

题目五 单词检索统计系统

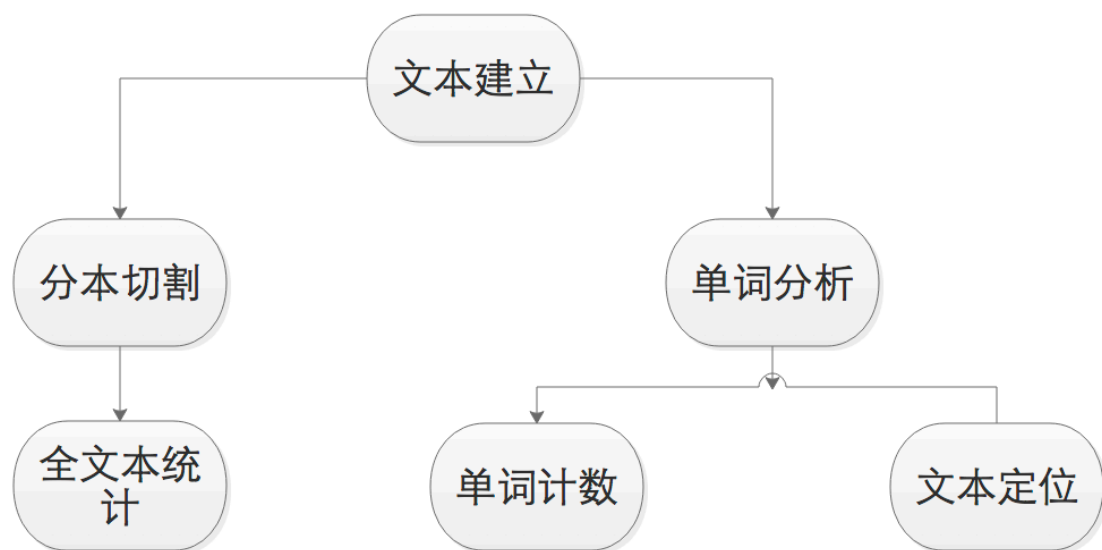
一.设计思路

从题目不难看出这是一个文本处理的系统。单词检索统计系统要求给定一个文本文件，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。

这个问题的核心是对字符串的处理。包括从一段文本中分离出指定的字符串，进行字符串的比较等操作，还涉及文件读写操作。

根据所要求的功能，可将系统分作如下功能块：

逻辑结构如下：



考虑到规范性与便捷性，使用C++ 的 `std::string` 类作为程序内部存储字符串的统一方式。

二.数据结构实现

1.字符串类型（string）

利用C++ STL 容器 `string` 作为从文件中读入的文本的统一存储方式。

2.字典类型（map）

定义了 `map<string, int>` 的字典类型，实现从单词 `string` 到 单词数目 `int` 映射，作为对统计结果存储的类型。

3.集合类型 (set)

定义了 `set <char>` 作为存储非单词字符的容器。C++ STL `set` 类具有不存储相同容器的优良特点，可以实现对非单词字符种类的统计。

4.二维向量 (vector< vector< > >)

定义了二维向量 `vector< vector<int> >` 作为存储对文本中某个单词分析结果的容器。C++ STL `vector` 类具动态扩展大小的优良特点，免去了繁琐的预处理工作。二维向量的第一个维度代表文本行号，第二个维度代表该行中目标单词出现的位置。各行中单词的数量通过 `vector<>` 的内部函数 `size ()` 得到，省去了额外的空间开销。

5.单词切分函数

```
void split_line( const string & str, vector<string> & words );    //单词拆分
```

将一行string类数据切分为单词数组

6.新建文本函数

```
void create_TXT(); //新建文本
```

新建txt文件，文件名和文件内容由用户输入

7.文本单词汇总函数

```
void word_list(string name);    //建立单词汇总信息列表
```

统计文件中单词的种类以及各个单词出现的次数

8.单词计数函数

```
int word_count(vector<vector<int>>& list); //单词计数
```

统计该单词在文件中出现次数

9.单词定位函数

```
void word_locate(vector<vector<int>>& list); //单词定位
```

格式化输出单词在各行出现的情况

10.单词分析函数

```
void word_analyse(string f,string word,vector<vector<int>>& ans); //单词分析
```

分析该单词在文件中出现的情况，记录该单词在各行出现的次数

11.辅助函数

```
void menu1(){
    cout<<"*****\n";
    cout<<"****文本文件单词的检索与计数****\n"
        <<"===== \n"
        <<"*      【1】 建立文本文档      * \n"
        <<"*      【2】 文本单词汇总      * \n"
        <<"*      【3】 单词定位          * \n"
        <<"*      【4】 退出              * \n"
        <<"===== \n";
}
```

打印主菜单。

```

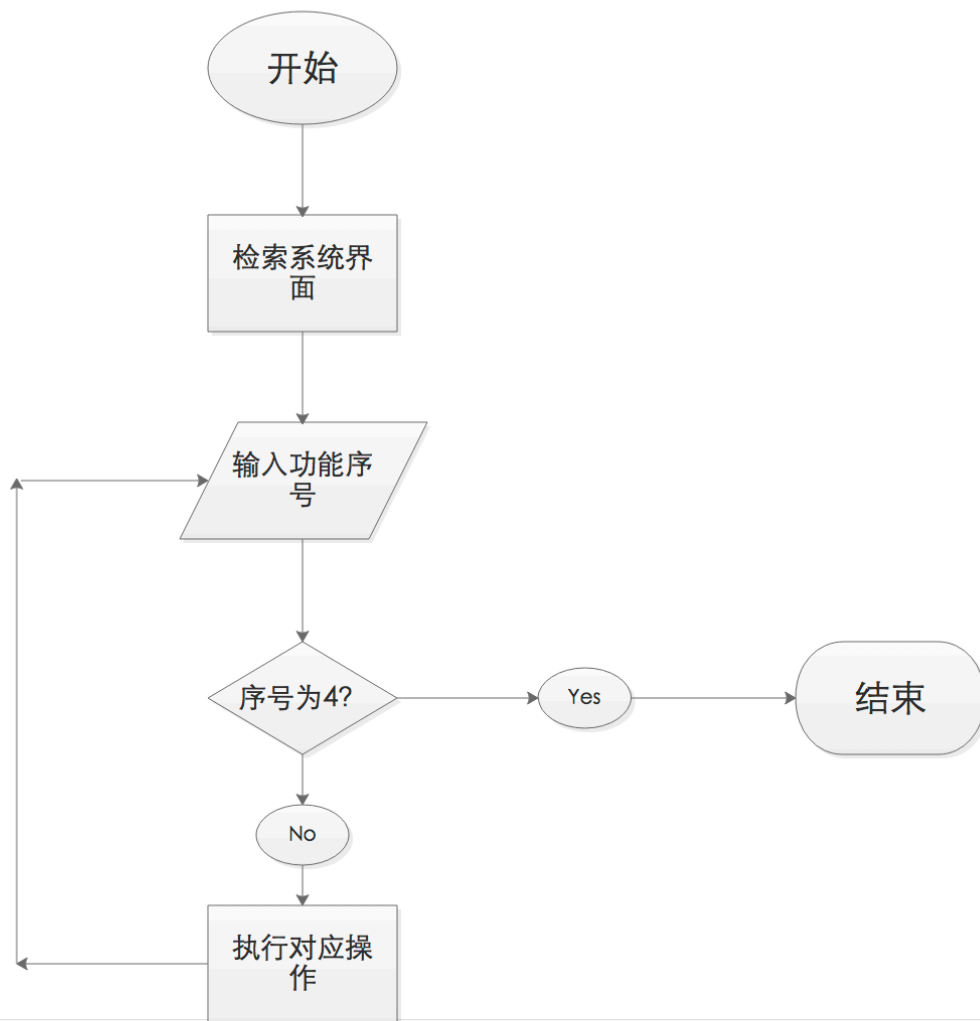
void menu2(){
    cout<<"=====\n"
    <<"|| 文本文件单词字符串的统计定位及定位 ||\n"
    <<"|| =====||\n"
    <<"||      a.      单词出现的次数      ||\n"
    <<"||                                          ||\n"
    <<"||                                          ||\n"
    <<"||      b.      单词出现的位置      ||\n"
    <<"||                                          ||\n"
    <<"===== \n";
}

```

打印功能3的菜单

三.系统实现

1.系统执行框架

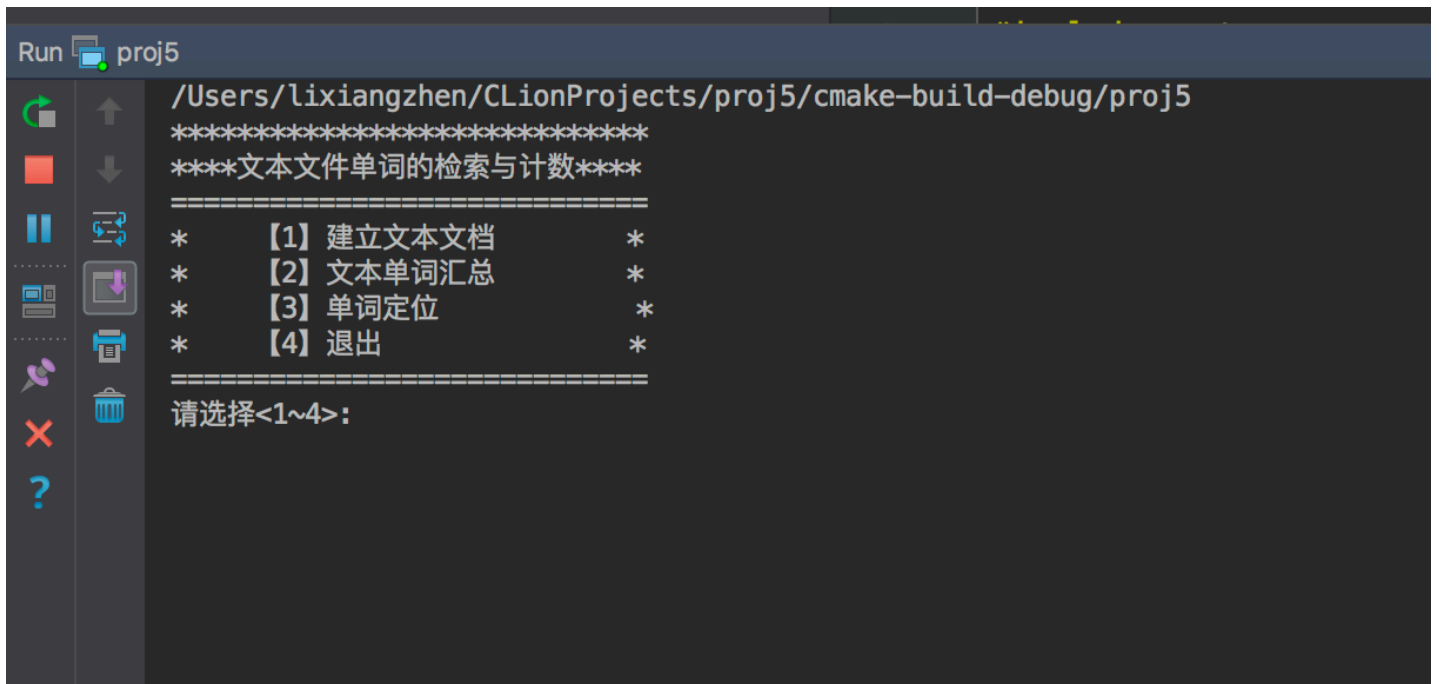


调用 `void menu1()` 函数打印主菜单，提示用户输入所需操作。
没有文本文件的情况下，用户首现需要新建文本文件。

核心代码如下：

```
menu1();        //菜单初始化
int choice;
cout<<"请选择<1~4>:";
cin>>choice;    //输入操作
string fname;
```

程序执行情况如下：



```
Run proj5
/Users/lixiangzhen/CLionProjects/proj5/cmake-build-debug/proj5
*****
***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>:
```

用户输入所选操作，利用 `switch--case` 语句判定不同操作，并跳转到相应位置。

核心代码如下：

```

while(choice != 4){    //根据操作号选择操作
    switch (choice){
        case 1:
            create_TXT();    //新建文件
            break;
        case 2:
            cout<<"请输入文件名: ";
            cin>>fname;
            word_list(fname);    //制作单词汇总表
            break;
        case 3:
            menu2();
            string word;

            cout<<"请输入a或b: ";
            char c;
            cin>>c;
            if(c == 'a'){
                cout<<"请输入文件名: ";
                cin>>fname;
                cout<<"请输入要统计计数的单词: ";
                cin>>word;
                vector<vector<int>> ans;    //记录对该单词的分析结果
                word_analyse(fname,word,ans);
                cout<<"单词"<<word<<"在文本文件"<<fname<<"中共出现"
                    <<word_count(ans)<<"次\n";
            }
            else if(c == 'b'){
                cout<<"请输入文件名: ";
                cin>>fname;
                cout<<"请输入要检索的单词: ";
                cin>>word;
                vector<vector<int>> ans;    //记录单词分析结果
                word_analyse(fname,word,ans);
                word_locate(ans);
            }
            else{
                cout<<"非法输入! \n";
            }
            break;
    }
    menu1();
    cout<<"请选择<1~4>:";
    cin>>choice;
}

```


程序执行情况如下：



2.建立文本文档

核心代码如下：

```

void create_TXT(){
    string name;
    cout<<"输入要建立的文件名: ";
    cin>>name;
    getchar();
    ofstream txt(name); //新建文件
    while(1){           //循环输入文件内容
        cout<<"请输入一行文本: ";
        string line;
        getline(cin,line);
        for(int i = 0;i < line.size();++i) //全部转化为小写字母, 解决大小写问题
            if(isalpha(line[i]))
                line[i] = tolower(line[i]);
        txt<<line<<endl;
        cout<<"输入结束吗? y or n :";
        char choice;
        cin>>choice;
        if(choice == 'y')
            break;
        getchar();
    }
    txt.close(); //文件建立完毕, 关闭文件流
    cout<<"建立文件结束!\n";
}

```

- 输入文件名
- 循环输入文件各行
- 处理大小写问题
- 结束

程序执行情况如下:

```
Run proj5
/Users/lixiangzhen/CLionProjects/proj5/cmake-build-debug/proj5
*****
****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
输入要建立的文件名: tang.txt
请输入一行文本: this is a good boy
输入结束吗? y or n : n
请输入一行文本: this is a bad boy
输入结束吗? y or n : n
请输入一行文本: this is a boy
输入结束吗? 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<1~4>:
```

3.单词分割

核心代码如下：

```

void split_line( const string & str, vector<string> & words )
{
    int i = 0;

    while( i != str.size() )
    {
        while( i != str.size() && !isalpha( str[i] ) )    //跳过非字母类型
            ++i;

        int j = i;
        while( j != str.size() && isalpha(str[j]))        //记录单词跨度
            ++j;
        if( i != j )
        {
            words.emplace_back(str.substr( i, j - i ));    //将单词记录进数组
            i = j;
        }
    }
}

```

- 调用库函数分析字符类型
- 将分好的单词压入数组
- 用 `i` , `j` 分割单词首尾

关联调用情况如下：

```

142     map<string,int> list;
143     set<char> st;          //建立集合类型统计非文本的种类，保证无重复
144     while (getline(file,line)){    //每次从文件中读入一行
145         vector<string> words;
146         split_line(line,words);    //调用分词函数处理改行
147         for(int i = 0;i < line.size();++i){    //记录非文本种类
148             if(!isalpha(line[i])&&!isspace(line[i]))
149                 st.insert(line[i]);
150         }
151         for(int i = 0;i < words.size();++i){

```

4.文本单词汇总

核心代码如下：

程序执行情况如下：

[illegible]

5.单词分析

核心代码如下：

```

void word_analyse(string f,string word,vector<vector<int>>& ans){
    ifstream file(f);
    string line;
    vector<int> positions; //记录每个单词出现的所有位置

    while (getline(file,line)){
        int beg = 0;
        int pos = 0;
        positions.clear();
        while(beg != string::npos){ //循环查找该行中该单词的所有出现情况
            beg = line.substr(pos).find(word);
            if(beg != string::npos){
                pos += beg;
                positions.emplace_back(pos); //记录下每个出现位置
                pos += word.size();
            }
        }
        ans.emplace_back(positions);    //将该行的出现位置记录
    }
}

```

- 循环查找该行中该单词的所有出现情况
- 记录下每个出现位置
- 将该行的出现位置记录

关联调用情况如下：

```

cout<<"请输入要统计计数的单词: ";
cin>>word;
vector<vector<int>> ans;    //记录对该单词的分析结果
word_analyse(fname,word,ans);
cout<<"单词"<<word<<"在文本文件"<<fname<<"中共出现"
    <<word_count(ans)<<"次\n";
}

```



```

        cin>>fname;
        cout<<"请输入要检索的单词: ";
        cin>>word;
        vector<vector<int>>> ans;    //记录单词分析结果
        word_analyse(fname,word,ans);
        word_locate(ans);
    }
    else{

```

6.单词出现次数

核心代码如下：

```

int word_count(vector<vector<int>>& list){
    int count = 0;
    for(int i = 0;i < list.size();++i){    //统计单词在各行数显次数，并累加
        count += list[i].size();
    }

    return count;
}

```

- 遍历分析结果
- 统计单词在各行次数
- 累加结果

程序执行情况如下：

请选择<1~4>:

核心代码如下：

```

void word_locate(vector<vector<int>>& list){
    if(list.size() == 0){          //处理单词不存在的情况
        cout<<"单词不存在! ";
        return;
    }
    for(int i = 0;i < list.size();++i){      //格式化输出单词出现情况
        if(list[i].size() != 0){
            cout<<"行号: "<<i+1<<"， 次数: "<<list[i].size()<<"， 起始位置分别为: ";
            for(int j = 0;j < list[i].size();++j){
                cout<<"第 "<<list[i][j]<<"个字符 ";
            }
            cout<<endl;
        }
    }
}

```

- 处理单词不存在的情况
- 格式化输出单词出现情况

程序执行情况如下：

```
单词boy在文本文件tang.txt中共出现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<1~4>: 3
=====
|| 文本文件单词字符串的统计定位及定位 ||
||=====||
||      a.   单词出现的次数      ||
||=====||
||      b.   单词出现的位置      ||
||=====||

请输入a或b: b
请输入文件名: tang.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<1~4>:
```

Build finished in 186ms (26 minutes ago)

四.测试

1.基本功能测试

测试用例

文件名: MJ.txt

文本内容:

we are the world!

we are the child!

we all like MJ

hahaha

建立文件

程序执行情况如下：

```
Run proj5
/Users/lixiangzhen/CLionProjects/proj5/cmake-build-debug/proj5
*****
***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
输入要建立的文件名: MJ.txt
请输入一行文本: we are the world!
输入结束吗? y or n :n
请输入一行文本: we are the child!
输入结束吗? y or n :n
请输入一行文本: we all like MJ
输入结束吗? y or n :n
请输入一行文本: hahah
输入结束吗? 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
```

统计所有单词

程序执行情况如下：


```
*****
***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

```
=====
|| 文本文件单词字符串的统计定位及定位 ||
||=====||
```

```
||      a.      单词出现的次数      ||
||=====||
```

```
||      b.      单词出现的位置      ||
||=====||
```

请输入a或b: a

请输入文件名: MJ.txt

请输入要统计计数的单词: we

单词we在文本文件MJ.txt中共出现3次

```
*****
***文本文件单词的检索与计数***
```

```
=====
*      【1】 建立文本文档      *
```

统计单个单词位置

程序执行情况如下：

```

单词we在文本文件MJ.txt中共出现5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<1~4>:3
=====
|| 文本文件单词字符串的统计定位及定位 ||
||=====||
||      a.      单词出现的次数      ||
||              ||
||              ||
||      b.      单词出现的位置      ||
||              ||
||=====||
请输入a或b: b
请输入文件名: MJ.txt
请输入要检索的单词: we
行号: 1, 次数: 1, 起始位置分别为: 第 0个字符
行号: 2, 次数: 1, 起始位置分别为: 第 0个字符
行号: 3, 次数: 1, 起始位置分别为: 第 0个字符
*****
****文本文件单词的检索与计数****
=====

```

程序退出

程序执行情况如下：


```
*****
***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>:4

Process finished with exit code 0
```

2.边界测试

单词大小写问题

测试用例

文件名：test1.txt

文本内容：

This is this too

程序执行情况如下：

```
Run proj5
/Users/lixiangzhen/CLionProjects/proj5/cmake-build-debug/proj5
*****
****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
输入要建立的文件名: test1.txt
请输入一行文本: This is this too
输入结束吗? 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<1~4>:2
请输入文件名: test1.txt
>>>>>>>>>单词<<<>>>>个数<<<<<<<<<
           is           1
           this         2
           too          1

>>>>>>>>>>>>>>>test1.txt的单词个数为4个

>>>>>>>>>>>>>>>test1.txt的非单词个数为0种
*****
```

非单词字符不重复统计

测试用例

文件名: test2.txt

文本内容:

one! tow! one* two*

程序执行情况如下:

```
Run proj5
/Users/lixiangzhen/CLionProjects/proj5/cmake-build-debug/proj5
*****
****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
输入要建立的文件名: test2.txt
请输入一行文本: one! tow! one* two*
输入结束吗? 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<1~4>:2
请输入文件名: test2.txt
>>>>>>>>>>单词<<<>>>>个数<<<<<<<<
           one           2
           tow           1
           two           1

>>>>>>>>>>>>>>test2.txt的单词个数为4个

>>>>>>>>>>>>>>test2.txt的非单词个数为2种
*****
```

查找的单词不存在

文件名: MJ.txt

文本内容:

we are the world!

we are the child!

we all like MJ

hahaha

查找对象：happy

程序执行情况如下：

```
/Users/lixiangzhen/CLionProjects/proj5/cmake-build-debug/proj5
*****
****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
=====
|| 文本文件单词字符串的统计定位及定位 ||
||=====||
||      a.    单词出现的次数      ||
||=====||
||      b.    单词出现的位置      ||
||=====||
请输入a或b: b
请输入文件名: MJ.txt
请输入要检索的单词: happy
单词不存在!
*****
```