

Lab1 统计文章中单词的数量

李仁杰 13307130279

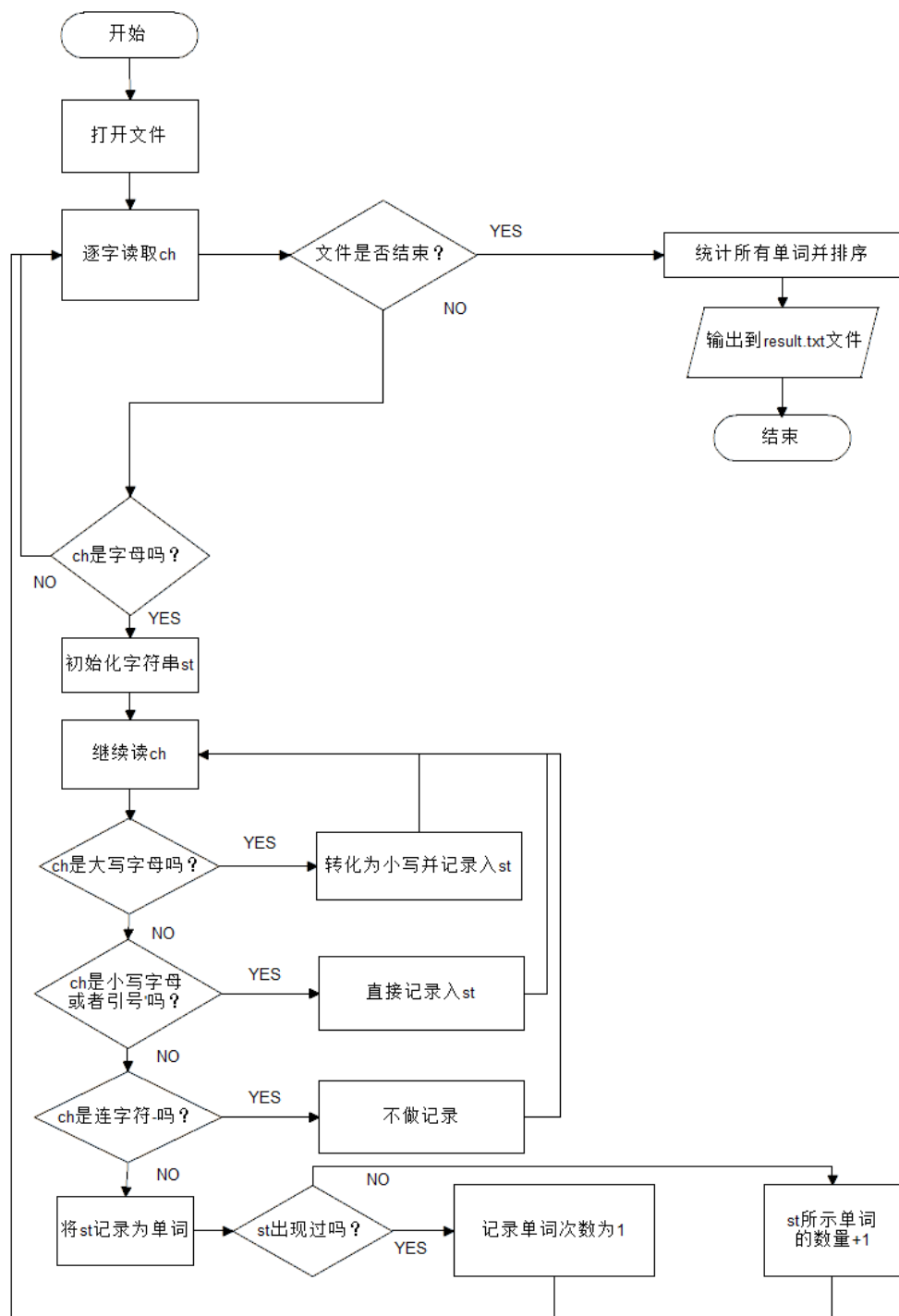
一、实验目的

1. 温习 C 语言的使用；
2. 练习数据结构和算法。

二、实验内容

统计文档“UnsortedFile.txt”中的单词，并按照出现的次数进行排序，出现次数相同的，字典序小的在前。输出到结果文件“SortedFile.txt”。

三、程序的框图



上面的程序框图是由 Diagram Designer 软件绘制。

四、程序的改进

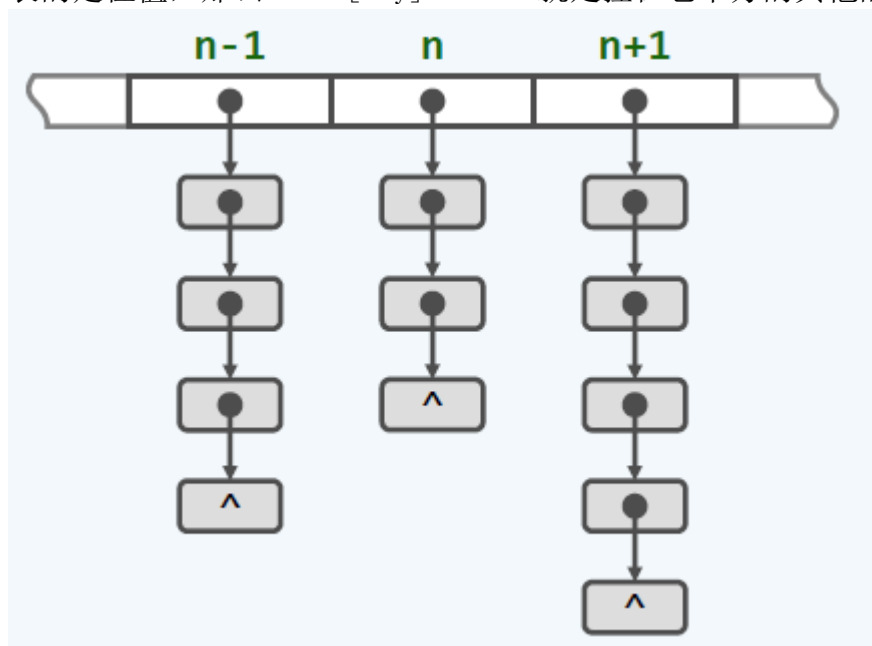
1. 单词的查重

当出现了一个新单词，如果用顺序查找的方式检查单词是否出现，那么一次查找最坏的情况就是 $O(n)$ ， n 为单词的数量。对于 n 个单词，时间复杂度为 $O(n^2)$ ，随着单词的数量增加，消耗的时间会非常巨大。于是这里引入哈希表进行查重。

哈希表的结构为：

```
struct List
{
    char str[length];
    int flag;
    struct List *next;
}; // List is a hashtable element.
```

其中 str 是字符串（单词），flag 是单词出现的次数，如果让 key 为一串哈希表的定位值，那么 $\text{hash}[\text{key}] \rightarrow \text{next}$ 就是挂在它下方的其他的链。如下图所示：



（图片来自清华大学邓俊辉的数据结构 2014 年春的教学课件）

不同的单词可能会有相同的 key 值，这里采用挂链的方式解决冲突。

对于 key 值的计算，我们采取随机数的办法，引入 sj 数组帮助计算：

```
srand(time(NULL));
for(i = 0; i < length; i++)
    sj[i] = rand() % MOD;
// sj[] is an array that can help to calculate the key value.
```

之后，对于一个长度为 len 的字符串 st，它的 key 值计算如下：

```
for(i = 0; i < len; i++)
    key = (key + st[i] - 'a') * sj[i] % MOD;
// It means that sum st[i]*sj[i] is the key value.
```

使用哈希表之后程序的查重复杂度的平均值大概为 $O(n/\text{MOD})$ ，其中 MOD 为哈希表的长度。

2. 单词的排序

如果采取冒泡排序、选择排序或者插入排序，那么程序的时间复杂度可能会达到 $O(n^2)$ ，其中 n 为单词的数量。为了提高这部分的效率，我们引入快速排序。

我们取单词出现的次数为第一关键字，字典序为第二关键字，进行快速排序 Qsort，时间复杂度便可下降到 $O(n\log_2 n)$ 。

```
void Qsort(int l,int r)
```

3. 程序运行时间的记录

首先我们加载 <time.h> 库，在程序开始的时候调用：

```
clock_t start, finish;  
double during;  
start = clock();
```

之后在程序结束之后调用：

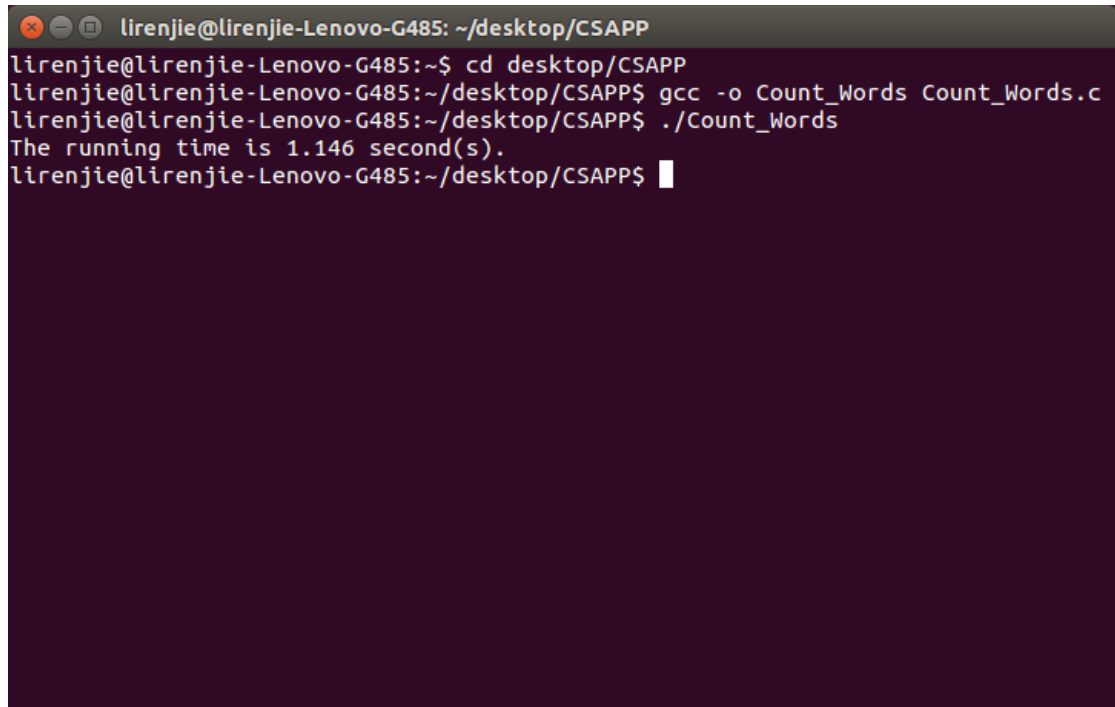
```
finish = clock();  
during = (double)(finish - start) / CLOCKS_PER_SEC;
```

即可得到程序的运行时间。

五、程序运行结果

运行结果在附件中的“result.txt”文件中。

运行总时间为：1.146 second(s)。（Linux Ubuntu 14.04 系统）



```
lirenjie@lirenjie-Lenovo-G485: ~/desktop/CSAPP  
lirenjie@lirenjie-Lenovo-G485:~$ cd desktop/CSAPP  
lirenjie@lirenjie-Lenovo-G485:~/desktop/CSAPP$ gcc -o Count_Words Count_Words.c  
lirenjie@lirenjie-Lenovo-G485:~/desktop/CSAPP$ ./Count_Words  
The running time is 1.146 second(s).  
lirenjie@lirenjie-Lenovo-G485:~/desktop/CSAPP$
```

六、实验总结

首先，在这次实验中，我温习了 C 语言的字符串、链表和排序的知识；其次，因为程序需要多次改进，为了记录程序的每一个版本，我学习了 github 的使用——这也使得我的效率显著提高，我的代码文件

“Count_Words.c” 也会在 Lab1 截止时间之后开源

（<https://github.com/lirenjie95/CSAPP>）；最后，我也在实验中基本掌握了 Linux 系统的使用。

七、源代码

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <time.h>  
/*-----*/  
#define MOD 900001  
#define MAXN 300000
```

```

#define length 80
struct List
{
    char str[length];
    int flag;
    struct List *next;
}; // List is a hashtable element.
struct List *hash[MOD];
char s[MAXN][length];
int sj[length], Num[MAXN];
FILE *fp;
/*-----*/
void Qsort(int l, int r)
{
    int i, j, t, u; char x[length], y[length];
    i = l; j = r; u = Num[(l+r)>>1];
    memset(x, 0, length); strcpy(x, s[(l+r)>>1]);
    while(i <= j)
    {
        while(Num[i] < u || (Num[i] == u && strcmp(x, s[i]) > 0)) i++;
        while(Num[j] > u || (Num[j] == u && strcmp(x, s[j]) < 0)) j--;
        if(i <= j)
        {
            t = Num[i]; Num[i] = Num[j]; Num[j] = t;
            strcpy(y, s[i]); strcpy(s[i], s[j]); strcpy(s[j], y);
            i++; j--;
        }
    }
    if(i < r) Qsort(i, r);
    if(l < j) Qsort(l, j);
}
/*-----*/
int main()
{
    int i, len, tot, key;
    struct List *u, *p;
    char ch, st[length];
    for(i = 0; i < MOD; i++)
    {
        hash[i] = (struct List *) malloc(sizeof(struct List));
        hash[i]->str[0] = '\0';
        hash[i]->flag = 0;
        hash[i]->next = NULL;
    } //memset
    tot = 0;
    srand(time(NULL));
    for(i = 0; i < length; i++)
        sj[i] = rand() % MOD;
    // sj[] is an array that can help to calculate the key value.
    fp = fopen("UnsortedFile.txt", "r");
    while(!feof(fp))
    {
        memset(st, 0, length);
        ch = getc(fp);
        if((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) // A new
word.

```

```

{
    len=0;

    while((ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z')||(ch=='\')||(ch=='-'
'))
    {
        if(ch >= 'A'&&ch <= 'Z')
            ch = ch+'a'-'A';
        if(ch == '-')
        {
            ch = getc(fp);
            continue;
        }
        st[len++] = ch;
        ch = getc(fp);
    }
    st[len] = '\0';
    while(st[len-1] == '\'){st[len]=0;len--;st[len]='\0';}
}
key = 0;
for(i = 0;i < len;i++)
    key = (key + st[i] - 'a') * sj[i] % MOD;
    // It means that sum st[i]*sj[i] is the key value.
key = (key + MOD) % MOD;
if(strlen(hash[key]->str) == 0)// hash[key] does not exist.
{
    strcpy(hash[key]->str,st);
    hash[key]->flag = 1;
    continue;
}
u=hash[key];
while(u)// hash[key] exists.
{
    if(strcmp(u->str,st) == 0)
    {
        u->flag++;
        break;
    }
    p = u;
    u = u->next;
}
// Finding a place to put the new hashtable.
if(!u)
{
    u = (struct List *)malloc(sizeof(struct List));
    u->next = NULL;
    strcpy(u->str,st);
    u->flag = 1;
    p->next = u;
}
}
fclose(fp);
for(i = 0;i < MOD;i++)
    if(strlen(hash[i]->str) != 0)
    {
        u = hash[i];
    }
}

```

```
        while(u)
        {
            tot++;
            strcpy(s[tot],u->str);
            Num[tot] = u->flag;
            u = u->next;
        }
    }
    Qsort(1,tot);
    fp = fopen("SortedFile.txt","w");
    for(i = 1;i <= tot;i++)
        fprintf(fp,"%s %d\n",s[i],Num[i]);
    fclose(fp);
    return 0;
}
```