项目文档

谈瑞

项目五：单词检索统计系统

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 专业班级：软件工程4班  电话：18936361545 | 学号：1452775  电子邮件：tanrui106@163.com | 课目：数据结构课程设计  个人网站：http://guitoubing.top |
|  |  |  |

目录

[项目简介 3](#_Toc496962493)

[项目概要 3](#_Toc496962494)

[项目功能及要求 3](#_Toc496962495)

[项目结构 4](#_Toc496962496)

[项目类的实现 5](#_Toc496962497)

[System类 5](#_Toc496962498)

[File类 6](#_Toc496962499)

[Word类 7](#_Toc496962500)

[主要代码分析 8](#_Toc496962501)

[System.cpp 8](#_Toc496962502)

[setOperator函数 8](#_Toc496962503)

[操作1：创建文件(setInputFile函数 && inputToFile函数) 8](#_Toc496962504)

[操作2：countWords函数 9](#_Toc496962505)

[操作3：检索与计数(locateOrCountWordsInFile函数 && locateWordsInFile函数 && locateWordsInFile函数 && countWordsInFile函数) 10](#_Toc496962506)

[项目的一些拓展知识 13](#_Toc496962507)

[setw函数 13](#_Toc496962508)

[getline函数 13](#_Toc496962509)

[istringstream字符串输入流 13](#_Toc496962510)

[ifstream.clear及ifstream.seekg 14](#_Toc496962511)

[运行测试 15](#_Toc496962512)

[几点不足 17](#_Toc496962513)

[单例模式未能成功使用 17](#_Toc496962514)

[代码冗余比较突出 17](#_Toc496962515)

# 项目简介

## 项目概要

给定一个文本文件，要求统计给定单词在文本中出现的总次数，并检索输出某个单词出现在文本中的行号、在该行中出现的次数以及位置。

## 项目功能及要求

本项目的设计要求可以分为三个部分实现：其一，建立一个文本文件，文件名由用户用键盘输入；其二，给定单词计数，输入一个不含空格的单词，统计输出该单词在文本中的出现次数；其三，检索给定单词，输入一个单词，检索并输出该单词所在的行号、该行中出现的次数以及在该行中的相应位置。

# 项目结构

本项目总体概略结构是在System使用带头结点的单链表来连接File实例，对于系统的操作例如选择操作类型、建立文件、取消操作等都放在System类中，而基于文件的操作例如对文件中单词计数、定位、判断单词是否存在等都放在File类中，而Word类主要存储一个单词及其在文件中的数量。同时，System类、File类、Word类为友元关系，以达到操作数据的目的。（如下图所示）

# 项目类的实现

## System类

private成员

public成员

|  |  |
| --- | --- |
| 类成员 | 作用 |
| friend class File; | 声明System类为File类的友元函数 |
| System() | System类默认构造函数 |
| ~System() | System类默认析构函数 |
| File\* fileExist(string \_file\_name); | 判断文件名为\_file\_name的文件是否存在于System的文件链中,若存在则返回该结点的File指针，否则返回NULL |
| bool systemEmpty(); | 判断System文件链是否为空（由于System类在构建时创建了头结点，因此文件链为空说明文件链只有头结点文件），若是返回true，否则返回false |
| void setOperator(); | 接受用户输入的操作数字，并返回相应操作 |
| void setInputFile(); | 操作1，接受用户输入的文件名建立文件结点，若结点已存在，则重新输入 |
| void inputToFile(string \_file\_name); | 寻找setInputFile函数中的文件所在结点，并首次向其输入数据，并不断与用户交互直到不需要继续输入 |
| void countWords(); | 操作2，接受用户输入的文件名，找到文件并转到File中进行计数操作（具体实现在File类的countWords函数中） |
| void locateOrCountWordsInFile(); | 操作3，继续接受用户输入的指令，a为计数调用locateWordsInFile函数，b为定位调用countWordsInFile函数 |
| void countWordsInFile(); | 操作3-a，接受用户输入的文件名，找到文件并转入File中对用户确定的单词进行计数（具体实现在File类的countWordsInFile函数中） |
| void locateWordsInFile(); | 操作3-b，接受用户输入的文件名，找到文件并转人File中对用户确定的单词进行定位及统计（具体实现在File类的locateWordsInFile函数中） |
| void printSystem(); | 在命令行中输出当前文件链（调试时使用） |
| File\* file\_list\_head; | 声明文件链的头结点 |
| File\* present\_file; | 声明文件链的当前结点 |
|  |  |

## File类

private成员

public成员

|  |  |
| --- | --- |
| 类成员 | 作用 |
| friend class System; | 声明System类为File类的友元函数 |
| File(); | File类默认构造函数 |
| File(string \_file\_name); | File类重载构造函数，通过参数\_file\_name初始化private中的文件输入和输出流 |
| ~File(); | File类默认析构函数 |
| Word\* wordExist(string \_word); | 判断文件中是否有单词\_word存在，若有返回该词的Word实例，否则返回NULL |
| void addToWordVec(string \_word); | 将单词\_word的Word实例添加到word\_vec中，若该实例存在，只需将其计数添加1即可，否则创建实例 |
| void countWords(); | 操作1，被System-countWords函数调用，将文件内所有单词添加到word\_vec中，并对所有单词总数量进行计数 |
| void countWordsInFile(); | 操作3-a，被System-countWordsInFile函数调用，在文件中检索用户确定的单词，并对其计数 |
| void locateWordsInFile(); | 操作3-b，被System-locateWordsInFile函数调用，在文件中检索用户确定的单词，返回单词的行号、数量、位置等信息 |
| void locateWordsInLine(string,  string, vector<int>&,  int&, vector<int>&); | 由File-locateWordsInFile函数调用，将文件拆分成行，检索每行中所查单词的行号、数量、位置等信息 |
| string file\_name; | 声明文件名 |
| ofstream out\_file; | 声明输出文件流 |
| ifstream in\_file; | 声明输入文件流 |
| File\* next\_file; | 声明下一个File结点 |
| vector<Word> word\_vec; | 声明存储文件中所需word的vector（容器） |
|  |  |

## Word类

private成员

public成员

public成员

|  |  |
| --- | --- |
| 类成员 | 作用 |
| friend class System; | 声明System类为Word类的友元函数 |
| friend class File; | 声明File类为Word类的友元函数 |
| Word(); | Word类的默认构造函数 |
| Word(std::string \_word, int \_count); | Word类的重载构造函数，通过参数\_word及\_count初始化Word实例的word值及count值 |
| ~Word(); | Word类的默认析构函数 |
| std::string word; | 声明Word实例的字符串数据 |
| int count; | 声明word出现的次数 |
|  |  |

# 主要代码分析

## System.cpp

### setOperator函数

/\*系统主界面，导航用户进行操作，通过switch操作符选择操作，对于错误操作要求用户重新输入\*/

void System::setOperator() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*文本文件单词的检索与计数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■" << endl

<< "■ 【1】建立文本文档 ■" << endl

<< "■ 【2】文本单词汇总 ■" << endl

<< "■ 【3】单词定位 ■" << endl

<< "■ 【4】退出 ■" << endl

<< "■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■" << endl;

int operate;

cout << "请选择操作（1~4）:";

cin >> operate;

switch (operate) {

case 1: {

setInputFile();

setOperator();

break;

}

case 2: {

countWords();

setOperator();

break;

}

case 3: {

locateOrCountWordsInFile();

setOperator();

break;

}

case 4: {

break;

}

default:

// printSystem();

cout << "操作输入有误，请重新选择操作." << endl;

setOperator();

break;

}

}

### 操作1：创建文件(setInputFile函数 && inputToFile函数)

/\*通过用户输入文件名来创建文件，要注意当创建若不加后缀也是可以进行打开、写入等操作的，但是建议创建的时候使用".txt"后缀。\*/

void System::setInputFile() {

cout << "请输入要建立的文件名：";

string \_file\_name;

cin >> \_file\_name;

while (fileExist(\_file\_name) != NULL) {

cout << "文件：" << \_file\_name << " 已存在！请重新输入文件名:";

cin >> \_file\_name;

}

File\* temp = new File(\_file\_name);

present\_file = file\_list\_head;

while (present\_file->next\_file != NULL){

present\_file = present\_file->next\_file;

}

present\_file->next\_file = temp;

temp->next\_file = NULL;

// printSystem();//临时查看System

inputToFile(\_file\_name);

//创建File实例，并链入System文件链表中，同时要求用户初始化文件（即输入文本）

}

/\*用于初始化文件，从标准输入流中读取字符串并加到文件流中，同时询问用户是否继续输入，若结束输入则进行下次主界面操作。\*/

void System::inputToFile(string \_file\_name) {

/\*用于初始化文件，从标准输入流中读取字符串并加到文件流中，同时询问用户是否继续输入，若结束输入则进行下次主界面操作。\*/

File\* temp = fileExist(\_file\_name);

char go\_on;

string input\_line;

cout << "请输入一行文本：";

getline(cin, input\_line);

getline(cin, input\_line);

//第一行getline用于清空输入流缓存的回车符，第二行才是用户输入的真正的文本

temp->out\_file << input\_line << endl;

cout << "结束输入嘛？y or n :";

cin >> go\_on;

while (go\_on != 'y' && go\_on != 'n') {

cout << "输入有误，结束输入嘛？y or n :";

cin >> go\_on;

}

while (go\_on == 'n') {

cout << "请输入一行文本：";

getline(cin, input\_line);

getline(cin, input\_line);

temp->out\_file << input\_line << endl;

cout << "结束输入嘛？y or n :";

cin >> go\_on;

}

if (go\_on == 'y') {

cout << "建立文件结束" << endl;

return;

}

}

### 操作2：countWords函数

/\*该函数有两个定义，一个在System类中，一个在File类中，通过System::countWords获取文件名获取实例而后调用该文件实例的File::countWords函数在文件中统计所有单词以及其个数。\*/

void System::countWords() {

string \_file\_name;

cout << "请输入文件名：";

cin >> \_file\_name;

while (!fileExist(\_file\_name)) {

cout << "该文件不存在，请重新输入文件名：";

cin >> \_file\_name;

}

present\_file = fileExist(\_file\_name);

present\_file->countWords();

}

void File::countWords() {

vector<Word>(word\_vec).swap(word\_vec);//清空之前留存的word

string temp;

while (in\_file >> temp) {

addToWordVec(temp);

}

cout << "》》》》》》》》》》》单词《《《《《》》》》》个数《《《《\n";

vector<Word>::iterator i\_vec = word\_vec.begin();

int word\_amount = 0;

for (; i\_vec != word\_vec.end(); i\_vec++) {

word\_amount += i\_vec->count;

cout << "\t" << right << setw(16) << i\_vec->word << "\t\t\t" << right << i\_vec->count << endl;

}

cout << "》》》》》》》》》》》》》》》" << file\_name << "的单词总数为" << word\_amount << "个！" << endl;

}

### 操作3：检索与计数(locateOrCountWordsInFile函数 && locateWordsInFile函数 && locateWordsInFile函数 && countWordsInFile函数)

/\*由用户选择操作3的子操作(a或b)，并进行相应操作\*/

void System::locateOrCountWordsInFile() {

char choice;

cout << "###############################################################" << endl;

cout << "####################文本文件单词的检索与计数###################" << endl;

cout << "∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷" << endl

<< "∷ 【a】 单词出现次数 ∷" << endl

<< "∷ ∷" << endl

<< "∷ 【b】 单词出现位置 ∷" << endl

<< "∷ ∷" << endl

<< "∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷∷" << endl;

cout << "请选择a或b：";

cin >> choice;

while (choice != 'a' && choice != 'b'){

cout << "输入错误，请重新输入（a或b）：";

cin >> choice;

}

if (choice == 'a'){

countWordsInFile();

}

else {

locateWordsInFile();

}

}

/\*countWordsInFile函数有两个定义，通过System::countWordsInFile获取文件名获取实例而后调用该文件实例的File::countWordsInFile函数，在文件中统计指定单词以及其个数。\*/

void System::countWordsInFile() {

string \_file\_name;

cout << "请输入文件名：";

cin >> \_file\_name;

while (!fileExist(\_file\_name)) {

cout << "该文件不存在，请重新输入文件名：";

cin >> \_file\_name;

}

present\_file = fileExist(\_file\_name);

present\_file->countWordsInFile();

}

void File::countWordsInFile() {

in\_file.clear();

in\_file.seekg(0);//清空输入流缓存，并将读指针返回到文件头

cout << "请输入要检索的单词：";

string input\_word, temp;

cin >> input\_word;

int amount = 0;

while (in\_file >> temp){

if (temp == input\_word) {

amount++;

}

}

cout << "单词" << input\_word << "在文本文件" << file\_name << "中共出现了" << amount << "次" << endl;

}

/\* locateWordsInFile函数有两个定义，通过System::locateWordsInFile获取文件名获取实例而后调用该文件实例的File::locateWordsInFile函数，再循环调用File::locateWordsInLine函数在文件中统计指定单词以及其个数。\*/

void System::locateWordsInFile() {

string \_file\_name;

cout << "请输入文件名：";

cin >> \_file\_name;

while (!fileExist(\_file\_name)) {

cout << "该文件不存在，请重新输入文件名：";

cin >> \_file\_name;

}

present\_file = fileExist(\_file\_name);

present\_file->locateWordsInFile();

}

void File::locateWordsInFile() {

in\_file.clear();

in\_file.seekg(0);//清空输入流缓存，并将读指针返回到文件头

cout << "请输入要检索的单词：";

string input\_word, out\_line;

cin >> input\_word;

vector<int> line\_nums, word\_index, amounts;

int line\_num = 0, amount = 0, index\_size = 0;

while (getline(in\_file, out\_line)) {

line\_num++;

amount = 0;

locateWordsInLine(out\_line, input\_word, word\_index, amount, amounts);

if (word\_index.size() != index\_size) {

line\_nums.push\_back(line\_num);

}

index\_size = word\_index.size();

}

if (line\_nums.size() == 0) {

cout << "文件" << file\_name << "里面没有单词" << input\_word << endl;

}

else {

vector<int>::iterator i\_line\_nums = line\_nums.begin(),

i\_word\_index = word\_index.begin(),

i\_amounts = amounts.begin();

for (; i\_line\_nums != line\_nums.end(); i\_line\_nums++, i\_amounts++) {

cout << "行号：" << \*i\_line\_nums << ",次数：" << \*i\_amounts << "起始位置分别为：";

for (int i = 0; i < \*i\_amounts; i++) {

cout << "第" << \*i\_word\_index+1 << "个字符 ";

i\_word\_index++;

}

cout << endl;

}

}

}

void File::locateWordsInLine(string out\_line, string input\_word, vector<int>& word\_index, int &amount, vector<int>& amounts) {

istringstream out\_line\_stream(out\_line);//创建字符串输入流

string temp;

int index = 0;

while (out\_line\_stream >> temp) {

if (temp == input\_word) {

word\_index.push\_back(index);

amount++;

}

index += temp.size() + 1;

}

if (amount != 0) {

amounts.push\_back(amount);

}

}

# 项目的一些拓展知识

在写项目时，可以说是边写边学，很多数据、结构、操作都是现查的，写完项目后也学到了很多以前不懂的技术。现在此记录以备。

## setw函数

在项目要求文档中有关于格式化输出的要求（虽然没有硬性要求，但是为了美观还是小小的研究了一下），在对操作2输出时要求所有单词右对齐（如图所示），于是Google了一下，发现了setw函数，原理是通过传入的n参数，将字符串不足n的部分填充空格（由于是右对齐所以是在左边填充），使用时需要包含头文件iomanip。当然用std::right控制符也是能达到同样的效果。

## getline函数

操作3-b中，需要对文件中的确定字符串进行定位、计数，且要分行统计，很容易想到了getline函数，但是在使用时发现程序把getline函数自动跳过了，经过查找发现是因为命令行的输入流缓冲没有被清空，导致之前输入的换行符被getline读取，从而导致直接跳过，这里的解决办法就是getline两次。

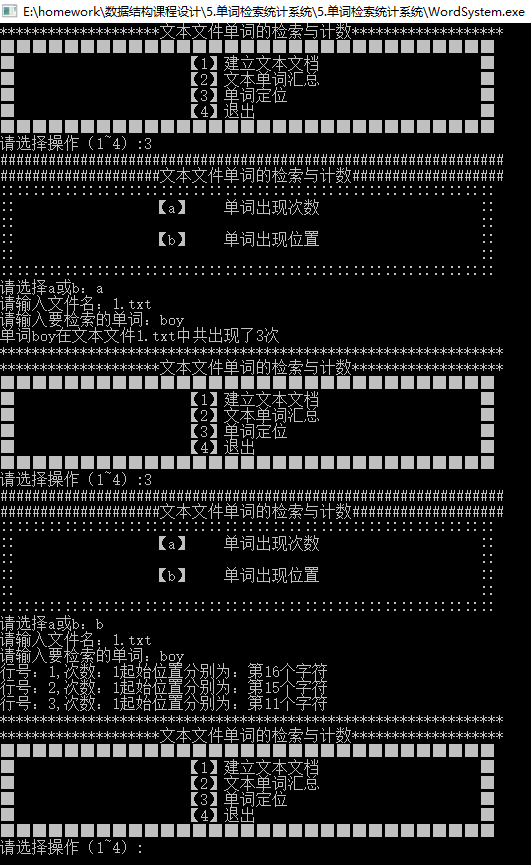
## istringstream字符串输入流

操作3-b中，需要逐行统计单词信息，而通过getline读取到的input\_line行字符串时，怎么把它拆分成一个个单词又是个问题。这里用到了字符串输入流istringstream，定义了字符串输入流，就可以像标准输入流一样读取单个字符串。

## ifstream.clear及ifstream.seekg

项目完成后在调试阶段，发现对同一个文件结点操作读取两次后会发现统计的数据全为0了，查看了一下创建的资源文件也没问题有内容的，后来查了一下知道是因为文件流在读取文件时会有一个与文件流相关联的读指针，每次执行读取时指针会自动后移（依据读取的数据结构迭代），相应的操作是seekg函数。而我在进行第一次操作时，读指针已经到了文件尾，这时没有初始化，导致第二次操作读不到数据直接结束（EOF），因此使用了clear函数清楚文件流缓冲区，再使用seekg函数设定读指针到文件头，即seekg（0）。

# 运行测试



# 几点不足

## 单例模式未能成功使用

项目筹划时期，构想的是将System类设计成单例模式，使用sitatic指针声明唯一一个指向System类实例的指针，并定义一个初始化该指针的函数（判断类是否已实例化，若是则返回该指针，否则新建指针），同时将构造函数、析构函数、重载赋值操作函数及拷贝函数均声明为私有变量防止指针重复定义，但在实现过程中由于知识掌握不足，在各种地方出现了指针无法操作，以及调用错误等，最终因力所不能及而放弃该模式选择了普通链表类。

## 代码冗余比较突出

代码编写时期，出现了很多冗余代码，例如在判断用户输入的文件名是否已存在时，在不同的函数中写了重复的代码而没有统一汇总打包成一个函数。

## 未能在程序运行结束后删除文件

在析构函数使用cstdio库中的remove函数删除文件，但是却总是找不到创建的文件，因而删除不了文件。