四川大学计算机学院

学生实验报告

班级 行政2班 姓名 严涵栩 学号 2023141410163

一、实验二:A Text Editor Imlementation

二、实验的目的和要求:

1.采用C++的ASCII码文件和串函数实现;

2.熟练掌握串运算的应用;

3.熟练掌握计算机系统的基本操作方法,了解如何编辑、编译、链接和运行一个C++程序;

4.上机调试程序,掌握查错、排错使程序能正确运行。

三、实验的环境:

**1. 硬件环境:** MacBook Air (芯片：Apple M2, 内存：16GB)

**2. 软件环境:** macOS, Visual Studio Code (VScode), g++ 编译器

四、算法描述:

1. **数据结构：**

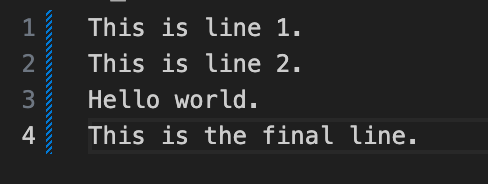
程序使用一个双向链表 (DLinkList) 作为文本内容的缓冲区。链表中的每一个节点 (Node) 均包含一个 std::string data 成员，用于存储文件中的一行文本。每个节点还包含 Node\* prev 和 Node\* next 指针，分别指向其前驱节点和后继节点。

这种结构（双向链表）的优势在于插入和删除操作效率高（时间复杂度为O(1)），非常适合模拟文本行的插入和删除。

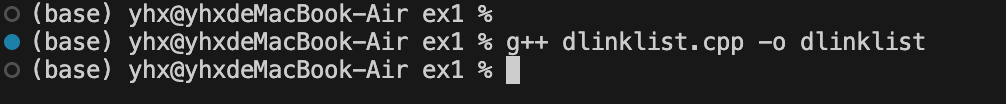
1. **核心类设计 (TextEditor)：**

TextEditor 类封装了所有编辑器的功能。它包含一个 DLinkList buffer 成员，用于存储所有文本行。一个 Node\* cursor 指针，用于指向用户当前正在操作的行。一个 int cursor\_lines 整数，用于追踪当前光标所在的行号。input\_filename 和 output\_filename 字符串用于存储读写的文件名。

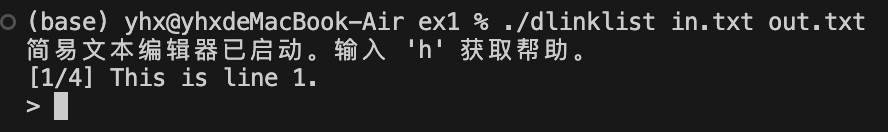
1. **程序流程：见algorithm.txt**
2. 源程序清单: 见dlinklist.cpp
3. 运行结果:
4. 准备输入文件

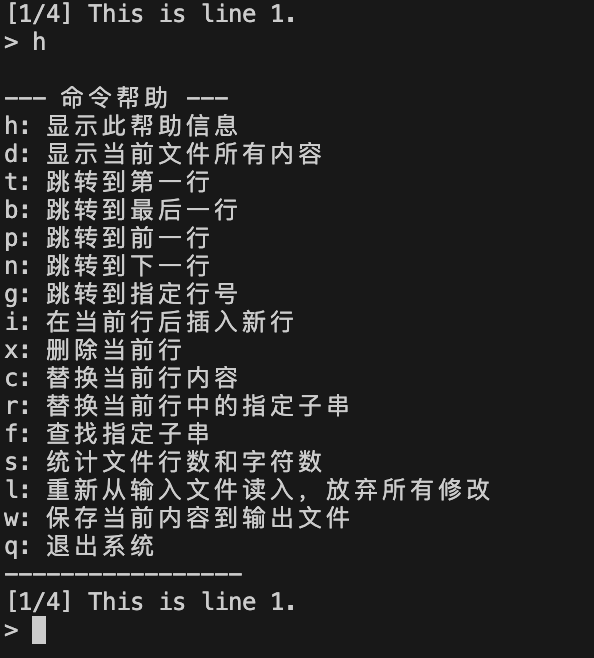


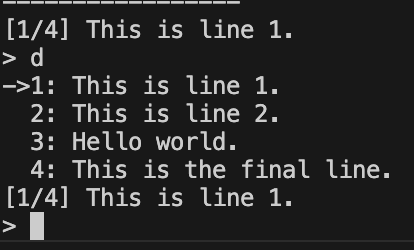
1. 编译程序

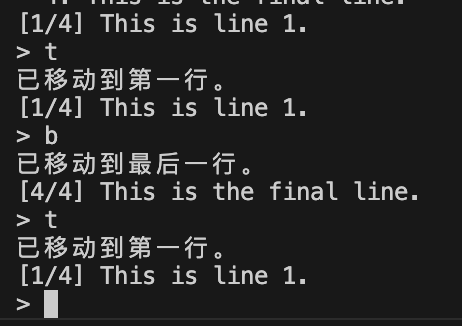


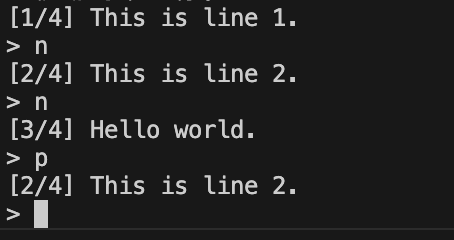
1. 运行程序并交互

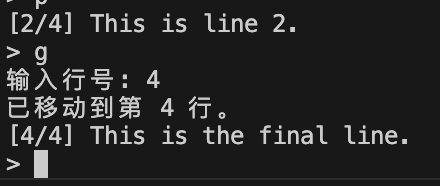


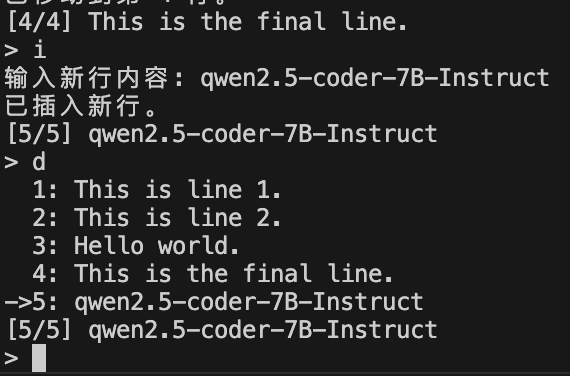


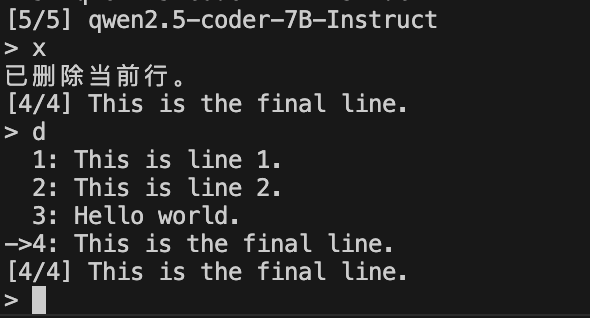


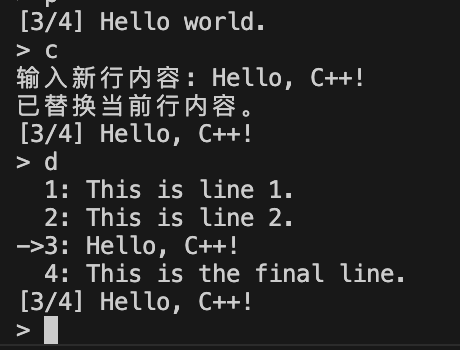


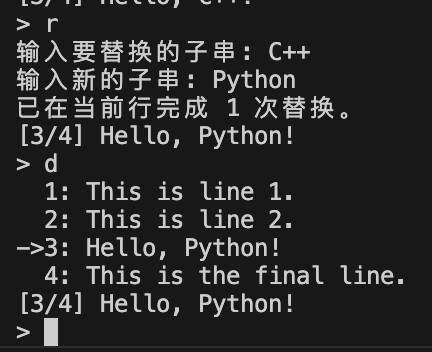


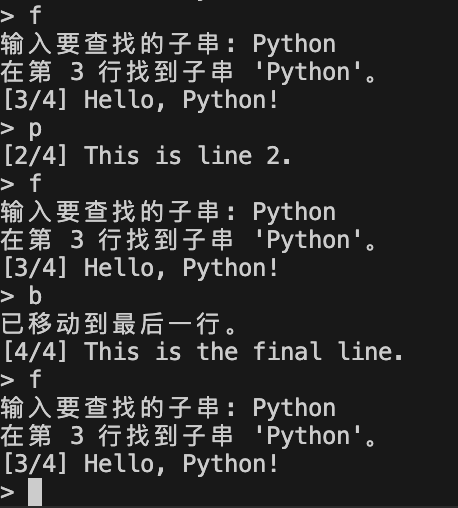




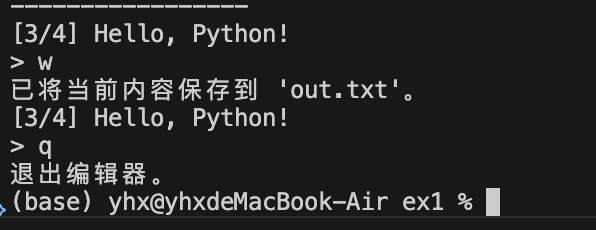


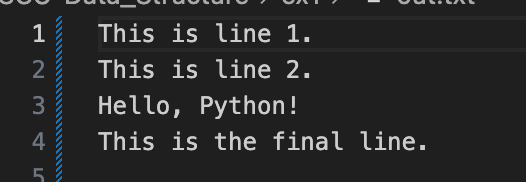












七、实验运行情况分析(包括算法、运行结果、运行环境等问题的讨论)。

**1. 算法分析：**

数据结构选择： 本实验的核心是使用双向链表存储文本行。这是一个非常合适的数据结构。

优点： 对行的插入和删除操作（insert\_after, delete\_node）非常高效，时间复杂度为 O(1)（假设已持有光标指针）。这远优于使用数组或 std::vector（插入/删除时间复杂度为 O(N)）。

缺点： 随机访问（如 move\_to\_line）效率较低，时间复杂度为 O(N)，因为它需要从当前位置开始遍历。

**字符串处理：** 实验要求熟练使用串运算。在 replace\_substring\_in\_current\_line 函数中，使用了 std::string::find 来定位子串，并使用 std::string::replace 来替换内容，这很好地满足了实验要求。getline 函数用于读取整行，避免了空格导致的中断。

1. **运行结果分析：**

从第六节的运行结果可以看出，程序的功能均已正确实现。

文件加载与保存 (l, w)： 程序能正确读入 input.txt，并通过 w 命令将修改后的缓冲区内容（包括插入、替换、删除）正确写入 output.txt。

导航 (t, b, p, n, g)： 光标可以正确地在首、尾、上一行、下一行和指定行之间跳转，cursor\_lines 始终保持正确。

编辑 (i, x, c, r): i (插入) 能在当前行后正确添加新节点。x (删除) 能正确删除当前节点，并智能地将光标移动到相邻节点（优先下一行，其次上一行），避免了指针悬空。c (替换行) 和 r (替换子串) 均能正确修改当前 cursor->data 的 std::string 内容。

统计 (s)： 能够正确遍历链表，统计总行数（buffer.length）和总字符数。

1. **运行环境与调试：**

环境： 程序在 macOS (M2芯片) 环境下，使用 g++ 编译器编译通过，没有出现平台相关错误，表明代码具有良好的可移植性。

调试： 在开发过程中，cin 和 getline 的混用是一个常见的陷阱。例如，在 run() 循环中，cin >> command 只读取了命令字符，留下了换行符在缓冲区。后续的 getline（如 insert\_line\_after 中）会立即读到这个换行符而导致输入失败。本程序使用了 cin.ignore(numeric\_limits<streamsize>::max(), '\n'); 来清空输入缓冲区，有效解决了这个问题。