**北京科技大学实验报告**

学院：计通学院 专业：物联网工程 班级：物联1602

姓名： 周康能 学号：41624657 实验日期： 2018年 5月 21 日

**实验名称：《数据结构》实验6综合应用**

**实验目的：能根据特定问题需求，综合应用学过的数据结构知识，分析建立计算模型（包括逻辑结构和物理结构）、设计算法和程序，并在设计中综合考虑多种因素，对算法的有效性进行分析。**

**实验内容：**

统计若干个大型英文txt文件（如英文小说）中所有单词出现的次数，并输出出现次数最多的前10个单词及其出现次数。假设单词字符定义为大小写字母、数字和下划线，其他字符均看作单词分隔符。

要求：

（1）自行设计合适的数据结构及相关算法；

（2）程序运行结束时将txt文件名以及统计结果写入磁盘；

（3）每次程序启动时（除了首次运行）将上次的结果读入内存、显示；

（4）能根据用户选择实现重新初始化、查找某单词出现次数、追加统计、退出等功能。

**问题分析与算法思路：**

根据题目要求，统计txt文件中单词出现的次数，用到的查找的数据结构为hash表，解决冲突的方式为链地址，关键码值的求得具有随机性。

排序的方式用的是快速排序。

**算法描述：**

typedef struct HashList

{

char word[20];

int count = 0; //计数器置为零

struct HashList\* next;

}HashList;

哈希表的定义

bool BuildHash(FILE \*file)

{

逐个字符去读取

求取关键码值

再放入到哈希表中的相应位置

}

建表函数

void QuickSort(HashList\*\* List, int l, int r)

{

快排

}

快排函数

**程序实现：**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Copyright(c) :Ellis

\* All rights reserved.

\*

\* 文件名称:USTB2018\_data\_structure\_test6

\* 简要描述:

\* 要求：题目：统计若干个大型英文txt文件（如英文小说）中所有单词出现的次数，并输出出现次数最多的前10个单词及其出现次数。

\*

\* 单词字符定义为大小写字母、数字和下划线，

\* 其他字符均看作单词分隔符；

\* 自行设计合适的数据结构及相关算法；

\* 程序运行结束时将txt文件名以及统计结果写入磁盘；

\* 每次程序启动时（除了首次运行）将上次的结果读入内存、显示；

\* 能根据用户选择实现重新初始化、查找某单词出现次数、追加统计、退出等功能。

\*

\*

\* 创建日期:20180519\_10:20

\* 作者:Ellis

\* 说明:完成所有任务

\*

\* 修改日期:

\* 作者:Ellis

\* 说明:

\* 注意：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 宏定义、头文件、命名空间区

\* 创建日期: 20180519

\* 作者 : Ellis

\* 说明 : 无

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define \_CRT\_SECURE\_NO\_DEPRECATE

#pragma warning(disable: 4996)

#include <string>

#include <stdlib.h>

#include <iostream>

#include <ctype.h>

#define NUM 9999 //后期需要改

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 结构体定义区

\* 创建日期: 20180519

\* 作者 : Ellis

\* 说明 : 链式哈希

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct HashList

{

char word[20];

int count = 0; //计数器置为零

struct HashList\* next;

}HashList;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 全局变量区

\* 创建日期: 20180519

\* 作者 : Ellis

\* 说明 :

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

HashList\* HashKeyList[NUM];

int Num\_word = 0; //计数器

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名称 : 函数定义区

\* 创建日期: 20180519

\* 作者 : Ellis

\* 说明 : 定义各种函数,集成化，容易看

\*

\* 修改日期:

\* 作者 :

\* 说明 :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Put\_in(char\* word); //插入函数

int Cal\_The\_Key\_Value(char\* word); //算关键值

bool BuildHash(FILE \*file); //创建哈希表

void PrintTheList(HashList\*\* HashKeyList); //打印出哈希表（测试用）

void WriteResult(HashList\*\* HashKeyList); //写结果入txt

void QuickSort(HashList\*\* List, int l, int r); //快排

void ReList(HashList\*\* HashKeyList, HashList\*\* sort\_hash); //将哈希表重构为一个无冲突的表

void Print\_Most(HashList\*\* sort); //打印出频率最高的十种频率对应的单词

void Seek\_for(); //追加搜索，哈希查找

void DataLastTime(); //显示上次的统计结果

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void main()

Author :Ellis

Date :20180519

Description :

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

DataLastTime(); //显示出上次统计数据

//-------------------------------------------//

//输入要打开的txt文件，并打开

//-------------------------------------------//

FILE \*file;

char name\_of\_txt[20];

printf("Please enter the name of the txt...\n");

scanf("%s", name\_of\_txt);

file = fopen(name\_of\_txt, "r");

//-------------------------------------------//

HashList\* sort\_hash[NUM];

if (!file)

{

printf("Could not open this txt!\n");

system("pause");

}

else

{

printf("Open the txt successfully!\n");

}

BuildHash(file);

//PrintTheList(HashKeyList);

//WriteResult(HashKeyList);

ReList(HashKeyList, sort\_hash);

//开始快排

QuickSort(sort\_hash, 0, Num\_word - 1);

WriteResult(sort\_hash);

//printf("%d", Num\_word);

//取最多十个

printf("////////////////////////////////");

Print\_Most(sort\_hash);

char t;

while (1)

{

printf("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

printf("\/\t\t\t\t控制台\t\t\t\t\t\/\n");

printf("\/\t\t\tCopyRight:\tEllis\t\t\t\t\/\n");

printf("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

printf("\/\t\t\t\tDescription:\t\t\t\t\/\n");

printf("\/\t\t\tEnter'A'to initialize the program.\t\t\/\n");

printf("\/\t\t\tEnter'B'to seek ont word.\t\t\t\/\n");

printf("\/\t\t\tEnter'C'to count more.\t\t\t\t\/\n");

printf("\/\t\t\tEnter'D'to exit.\t\t\t\t\/\n");

printf("/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/\n");

//scanf("%d", &t);

getchar(); //清回车

t = getchar();

if (t == 'A')

{

remove("结果.txt");

printf("Remove 结果.txt successfully.\n");

}

else if (t == 'B')

{

Seek\_for();

continue;

}

else if (t == 'C')

{

//---------------------------------//

//防止出错的初始化

//---------------------------------//

for (int j = 0; j < NUM;j++)

{

HashKeyList[j] = NULL;

}

Num\_word = 0;

//---------------------------------//

FILE \*filee;

char name[20];

printf("Please enter the name of the txt...\n");

scanf("%s", name);

filee = fopen(name, "r");

HashList\* sort\_hash1[NUM];

if (!filee)

{

printf("Could not open this txt!\n");

system("pause");

}

else

{

printf("Open the txt successfully!\n");

}

BuildHash(filee);

//PrintTheList(HashKeyList);

//WriteResult(HashKeyList);

ReList(HashKeyList, sort\_hash1);

//开始快排

QuickSort(sort\_hash1, 0, Num\_word - 1);

WriteResult(sort\_hash1);

//取最多十个

printf("////////////////////////////////");

Print\_Most(sort\_hash1);

}

else if (t == 'D')

{

//free(HashKeyList);

//free(sort\_hash);

\*sort\_hash = NULL;

system("pause");

return 0;

}

else

{

printf("Error input!Please try again!\n");

continue;

}

}

system("pause");

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:int Cal\_The\_Key\_Value(char\* word)

Author :Ellis

Date :20180519

Description :

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Cal\_The\_Key\_Value(char\* word)

{

char\* p = word;

int key = \*p; //良好的随机性

while (\*p != '\0')

{

key = key + \*p;

p++;

}

key = key % 100;

return key;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:bool BuildHash(FILE \*file)

Author :Ellis

Date :20180519

Description :建立哈希表

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool BuildHash(FILE \*file)

{

int k = 0;

char ch;

char word[20];

int flag = 0;

while (

(ch = fgetc(file)) != EOF //逐个字符去读取

)

{

//-------------------------------------------------

if (ch >= 'A' && ch <= 'z')

{

ch = tolower(ch); //读到字母的操作，将其转化为全为小写的

word[k] = ch; //标志位计为1，为下一步一同构成字母的符号和数字的检测提供方法

k++;

flag = 1;

}

//------------------------------------------------

else if (

(

ch == '\_'

||

(

ch <= '9'

&&

ch >= '0'

)

)

&&

flag == 1

)

{

word[k] = ch;

k++;

}

else

{

if (flag == 0)

{

continue;

}

else

{

//------------------------------------

//插入操作（结尾是\0,且标志位符合），注意插入后要清标志位和置空word

//------------------------------------

word[k] = '\0';

Put\_in(word);

k = 0;

//\*word = NULL;

memset(word, 0, sizeof(word));

flag = 0;

//------------------------------------

}

}

}

fclose(file); //文件有开一定要有闭

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void Put\_in(char\* word)

Author :Ellis

Date :20180519

Description :插入函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Put\_in(char\* word)

{

int order = Cal\_The\_Key\_Value(word); //算出关键码值

HashList\* mid = HashKeyList[order]; //防止对哈希表的改变，取一个中间变量

while (1)

{

if (!mid)

{

break;

}

if (strcmp(word, mid->word) == 0) //如果此值对应的表上没有数据，申请内存并插入

{ //如果有其他的数据存在，插到表头

mid->count++; //如果此数已经存在，count++

//break;

return;

}

mid = mid->next;

}

HashList\* fact = (HashList\*)malloc(sizeof(HashList));

fact->next = HashKeyList[order];

HashKeyList[order] = fact;

fact->count = 1;

strcpy(fact->word, word);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void PrintTheList(HashList\*\* HashKeyList)

Author :Ellis

Date :20180519

Description :打印哈希表（测试用的）

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PrintTheList(HashList\*\* HashKeyList)

{

if (!HashKeyList)

{

return;

}

for (int i = 0; i < NUM - 1; i++)

{

HashList\* mid = HashKeyList[i];

while (mid)

{

printf("%s\t\t", mid->word);

printf("%d\n", mid->count);

mid = mid->next;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void WriteResult(HashList\*\* HashKeyList)

Author :Ellis

Date :20180519

Description :写入函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteResult(HashList\*\* HashKeyList)

{

FILE \*file;

char name\_of\_txt[20] = "结果.txt";

file = fopen(name\_of\_txt, "w");

if (!HashKeyList)

{

fprintf(file, "无结果！");

return;

}

//for (int i = 0; i < NUM-1; i++)

for (int i = 0; i < Num\_word; i++)

{

HashList\* mid = HashKeyList[i];

fprintf(file, "%s\t\t", mid->word); //注意fprintf的使用

//printf("\t");

fprintf(file, "%d\n", mid->count);

//mid = mid->next;

}

printf("写入成功！\n");

fclose(file); //一定记住要关闭

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void QuickSort(HashList\*\* List, int l, int r)

Author :Ellis

Date :20180519

Description :快排递归法

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void QuickSort(HashList\*\* List, int l, int r)

{

int i, j;

HashList\* x;

HashList\* mid = NULL;

if (l < r)

{

i = l;

j = r;

x = List[i]; //取0位置

//x = List[i]->count;

while (i < j)

{

while (i < j && List[j]->count >= (x->count))

{

j--;

}

if (i < j)

{

List[i] = List[j];

i++;

}

while (i < j && List[i]->count <= (x->count))

{

i++;

}

if (i < j)

{

List[j] = List[i];

j--;

}

}

//mid = List[i] ;

//List[i] = x;

//List[l] = mid;

List[i] = x;

QuickSort(List, l, i - 1);

QuickSort(List, i + 1, r);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void ReList(HashList\*\* HashKeyList, HashList\*\* sort\_hash)

Author :Ellis

Date :20180519

Description :重构哈希表

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReList(HashList\*\* HashKeyList, HashList\*\* sort\_hash)

{

if (!HashKeyList)

{

return;

}

//整理数据到一个无next的哈希表

int k = 0;

for (int i = 0; i < NUM - 1; i++)

{

if (HashKeyList[i])

{

HashList\* mid = HashKeyList[i];

//HashList\* st = sort\_hash[k];

sort\_hash[k] = mid;

//strcpy(sort\_hash[k]->word, mid->word);

//st->count = mid->count;

k++;

Num\_word++;

while (mid->next)

{

mid = mid->next;

sort\_hash[k] = mid;

k++;

Num\_word++;

//strcpy(st->word, mid->word);

//st->count = mid->count;

}

}

else

{

continue;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void Print\_Most(HashList\*\* sort)

Author :Ellis

Date :20180519

Description :打印出出现频率最高的十个频率对应的单词

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Print\_Most(HashList\*\* sort)

{

int counting = 1;

printf("\n");

printf("频率最高的：\n");

for (int i = Num\_word - 1; i > 0; i--)

{

if (counting == 10)

{

break;

}

if (i < Num\_word && sort[i]->count != sort[i - 1]->count)

{

counting++;

}

printf("%s\t\t%d\n", sort[i]->word, sort[i]->count);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void Seek\_for()

Author :Ellis

Date :20180519

Description :进一步查询函数

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Seek\_for()

{

printf("Enter the word you want to seek for ...\n");

int flag = 0;

char name[20];

scanf("%s", &name);

int s = Cal\_The\_Key\_Value(name); //计算关键码值，查找

HashList\* mid = HashKeyList[s]; //若不存在重试

while (mid)

{

if (!strcmp(mid->word, name))

{

printf("%s\t%d\n", name, mid->count);

flag = 1;

}

mid = mid->next;

}

if (flag ==0)

{

printf("未找到！请重试！\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:void DataLastTime()

Author :Ellis

Date :20180519

Description :输出上次的统计结果，不存在的话（已经被清了）就不显示

Inputs :

Outputs :

Notes :

Revision :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DataLastTime()

{

FILE \*file;

char name[20] = "结果.txt";

if (file = fopen(name, "r"))

{

char c;

printf("上一次的统计数据:\n");

while ((c = fgetc(file)) != EOF)

{

printf("%c", c);

}

fclose(file);

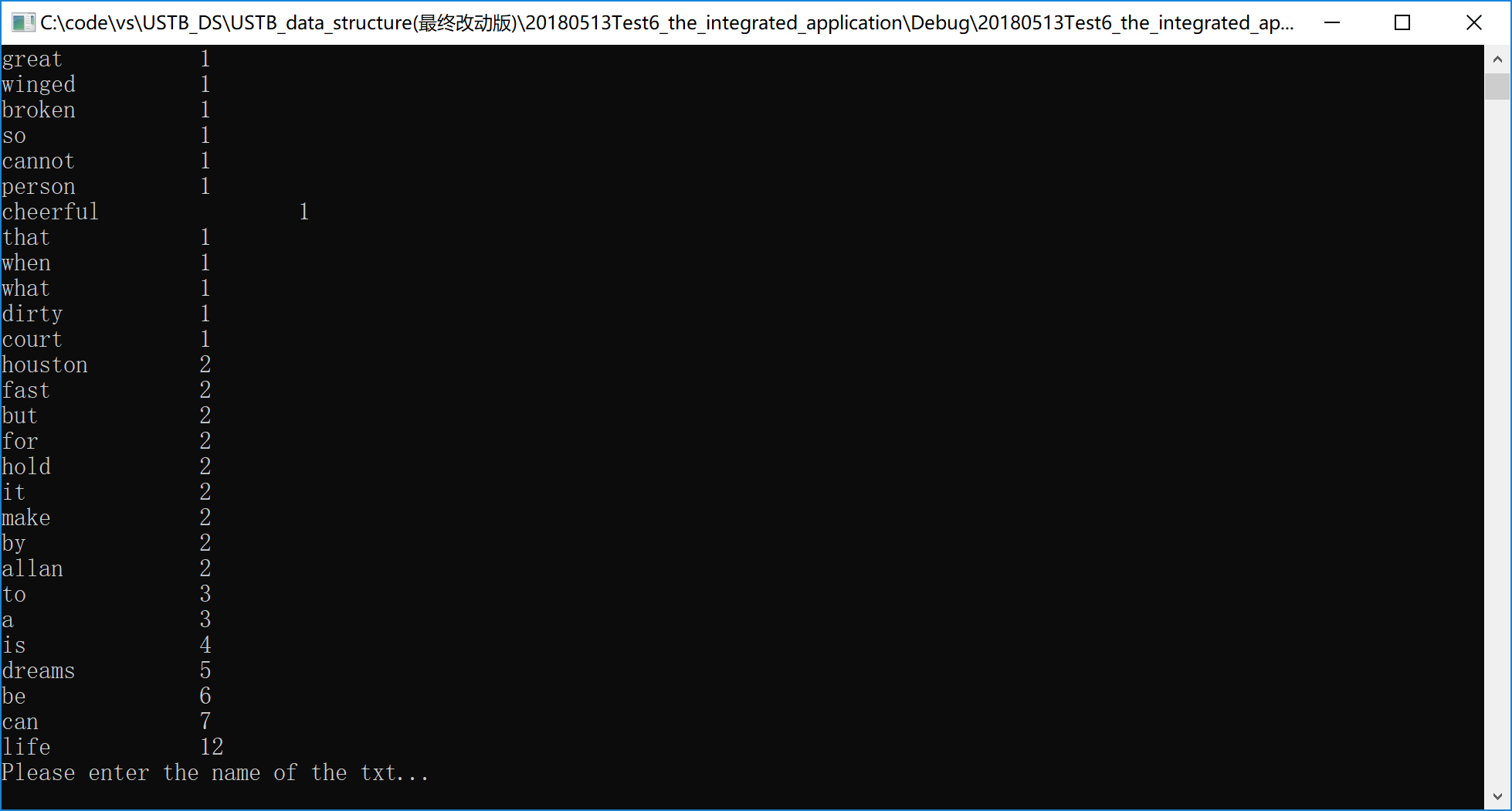
}

}

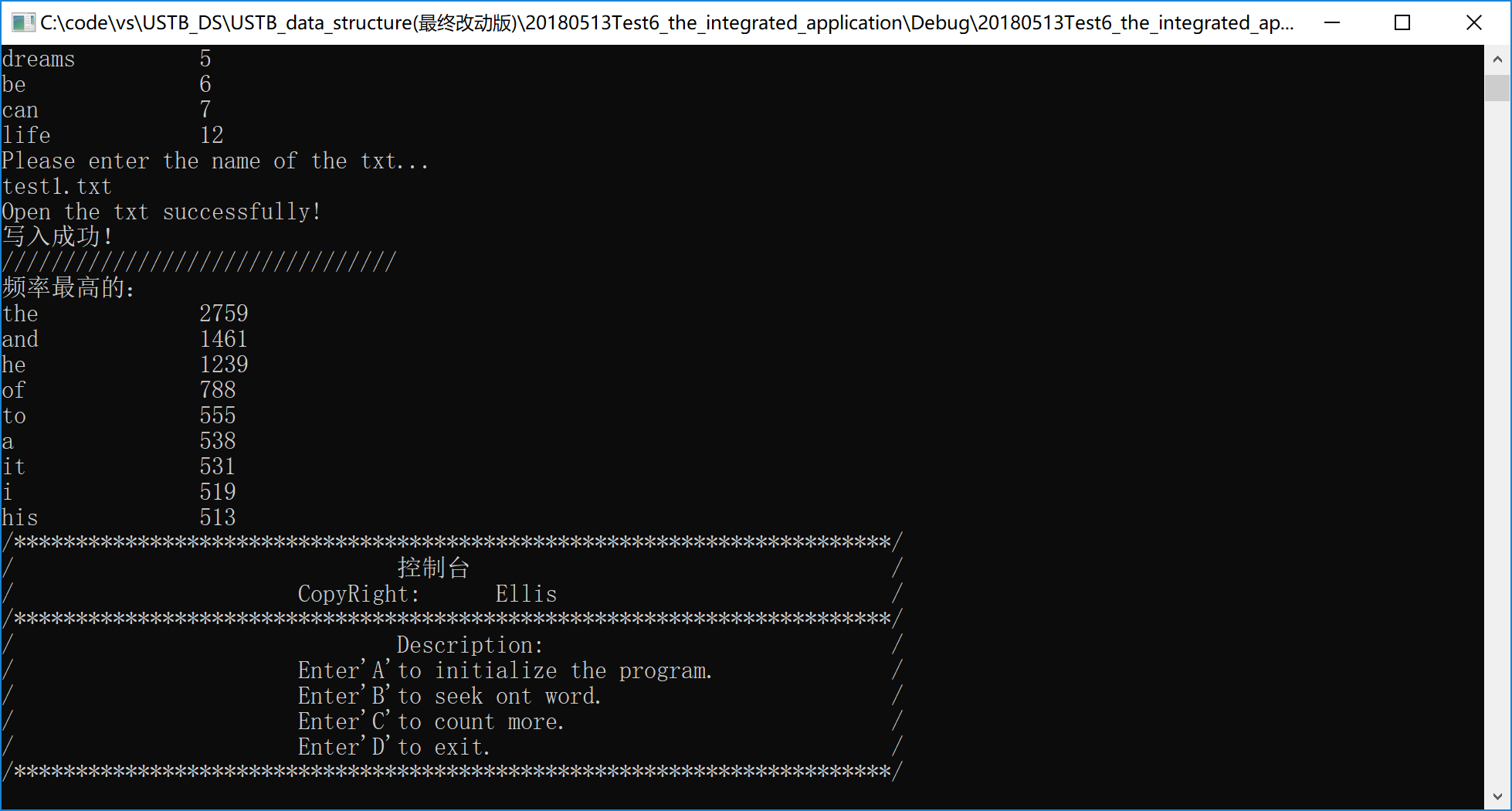
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**测试：**

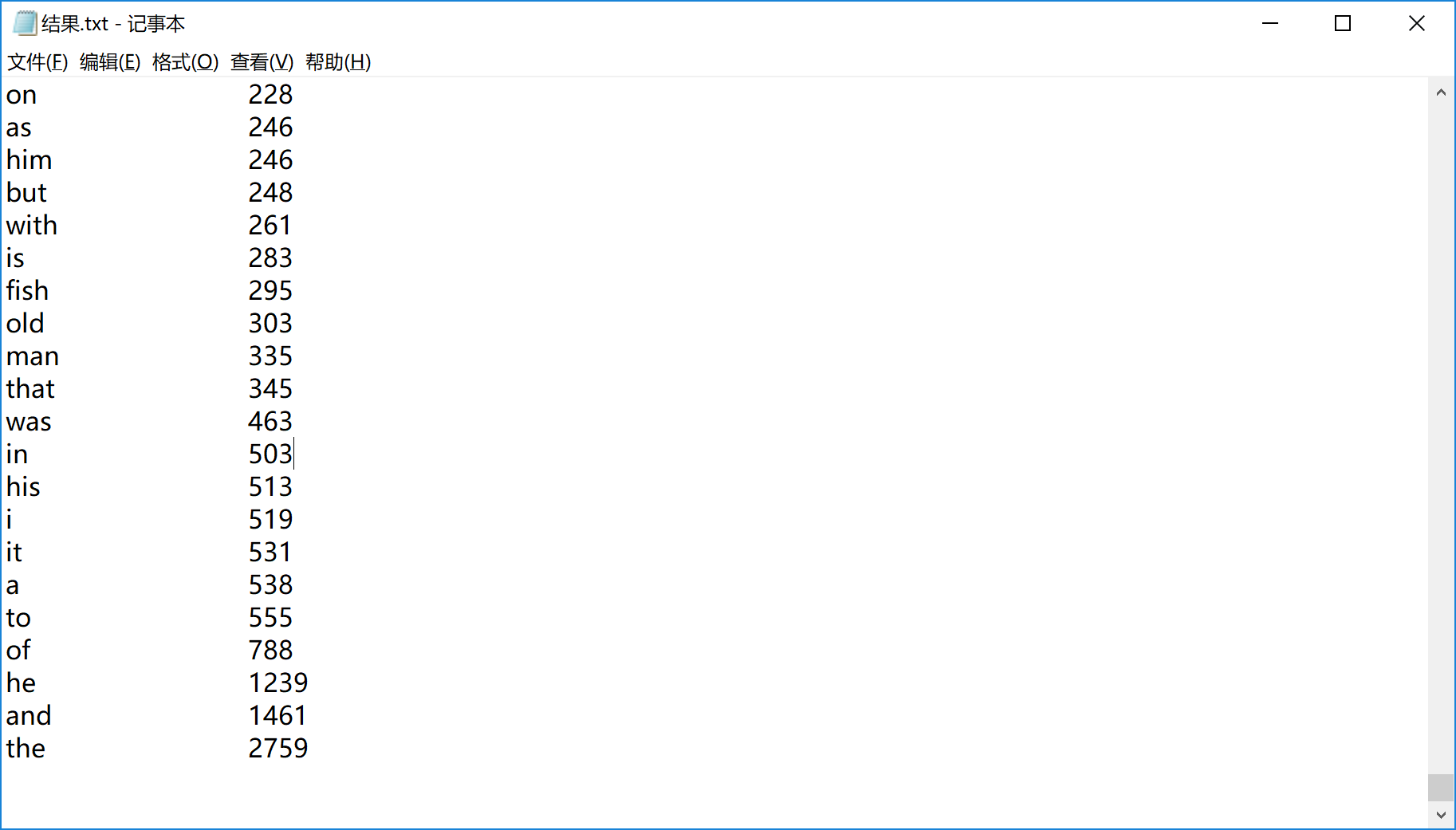
测试每次运行前都能显示出上次统计结果：



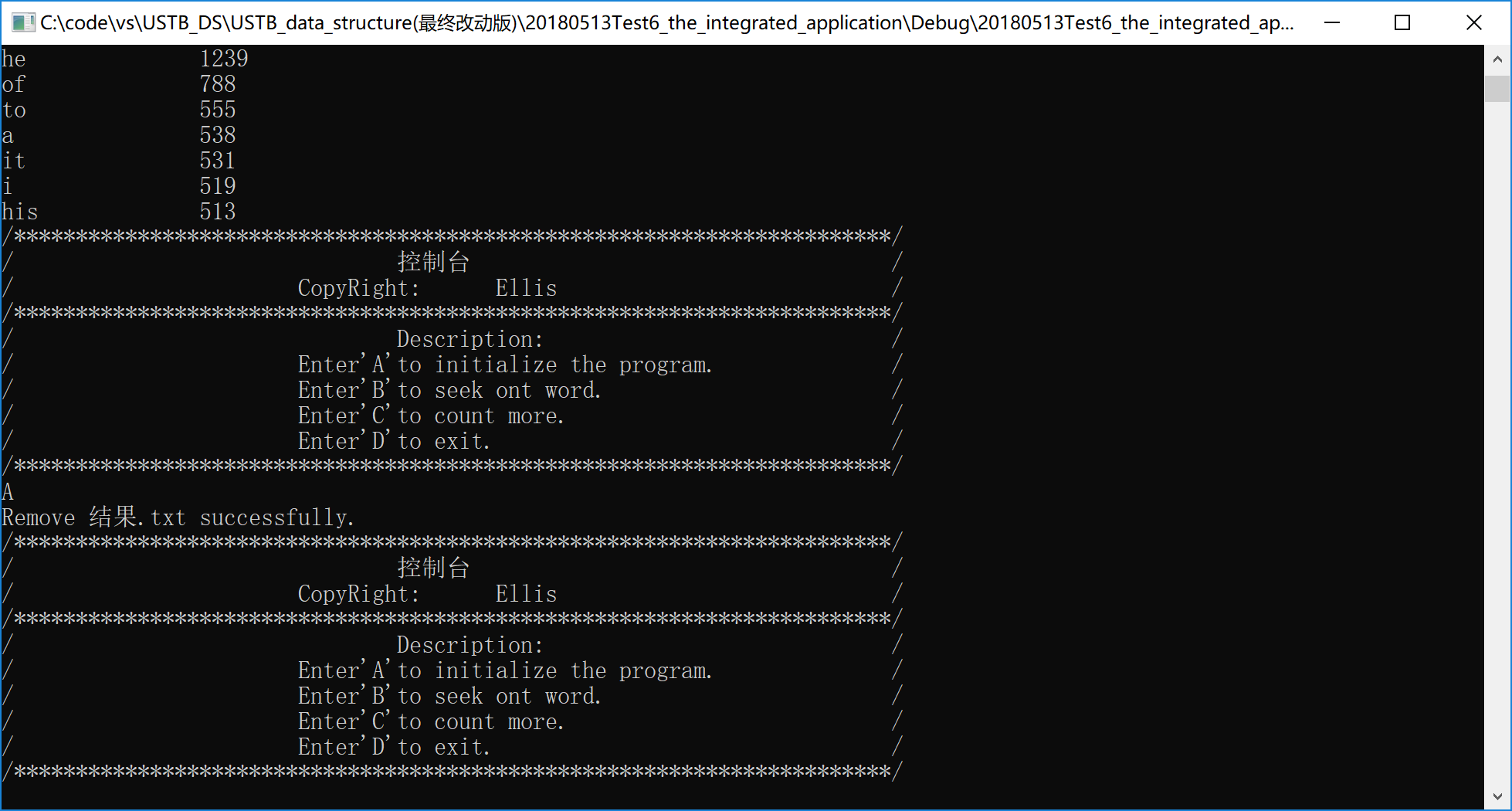
测试统计：



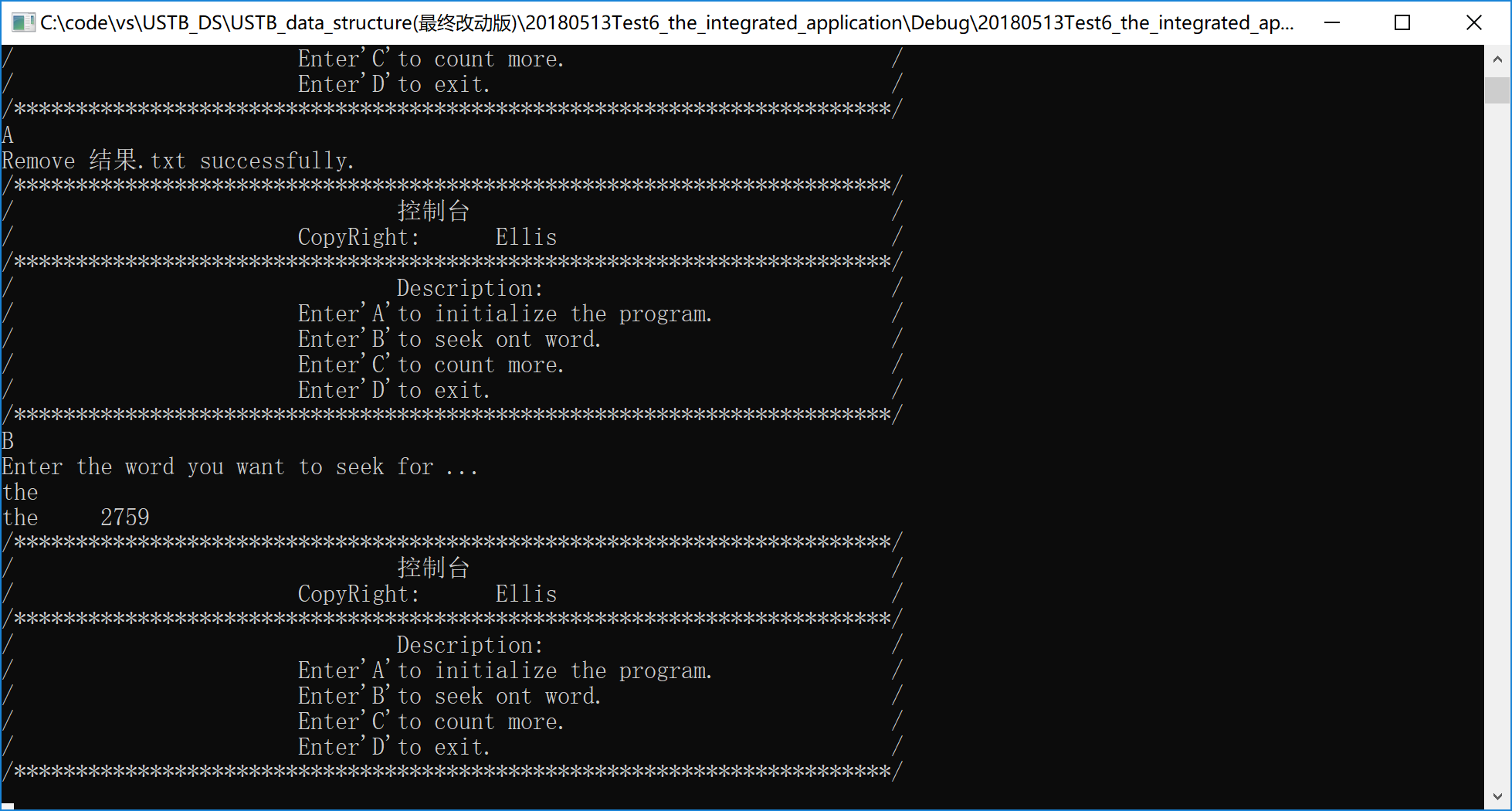
测试已经把结果存入磁盘：



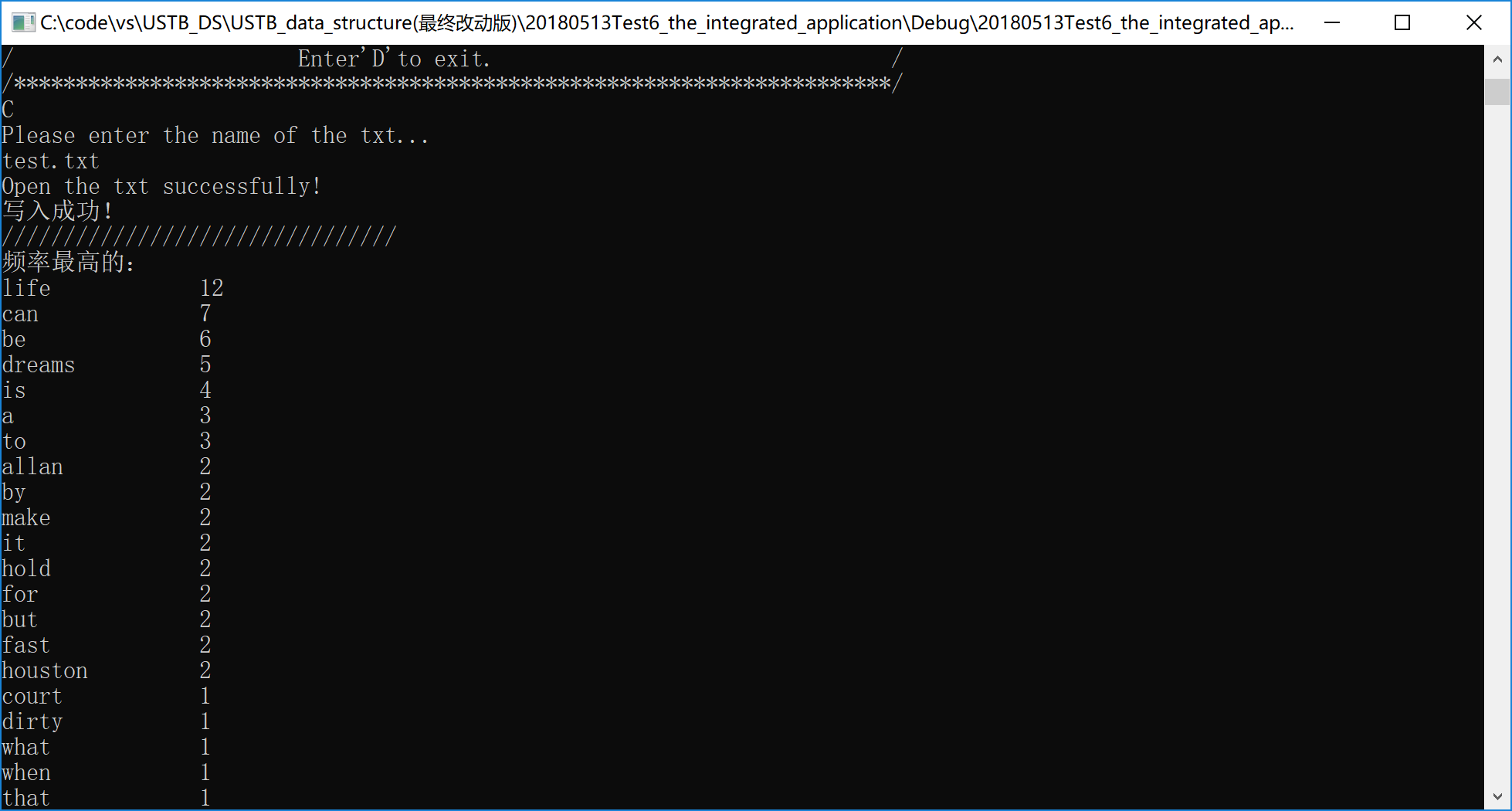
测试重新初始化：



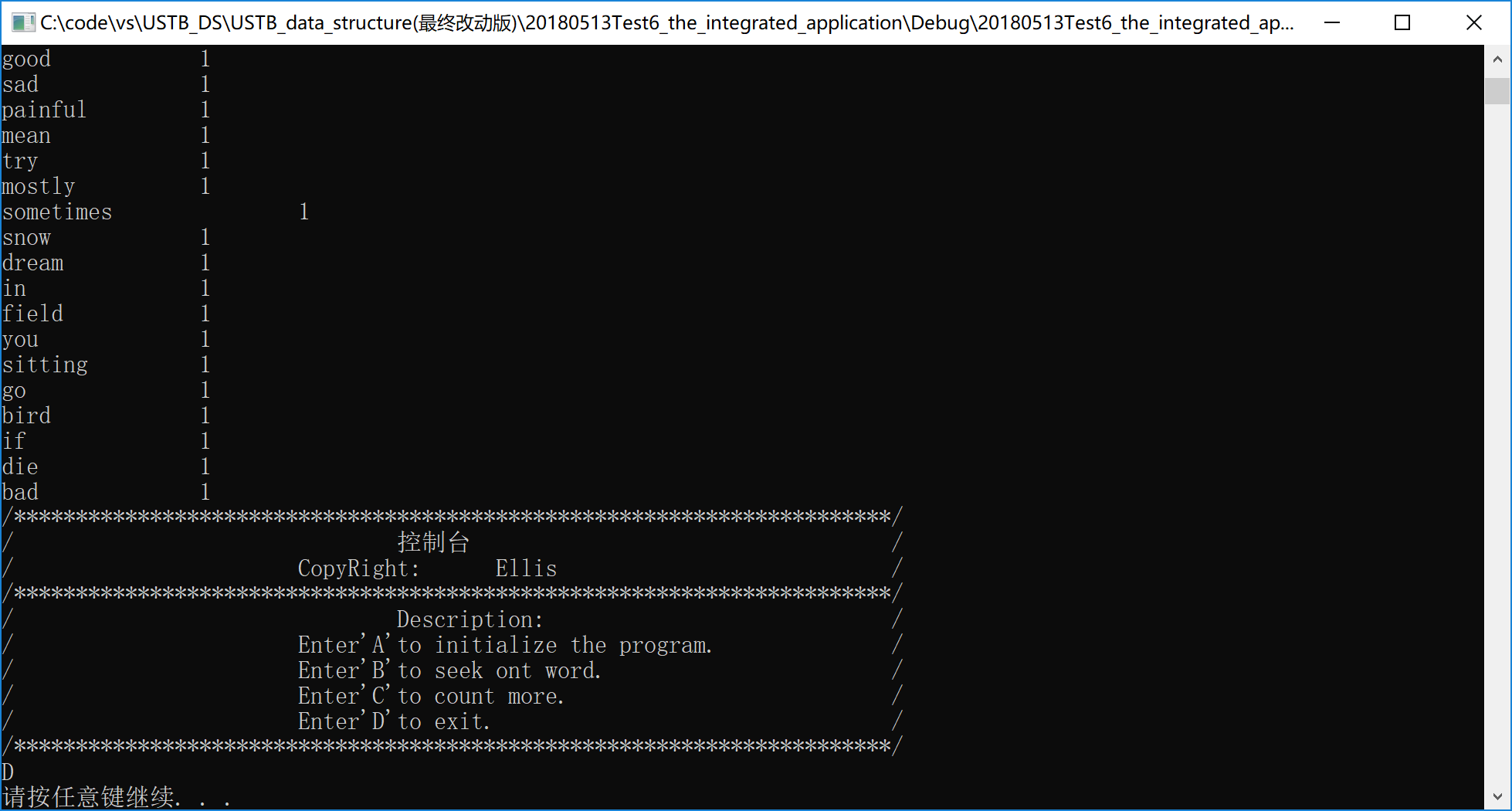
测试查找某单词次数：



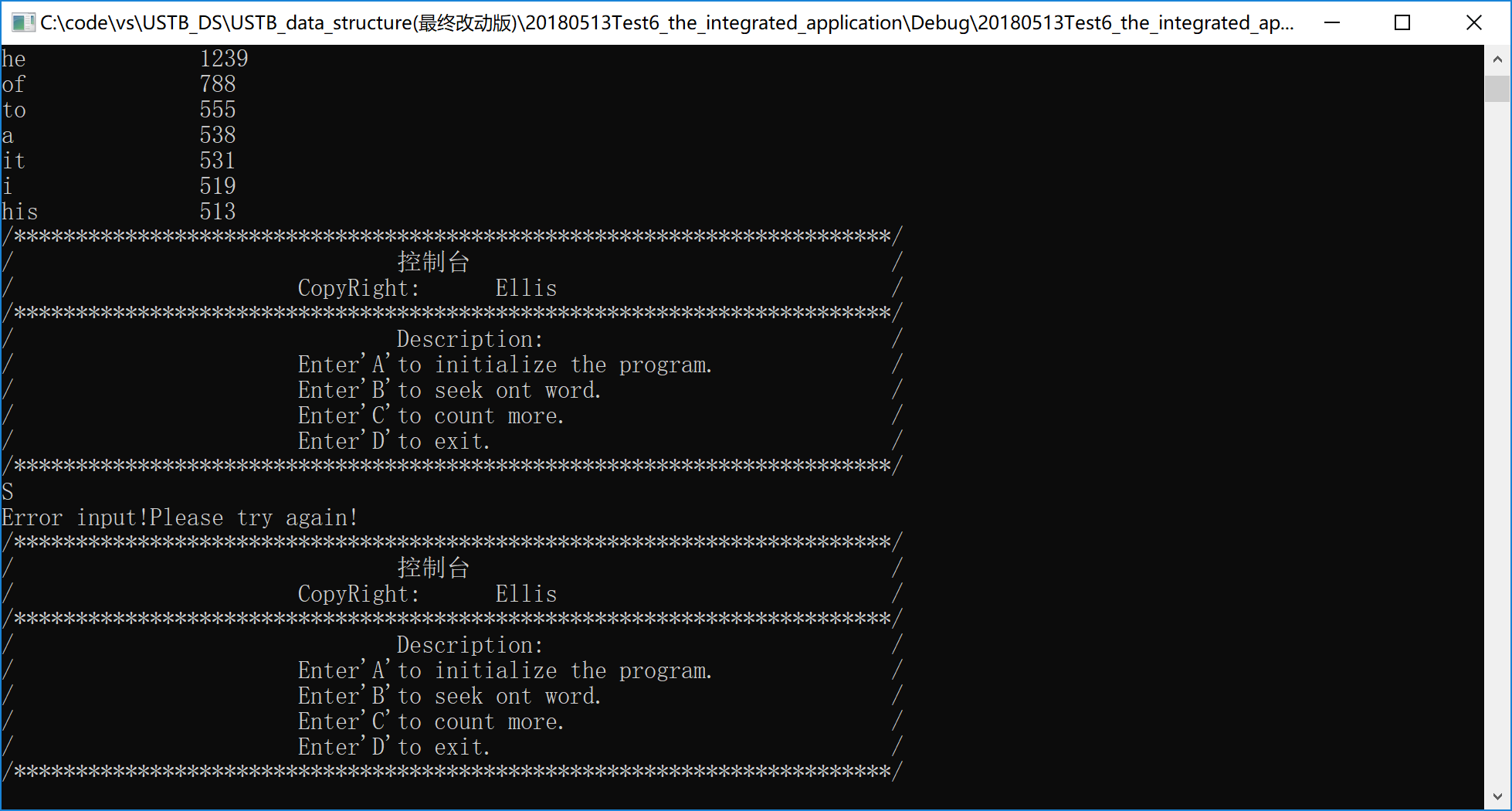
测试追加统计：



测试退出：



测试错误输入：



**算法的有效性分析：**

时间空间复杂度：hash时间复杂度O(log2n),快排时间复杂度O(n^2)，总体复杂度O(n^2).空间复杂度合适

有效性：算法正确，数据结构良好，容易阅读，测试完整且正确

不足：排序时用快排太慢了，要把所有的都遍历一遍，应该选用堆排序