单词检索统计系统

1.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。

2.项目功能要求

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。

3.核心代码及其功能

- **新建文本并输入文本信息**
  这里我们使用`ofstream`类来新建一个文本文件，然后每次先将要输入的一行字符放入string中，然后再输入到文本中，最后关闭文本。

```
void Create(){
    string fileName;
    cout << "请输入要新建的文件名：";
    cin >> fileName;
    ofstream fout(fileName);            //创建待写入数据文件
    bool goOn = 1;
    while (goOn) {
        cout<<"请输入一行文本:";
        getchar();
        string line;
        char yOrn;
        getline(cin,line);
        fout << line+'\n';
        cout << "结束输入吗? y or n:";
        cin >> yOrn;
        if (yOrn == 'y') {
            break;
        }
    }
    fout.close();
}
```

- **切分单词**
  由于在读入文本的时候每次是读入一行，所以需要根据‘ ’也就是space来切分一行string，python里提供了split函数，但是由于c++没有提供这个函数，所以我手写了一个切分的函数。返回一个保存切分出来的单词的vector，以参数c为切分标准，切分string s。

```
void splitString(const string& s, vector<string>& v,char c){
    size_type pos1, pos2;
    pos2 = s.find(c);                   //找到c出现的位置
    pos1 = 0;
    while(string::npos != pos2)         //如果没有找到string的末尾则继续寻找
    {
        v.push_back(s.substr(pos1, pos2-pos1));     //将遇到space之前的字符作为一个单词放入vector中
        pos1 = pos2 + 1;                            //从下一个位置开始继续寻找c出现的位置
        pos2 = s.find(c, pos1);
```

```
    }
    if(pos1 != s.length())                          //将最后一段字符作为一个单词放入vector中
        v.push_back(s.substr(pos1));
}
```

- **文本单词汇总**
  每次读入一行单词，然后调用splitString函数将该行按照空格切分成单词，然后对每个单词使用map进行统计，最后遍历整个map输入统计的效果。这里需要对切分后的单词进行检查单词末尾是否带有'.'，如果有则需要将其去除。

```
void Summary(string fileName){
    fileSum.clear();            //将map清空
    int wordCount=0;            //统计所有单词量
    string line;                //每次读入一行句子
    string word;                //存放每行切分的所有单词
    while(getline(openFile,line)){
        vector<string>word;
        splitString(line,word,' ');
        wordCount+=word.size();
        for(auto i:word){           //如果单词的末尾是'.'就将其去除
            if(i[i.size()-1]=='.'){
                i.erase(i.end()-1);
            }
            fileSum[i]++;           //使用map统计总单词量
        }
    }
    cout<<"*******单词计数*********"<<endl;
    cout<<"单词        次数"<<endl;
    for(auto i:fileSum){
        cout<<i.first<<"       "<<i.second<<endl;
    }
    cout<<fileName<<"的单词总数为"<<wordCount<<endl;
}
```

- **对给定单词进行计数**
  这里我们采用ifstream的方式来读入文件，声明一个ifstream变量`openFile`，然后每次从openFile读入一个单词记录在temp的string中，如果该string等于我们之前输入的word就将计数器加一，直到读到文件末尾处。

```
void Count(){
    int countNum=0;
    cout<<"please input the word you want to count:";
    string word;                        //保存输入的单词
    cin>>word;
    string temp;                //保存从文本中每次读入的单词
    while(openFile>>temp){
        if(temp==word){         //每次出现一次就将计数器加1
            countNum++;
        }
    }
    cout<<"该单词出现了"<<countNum<<"次"<<endl;
}
```

- **搜索给定的单词**
  首先将要搜索的单词保存至word变量中，这里我们采用每次用getline读取一行内容，然后再对在该行进行查找单词。lineNum为目前的行数，pos为在该行查找到的位置，使用了string的内置函数find查找单词,并且需要考虑到使用find函数仅查找子串可能找到一个包含该单词的单词，所以需要添加条件找到的单词后面的一个字符必须是' '或者'\n'。

```
void Find(){
    cout<<"please input the word you want to count:";
    string word;                //保存想要搜索的单词
    cin>>word;
    int lineNum=1,pos=0;
    string line;
```

```
    while(getline(openFile,line)){      //每次读入一行
        while((pos=line.find(word,pos))!=string::npos
            &&(line[pos+word.size()]==' '||line[pos+word.size()]=='\n')){
                                                        //在每一行中重复寻找是否存在该单词
                                                        //pos代表当前行寻找到的位置
            cout<<"find word "<<'\''<<word<<'\''<<" at line "<<lineNum<<",column "<<pos<<endl;
            pos+=word.size();
        }
        lineNum++;
        pos=0;                          //将每一行寻找到的位置重置为0
    }
}
```

- **读入失败时重新询问用户输入**
  这里同样考虑了用户输入了非法输入或者未打开文件时重新询问用户的操作：

```
bool checkOption(){              //询问用户是否想重新输入
    cout<<"Do you want to reinput?(y/n)"<<endl;
    char option;
    cin>>option;
    if(option=='y'){
        return 1;
    }
    else {
        return 0;
    }
}
```

4.项目实例

> a.txt内容
> this is a boy and he is a good boy
> you are a student

- **建立文本文档**

```
/Users/kirito/CLionProjects/untitled/cmake-build-debug/untitled
****************************************
********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):1
请输入要新建的文件名：b.txt
请输入一行文本:you are a good boy.
结束输入吗? y or n:n
请输入一行文本:he is a good student.
结束输入吗? y or n:y
```

- **打开文件失败重新询问用户输入**

```
****************************************
[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):2
请输入要打开的文件名：c.txt
can not find/open this file
```

```
Do you want to reinput?(y/n)
y
```

- **文本单词汇总**

```
请输入要打开的文件名：a.txt
*******单词计数*********
单词          次数
a       3
and       1
are       1
boy       2
good       1
he       1
is       2
student       1
this       1
you       1
a.txt的单词总数为14
```

- **单词定位之单词出现次数统计**

```
****************************************
[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
请输入要打开的文件名：a.txt
文本文件单词的定位统计及定位
a.单词出现次数
b.单词出现位置
请输入a或b:a
please input the word you want to count:good
该单词出现了1次
****************************************
[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
请输入要打开的文件名：b.txt
文本文件单词的定位统计及定位
a.单词出现次数
b.单词出现位置
请输入a或b:a
please input the word you want to count:good
该单词出现了2次
```

- **单词定位之单词出现位置**

```
****************************************
[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
请输入要打开的文件名：b.txt
文本文件单词的定位统计及定位
a.单词出现次数
b.单词出现位置
请输入a或b:b
please input the word you want to count:good
find word 'good' at line 1,column 10
```

```
find word 'good' at line 2,column 8
```

- **退出程序**

```
**************************************
[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):4

Process finished with exit code 0
```