组号: 12



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：2023-2024 年冬季 组 长： 孔馨怡 学 号： 22122128 指导教师： 朱能军 成绩评定： （教师填写）

二〇二四年一月二日

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **小组信息** | | | | |
| 登记序号 | 姓名 | 学号 | 贡献比 | 签名 |
| 1 | 孔馨怡 | 22122128 | 33.3% | 孔馨怡 |
| 2 | 高语涵 | 22120828 | 33.3% | 高语涵 |
| 3 | 杨利亚 | 22122418 | 33.3% | 杨利亚 |

|  |  |
| --- | --- |
| **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 | Chapter 3---双向链表的应用：发简历 |
| 实验二 | Chapter 4---队列/栈的应用：车厢调度问题 |
| 实验三 | Chapter 5---模式匹配的 KMP 算法：文学研究助手 |
| 实验四 | (*实验题目*) |

实验一

一、**实验题目：文学研究助手**

**1.问题描述**

文学研究人员需要统计英文小说中某些词出现的次数和位置。试编写一个实现 这一 目标的文字统计系统，称为“文学研究助手”。

**2.基本要求**

英文小说存于一个文本文件中，并假设小说中的单词一律不跨行，每行的长度 不超过 120 个字符，待统计的词汇集合要一次输入完毕。要求对英文小说扫描 一遍就完成统计工作。程序的输出结果是每个单词的出现次数和出现位置所在 行的行号。其格式自行设计。

**3.输入数据**

输入数据包括两部分，第一部分是要统计的单词，不超过 100 个，单词之间用 空格 分隔；第二部分是被统计的文章，可以考虑把这两部分内容放在一个文件 中。例如：以某一 C++源程序模拟英文小说，用 C++程序设计语言的保留字集 作为待统计的词汇集。

**4.输出数据**

对出现在文章中的要统计的单词，输出其在文章中出现的次数和所在的行号。

**二、实验内容**

该实验的目标是设计一个文学研究助手程序，用于统计英文小说中指定单词 的出现次数和位置。程序的大致流程是:首先读取包含小说文本的文件，然后用 户输入要统计的单词集合。接着， 程序扫描文本，统计每个单词在文本中的出现 次数和位置，并最终输出结果。通过利用 KMP 算法，巩固模式匹配的思想，深入 理解其规律和特点。

**三、解决方案**

1、算法设计（主要描述数据结构、算法思想、主要操作、用例分析、改进方法 等）

(1)数据结构

本次实验利用 SearchItem 结构体和 vector 的 lines 向量和 search 向量来实 现。

(2)算法思想

程序通过 fs 指令打开指定的文本文件进行逐行读取，并将每一行存储在 vector<string>类型的向量中。接着将用户输入的需要统计的单词集合存入 vector<SearchItem>类型的向量中。接着进行单词匹配， 其操作包括遍历每一行 文本，逐字符扫描、对于字符进行类型检查、更新单词出现次数及输出匹配信息。

(3)主要操作

1.文件读取：

l 使用 fstream 打开指定的文本文件。

l 若文件不存在或打开失败，则提示用户重新输入文件名。

l 使 用 getLine 逐 行 读 取 文 本 内 容 ， 并 将 每 一 行 的 内 容 存 储 在

vector<string> 类型的 lines 向量中。 2.用户输入部分：

l 用户首先输入文本文件的文件名，程序尝试打开文件。

l 如果文件不存在或打开失败，提示用户重新输入文件名，直到成功打开文件。

l 用户随后输入要统计的单词集合，以空格分隔，输入 “#” 表示结束。

l 这些单词以及其相关信息被存储在 vector<SearchItem> 类型的 search 向量中。

3.单词匹配：

l 遍历 lines 向量中的每一行，逐字符扫描文本。

l 对于每个字符，检查是否为字母或单引号。如果是字母， 将大写字母转换为 小写字母，然后逐个检查每个待匹配的单词。

l 在分隔符处（逗号、空格等） ，检查前一个字母是否匹配成功，若是则更新 出现次数和输出匹配信息。

4.输出结果部分：

l 在匹配过程中，输出每个单词在文章中的出现次数和所在行的行号。

l 最后输出每个单词总共出现的次数。 5.程序结束：

l 用户完成输入和查找后，程序输出统计结果，包括每个单词在文章中的出现 次数和所在行的行号，以及每个单词总共出现的次数。

l 程序执行完毕，结束运行。

2、源程序代码（要求有必要注释、格式整齐、命名规范，利于阅读）

打开文本文件： 读取文本：



输入并读取需要查找的单词：

//输入“#”表示结束输入操作



查找单词：

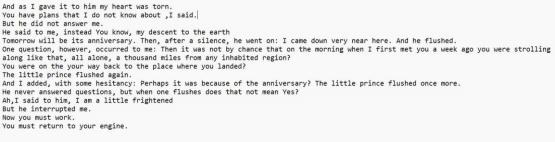




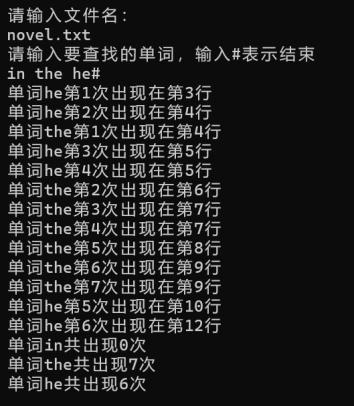
结果输出：

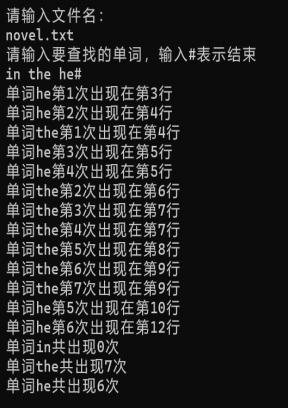


3、实验结果（展示实验结果、测试情况、结果分析等） 测试文本内容：



输入和输出：





4、算法分析（对算法空间、时间效率进行必要分析，可能的改进建议等）

（1）时间复杂度

l 文件读取部分：逐行读取文本内容，每行的字符数平均为常数（假设文本 行长度不超过 120 个字符），时间复杂度为 O(n)，其中 n 为文本总字符 数。

l 用户输入部分：输入的单词数量为常数，时间复杂度为 O(1)。

l 单词匹配部分：遍历每一行，对于每个字符逐个检查每个单词，时间复杂 度为 O(nmk)，其中 n 为行数，m 为单词的平均长度，k 为要查找的单词 数量。

l 输出结果部分：输出每个单词的次数和行号，时间复杂度为 O(k)，其中 k 为要查找的单词数量。

总体时间复杂度为 O(nmk)。

（2）空间复杂度

l lines 向量用于存储每一行的内容，其空间复杂度为 O(n)，其中 n 为文 本总行数。

l search 向量用于存储用户输入的单词信息，其空间复杂度为 O(k)，其中 k 为要查找的单词数量。

l SearchItem 结构体占用常数空间。 总体空间复杂度为 O(n + k)。

（3）改进建议

1.可以考虑使用更为高效的匹配算法以提高程序性能。

2.引入异常处理机制，处理文件打开失败等异常情况，而不是简单地输出“文件 打开失败”的提示字样。

3.在遇到分隔符时，可以把所有待查找单词按照当前已匹配成功的位置从小到大 排序，这样只需要找到第一个已匹配成功的单词并更新出现次数，不需要遍历整 个待查找单词的 vector。

4.在判断一个待查找单词是否还可能被匹配成功时，如果已经发现它无法匹配成 功，可以把它从 vector 中删除，这样可以避免循环次数过多导致性能下降。

但不能在循环遍历 vector 的时候修改 vector 的大小，可以另外新建一个 vector 存放还需要检查的单词。

5、总结与心得（主要描述实验过程中存在的问题、原因、解决方法、收获、对 实验内容的其他应用思考等）

（1）问题与解决

①实验起初不清楚如何打开文件以及常出现文件打开失败的情况，后续学习到使 用 fstream 流来打开和读取文件内容；

②在单词匹配过程中，通过使用结构体 SearchItem 记录每个单词的匹配状态， 是的匹配过程更加清晰高效。

（2）对其他应用的思考

在实验以外的题解过程中，要学会利用模式匹配的思想快速进行相应内容的查 找和匹配，也可以寻求更加高效的算法。