

目录

目录

1.项目背景

2.需求分析

3.设计思路

读取文本

功能实现

4.核心代码说明

函数接口说明

类说明

函数功能说明

word类成员函数

构造函数

默认构造函数

带单词名的构造函数

成员函数

比较单词

增加词频

获取单词名

非成员函数

全文单词计数

行信息分析

5.使用说明及函数功能演示

文档说明

打开文档

输入单词

功能选择

1.项目背景

文本文件的处理是计算机的常用功能之一，在日常工作生活中，我们常常要处理各种形式的文档，一个好的文本处理程序能够在很大程度上提高文档操作的效率与正确率，帮助使用者提供工作效率。

2.需求分析

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。

本项目的设计要求分为三个部分实现：

- 建立一个文本文件，文件名由用户用键盘输入
- 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
- 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.设计思路

读取文本

使用循环读取文本内容，每次读入一个char，以eof作为循环结束条件，读取文本时，利用ASCII码对文本进行处理，将读取到的内容先存入char数组

- 如果字符数值在[65,90]（即为A~Z），则直接+32转换为小写字母
- 如果字符数值也不再[97,122]，说明为空格或是其他符号，直接转换为 `\0` 作为断字符

功能实现

在读取文本的同时就可以对文本进行操作，由于需求实现的功能都是对指定单词进行操作，所以不需要保存此前读入的内容

实现全文单词计数：

- 将char数组转换为string类型，并与此前输入的string类型的指定单词对比，如果相等，则计数器+1

检索单词输出行号、次数、位置：

- 计数方法于与全文单词计数相同
- 行号则通过 `\n` 符号来判断
- 行中位置用该行第k个单词的形式表示，而每读取到 `\0` 则 增加一次单词计数，每读取到 `\n` 则输出指定单词在该行的信息

4.核心代码说明

函数接口说明

返回值类型	成员函数名	参数	属性	功能
\	word	()	word成员函数 public	默认构造函数
\	word	(string &r)	word成员函数 public	构造函数
bool	compare	(string &r)	word成员函数 public	与指定单词比较
void	addFrequency	()	word成员函数 public	单词计数+1
void	getName	()	word成员函数 public	获取存储单词名
ostream&	operator<<	(ostream &os, const word &obj)	word友元函数	输出单词信息
void	CountFrequency	(string file_name,string target_word)	非成员函数	全文单词计数
void	AnalyseInRow	(string file_name, string target_word)	非成员函数	分析单词行信息

类说明

使用 `word` 存储单词相关信息

```
1 //Create a class to save words
2 class word {
3 private:
4     int frequency;
5     string name;
6 public:
7     //Default constructor
8     word();
9     //Overload constructor
10    word(string &r);
11    //Comepare name and r
12    bool compare(string &r);
13    //Add this word's frequency by 1
14    void addFrequency();
15    //Output this word's information
16    friend ostream& operator<<(ostream &os, const word &obj);
17    //Return this word's name
18    string getName();
```

```
19 };
```

私有成员变量 `frequency` 用于存储该单词出现的频率， `name` 用于存储该单词名称

公有成员函数功能见注释

函数功能说明

word类成员函数

构造函数

默认构造函数

```
1 word::word() {  
2     frequency = 0;  
3 }
```

带单词名的构造函数

```
1 word::word(string &r) {  
2     frequency = 0;  
3     name = r;  
4 }
```

成员函数

比较单词

```
1 bool word::compare(string &r) {  
2     if (name == r) {  
3         this->addFrequency();  
4         return true;  
5     }  
6     else  
7         return false;  
8 }
```

用于比较成员变量中的单词与参数传入单词的异同，相同返回true，否则返回false

增加词频

```
1 void word::addFrequency() {  
2     frequency++;  
3 }
```

成员变量frequency+1

获取单词名

```

1 //Overload <<
2 ostream& operator<<(ostream &os, const word &obj) {
3     os << obj.name << ": " << obj.frequency<<" times ";
4     return os;
5 }

```

非成员函数

全文单词计数

```

1 //Count the word's frequency
2 void CountFrequency(string file_name,string target_word) {
3     //Open file in the function,to make sure every time the function is called,the buffer is
    new
4     ifstream fin(file_name.c_str());
5     if (fin.good() == 0) {
6         cout << "Open file failed!" << endl;
7         return;
8     }
9     char *str = new char[100];
10    word* key = new word(target_word);
11
12    while (!fin.eof()) {
13        int i = 0;
14        bool flag = 0;
15        while (fin.get(str[i++])) {
16            if (str[i - 1] >= 65 && str[i - 1] <= 90)
17                str[i - 1] += 32;
18            else if (str[i - 1] < 97 || str[i - 1]>122) {
19                str[i - 1] = '\0';
20                break;
21            }
22        };
23        str[i] = '\0';
24        string temp2(str);
25
26        if (target_word == temp2)
27            key->addFrequency();
28    }
29    cout << *key;
30 }

```

首先利用文件流打开文本文档，以eof()作为结束判断，使用循环读取内容，每次读取一个char，利用ASCII码对读取的char进行处理，使其全部转换为小写字母，标点符号用断字符代替，再利用string类型重载的以char*为参数的构造函数，将读取到的单词转化为string类型，使用string类型重载过的=号进行判断，如果相同，在调用该单词的addFrequency函数添加一次频率

行信息分析

```

1 //Find the word's row number,its frequency in the row and position in the row
2 void AnalyseInRow(string file_name, string target_word) {
3     int line = 1;

```

```

4     int word_time = 0;
5     int position_counter = 1;
6     vector<int> position;
7     //Open file in the function,to make sure every time the function is called,the buffer is
new
8     ifstream fin(file_name.c_str());
9     if (fin.good() == 0) {
10         cout << "Open file failed!" << endl;
11         return;
12     }
13     char *str = new char[100];
14
15     while (!fin.eof()) {
16         int i = 0;
17         bool flag = 0;
18         while (fin.get(str[i++])) {
19             //Once get '\n',output information of this line and reload for the next line
20             if (str[i - 1] == '\n') {
21                 cout << target_word << " have appeared " << word_time << " times in line "
<< line;
22
23                 cout << " and the positions are: "<<endl<<endl;
24                 for (auto iter = position.begin(); iter != position.end(); iter++)
25                     cout << *iter << " ";
26                 cout << endl;
27                 line++;
28                 word_time = 0;
29                 position_counter = 1;
30                 position.clear();
31             }
32             if (str[i - 1] >= 65 && str[i - 1] <= 90)
33                 str[i - 1] += 32;
34             else if (str[i - 1] < 97 || str[i - 1]>122) {
35                 str[i - 1] = '\0';
36                 break;
37             }
38         };
39         str[i] = '\0';
40         string temp2(str);
41         if (target_word == temp2) {
42             word_time++;
43             position.push_back(position_counter);
44         }
45         position_counter++;
46     }
47 }

```

读取文本方法与上相同，新增行计数 `line` 以及对 `\n` 的判断，用于获取行号，每次读取到 `\n`，说明换行，输出当前行信息，同时行计数加1

词计数 `position_counter` 用于统计检索单词在该行的位置，每读取一个单词，计数+1，结束一行后计数清零

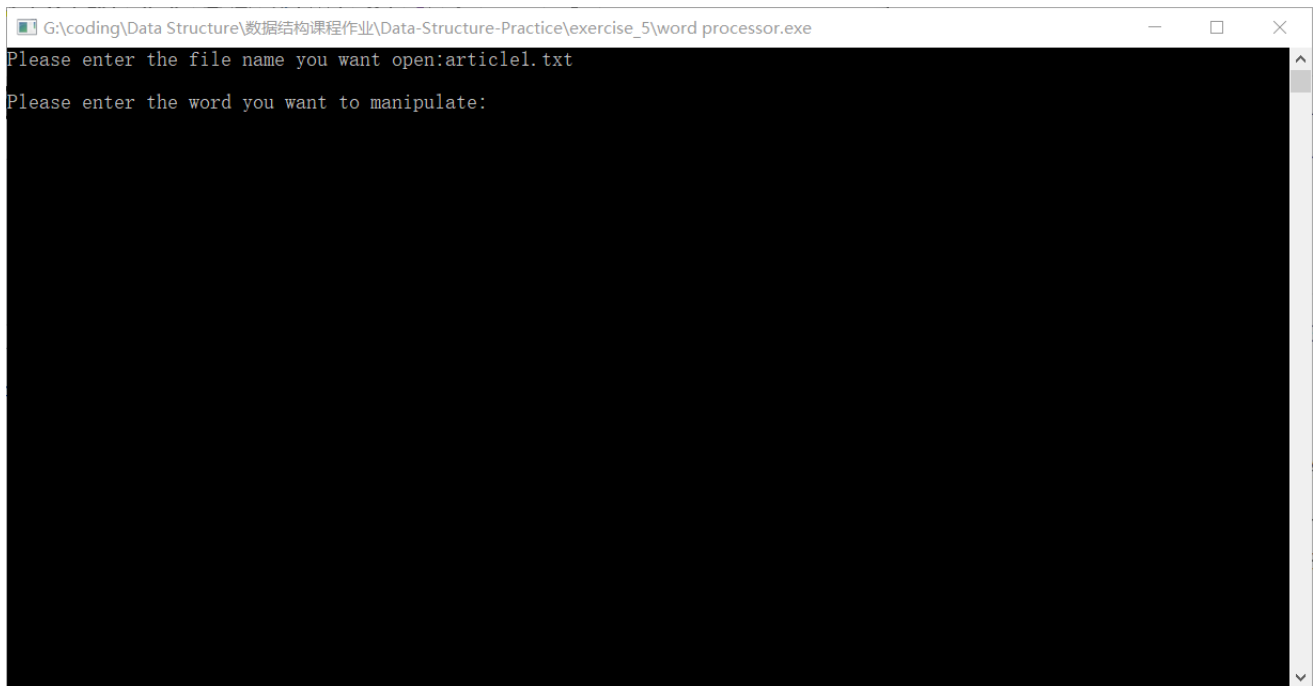
5.使用说明及函数功能演示

文档说明

处理的文档应该保存为txt格式，文档中的换行处理按照win系统下记事本中换行为准，文档保存在和 `word_processor.exe` 相同目录下

打开文档

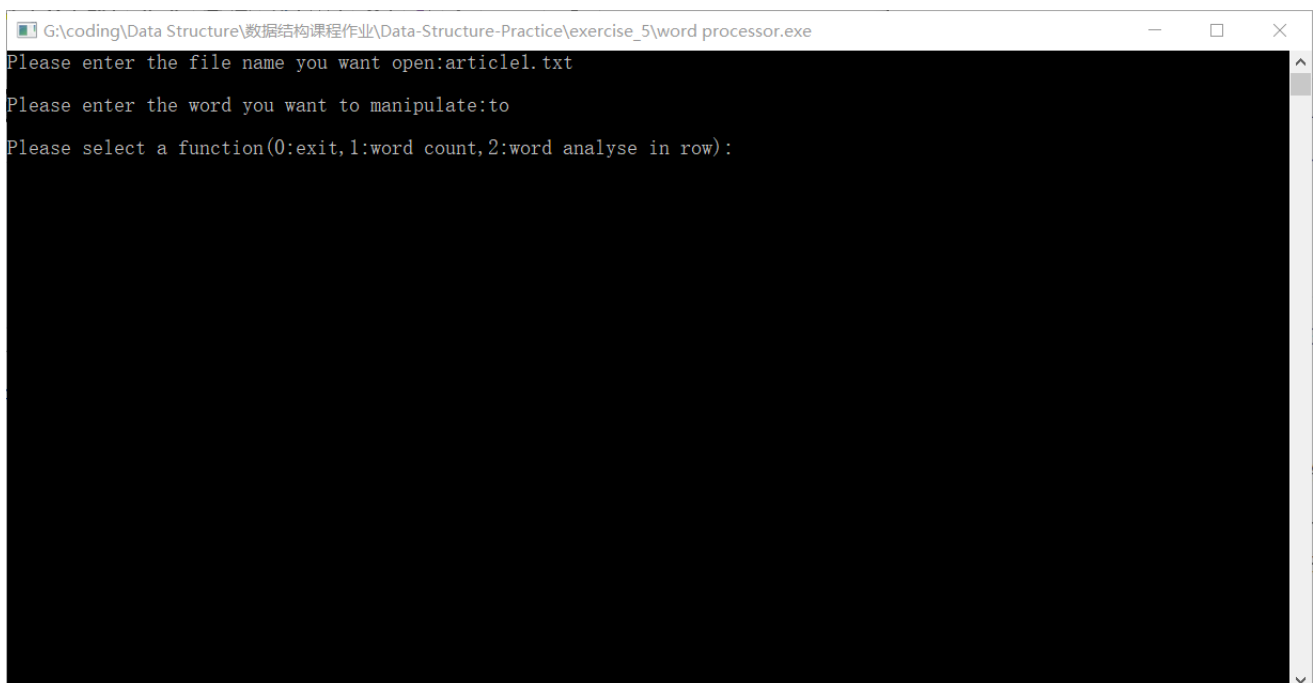
双击运行 `word_processor.exe`，系统提示输入文档名，需输入全称（包括后缀），若成功打开，则会提示输入要处理的单词，若打开失败，则会提示“Open file failed!”



```
G:\coding\Data Structure\数据结构课程作业\Data-Structure-Practice\exercise_5\word processor.exe
Please enter the file name you want open:article1.txt
Please enter the word you want to manipulate:
```

输入单词

按照提示输入需要处理的单词，输入完成后会出现功能选择提示



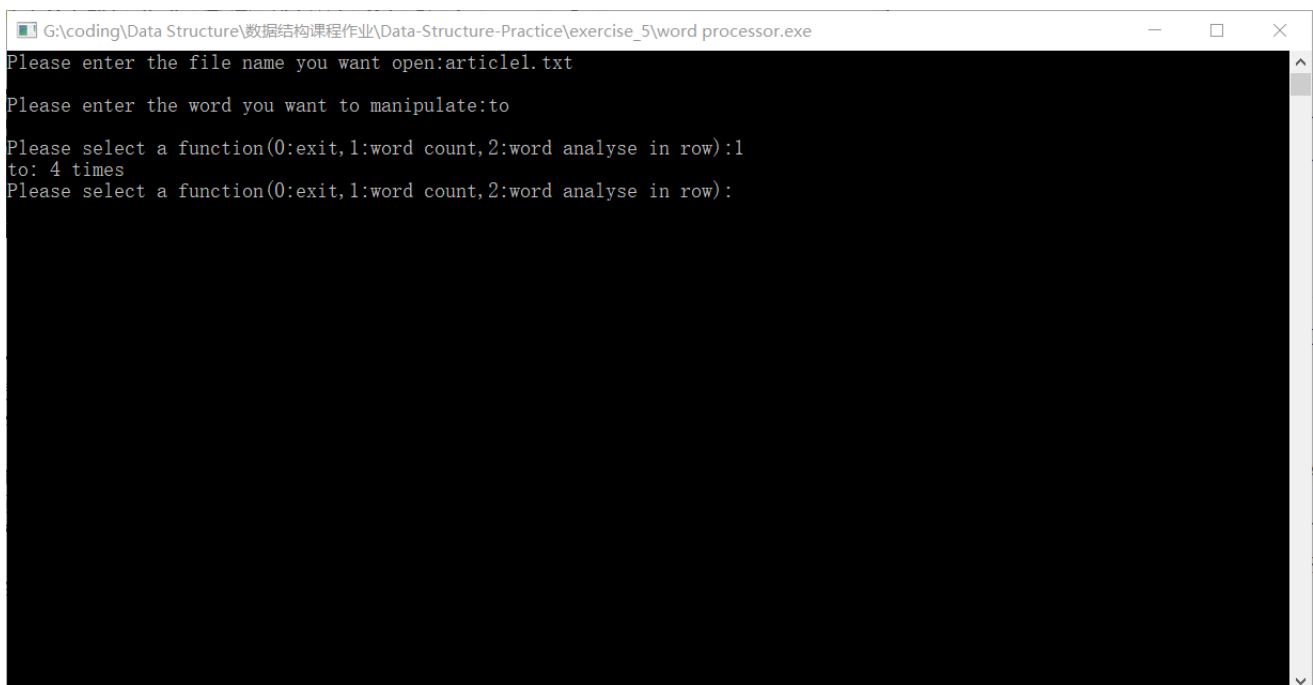
```
G:\coding\Data Structure\数据结构课程作业\Data-Structure-Practice\exercise_5\word processor.exe
Please enter the file name you want open:article1.txt
Please enter the word you want to manipulate:to
Please select a function(0:exit,1:word count,2:word analyse in row):
```

功能选择

数字	功能
0	退出程序
1	全文中单词计数
2	检索单词及行信息分析

按照表格所示输入对应功能数字

输入1：统计输出此前输入单词在文本中的出现次数



```
G:\coding\Data Structure\数据结构课程作业\Data-Structure-Practice\exercise_5\word processor.exe
Please enter the file name you want open:article1.txt
Please enter the word you want to manipulate:to
Please select a function(0:exit,1:word count,2:word analyse in row):1
to: 4 times
Please select a function(0:exit,1:word count,2:word analyse in row):
```

输入2：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


```
G:\coding\Data Structure\数据结构课程作业\Data-Structure-Practice\exercise_5\word processor.exe
Please enter the file name you want open:article1.txt
Please enter the word you want to manipulate:to
Please select a function(0:exit,1:word count,2:word analyse in row):1
to: 4 times
Please select a function(0:exit,1:word count,2:word analyse in row):2
to have appeared 3 times in line 1 and the positions are:
10 17 19
to have appeared 1 times in line 2 and the positions are:
13
to have appeared 0 times in line 3 and the positions are:

Please select a function(0:exit,1:word count,2:word analyse in row):
```

输入0: 退出程序

需要再次启动则重新打开exe文件