#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define LEN sizeof(struct myword)

struct myword

{

int num;

char \*word;

struct myword \*next; /\* 指向下一个节点 \*/

};

void main()

{

/\*读取infile\_name文件，用find\_words找到单词，存储到head链表，最后将链表输出到文件\*/

void print\_link\_to\_file(struct myword \*head); //输出链表到文件

struct myword \*find\_words(FILE \*input); //找单词，输出单词链表

char infile\_name[]={"word.txt"};

struct myword \*head;

FILE \*input;

input = fopen(infile\_name, "r"); //打开文件

if (input == NULL)

{

printf("打开文件%s失败\n",infile\_name);

exit(0);

}

head = find\_words(input); //找单词

fclose(input); //关闭文件

print\_link\_to\_file(head); //将链表输出到文件

}

void print\_link\_to\_file(struct myword \*head)

{

/\*输入链表，将链表内容输出到input.txt\*/

printf("NOW,print LINK......\n");

struct myword \*p;

FILE \*output;

char outfile\_name[]={"output.txt"};

output = fopen(outfile\_name, "w");

if (output == NULL)

{

printf("打开文件%s失败\n",outfile\_name);

exit(0);

}

p = head->next;

do

{

fprintf(output,"%s,%d\n",p->word,p->num);

printf("%s,%d\n",p->word,p->num);

p = p->next;

}while(p!=NULL);

fclose(output);

printf("Done!");

}

struct myword \* new\_word(char word[])

{

/\*输入一个单词数组，返回一个myword结构体\*/

struct myword \*tmp;

char \*t\_word;

tmp=(struct myword \*) malloc(LEN); //给一个新的节点申请空间

//动态申请一个空间来保存myword->word，因为这本身是个指针，需要指向一个数组来保存

t\_word = (char \*) malloc(strlen(word));

strcpy(t\_word,strlwr(word));

tmp->word = t\_word;

tmp->num = 1;

tmp->next=NULL;

printf(tmp->word);

printf("\n");

return tmp;

}

struct myword \*instert\_word(struct myword \*tmp,struct myword \*head)

{

/\*将新的tmp节点插入到head对应的位置

->遍历所有节点

# head空 放到第一个节点

# 与当前节点相同 当前节点的num+1

# 比当前节点大 如果比下一个节点小，则插在当前节点后

如果比下一个节点大 ，则遍历下一个节点

# 比第一个节点小 放在第一个节点前，头结点后

【因为链表是有序的，且在上一情况中处理，

在后面不存在比当前节点小的情况】

\*/

struct myword \*cur;

if (head==NULL)

{

head = (struct myword \*) malloc(LEN);

head->next = tmp;

}

else

{

cur = head->next;

while (cur!=NULL)

{

if (strcmp(tmp->word,cur->word)<0)

{

//小于第一个值 ,因为链表有序，如果比第一个小，那就是最小的

tmp->next = cur;

head->next = tmp;

break;

}

if (strcmp(tmp->word,cur->word)==0)

{

//等于当前值

cur->num++;

free(tmp); //这个节点已经存在，释放tmp空间

break;

}

else if (strcmp(tmp->word,cur->word)>0)

{

//大于当前值

if (cur->next!=NULL)// && (tmp->word,cur->next->word)<0

{

//后面不为空

if (strcmp(tmp->word,cur->next->word)<0)

{

//小于下一个值 ,插入当前节点之后

tmp->next = cur->next;

cur->next = tmp;

break;

}

//大于等于下一个值

cur = cur->next;

}

else

{

//后面为空

cur->next= tmp;

break;

}

}

}

}

return head;

}

struct myword \*find\_words(FILE \*input)

{

/\*输入文件指针，找到单词，生成链表，输出链表头head，

从文件中一个字符一个字符的读取，如果遇到字母则将其存入

字符数组中，直到遇到非字母，

【在数组的后面加入'\0',以此来分割数组中的

单词，因为前一个单词可能比当前单词长】

则数组中的就是一个单词，

生成节点，并插入相应位置。

用字符数组保存当前单词\*/

struct myword \*head, \*tmp;

struct myword \*instert\_word(struct myword \*tmp,struct myword \*head); //插入节点

struct myword \*new\_word(char word[]); //生成一个新的节点，返回指针

char ch;

char word[25];

int i=0;

head = NULL;

while (! feof(input))

{

ch = fgetc(input);

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

{

word[i]=ch;

i++;

}

else

{

word[i]='\0';

if (strlen(word) !=0)

{

tmp = new\_word(word);

head = instert\_word(tmp,head);

}

i=0;

}

}

return head;

}