北京郵電大學

# 实验报告


题目：实验二：线性表-栈-串综合应用实验

班　　级：2022211320

学　　号：2022211683

姓　　名：张晨阳

学　　院：计算机学院（国家示范性软件学院）

2023 年 10 月 28 日

# 一、实验目的

1. 熟悉线性表的实现与应用；
2. 熟悉栈的实现与应用；
3. 熟悉串的实现与应用；
4. 培养根据实际问题合理选择数据结构的能力；
5. 学习自己查找相关资料以解决实际问题的能力。

# 二、实验环境

1. Windows 11
2. Visual Studio Code 1.83.1
3. x86-64 gcc 10.3

# 三、实验内容（简单行编辑程序）

**问题描述：**

文本编辑程序是利用计算机进行文字加工的基本软件工具，实现对文本文件的插入、删除等修改操作。限制这些这些操作以行为单位进行的编辑程序称为行编辑程序。本实验将综合利用所学知识，设计并实现一个数据结构用来在计算机内存中建立一个行文本文件的映像，并实现后面要求的一些基础操作命令，为简化实验过程，可假设编辑处理的文本文件为 C/C++源代码文件，且文件中均为 ASCII 码字符，不含中文等多字节字符。

**要求：**

- 功能 1：实现打开并读入文件的命令，执行该命令可以将指定文件的内容读取到内存中自己建立的数据结构中。
- 功能 2：在屏幕上输出指定范围的行，执行该命令可以将已经读入内存的文本文件的某几行显示到屏幕上。
- 功能 3：行插入，执行该命令可以在指定位置插入行。
- 功能 4：行删除，执行该命令可以删除一行或连续的几行。
- 功能 5：行内文本插入，执行该命令可以在某一行的某个字符处插入一个或多个字符。
- 功能 6：行内文本删除，执行该命令可以在某一行的某个字符处删除若干字符。
- 功能 7：文本查找，执行该命令可从指定行开始查找某个字符串第一次出现的位置。
- 功能 8：文本文件保存，执行该命令可以将修改后的文本内容写入到一个新的文件中。

  **附加要求：**

  检查源代码文本文件中的{}()是否匹配。

# 四、实验步骤（基础80分+附加10分）

初始 text.txt 文件内容为

```
1   12345
2   abcde
3   67890
4   fghij
5   bupt+2022
6   2022-bupt
```

## 数据结构选择：

1. 用线性链表（Linked List）处理文本文件。每个节点可以表示文件中的一行文本，具有指向前一行和后一行的指针，这样可以轻松进行插入和删除操作。另外，每个节点可以存储一行文本内容。

```
// 定义 LineNode 结构
struct LineNode
{
    char *text;             // 存储一行文本内容
    struct LineNode *prev; // 指向前一行的指针
    struct LineNode *next; // 指向后一行的指针
};
```

双向链表结构

2. 用栈（Stack）进行{}()的匹配：每当遇到一个{或(时，将其推入栈中，当遇到相应的右括号时，弹出栈顶元素，并检查是否匹配。

```
struct Stack
{
    char data;
    struct Stack *next;
};
```

栈结构

## 功能 1

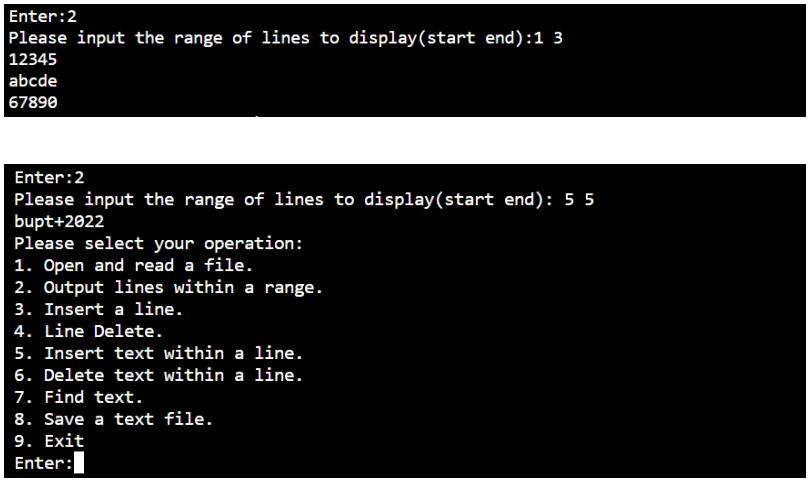
- 首先输入 `1` 选择进行功能 1 的操作；
- 用 `fopen` 以只读方式打开文件 test. txt，若打开成功，`file` 将指向该文件；否则，`file` 为 `NULL` 表示文件打开失败并退出程序；
- 定义字符串数组 `line` 来存储从文件中读取的每一行文本；
- 使用 `fgets` 函数从文件中读取一行文本并存储于 `line`；
- 调用 `strdup` 函数将 `line` 中的内容复制到当前节点的 text 中；
- 设置新节点的下一个节点指针为空，然后判断是否为头节点，若不是，将新节点连接到尾节点，同时新节点的前一个节点指针指向尾节点，最后更新尾节点指针为新节点；
- 执行上述步骤直到文件结束。
- 运行截图如图：

```
PS D:\VSCODE\C_C++\Lab02> .\CODE.exe
Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:1
The file has been successfully opened and read into memory!
```

功能1

## 功能 2

- 接收两个参数，分别是需要输出行的开始行数 `start` 和结尾行数 `end`；
- `lineNum` 用于跟踪当前行号，初始值为 1；
- `struct LineNode *current` 用于遍历链表，初始值为链表的头节点（`head`）。
- 使用 `while` 循环来遍历链表，同时检查两个条件：当前节点不为空（`current != NULL`）且当前行号不超过结束行号。
- 在循环中，首先检查当前行号是否在指定的范围内，若是，则使用 `printf` 函数打印当前节点的文本内容。
- 然后，将 `current` 指针移动到下一个节点，以继续遍历链表。
- 最后，递增 `lineNum`，以便在下一次迭代时跟踪下一个行号。
- 运行截图如图：




功能2

## 功能 3

- 首先 `currentLine` 用于跟踪当前行号，初始化为1；
- 创建一个新的链表节点 `newNode`，并为其分配内存，然后将新文本内容 `newText` 复制到 `newNode->text` 中；
- 使用 `while` 循环遍历链表，同时检查当前行号是否等于要插入的行号 `lineNumber`；
- 如果当前行号等于要插入的行号，将 `newNode` 插入到链表中，并进行必要的指针更新操作，以保持双向链表的正确性；
- 如果行号大于链表的行数，则将新节点追加到链表的末尾；
- 运行截图如图：（分别在第一行与第三行插入了新行，并用功能 2 显示）

```
Enter:2
Please input the range of lines to display(start end):1 4
12345
abcde
67890
fghij

Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:3
Please enter the line number where the insertion is: 1
Insert Successfully!

Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:3
Please enter the line number where the insertion is: 3
Insert Successfully!

Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:2
Please input the range of lines to display(start end):1 4
This is a new line.
12345
This is a new line.
abcde
```

功能3

## 功能 4

- `struct LineNode *current` 用于遍历链表，初始值为指向链表头的指针，由于需要修改 `head` 指针，所以参数是指向指针的指针 (`**head`)；
- `currentLine` 用于跟踪当前行号，初始化为1；
- 使用 `while` 循环遍历链表，同时检查当前行号是否在要删除的范围内；
- 如果当前行号在范围内，执行以下操作：
  - 更新前一个节点的 `next` 指针，以跳过当前节点。
  - 更新后一个节点的 `prev` 指针，以跳过当前节点。
  - 如果当前节点是链表的头节点，更新 `head` 指针，使其指向下一个节点。
  - 释放当前节点的文本内容和节点本身的内存，以防止内存泄漏。
- 如果当前行号大于 `end`，表示删除操作已经完成，输出 "Delete Successfully!" 并返回函数。
- 运行截图如图：（删除第 1 行与第 2 行）

```
Enter:2
Please input the range of lines to display(start end):1 5
12345
abcde
67890
fghij
bupt+2022

Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:4
Please input the range of lines you wanna delete(start end):1 2
Delete Successfully!

Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:2
Please input the range of lines to display(start end):1 3
67890
fghij
bupt+2022
```

功能4

## 功能 5

- 首先遍历链表找到指定行的节点；
- 然后为新文本分配足够的内存空间，包括原始文本长度、插入文本长度和结尾空字符的空间；
- 使用 `strncpy` 将当前文本的一部分复制到新文本中，位置在指定的插入位置之前，将要插入的文本复制到指定位置，将当前文本中剩余部分复制到新文本中已插入文本之后的位置；
- 释放原始文本的内存，并将当前节点的文本指针指向新文本。
- 最后，打印出一条插入成功的指令。
- 运行截图如图：（在第 1 行字符位置 2 处 (理解为 line1[2])插入"abd"）

```
Enter:5
Please enter the line number where the insertion is: 1
Please enter the position where the insertion is:2
Please enter the text to insert:abd
Text Insert Successfully!

Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:2
Please input the range of lines to display(start end):1 1
12abd345
```

功能5

## 功能 6

- 首先遍历链表找到指定行的节点；
- 若删除长度超过了当前文本的剩余长度，将长度调整为合适的值；
- 为新文本分配足够的内存空间，包括删除指定长度后的剩余文本和结尾空字符的空间；

- 使用 `strncpy` 将删除位置之前的文本复制到新文本中，再将删除长度之后的文本复制到新文本中；
- 最后，打印出删除成功的指令。
- 运行截图如图：（删除了第 2 行字符位置 2 之后的所有字符）

```
Enter:6
Please enter the line number where the deletion is: 2
Please enter the position where the deletion is:2
Please enter the text length to delete:10
Text Delete Successfully!

Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:2
Please input the range of lines to display(start end):1 2
12345
ab
```

功能6

## 功能 7

- 首先遍历链表找到指定行的节点；
- 使用 `strstr` 函数在当前行的文本中查找指定的字符串；
- 若找到字符串，则计算它首次出现的位置并打印。位置的计算基于指针算术，即 `found` 指针位置减去 `current->text` 指针位置，再加 1；
- 若没有找到指定的字符串，打印"NotFound!"。
- 运行截图如图：（分别查找了第 1 行"34", 第 2 行"asf"）

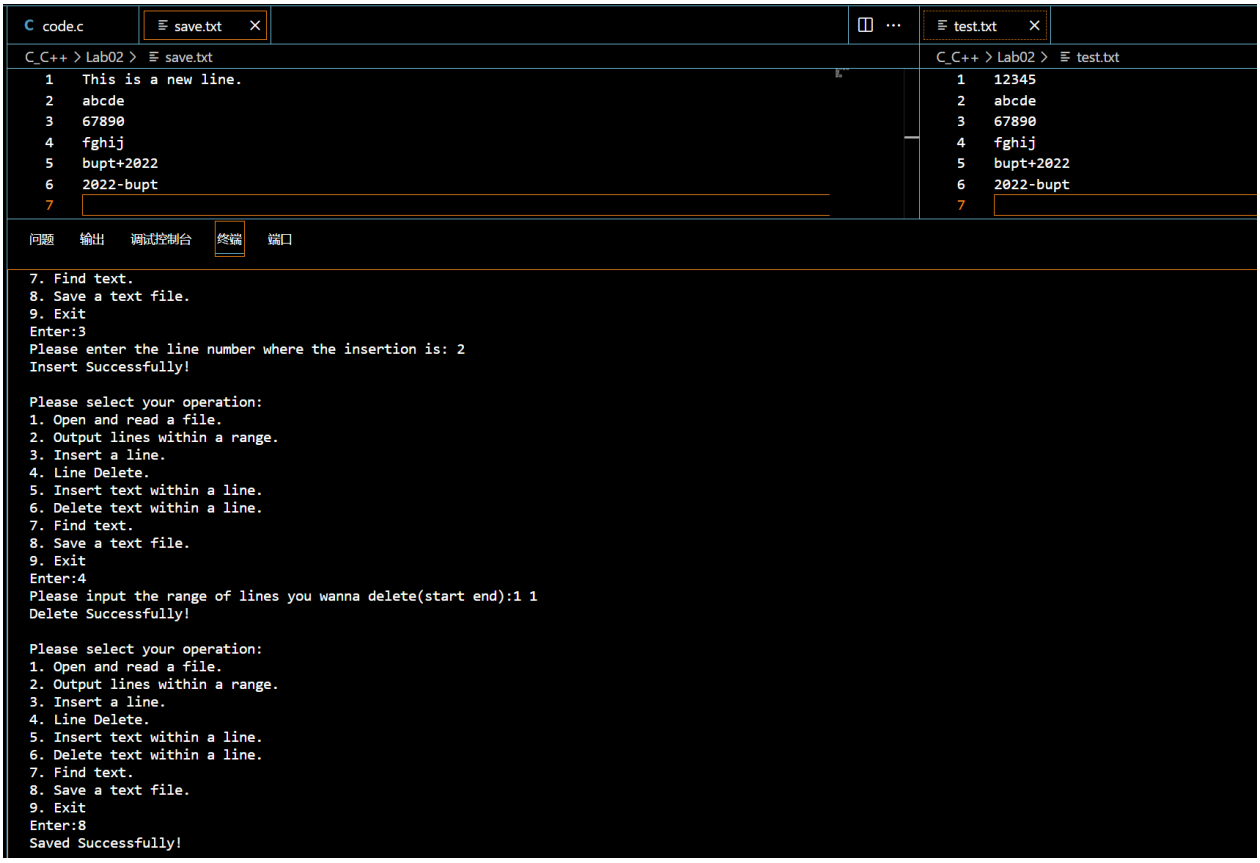
```
Enter:7
Please enter the line number where the search is: 1
Please enter the text you wanna search:34
The first position is 3

Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:7
Please enter the line number where the search is: 2
Please enter the text you wanna search:asf
NotFound!
```

功能7

## 功能 8

- `FILE *file = fopen("save.txt", "w");` 打开一个文件 "save. txt" 来保存修改后的内容；
- 遍历链表中的每个节点，将每个节点的文本内容使用 `fputs` 函数写入到文件中；
- 通过 `current = current->next` 移动到链表的下一个节点，以便继续写入下一行的文本；
- 循环完成后，使用 `fclose(file)` 关闭文件；

- 最后，输出 "Saved Successfully!" 表示保存操作成功。
- 运行截图如图：（在第 2 行插入了新行，删除了第 1 行）

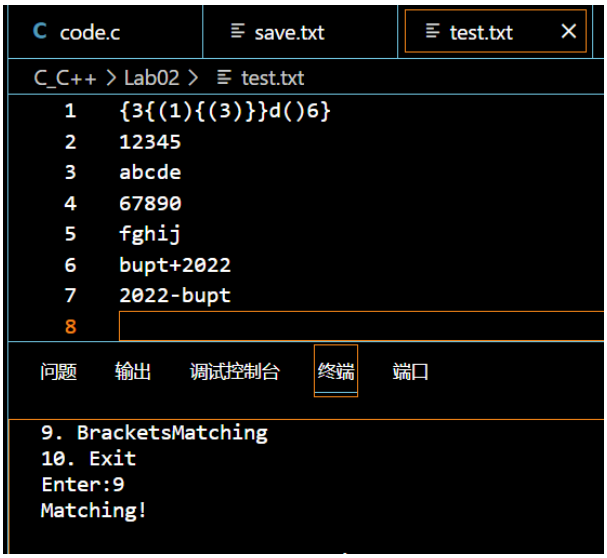



功能8

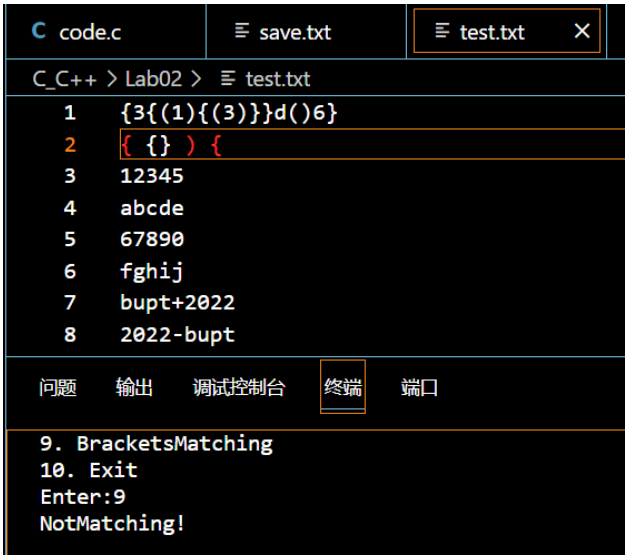
## 附加要求

- 建一个栈（`struct Stack`），用于存储遇到的开括号，以便后续匹配；
- 循环遍历文本的每一行，使用 `node` 表示当前行的节点，在每行中，使用 `line` 表示当前行的文本，然后遍历文本的每个字符；
- 如果字符是 { 或 (，将其推入栈中以等待匹配；
- 如果字符是 } 或 )，尝试与栈中的括号匹配。如果匹配失败，输出 "NotMatching!" 并返回，表示括号不匹配；
- 如果成功遍历了整个文本，且栈为空，说明所有括号都匹配，输出 "Matching!"。

- 运行截图如图：（测试 1 是匹配的情况，测试 2 是不符合的情况，均通过）




测试1




测试2

# 五、实验分析和总结（10 分）

## 综合分析

### 思路设想

**功能 1：读入文件**

读入文件时，可以将每一行文本创建一个 `LineNode` 节点，并将其插入链表中，建立一个文本文件的映像。

**功能 2：输出指定范围的行**

要输出指定范围的行，只需遍历链表，从链表的头部或指定位置开始，输出所需行数的文本内容。

**功能 3：行插入**

行插入操作可以通过创建新的 `LineNode` 节点并调整链表指针来实现。在链表中找到要插入的位置，然后插入新节点即可。

**功能 4：行删除**

行删除操作可以通过找到要删除的行，并调整链表指针来实现。删除节点后，释放相关内存。

**功能 5：行内文本插入**

行内文本插入可以在 `LineNode` 节点的文本内容中插入字符，类似于字符串操作。

**功能 6：行内文本删除**

行内文本删除可以在 `LineNode` 节点的文本内容中删除字符，也类似于字符串操作。

**功能 7：文本查找**

文本查找可以遍历链表，从指定行开始，查找指定字符串，如果找到，返回位置信息。

**功能 8：文本文件保存**

文本文件保存可以通过遍历链表，逐行写入到一个新的文件中来实现。

**附加要求：检查{}()是否匹配**

检查源代码文本文件中的{}()是否匹配需要使用栈（Stack）数据结构。每当遇到{或(时，将其推入栈中，当遇到相应的闭括号时，弹出栈顶元素，并检查是否匹配。如果栈在处理完所有字符后为空，那么括号匹配。

对于链表不涉及字符串的操作，时间复杂度为 $O(n)$（$n$ 是链表节点数量，即行数），对于栈的操作以及链表涉及字符串的操作，时间复杂度为 $O(nm)$ ($n$ 为行数，$m$ 为字符个数)。

# 所遇问题及心得

## 功能 1

**问题 1_无法打开文件**：试图用字符串存储文件名作为参数传递给 `fopen`,但发现无法打开.

```
// char filename[100];
// printf("请输入文件名：");
// scanf("%s", filename);
// openFile(filename);
```

经过查询发现此方法需要输入完整的文件路径，考虑实验演示的方便，故改为直接打开"test. txt"文件。

## 功能 2

**问题 2_输出行格式错误：** 当直接使用 `printf("%s\n", current->text);` 输出时，每行之间会多出一行。

```
Please input the range of lines to display(start end):1 3
123456

abcdef

bupt2022
```

分析原因是在功能 1 向内存读取文件内容时，读入了换行符，导致输出时连续换行两次。修改：删去 `\n`。

## 功能 3

**问题 3_插入行前插和后插的选择：** 最初的插入只考虑了头节点的特殊情况，导致在第 1 行前插入（插入行为 1）正确，在第 2行插入（插入行为 2）变为了后插（插在了第 3行）。

```
Enter:3
Please enter the line number where the insertion is: 2
Please select your operation:
1. Open and read a file.
2. Output lines within a range.
3. Insert a line.
4. Line Delete.
5. Insert text within a line.
6. Delete text within a line.
7. Find text.
8. Save a text file.
9. Exit
Enter:2
Please input the range of lines to display(start end):1
3
12345
abcde
This is a new line.
```

经思考，发现是对于头节点和非头节点的插入，使用了同一种方法，导致没有弄清楚 `current` 节点的指向。修改：使用 `if` 语句将头节点情况与普通情况分开插入，具体操作见源代码。

## 心得

这次实验加深了我对链表，栈以及字符串的理解，特别是对于链表的结构清晰明了了很多，对于他们的操作也熟悉了不少。

同时也明白了对于链表的操作，在不熟练的情况下，可以通过画图来帮助自己解决问题，切忌对着代码空想，很容易产生未曾设想的问题。

# 六、程序源代码（10 分）

见附件 code. c 和 stack. h