

Assignment_5 单词检索统计系统

一、项目简介：

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。

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。

二、构建设想：

单词检索系统要求实现的功能比较多,所以具体实现起来还是很有很多不同的选择的，第二部分单词计数问题我是用了map来计数，int表示单词出现的次数。第三部分输出单词所在的行号则使用了类来按行读取数据,实现输出单词所在行号以及该行中出现的相应的位置。

三、程序设计：

1. 程序的类与结构组织

程序中使用了Dataset类来实现整个检索系统的组织，几个public的函数也代表着应用所要求的不同的功能，如下会 对于不同函数进行解释,先见Dataset类的声明:

```

class Dataset{

private:

    int                _command;           //命令
    int                _currentLine;       //目前所在行
    int                _wordCount;         //单词计数
    int                _nonWordCount;      //非单词计数
    string              _currentWords;     //目前的单词串
    map<string,int>     _wordMap;          //单词地图

public:

    Dataset():_command(0){ }              //构造函数
    void createFile();                     //创建文本文件
    void printMenu();                      //打印菜单
    void textCount();                      //统计单词与非单词个数
    void wordLocate();                     //统计单词出现的次数与位置
    void executeCommand();                 //执行各项命令
    void wordPosition(string fileName);    //返回单词的位置
    void wordCount(string fileName);       //单词数量统计

};

```

2.程序类中函数的实现

(1) 文本创建函数 createFile() :

文本创建函数的主要功能就是创建一个存储数据的txt文本然后将输入的信息存储在文本中,如下是文本创建函数的具体实现方式:

```

void Dataset::createFile()
{
    string fileName,line;
    cout << "输入要建立的文件名" << endl;
    cin >> fileName;
    ofstream file;
    file.open(fileName);

    char command = 'n';
    while (command == 'n')
    {
        cout << "请输入一行文本:";
        getline(cin,line);
        getline(cin,line);
        file << line << endl;
        cout << " 输入结束吗? y or n :";
        cin >>command;
    }
    cout << "建立文件结束!" << endl;

    file.close();
}

```

(2) 单词计数函数 wordCount() :

单词计数的主要思路是对将文本中的内容遍历一遍,然后将非单词 与单词进行分别计数, 如下是函数的具体实现方式:

```

void Dataset::wordCount(string fileName)
{
    ifstream file;
    file.open(fileName);
    while (getline(file, _currentWords))
    {
        _currentLine++;
        //建立一个stringstream的对象 并将刚才读取的line (字符串) 的内容放在ss里面
        stringstream ss(_currentWords);
        string word;
        while (ss >> word)
        {
            if (word[0] >= 'A' && word[0] <= 'Z' || word[0] >= 'a' &&
word[0] <= 'z')
                _wordCount++;
            else
                _nonWordCount++;

            if (_wordMap[word] == 0)
                _wordMap[word] = 1;
            else
                _wordMap[word]++;
        }
    }
    file.close();
}

```

(3) 单词位置统计函数 wordPosition() :

wordPosition与wordCount使用的并不是一个逻辑,wordPosition是对文本中的每一行进行读取,并且对被搜索单词每行中的位置进行记录,最后进行输出, 如下是wordPosition函数的具体实现逻辑:

```

void Dataset::wordPosition(string fileName)
{
    string word;
    cout << "要检索的单词:";
    cin >> word;
    _currentLine = 0;
    int position = 0;
    ifstream file;
    file.open(fileName);

    vector<int> _wordPosition;
    while (getline(file, _currentWords))
    {
        position = 1;
        _wordPosition.clear();
        _currentLine++;
        _wordCount = 0;
        stringstream ss(_currentWords);
        //建立一个stringstream的对象 并将刚才读取的line（字符串）的内容放在ss里面
        string tempWord;
        while (ss >> tempWord)
        {
            if(word == tempWord)
            {
                _wordCount++;
                _wordPosition.push_back(position);
            }
            position += tempWord.size() + 1;
        }
        if(_wordCount != 0)
        {
            cout << "行号:" << _currentLine << ",出现次数为"<< _wordCount << ",
            起始位置分别为：第";
            for(auto iter = _wordPosition.begin(); iter !=
            _wordPosition.end(); iter++)
                cout << " " << *iter;
            cout << "个字符" << endl;
        }
    }
    file.close();
}

```

(4) 文本文件字串的定位统计及定位 wordLocate():

这个函数的功能主要是实现检索系统要求的字串的定位统计及定位的功能,其会调用wordPosition与wordCount函数来实现单词出现的位置与单词出现次数的功能.

```
void Dataset::wordLocate()
{
    cout << "===== " << endl;
    cout << "||          文本文件字串的定位统计及定位          ||" << endl;
    cout << "||=====||" << endl;
    cout << "||\t\t" << "a.   单词出现的次数   " <<" \t\t||"<<endl;
    cout << "||\t\t\t\t\t\t\t\t\t\t\t\t||"<<endl;
    cout << "||\t\t\t\t\t\t\t\t\t\t\t\t||"<<endl;
    cout << "||\t\t" << "b.   单词出现的位置" <<" \t\t\t||"<<endl;
    cout << "||\t\t\t\t\t\t\t\t\t\t\t\t||"<<endl;
    cout << "===== " << endl;

    string command;
    cout << "请输入a或b:";
    cin >> command;
    string word;
    if( command == "a")
    {
        string fileName;
        cout << "请输入文本文件名:";
        cin >> fileName;

        cout << "请输入要统计计数的单词:" ;
        cin >> word;
        _wordMap.clear();
        wordCount(fileName);
        cout << "单词" << word << "文本文件" << fileName << "中共出现" <<
        _wordMap[word] << "次" << endl;
    }
    else if(command == "b")
    {
        string fileName;
        cout << "请输入文本文件名:";
        cin >> fileName;
        wordPosition(fileName);
    }
    else
        cout << "输入错误，已退出！" << endl;
}
```

(5) 执行指令函数 executeCommand():

此函数是用来调用其他成员函数，其中包含了单词检索系统的主循环。

```

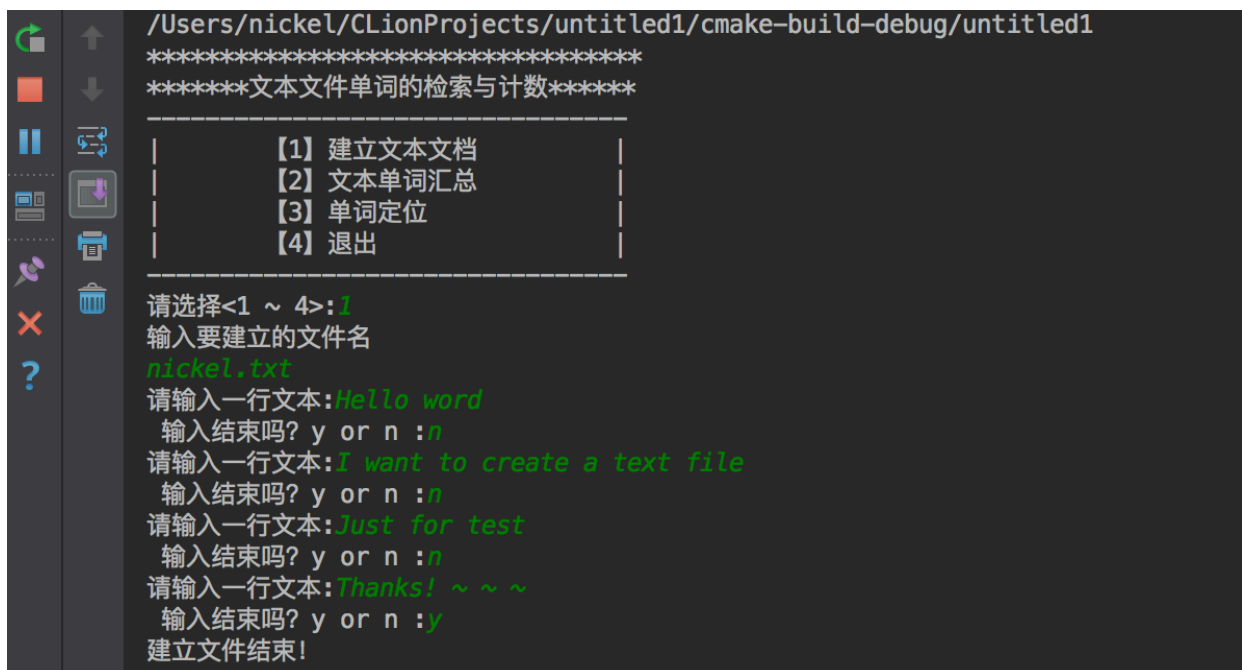
void Dataset::executeCommand()
{
    while(_command != 4)
    {
        printMenu();
        cout << "请选择<1 ~ 4>:";
        cin >> _command;
        switch (_command)
        {
            case 1:
                createFile();
                break;
            case 2 :
                textCount();
                break;
            case 3:
                wordLocate();
                break;
            default:
                cout << "输出错误,请重新输入" << endl;
                break;
        }
    }
}
}

```

四、程序功能性测试：

1. 文本文档创建功能测试：

输入1选择建立文本文档，然后按行输入文本当输入y的时候即可结束建立文件，此时文件建立成功！



```

/Users/nickel/CLionProjects/untitled1/cmake-build-debug/untitled1
*****
*****文本文档单词的检索与计数*****
-----
|          [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<1 ~ 4>:1
输入要建立的文件名
nickel.txt
请输入一行文本:Hello word
  输入结束吗? y or n :n
请输入一行文本:I want to create a text file
  输入结束吗? y or n :n
请输入一行文本:Just for test
  输入结束吗? y or n :n
请输入一行文本:Thanks! ~ ~ ~
  输入结束吗? y or n :y
建立文件结束!

```

2.文本单词汇总功能测试:

选择2并且随后输入文件名,输入文件名后出现单词的个数以及单词总数与非单词总数的汇总:

[illegible]

3.统计单词出现的次数功能测试1:

统计某个单词在文档中出现的次数,测试1为当单词在文档中存在时:


```
输入错误，已退出！
*****
*****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<1 ~ 4>:3
=====
||          文本文件字串的统计及定位          ||
||=====||
||          a.    单词出现的次数              ||
||              ||
||          b.    单词出现的位置              ||
||              ||
||=====||
请输入a或b:a
请输入文本文件名:nickel.txt
请输入要统计计数的单词:I
单词I文本文件nickel.txt中共出现1次
*****
```

4.统计单词出现的次数功能测试2:

统计某个单词在文档中出现的次数,测试2为当单词在文档中不存在时:

```
/Users/nickel/CLionProjects/untitled1/cmake-b
*****
*****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<1 ~ 4>:3
=====
||          文本文件字串的统计及定位          ||
||=====||
||          a.    单词出现的次数              ||
||              ||
||              ||
||          b.    单词出现的位置              ||
||              ||
||=====||
请输入a或b:a
请输入文本文件名:nickel.txt
请输入要统计计数的单词:hello
单词hello文本文件nickel.txt中共出现0次
~~~~~
```

5.统计单词出现的次数功能测试：

选择操作3与操作b 输入文件名之后选择要统计的单词,会自动显现出单词出现的行号以及出现的次数与起始位置：

```
*****
*****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<1 ~ 4>:3
=====
||          文本文件字串的统计及定位          ||
||=====||
||          a.    单词出现的次数              ||
||          ||
||          b.    单词出现的位置              ||
||          ||
||=====||
请输入a或b:b
请输入文本文件名:nickel.txt
要检索的单词:I
行号:2,出现次数为1, 起始位置分别为: 第 1个字符
*****
```

五、程序健壮性测试：

1.考试系统的初始化功能健壮性测试：

程序将会保证初始化时的数字大于等于1， 否则就提示错误：



2.考试系统插入功能健壮性测试：

程序将会保证插入时候的数字大于等于0， 否则就提示错误：



3.考试系统的删除功能健壮性测试：

程序如果找不到被删除的同学就提示错误：



4.考试系统的查找功能健壮性测试：

程序如果找不到被删除的同学就提示错误：

