**数据结构课程设计《关键字检索系统》项目说明文档**

2253893 苗君文 软件工程

**1. 项目简介**

**1.1. 项目背景**

建立一个文本文件，文件名由用户用键盘输入，输入一个不含空格的关键字，统计输出关键字在文本中的出现次数。

**1.2. 项目要求**

本项目的设计要求可以分成两个部分实现：首先建立一个文本文件，文件名由用户用键盘输入；然后输入一个不含空格的关键字，统计输出该单词在文本中的出现次数。

**1.3. 输入格式**

本项目首先需要用户输入文件名，若输入的文件名不带.txt后缀，则程序会为其加上.txt后缀。然后，用户需要输入英文文本，用户可以输入多段英文文本，但每一段不得超过1000个字符，否则会被要求重新输入本段内容。用户需要输入“^”以结束文本的输入。最后，用户需要输入只含英文文本且不含空格的关键字，不符合要求也需要重新输入。

**1.4. 输出格式**

本项目针对输入有详细的输入错误处理，若用户输入错误，则程序会输出相关提示信息，并要求重新输入。除此之外，程序会输出关键字在文件内的文本中出现的次数。

**2. 设计思路**

**2.1. 数据结构设计**

本项目要求能查找关键字在文本中出现的次数，因此设计了String字符串数据结构，从而更方便地处理和操作字符串。通过自定义String类，可以根据项目需求添加特定的功能，例如计算失效函数、带失效函数的KMP匹配等，这从而能够更好地满足项目的特定需求。String类中的功能包括字符串的复制、连接、获取子串、计算失效函数、KMP 匹配等。这些功能在实现关键字检索系统时非常重要。而且，String类封装了字符串的各种处理操作，使得在项目中对字符串的处理更加方便和清晰。通过使用String，可以提高代码的可读性和可维护性也满足项目对字符串处理的需求。

**2.2. 类设计**

为了实现关键字检索系统的功能，设计了String类，负责存储并做出与字符串相关的一系列操作。String类的成员变量及成员函数如下：

成员变量：

* char\* ch: 字符串的存储数组。
* int curLen: 当前字符串的长度。
* int f[maxLen]: 失效函数数组。

成员函数：

* String(const String& ob): 复制构造函数，从已有串 ob 复制。
* String(const char\* init): 复制构造函数，从已有字符数组 init 复制。
* String(): 构造函数，创建一个空串。
* ~String(): 析构函数，释放动态分配的内存。
* int Length()const: 返回当前串的长度。
* String& operator()(int pos, int len): 从串中第 pos 个位置起连续提取 len 个字符形成子串。
* int operator==(const String& ob) const: 重载操作符，判断两个串是否相等。
* int operator!=(const String& ob) const: 重载操作符，判断两个串是否不相等。
* int operator!() const: 重载操作符，判断当前串是否为空。
* String& operator=(const String& ob): 重载操作符，串赋值。
* String& operator+=(const String& ob): 重载操作符，串连接。
* char& operator[](int i): 重载操作符，取当前串 \*this 的第 i 个字符。
* void getline(istream& is, String& str): 从输入流中读取一行。
* bool getlineFile(ifstream& inFile, String& str): 从文件输入流中读取一行。
* void fail(): 计算失效函数。
* int fastFind(String& pat) const: 带失效函数的 KMP 匹配算法。
* const char\* getCharArray() const: 获取字符数组。

**2.3. 全局函数设计**

* writeToFile 函数：从用户获取文本输入，写入到指定的文件。
* readAndOutputFile 函数：从指定文件读取文本，并输出到控制台。然后，调用 fastFind 函数计算关键字在文本中的出现次数。

**2.4. 主程序设计**

首先，获取文件名，提示用户输入文件名，包含文件扩展名（.txt）。

检查输入文件名是否包含文件扩展名，如果没有，则自动添加。

接着，写入文本到制定文件中，调用writeToFile函数，该函数实现了从用户获取文本输入并写入到指定文件的功能。在用户输入文本时，通过自定义的customGetline函数获取一行文本，直到用户输入 ^ 并回车结束。随后，将用户输入的文本写入到指定的文件中。

接下来，提示用户输入关键字，确保关键字只包含字母字符。使用自定义的 onlyAlpha 函数检查关键字是否只包含字母。

最后，读取文件并输出结果。调用readAndOutputFile函数，该函数从指定文件中读取文本并输出到控制台。在读取文本时，使用自定义的customGetlineFile函数获取一行文本。调用使用了KMP算法的fastFind函数计算关键字在文本中的出现次数，并输出到控制台。

从而，主程序能够完成用户与关键字检索系统的交互，包括文件的输入和输出，以及关键字在文本中的匹配统计。

**2.5. 输入错误处理设计**

**2.5.1. 单个参数的输入错误处理**

在询问用户是否需要继续运行程序时，使用 while 循环不断尝试获取用户输入，直到输入满足要求。使用 cin.fail() 及min、max来判断输入是否出错，检查输入是否在有效范围内。如果输入无效，则输出错误信息，清除输入缓冲区并忽略之后的字符，并重新输出输入的提示信息。通过一个字符来获取输入一个数之后的字符以检查输入字符的个数，若不正确，也输出错误信息，并重新输出输入的提示信息，执行相应清除操作。如果输入有效，则跳出循环。

**2.5.2. onlyAlpha()函数**

在检查输入的关键字中只有字母字符时，使用onlyAlpha()函数，该函数遍历检查字符串中的每个字符，若其中含有非字母字符，则直接返回false，否则遍历完全部字符后，返回true。

**2.5.3. 输入英文文本时处理存储分配错误**

在writeToFile函数中，程序要求用户输入英文文本，用户输入一个回车则将之前输入的内容视为一段。在输入每段内容时，通过customGetline函数读取输入的一段内容，用size函数获取输入的总长度，如果输入的长度大于maxLen，即1000，则会输出“存储分配错!”，并要求用户重新输入该段内容。

**3. 功能实现**

**3.1. fastFind(String& pat) const函数**

FastFind函数是本项目中的关键，使用了一种高效的字符串匹配算法，即KMP算法，它通过构建失效函数，能够在匹配失败时快速确定下一次匹配的起始位置，从而提高整体匹配效率。fastFind函数实现步骤具体如下：

函数开始时，首先初始化两个指针posP和posT，分别指向关键字pat和当前字符串ch的起始位置。然后，获取关键字pat和当前字符串ch的长度，分别用lengthP和lengthT表示。调用pat.fail()计算关键字的失效函数。失效函数的计算是一个关键步骤，它根据关键字的自身结构，在匹配失败时告诉算法应该从哪里开始重新匹配。失效函数fail通过遍历关键字的每个字符，计算对应位置的失效值f。f[j]表示关键字中以第j个字符结尾的子串的最长公共前缀后缀的长度。在匹配过程中，当关键字中的某个字符与当前字符串不匹配时，通过失效函数的值，快速调整关键字的位置。计算完f以后，设置一个计数器count用于统计关键字在当前字符串中出现的次数。而后，使用循环开始匹配，检查关键字pat的当前字符是否与当前字符串ch的当前字符相等，如果相等，继续比较下一个字符。如果posP指向了关键字的末尾，表示整个关键字已匹配成功。此时增加匹配次数计数器count，并通过失效函数调整posP的位置。字符不匹配且关键字的第一个字符不匹配，则移动当前字符串的指针 posT。若字符不匹配，则调整关键字的位置，根据失效函数调整 posP 的位置，使其指向关键字的下一个可能匹配的位置。最后，循环结束后，返回匹配次数count。

**3.2. 文件相关函数功能**

**3.2.1. writeToFile函数**

该函数用于将用户输入的文本逐行写入到指定文本文件的函数。其实现步骤如下：

首先通过ofstream类型的对象outFile打开指定的文件。然后，在循环中，通过customGetline函数逐行获取用户输入的文本内容。如果用户输入的行为一个 ^，表示输入结束，跳出循环。同时，需要检查当前行的长度是否超过预定的最大长度 maxLen。如果超过，提示错误并要求重新输入。符合要求则将有效的行写入到文件中。在文件写入后，关闭文件流，并输出提示信息，说明文本已保存在指定的文本文件中。

**3.2.2. readAndOutputFile函数**

该函数用于从指定的文本文件中读取文本内容，并输出这段内容，同时同济关键字在文本中的出现次数。其实现步骤如下：

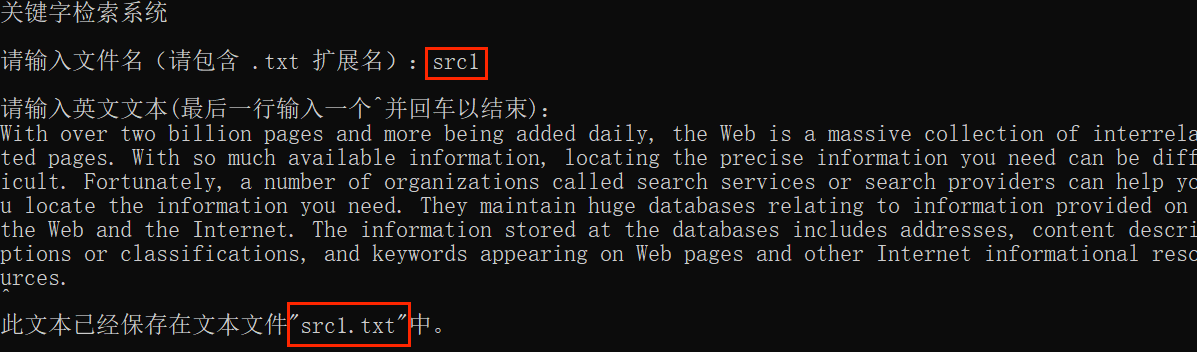
首先，通过ifstream类型的对象inFile打开指定的文本文件。使用循环，通过customGetlineFile函数逐行读取文件内容，同时输出到控制台。在读取文件的同时，调用fastFind函数统计关键字在当前行的出现次数。读取文件内容完成后，关闭文件流，并输出关键字的出现次数。

**4. 测试结果**

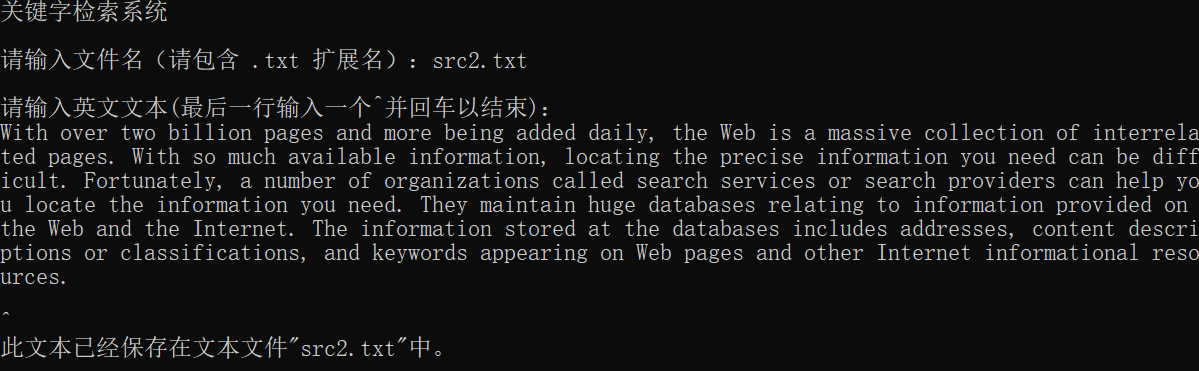
**4.1. 输入文件名**

本程序确保用户输入带有“.txt”扩展名的有效文件名。如果输入的文件名没有扩展名，它会附加“.txt”到文件名，保证程序正常运行。

（1）输入不带有“.txt”的文件名，程序会加上后缀名，保证正常运行。



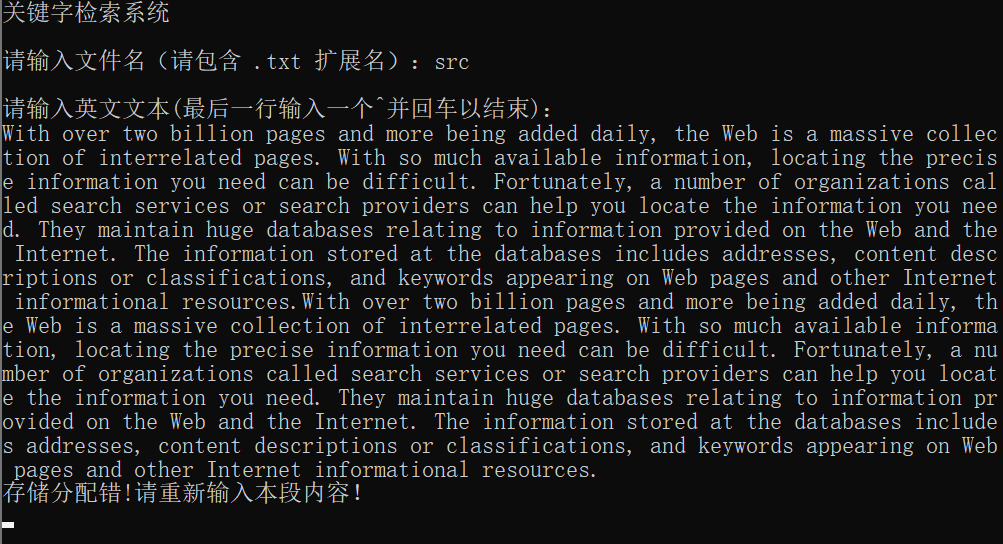
（2）输入带有“.txt”的文件名，程序也可以正常运行。



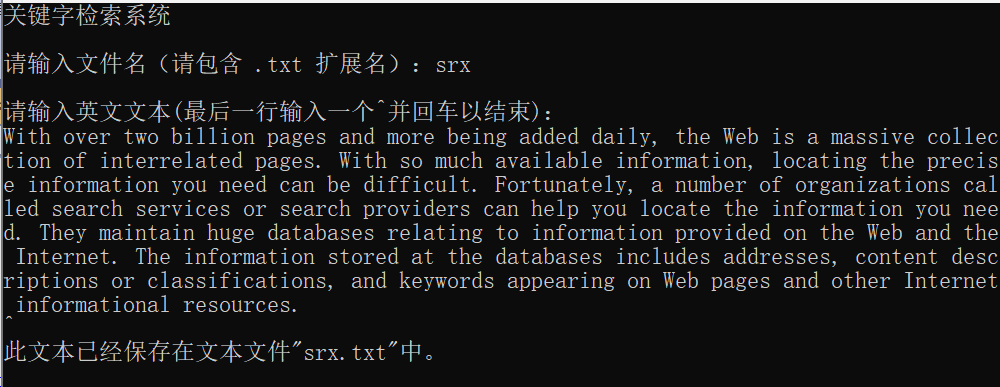
**4.2. 输入英文文本内容（最后一行输入一个^并回车以结束）**

本程序支持输入多段内容，但每段内容不得超过1000个字符，否则会提示用户重新输入此段内容。

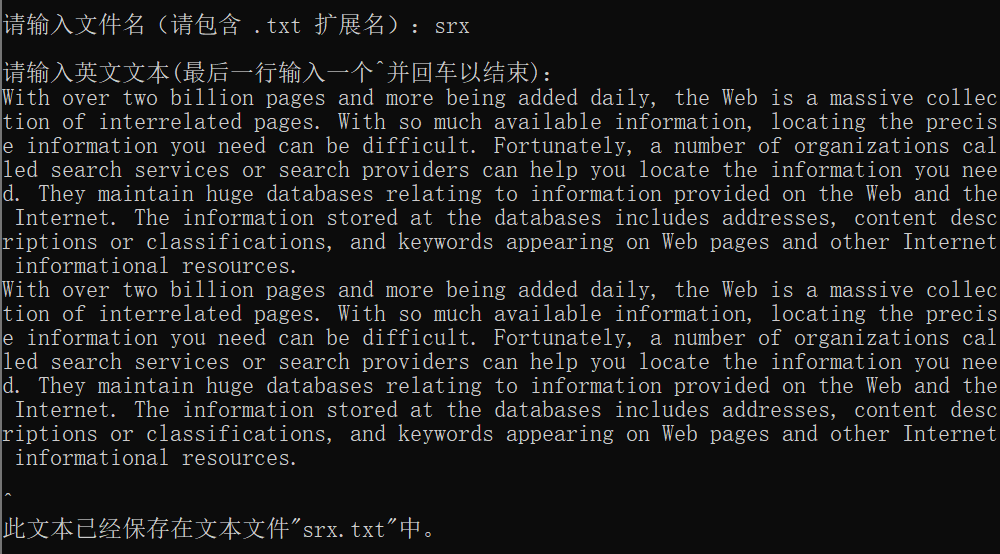
（1）输入一段超过1000字符的英文内容，提示“存储分配错误”，要求重新输入。



（2）本程序支持输入单段英文文本。



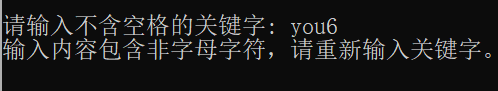
（3）本程序支持输入多段英文文本。



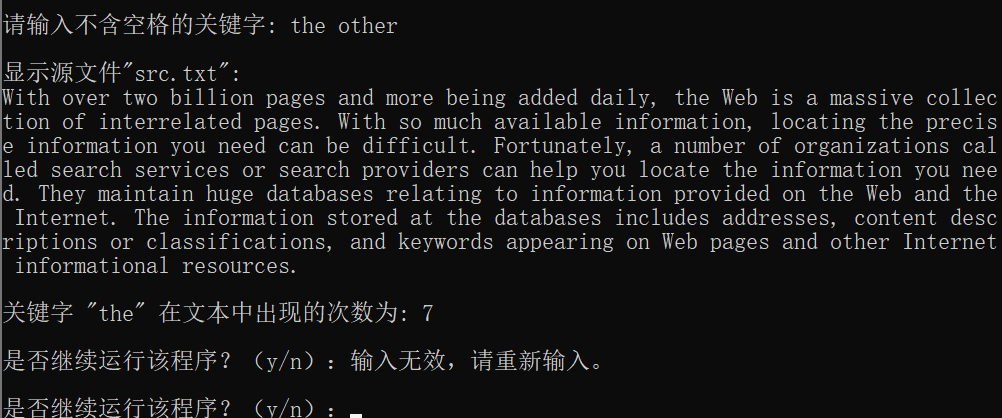
**4.3. 输入关键字。**

本程序设定输入的关键字必须为英文文本，且不含空格，若含有空格，则取第一个空格之前的文本内容。若含有英文字符以外的内容，则程序会提示用户重新输入。

（1）输入含有英文字符以外的字符。



（2）输入含有空格，则取首个空格之前的文本。



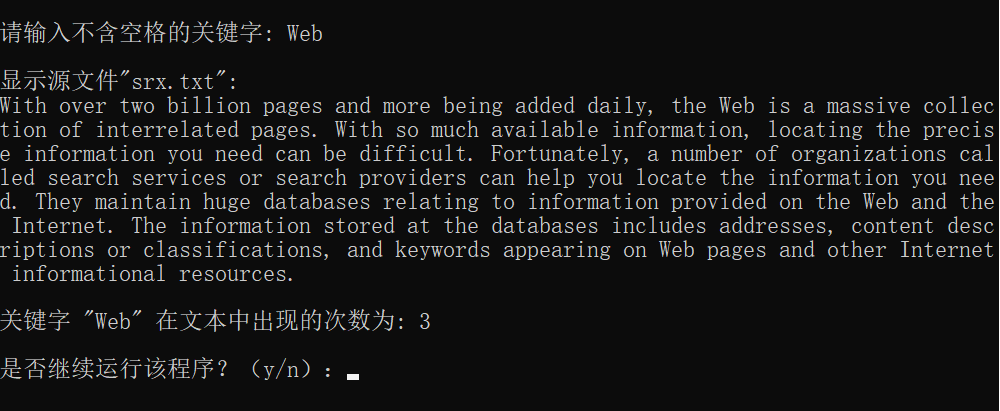
（3）重新输入后输入正确



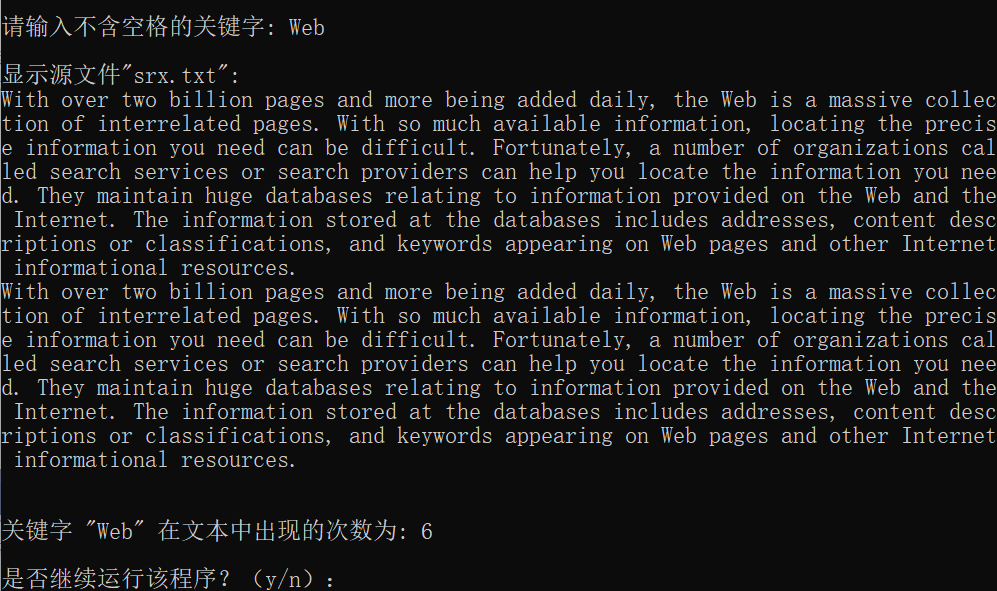
**4.4. 输出源文件内容及关键字在文本中出现的次数**

本程序设定只有与关键字单词相同才算作一次出现，例如关键字为“the”，则文本中出现的“other”中的“the”就不算作一次出现，这通过在KMP快速查找算法中增加限制条件得以实现。同时，多段文本中的关键字的出现次数也可以实现。

（1）单段文本的输出结果。



（2）多段文本的输出结果。



**4.5. 输入以确定是否继续运行程序**

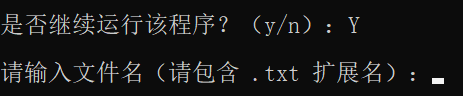
（1）输入内容非法



（2）输入个数非法



（3）输入正确（大小写均可）即可执行相应的操作



**4.6. 程序的测试结果总览（此处仅展示输入正确有效的情况）**

