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 学号：1452775 课目：数据结构课程设计
电话：18936361545 电子邮件：tanrui106@163.com 个人网站：http://guitoubing.top

目录

项目简介

项目概要

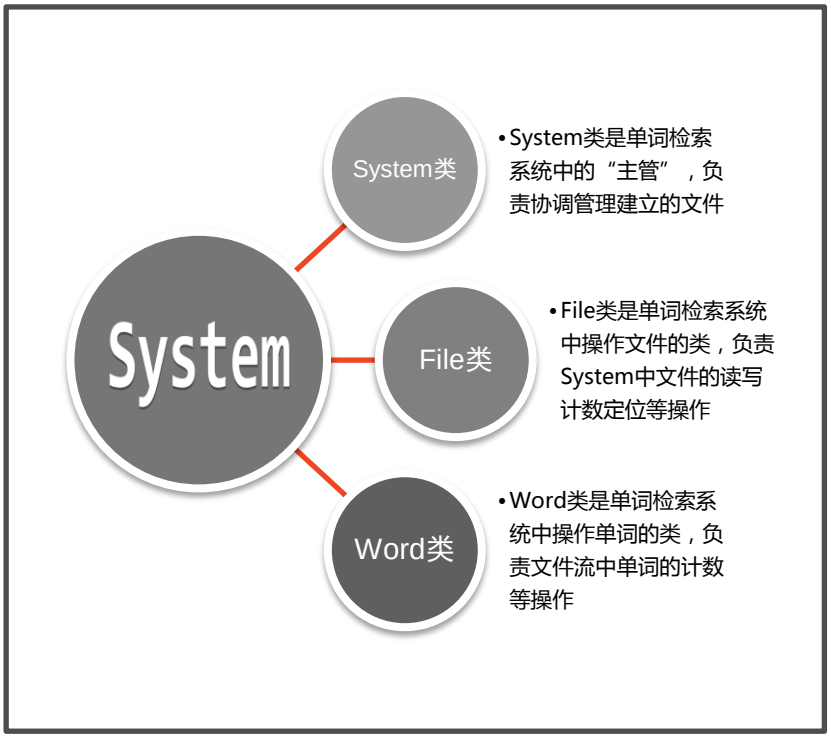
给定一个文本文件，要求统计给定单词在文本中出现的总次数，并检索输出某个单词出现在文本中的行号、在该行中出现的次数以及位置。

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

	类成员	作用
public 成员	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）
private 成员	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

	类成员	作用
public 成员	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
private 成员	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

	类成员	作用
public 成员	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
private 成员	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();
	}
```

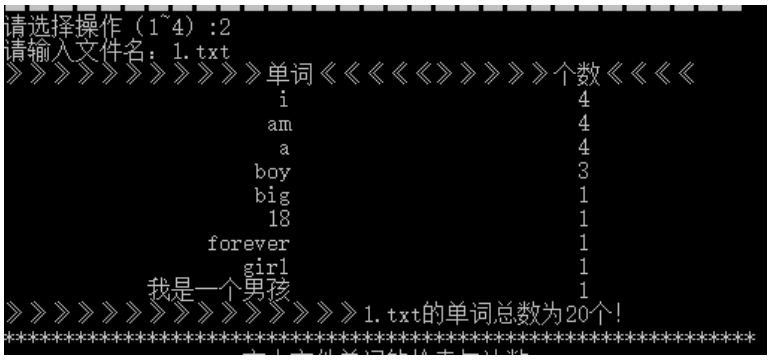
```
        if (line_nums.size() == 0) {
                cout << "文件" << file_name << "里面没有单词" << input_word << endl;
        }
        else {
                vector<int>::iterator i_line_nums = line_nums.begin(),
                        i_word_index = word_index.begin(),
                        i_amounts = 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。

```
getline(cin, input_line);
getline(cin, input_line);
```

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

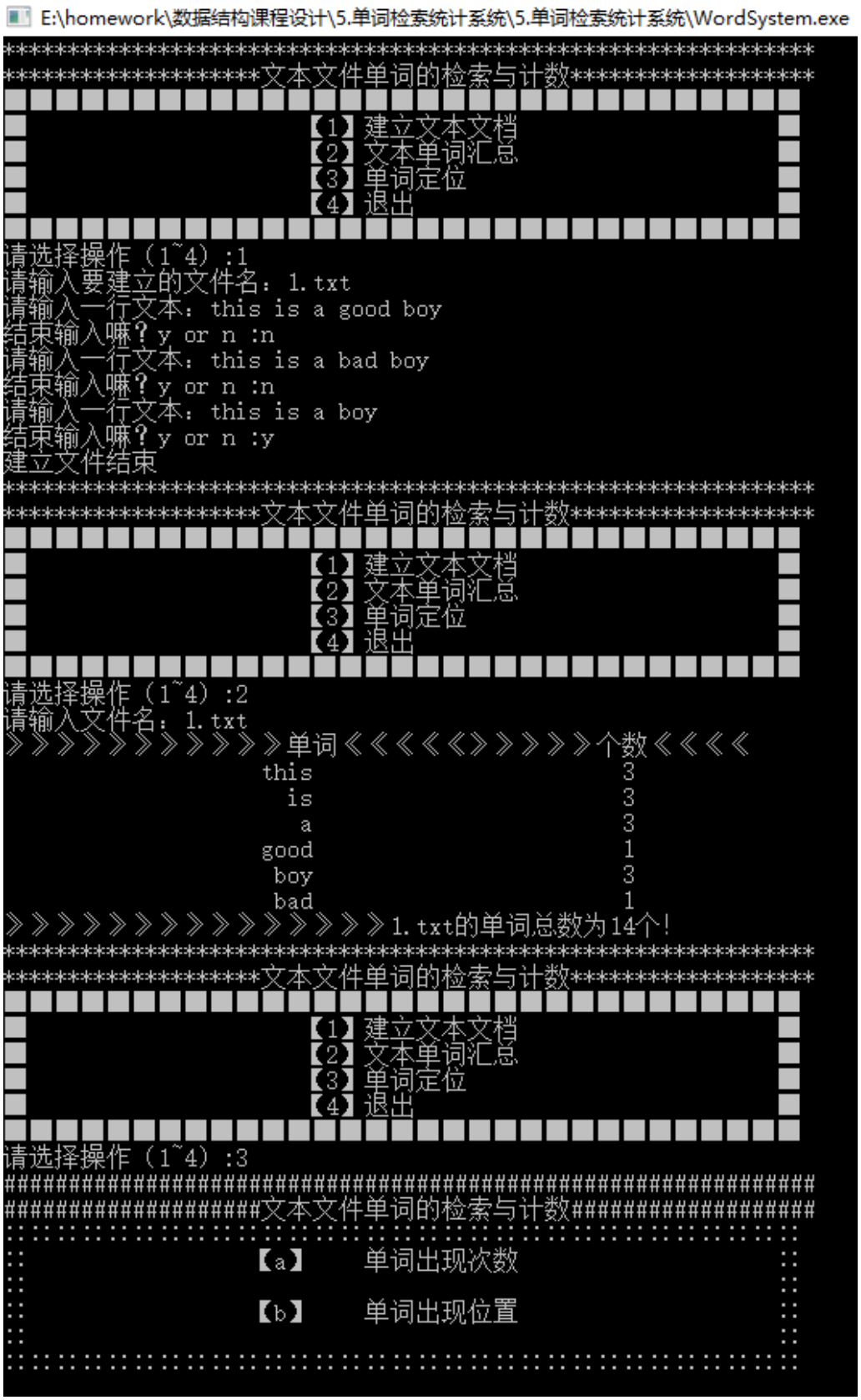
项目完成后在调试阶段，

```
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
```

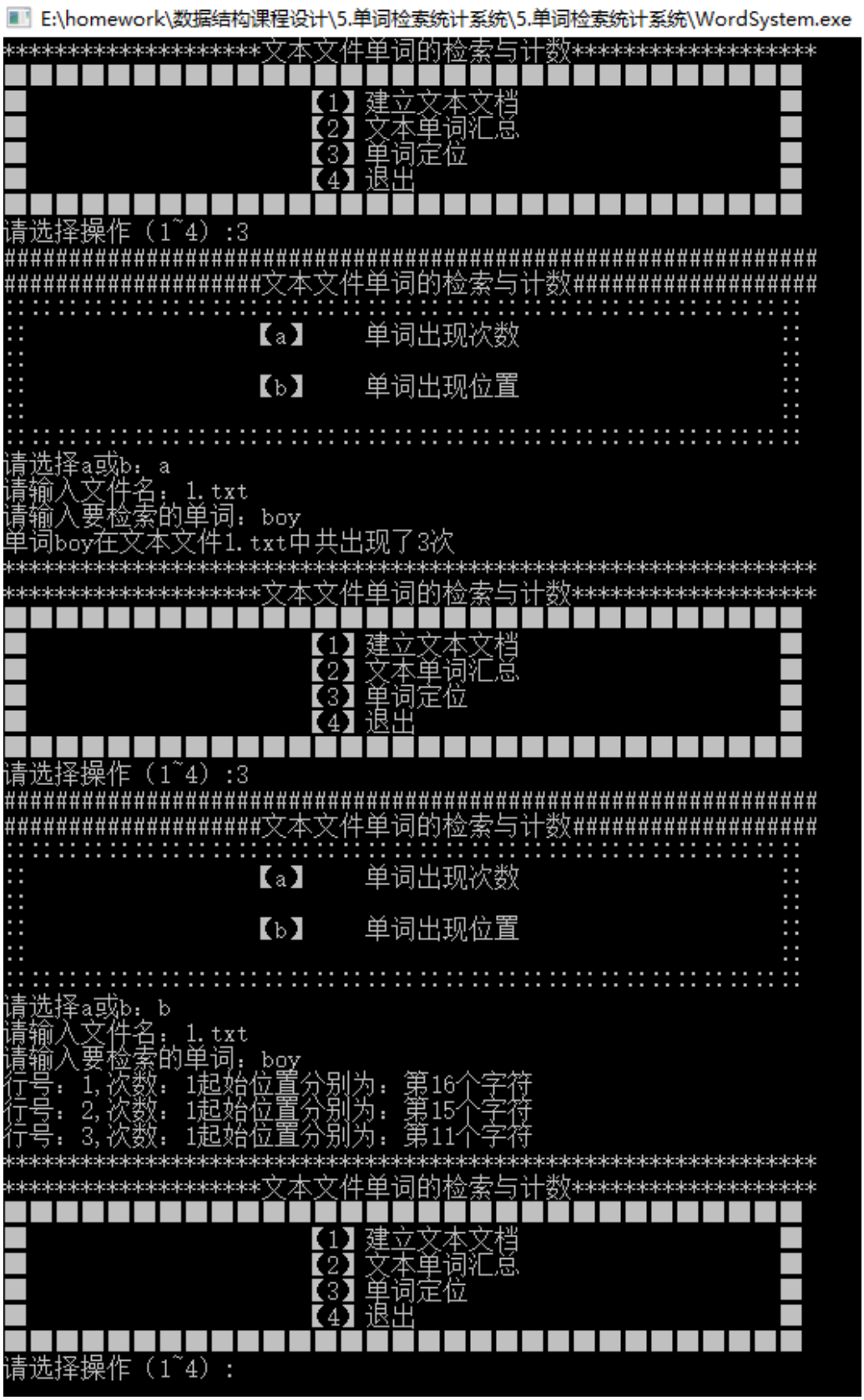
发现对同一个文件结点

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


E:\homework\数据结构课程设计\5.单词检索统计系统\5.单词检索统计系统\WordSystem.exe
**********文本文件单词的检索与计数**********
【1】建立文本文档
【2】文本单词汇总
【3】单词定位
【4】退出
请选择操作（1~4）:1
请输入要建立的文件名：1.txt
请输入一行文本：this is a good boy
结束输入嘛？y or n :n
请输入一行文本：this is a bad boy
结束输入嘛？y or n :n
请输入一行文本：this is a boy
结束输入嘛？y or n :y
建立文件结束
**********文本文件单词的检索与计数**********
【1】建立文本文档
【2】文本单词汇总
【3】单词定位
【4】退出
请选择操作（1~4）:2
请输入文件名：1.txt
》》》》》》》》》》单词《《《《《》》》》》个数《《《《《
this 3
is 3
a 3
good 1
boy 3
bad 1
》》》》》》》》》》》》》》》》》》1.txt的单词总数为14个!
**********文本文件单词的检索与计数**********
【1】建立文本文档
【2】文本单词汇总
【3】单词定位
【4】退出
请选择操作（1~4）:3
##########文本文件单词的检索与计数##########
【a】 单词出现次数
【b】 单词出现位置

运行测试


E:\homework\数据结构课程设计\5.单词检索统计系统\5.单词检索统计系统\WordSystem.exe
**********文本文件单词的检索与计数**********
【1】建立文本文档
【2】文本单词汇总
【3】单词定位
【4】退出
请选择操作（1~4）:3
##########文本文件单词的检索与计数##########
【a】 单词出现次数
【b】 单词出现位置
请选择a或b：a
请输入文件名：1.txt
请输入要检索的单词：boy
单词boy在文本文件1.txt中共出现了3次
**********文本文件单词的检索与计数**********
【1】建立文本文档
【2】文本单词汇总
【3】单词定位
【4】退出
请选择操作（1~4）:3
##########文本文件单词的检索与计数##########
【a】 单词出现次数
【b】 单词出现位置
请选择a或b：b
请输入文件名：1.txt
请输入要检索的单词：boy
行号：1,次数：1起始位置分别为：第16个字符
行号：2,次数：1起始位置分别为：第15个字符
行号：3,次数：1起始位置分别为：第11个字符
**********文本文件单词的检索与计数**********
【1】建立文本文档
【2】文本单词汇总
【3】单词定位
【4】退出
请选择操作（1~4）:

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。