题目：任给一个英文资料的文本文件，要求编程完成以下功能:

统计该文本文件的所有单词(每个单词不包括空格及跨行，单词由字母序列构成且不区分大小写），

并按字典顺序排列存储在一个文件里，

统计每个单词在文本文件中出现的总次数、频度（该单词出现的总次数/所有单词出现次数之和）、

在文本文件中首次出现的行号及位置。

代码：

# define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct word

{

char s[30];//单词

int sum;//总次数

float p;//频度

int l;//行号

int n;//位置

}word;

int main()

{

word a[300];//文件最多300词

int i = 0;//单词序数

int j = 0;//每个单词的位数

for (i = 0; i < 300; i++)//初始化

{

for (j = 0; j < 30; j++)

{

a[i].s[j] = '\0';

a[i].sum = 1;//后续加和次数用

}

}

i = 0;

j = 0;

int line = 1;//行数

int n = 1;//一行中的位置

char c = 1;//初始化

FILE\* fp = fopen("input.txt", "r");

while (!feof(fp))

{

c = fgetc(fp);//读一个字符

if ((c >= 65 && c <= 90) || (c >= 97 && c <= 122))//读到英文字母

{

if (c >= 65 && c <= 90)

{

c = c + 32;//大写转小写

}

a[i].s[j] = c;

j++;

}

else//读到非英文字符，往前再读一个字符，避免有两个及更多非英文字符

{

//再往前读一个字符

do {

c = fgetc(fp);

if (c == -1)//-1代表读到文件尾部，则跳出循环

{

break;

}

else if ((c >= 65 && c <= 90) || (c >= 97 && c <= 122))

{

fseek(fp, -1, SEEK\_CUR);//读到英文字符就回退一个，不然下一单词就少一个首字母

break;

}

else if (c == '\n')//读到回车进行赋值

{

a[i].s[j] = '\0';

a[i].l = line;

a[i].n = n;

line++;

n = 1;

break;

}

} while (!(c >= 65 && c <= 90) || (c >= 97 && c <= 122));

if (c != '\n')

{

a[i].s[j] = '\0';

a[i].l = line;

a[i].n = n;

n++;

}

j = 0;

for (int t = 0; t < i; t++)//统计该单词出现次数

{

if (strcmp(a[i].s, a[t].s) == 0)

{

if (i > t)//留最先出现的位置

{

a[i].l = a[t].l;

a[i].n = a[t].n;

}

a[i].sum += a[t].sum;

a[t].sum = 0;//把前面的sum赋值零，便于后面输出

}

}

i++;

}

}

fclose(fp);

//计算频度

printf("%d ", i);

for (int k = 0; k < i; k++)

{

a[k].p = a[k].sum / (float)i;

}

//排序

word d;

for (int k = 0; k < i; k++)

{

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

{

if (strcmp(a[j].s, a[j + 1].s) > 0)

{

d = a[j];

a[j] = a[j + 1];

a[j + 1] = d;

}

}

}

//输出

fp = fopen("new.txt", "w+");

fprintf(fp, "%d\n", i);

for (int t = 0; t < i; t++)

{

if (a[t].sum)//前边sum置零的判断条件

{

fprintf(fp, "%-12s ", a[t].s);

fprintf(fp, "出现次数%d ", a[t].sum);

fprintf(fp, "第%d行 ", a[t].l);

fprintf(fp, "第%d个 ", a[t].n);

fprintf(fp, "频度%f \n", a[t].p);

}

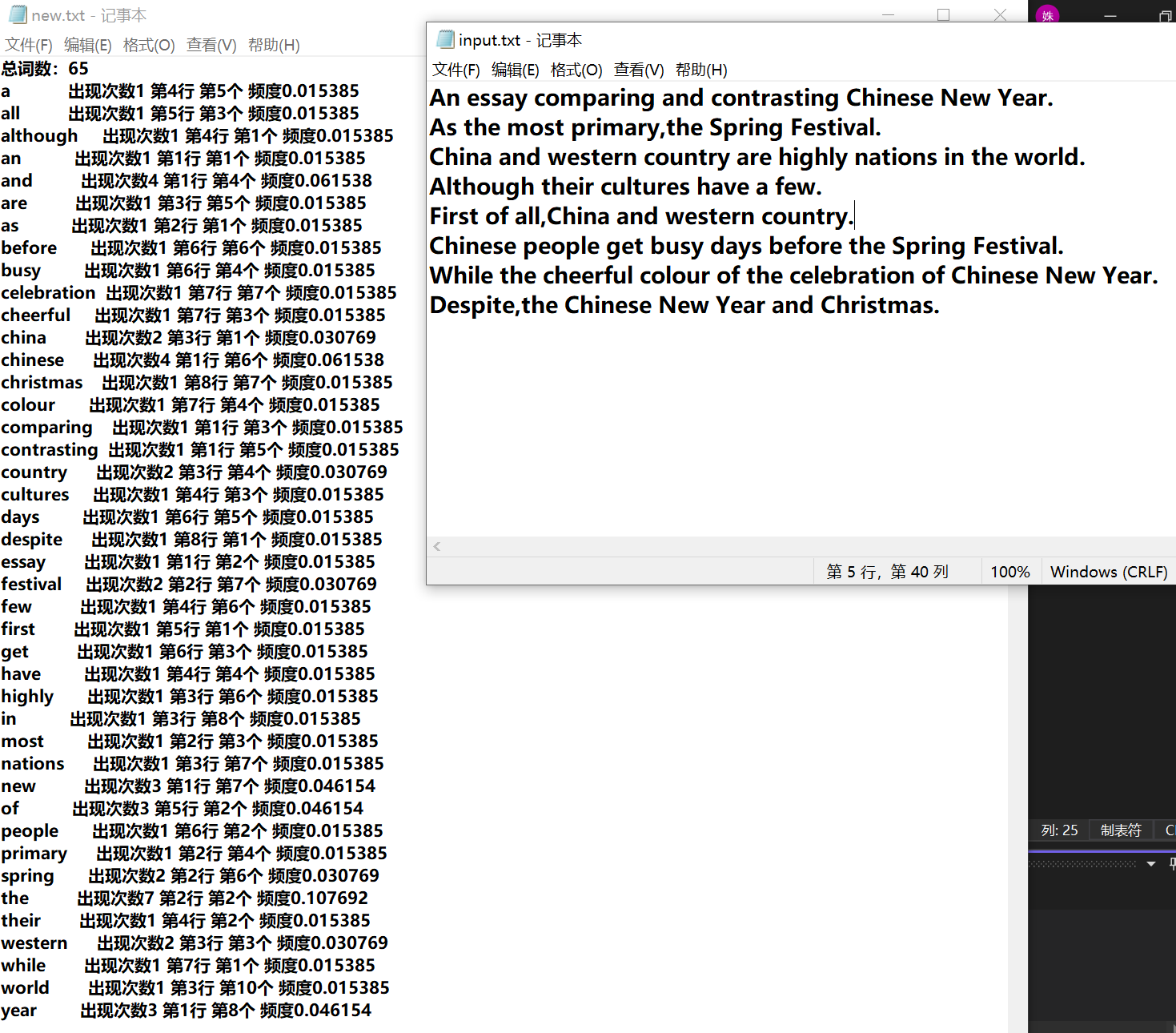
}

fclose(fp);

return 0;

}

运行结果：



总结：单词总个数统计：在下文提到的单词出现频率统计的过程结束之后，将每个单词出现的次数相加就是单词的总个数。

单词之间有多个空格。在进行判断时，若空格之后仍然是空格，则多个空格作为一个空格进行处理。

单词出现频率统计：逐行扫描文本，将不同的单词放在一个字符数组中。在扫描过程中，每遇到一个与已保存在数组中的某一个单词相同的单词的时候，该单词的计数加1。

按字典顺序排序：将数组中的每个元素（即单词）采用冒泡法排序，最后输出详细的单词出现频率。