
35. cout << "主目录构建成功！" << endl;
36. }
38. **void** menu\_Control() {//正常工作状态，循环菜单栏
39. **do** {
40. cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n1.建立文件\n2.打开文件\n3.删除文件\n4.关闭文件\n5.读文件\n6.写文件" << endl;
41. cout << "请输入您想选择的功能：";
42. **int** direction;
43. cin >> direction;
44. **switch** (direction) {
45. **case** 1:file\_Build(); **break**;
46. **case** 2:file\_Open(); **break**;
47. **case** 3:file\_Delete(); **break**;
48. **case** 4:file\_Close(); **break**;
49. **case** 5:file\_Read(); **break**;
50. **case** 6:file\_Write(); **break**;
51. }
52. } **while** (1 > 0);
53. }
55. **void** file\_Build() {//函数·建立文件
56. **int** temp;
57. cout << "\n目前有" << num << "个用户，分别是：";
58. **for** (**int** i = 0; i <= num - 1; i++) {
59. cout << "用户" << i + 1 << "\t";
60. }
61. cout << "\n请选择在哪个用户文件夹里建立文件：";
62. cin >> temp;
63. **while** (temp > num || temp <= 0) {
64. cout << "输入非法，请重新输入：";
65. cin >> temp;
66. }
67. p = Head\_MFD[temp - 1].next;
68. q = **new** File;
69. cout << "请输入新建文件名：";
70. cin >> q->name;
71. cout << "请输入文件内容：";
72. cin >> q->content;
73. q->next = NULL;
74. q->opened = 1;
75. **if** (p == NULL) {//直接插文件
76. Head\_MFD[temp - 1].next = q;
77. }
78. **else** {//文件夹中存在文件的情况，在链尾插
80. **if** (!file\_Check(p, q->name)) {
81. cout << "该目录中已存在该文件。\n";
82. **return**;
83. }
85. **while** (p->next != NULL) {
86. p = p->next;
87. }
88. p->next = q;
89. }
90. cout << "文件创建完毕！位于目录" << temp << "中\n";
91. }
93. **void** file\_Open() {//函数·打开文件
94. **int** choice;
95. **char** name[20];
96. cout << "请问你想打开哪个用户文件夹中的文件（输入数字)：";
97. cin >> choice;
98. **if** (Head\_MFD[choice - 1].next == NULL) {
99. cout << "该文件夹中不存在文件。\n";
100. **return**;
101. }
102. cout << "请问你想打开用户" << choice << "中哪个文件？（输入名字）：";
103. cin >> name;
104. p = Head\_MFD[choice - 1].next;
105. **while** (p != NULL) {
106. **if** (!strcmp(p->name, name)) {
107. p->opened = 1;
108. **break**;
109. }
110. p = p->next;
111. }
112. **if** (p == NULL) {
113. cout << "未找到该文件，无法打开。\n";
114. **return**;
115. }
116. cout << "已打开" << p->name << "文件，内容是" << p->content << "！\n";
117. }
119. **void** file\_Delete() {//函数·删除文件
120. **int** choice;
121. **char** name[20];
122. cout << "请问你想删除哪个用户文件夹中的文件（输入数字)：";
123. cin >> choice;
124. **if** (Head\_MFD[choice - 1].next == NULL) {
125. cout << "该文件夹中不存在文件。\n";
126. **return**;
127. }
128. cout << "请问你想删除用户" << choice << "中哪个文件？（输入名字）：";
129. cin >> name;
130. p = Head\_MFD[choice - 1].next;
131. **if** (!strcmp(p->name, name)) {//第一个位置就是要找的文件
132. Head\_MFD[choice - 1].next = Head\_MFD[choice - 1].next->next;
133. }
134. **else** {//要找的文件在后面，或者不存在
135. q = p->next;
136. **while** (q != NULL) {
137. **if** (!strcmp(q->name, name)) {
138. p->next = q->next;
139. **break**;
140. }
141. }
142. **if** (q == NULL) {
143. cout << "未找到该文件，无法删除。\n";
144. **return**;
145. }
146. }
147. cout << "已删除" << q->name << "文件！\n";
148. }
150. **void** file\_Close() {//函数·关闭文件
151. **int** choice;
152. **char** name[20];
153. cout << "请问你想关闭哪个用户文件夹中的文件（输入数字)：";
154. cin >> choice;
155. **if** (Head\_MFD[choice - 1].next == NULL) {
156. cout << "该文件夹中不存在文件。\n";
157. **return**;
158. }
159. cout << "请问你想关闭用户" << choice << "中哪个文件？（输入名字）：";
160. cin >> name;
161. p = Head\_MFD[choice - 1].next;
162. **while** (p != NULL) {
163. **if** (!strcmp(p->name, name)) {
164. p->opened = 0;
165. **break**;
166. }
167. p = p->next;
168. }
169. **if** (p == NULL) {
170. cout << "未找到该文件，无法关闭。\n";
171. **return**;
172. }
173. cout << "已关闭" << p->name << "文件"  << "！\n";
174. }
176. **void** file\_Read() {//函数·读文件
177. file\_Open();
178. }
180. **void** file\_Write() {//函数·写文件
181. **int** choice;
182. **char** name[20];
183. cout << "请问你想修改哪个用户文件夹中的文件（输入数字)：";
184. cin >> choice;
185. **if** (Head\_MFD[choice - 1].next == NULL) {
186. cout << "该文件夹中不存在文件。\n";
187. **return**;
188. }
189. cout << "请问你想修改用户" << choice << "中哪个文件？（输入名字）：";
190. cin >> name;
191. p = Head\_MFD[choice - 1].next;
192. **if** (!strcmp(p->name, name)) {//第一个就找到文件
193. cout << "请您输入修改后的文件名：";
194. cin >> p->name;
195. cout << "请您输入修改后的文件内容：";
196. cin >> p->content;
197. p->opened = 1;
198. }
199. **else** {//要找的文件不在第一个（在后面的位置或不存在）
200. **while** (p != NULL) {
201. **if** (!strcmp(p->name, name)) {
202. cout << "请您输入修改后的文件名：";
203. cin >> p->name;
204. cout << "请您输入修改后的文件内容：";
205. cin >> p->content;
206. p->opened = 1;
207. **break**;
208. }
209. p = p->next;
210. }
211. **if** (p == NULL) {
212. cout << "未找到该文件，无法打开。\n";
213. **return**;
214. }
215. }
216. cout << "已修改" << p->name << "文件，内容是" << p->content << "！\n";
217. }
219. **bool** file\_Check(File \*l,**char** s[]) {//函数·创建文件时查重
220. **while** (l != NULL) {
221. **if** (!strcmp(l->name, s)) {
222. **return** **false**;
223. }
224. l = l->next;
225. }
226. **return** **true**;
227. }
228. };
230. **int** main()
231. {
232. cout << "欢迎来到多用户文件管理系统！\n设计者：陈四贵 仲响 周泽江\n";
233. Catalog \*test = **new** Catalog();
234. test->menu\_Control();
235. }