中南大学

数据结构试验报告

题 目 实验三 串及其应用

学生姓名 张子洋

学 号 8208221223

指导老师 郑瑾

学 院 计算机学院

专业班级 计算机与通信类2212班

完成时间 2023.5

指导老师评定 签名

1.文学研究助手（设计性实验）

1. 需求分析

需求：

英文小说存于一个文本文件中。待统计的词汇集合要一次输入完毕，即统计工作必须在程序的一次运行之后就全部完成。程序的输出结果是每个词的出现次数和出现位置的行号，格式自行设计。

分析：

1. 输入的形式和输入值的范围：将英语文章按屏幕提示放入文件中，输入查找个数为int型数字，输入英文单词为字符串形式。
2. 输出的形式：输出int型数字，放在文件“统计词汇的结果.txt”中。
3. 程序所能达到的功能：统计要查找的单词在文件中的个数。
4. 测试数据：

* 以程序源代码作为英语文章放入文件中
* 查找3个单词：include while break

1. 概要设计
2. 为了实现程序功能，需要定义串的抽象数据类型。

ADT String {

数据对象：D＝{ ai |ai∈CharacterSet, i=1,2,...,n, n≥0 }

数据关系：R1＝{ < ai-1, ai > | ai-1, ai ∈D, i=2,...,n }

typedef struct WORD{

char letter;

struct WORD \*next;

}word; //单词节点

typedef struct{

word \*first;

word \*last;

}row; //行结构节点

typedef struct Row{

row r;

struct ROW \*next;

}ROW; //行结构

typedef struct{

ROW \*first;

ROW \*last;

}article; //文章结构

基本操作：

StrAssign (&T, chars)

初始条件：chars 是字符串常量。

操作结果：把 chars 赋为 T 的值。

StrCopy (&T, S)

初始条件：串 S 已存在。

操作结果：由串 S 复制得到串 T。

DestroyString (&S)

初始条件：串 S 已存在。

操作结果：串 S 被销毁。

StrEmpty (S)  
 初始条件：串S已存在。  
 操作结果：若 S 为空串，则返回 TRUE，否则返回 FALSE。

StrLength (S)  
 初始条件：串 S 已存在。  
 操作结果：返回 S 的元素个数，称为串的长度。

StrCompare (S, T)  
 初始条件：串 S 和 T 已存在。  
 操作结果：若S > T，则返回值> 0；   
 若S = T，则返回值=0；   
 若S < T，则返回值<0。

Concat (&T, S1, S2)  
 初始条件：串 S1 和 S2 已存在。  
 操作结果：用 T 返回由 S1 和 S2  
 联接而成的新串。

SubString (&Sub, S, pos, len)

初始条件：串S 已存在，1<=pos<=StrLength(S)

且 0<=len<=StrLength(S)-pos+1。

操作结果： Sub 返回串 S 的第 pos 个字符起长度为 len 的子串。

Index (S, T, pos)  
 初始条件：串S和T已存在，T是非空串，  
 1≤pos≤StrLength(S)。  
 操作结果：若主串 S 中存在和串 T 值相同的子串, 则返回它在主串 S

中第pos个字符之后第一次出现的位置；否则函数值为0。

Replace (&S, T, V)  
 初始条件：串S, T和 V 均已存在，且 T 是非空串。  
 操作结果：用 V 替换主串 S 中出现的所有与（模式串）T   
 相等的不重叠的子串。

StrInsert (&S, pos, T)  
 初始条件：串S和T已存在，1≤pos≤StrLength(S)＋1。  
 操作结果：在串S的第pos个字符之前插入串T。

StrDelete (&S, pos, len)

初始条件：串S已存在，1≤pos≤StrLength(S)-len+1。

操作结果：从串S中删除第pos个字符起长度为len的子串。

ClearString (&S)  
 初始条件：串S已存在。  
 操作结果：将S清为空串。  
}ADT String

1. 本程序包含函数：

int\* get\_next(char \*p,int \*next){ //求next数组

int length(char \*p) //求字符串长度

int Index(char \*S,char \*P,int pos,int \*next) //KMP算法

1. 详细设计

文学研究助手.c：

1. #include <sys/types.h>
2. #include <sys/stat.h>
3. #include <fcntl.h>
4. #include <unistd.h>
5. #include <stdlib.h>
6. #include <stdio.h>
7. #include <string.h>
8. #include <ctype.h>
9. int keynum;//关键词数量
10. char \*readbuf;//读入缓冲区（用来读写所存的文件）
11. int hang,lie;//确定行和列
12. int length(char \*p){ //求字符串长度
13. int i=1;
14. while(p[i]){
15. i++;
16. }
17. return i-1;
18. }
19. int\* get\_next(char \*p,int \*next){ //求next数组
20. int i,k;
21. i=1,k=0;
22. next[1]=0;
23. while(i<length(p)){
24. if(k==0||p[i]==p[k]){
25. ++i;
26. ++k;
27. if(p[i]!=p[k]){
28. next[i]=k;}
29. else{
30. next[i] = next[k];
31. }
32. }
33. else{
34. k=next[k];
35. }
36. }
37. return next;
38. }
39. int Index(char \*S,char \*P,int pos,int \*next) //KMP算法
40. {
41. int i=pos,j=1;
42. while(i<=length(S)&&j<=length(P))
43. {
44. if(j==0||S[i]==P[j])
45. {
46. ++i;
47. ++j;
48. }
49. else
50. {
51. j=next[j];
52. }
53. }
54. if(j>length(P))
55. {
56. return (i-length(P));
57. }
58. else
59. {
60. return 0;
61. }
62. }
63. int main()
64. {
65. int i,j;
66. char \*\*p;
67. int \*\*next;
68. int \*\*pnext;
69. printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*欢迎使用文学助手系统\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");
70. while(1){
71. printf("请输入你需要查询的关键词数量\n");
72. scanf("%d",&keynum);
73. p = (char\*\*)malloc(sizeof(char\*)\*keynum);//动态开辟二维数组 用来保存字符串
74. next = (int\*\*) malloc(sizeof(int\*)\*keynum);//动态开辟二维数组 用来保存每个字符串的next数组
75. pnext = (int\*\*) malloc(sizeof(int\*)\*keynum);//用来承接函数返回值
76. for(i=0;i<keynum;i++){
77. next[i] = (int\*)malloc(sizeof(int)\*20);
78. }
79. for(i=0;i<keynum;i++){
80. pnext[i] = (int\*)malloc(sizeof(int)\*20);
81. }
82. for(i=0;i<keynum;i++){
83. p[i] = (char \*)malloc(sizeof(char)\*20);
84. }
85. printf("请输入你需要查询的关键词\n");
86. for(i=0;i<keynum;i++){
87. scanf("%s",&p[i][1]); //输入字符串
88. }
89. printf("你所查询的关键词如下:\n");
90. for(i=0;i<keynum;i++){
91. printf("%s",&p[i][1]); //查询字符串
92. printf("\n");
93. }
94. for(i=0;i<keynum;i++){
95. pnext[i] = get\_next(p[i],next[i]);
96. printf("%s的next数组为：",&p[i][1]);
97. for(j=1;j<strlen(p[i]);j++){
98. printf("%d ",pnext[i][j]);
99. }
100. printf("\n");
101. }
102. readbuf = (char\*)malloc(sizeof(char)\*1024);
103. for(i=0;i<keynum;i++){
104. int row=0,col =0;
105. int k=0;
106. int count = 0;
107. FILE \*fp; //创建一个文件
108. fp=fopen("1.txt","r");
109. if(fp==NULL){
110. printf("open file\n");
111. }
112. while(!feof(fp)){
113. fgets(readbuf,1024,fp);
114. row++;
115. col = Index(readbuf,p[i],0,next[i]); //返回文件的列
116. if(col!=0){
117. printf("%s的行为%d,列为%d",&p[i][1],row,col+1);
118. k++;
119. }
120. while(col!=0){
121. col = Index(readbuf,p[i],col+1,next[i]);
122. if (col!=0)
123. {
124. k++;
125. printf(",%d",col+1);//若匹配成功，则打印列号
126. }else{
127. printf("\n");
128. }
129. }
130. }
131. count=count+k;
132. // if (k)
133. // {
134. // printf("\n"); //输出格式控制
135. // }
136. fclose(fp);
137. if(count){
138. printf("%s共出现%d次\n",&p[i][1],count);
139. printf("\n");
140. }
141. else{
142. printf("No Found\n");
143. printf("\n");
144. }
145. }
146. }

}

4.调试分析

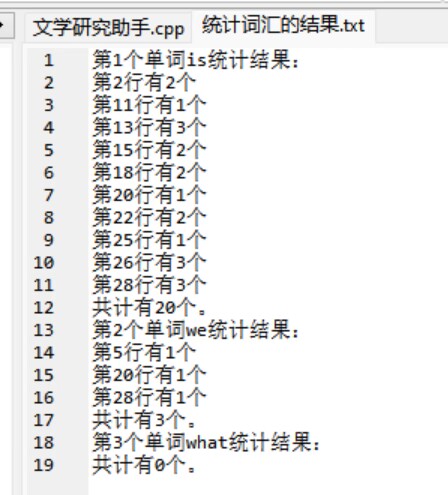
1. 采用IDE中自带的调试功能进行调试，手动添加断点和查看程序。
2. 对设计和编码的讨论和分析。该程序实现了顺序栈的操作。分析程序代码的质量，主要从以下几个方面考虑。 

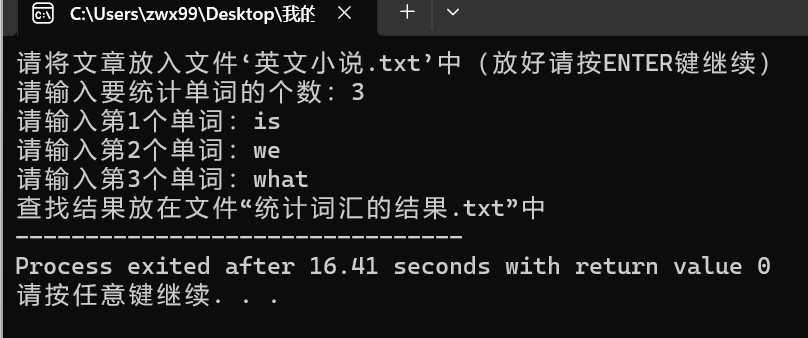
* 正确性。在一定的数据范围内，该程序能实现所需功能，所以正确性是没有问题的。
* 健壮性。在一定的数据输入范围内，该程序能较好的实现链表的操作。但是如果输入数 据非法，该程序还是可能会产生一些预想不到的输出结构，或是不做任何处理。所以， 该程序的健壮性有待进一步的提高。要综合考虑一些情况，当输入有误时，应返回一个 表示错误的值，并中止程序的执行，以便在更高的抽象层次上进行处理。

1. 使用说明

按照屏幕提示，将英语文章放入对应文件，按ENTER键继续，输入要查找单词的个数，依次输入要查找的单词，结果放在文件“统计词汇的结果.txt” 中。

1. 测试程序的运行结果





1. 心得体会

通过本次实验，我通过上课时老师的讲解，自己打出了kmp算法，虽然不如课本和老师给的代码简洁高效，但却使我对kmp算法有了更深的理解，原来上课时云里雾里的东西现在明白了许多，还是只有多打代码才能真正学好数据结构。

附录：源程序文件清单

各程序源代码文件随本实验报告电子版一起打包，存放在文件夹文学研究助手中。

文件清单如下：

文学研究助手.c…………………………………………文学研究助手的实现

文章结构链表.h…………………………………………文章结构链表与其基本操作的头文件

英文小说.txt………………………………………………存放输入数据的文件

统计词汇的结果.txt……………………………………存放程序运行结构数据的文件

2.简单行编辑程序（综合性实验）

1. 需求分析

需求：

文本编辑程序是利用计算机进行文字加工的基本软件工具，实现对文本文件的插入、删除等修改操作。限制这些操作以行为单位进行的编辑程序称为行编辑程序。 被编辑的文本文件可能很大，全部读入编辑程序的数据空间（内存）的做法既不经济也不总能实

现。一种解决方法是逐段地编辑，任何时刻只把待编辑文件的一段放在内存，称为活区。试按照这种 方法实现一个简单的行编辑程序。设文件每行不超过320个字符，并且很少超过80字符。

基本要求

实现以下4条基本编辑命令。

(1) 行插入。格式：i<行号><回车><文本><回车>。将<文本＞插入活区中第<行号＞行之后。

(2) 行删除。格式：d<行号1>[□ <行号2>]<回车>。删除活区中第<行号1＞行（到第<行号2＞行）。

两种格式的例子如“d10↙”和“d10□ 14↙”。

(3) 活区切换。格式：n<回车>。将活区写入输出文件，并从输入文件中读入下一段，作为新的活

区。

1. 活区显示。格式：p<回车>。逐页地（每页20行）显示活区内容，每显示一页之后请用户决定是否继续显示以后各页（如果存在）。印出的每一行要包含该行的行号和一个空格符，行号固定占4位， 增量为1。 各条命令中的行号均须在活区中各行行号范围之内，只有插入命令的行号可以等于活区第一行行号减1，表示插入当前屏幕中第一行之前，否则命令参数非法。

实现提示 (1) 设活区的大小用行数activemaxlen(可设为100)来描述.考虑到文本文件行长通常为正态分 布,且峰值为60～70,用320×activemaxlen大小的字符数组实现存储将造成大量浪费.可以以标准行 块为单位为各行分配存储,每个标准行块含81个字符.这些行块可以组成一个数组,也可以利用动态 链表连接起来.一行文字可能占多个行块.行尾可用一个特殊的ASCII字符(如012)标识.此外,还 应记住活区起始行号,因为行插入将引起随后各行行号的顺序下推.

1. 初始化过程包括请用户提供输入文件名(空串表示无输入文件)和输出文件名,两者不 能相同.然后尽可能多地从输入文件中读入各行,但不超过activemaxlen−x.x的值可以自定, 例如20.
2. 在执行行插入命令的过程中,每接收到一行时要检查活区大小是否已达到activemaxlen.如 果是,则为了在插入这一行之后仍保持活区大小不超过activemaxlen,应将插入点之前活区部分的第 一行输出到输出文件中.若插入点为第一行之前,则只得将新插入的这一行输出. (4) 若输入文件尚未读完,活区切换命令可将原活区中最后几行留在活区顶部,以保持阅读连续 性；否则,它意味着结束编辑或开始编辑另一个文件.

(5) 可令前3条命令执行后自动调用活区显示

1. 概要设计
2. 为了实现程序功能，需要定义串的抽象数据类型。

ADT String {

数据对象：D＝{ ai |ai∈CharacterSet, i=1,2,...,n, n≥0 }

数据关系：R1＝{ < ai-1, ai > | ai-1, ai ∈D, i=2,...,n }

typedef struct WORD{

char letter;

struct WORD \*next;

}word; //单词节点

typedef struct Word{

char \*ch;

struct Word \*next;

}word; //行字符串

typedef struct Row{

word \*r;

int number;

struct ROW \*next;

}ROW; //行结构

typedef struct{

ROW \*first;

ROW \*last;

}paragraph; //段结构

基本操作：

StrAssign (&T, chars)

初始条件：chars 是字符串常量。

操作结果：把 chars 赋为 T 的值。

StrCopy (&T, S)

初始条件：串 S 已存在。

操作结果：由串 S 复制得到串 T。

DestroyString (&S)

初始条件：串 S 已存在。

操作结果：串 S 被销毁。

StrEmpty (S)  
 初始条件：串S已存在。  
 操作结果：若 S 为空串，则返回 TRUE，否则返回 FALSE。

StrLength (S)  
 初始条件：串 S 已存在。  
 操作结果：返回 S 的元素个数，称为串的长度。

StrCompare (S, T)  
 初始条件：串 S 和 T 已存在。  
 操作结果：若S > T，则返回值> 0；   
 若S = T，则返回值=0；   
 若S < T，则返回值<0。

Concat (&T, S1, S2)  
 初始条件：串 S1 和 S2 已存在。  
 操作结果：用 T 返回由 S1 和 S2  
 联接而成的新串。

SubString (&Sub, S, pos, len)

初始条件：串S 已存在，1<=pos<=StrLength(S)

且 0<=len<=StrLength(S)-pos+1。

操作结果： Sub 返回串 S 的第 pos 个字符起长度为 len 的子串。

Index (S, T, pos)  
 初始条件：串S和T已存在，T是非空串，  
 1≤pos≤StrLength(S)。  
 操作结果：若主串 S 中存在和串 T 值相同的子串, 则返回它在主串 S

中第pos个字符之后第一次出现的位置；否则函数值为0。

Replace (&S, T, V)  
 初始条件：串S, T和 V 均已存在，且 T 是非空串。  
 操作结果：用 V 替换主串 S 中出现的所有与（模式串）T   
 相等的不重叠的子串。

StrInsert (&S, pos, T)  
 初始条件：串S和T已存在，1≤pos≤StrLength(S)＋1。  
 操作结果：在串S的第pos个字符之前插入串T。

StrDelete (&S, pos, len)

初始条件：串S已存在，1≤pos≤StrLength(S)-len+1。

操作结果：从串S中删除第pos个字符起长度为len的子串。

ClearString (&S)  
 初始条件：串S已存在。  
 操作结果：将S清为空串。  
}ADT String

1. 本程序包含函数：

* 主函数main()；
* void insertLine(ActiveRegion \*activeRegion, int lineNum, char \*text)；;
* void deleteLines(ActiveRegion \*activeRegion, int startLine, int endLine)添加行void switchActiveRegion(ActiveRegion \*activeRegion, char \*outputFile)
* void displayActiveRegion(ActiveRegion \*activeRegion)

1. 详细设计

简单行编辑程序.c：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_LINE\_LENGTH 320

#define MAX\_LINES 100

#define PAGE\_SIZE 20

typedef struct {

char lines[MAX\_LINES][MAX\_LINE\_LENGTH];

int startLine;

int endLine;

} ActiveRegion;

void insertLine(ActiveRegion \*activeRegion, int lineNum, char \*text) {

if (lineNum > activeRegion->endLine || lineNum < activeRegion->startLine - 1) {

printf("Invalid line number!\n");

return;

}

if (activeRegion->endLine == MAX\_LINES - 1) {

printf("Active region is full! Cannot insert line.\n");

return;

}

int insertIndex = lineNum - activeRegion->startLine + 1;

memmove(activeRegion->lines[insertIndex + 1], activeRegion->lines[insertIndex],

sizeof(char) \* (MAX\_LINE\_LENGTH \* (activeRegion->endLine - lineNum + 1)));

strncpy(activeRegion->lines[insertIndex], text, MAX\_LINE\_LENGTH);

activeRegion->endLine++;

}

void deleteLines(ActiveRegion \*activeRegion, int startLine, int endLine) {

if (startLine > activeRegion->endLine || endLine < activeRegion->startLine) {

printf("Invalid line range!\n");

return;

}

int deleteStartIndex = startLine - activeRegion->startLine;

int deleteCount = endLine - startLine + 1;

memmove(activeRegion->lines[deleteStartIndex], activeRegion->lines[deleteStartIndex + deleteCount],

sizeof(char) \* (MAX\_LINE\_LENGTH \* (activeRegion->endLine - endLine)));

activeRegion->endLine -= deleteCount;

}

void switchActiveRegion(ActiveRegion \*activeRegion, char \*outputFile) {

FILE \*file = fopen(outputFile, "w");

if (file == NULL) {

printf("Failed to open output file!\n");

return;

}

for (int i = 0; i <= activeRegion->endLine; i++) {

fprintf(file, "%s\n", activeRegion->lines[i]);

}

fclose(file);

activeRegion->startLine = activeRegion->endLine + 1;

activeRegion->endLine = activeRegion->startLine - 1;

}

void displayActiveRegion(ActiveRegion \*activeRegion) {

int currentPage = 0;

int totalPages = (activeRegion->endLine - activeRegion->startLine + 1) / PAGE\_SIZE;

while (currentPage <= totalPages) {

printf("Page %d:\n", currentPage + 1);

for (int i = 0; i < PAGE\_SIZE; i++) {

int lineIndex = activeRegion->startLine + currentPage \* PAGE\_SIZE + i;

if (lineIndex <= activeRegion->endLine) {

printf("%4d %s\n", lineIndex, activeRegion->lines[lineIndex - activeRegion->startLine]);

}

}

char choice;

if (currentPage < totalPages) {

printf("Press 'Enter' to continue to the next page, or 'q' to quit displaying.\n");

} else {

printf("End of active region. Press 'q' to quit displaying.\n");

}

choice = getchar();

if (choice == 'q') {

break;

}

currentPage++;

}

}

int main() {

char inputFile[100], outputFile[100];

ActiveRegion activeRegion;

activeRegion.startLine = 0;

activeRegion.endLine = -1;

printf("Enter the input file name: ");

fgets(inputFile, sizeof(inputFile), stdin);

inputFile[strcspn(inputFile, "\n")] = '\0';

printf("Enter the output file name: ");

fgets(outputFile, sizeof(outputFile), stdin);

outputFile[strcspn(outputFile, "\n")] = '\0';

FILE \*file = fopen(inputFile, "r");

if (file == NULL) {

printf("Failed to open input file!\n");

return 1;

}

char line[MAX\_LINE\_LENGTH];

while (fgets(line, sizeof(line), file) && activeRegion.endLine < MAX\_LINES - 1) {

line[strcspn(line, "\n")] = '\0';

strncpy(activeRegion.lines[++activeRegion.endLine], line, MAX\_LINE\_LENGTH);

}

fclose(file);

char command;

do {

printf("\nEnter a command (i-insert, d-delete, n-switch, p-display, q-quit): ");

command = getchar();

getchar(); // Consume the newline character

switch (command) {

case 'i': {

int lineNum;

printf("Enter line number: ");

scanf("%d", &lineNum);

getchar(); // Consume the newline character

char text[MAX\_LINE\_LENGTH];

printf("Enter text to insert: ");

fgets(text, sizeof(text), stdin);

text[strcspn(text, "\n")] = '\0';

insertLine(&activeRegion, lineNum, text);

displayActiveRegion(&activeRegion);

break;

}

case 'd': {

int startLine, endLine;

printf("Enter start line number: ");

scanf("%d", &startLine);

getchar(); // Consume the newline character

printf("Enter end line number (optional): ");

if (scanf("%d", &endLine) != 1) {

endLine = startLine;

getchar(); // Consume the newline character

} else {

getchar(); // Consume the newline character

}

deleteLines(&activeRegion, startLine, endLine);

displayActiveRegion(&activeRegion);

break;

}

case 'n':

switchActiveRegion(&activeRegion, outputFile);

printf("Switched active region.\n");

break;

case 'p':

displayActiveRegion(&activeRegion);

break;

case 'q':

printf("Quitting...\n");

break;

default:

printf("Invalid command! Please try again.\n");

break;

}

} while (command != 'q');

return 0;

}

1. 调试分析
2. 采用IDE中自带的调试功能进行调试，手动添加断点和查看程序。
3. 对设计和编码的讨论和分析。该程序实现了城市链表的操作。分析程序代码的质量，主要从以下几个方面考虑。 

* 正确性。在一定的数据范围内，该程序能实现所需功能，所以正确性是没有问题的。
* 健壮性。在一定的数据输入范围内，该程序能较好的实现链表的操作。但是如果输入数 据非法，该程序还是可能会产生一些预想不到的输出结构，或是不做任何处理。所以， 该程序的健壮性有待进一步的提高。要综合考虑一些情况，当输入有误时，应返回一个 表示错误的值，并中止程序的执行，以便在更高的抽象层次上进行处理。

1. 使用说明

按照屏幕提示，将数据放入对应文件中，输入文件名，在输入输出文件名，根据屏幕提示进行输入，即可得到想要的结果。

1. 测试程序的运行结果



1. 心得体会

在实现简单行编辑的过程中，我们需要掌握一些基本的 C 语言语法及相关的函数库，如字符处理函数 ctype.h、字符串处理函数 string.h、文件操作函数 stdio.h 等等，同时我们还需要养成细心、耐心、严谨的编程思维和代码规范。

在编写简单行编辑器的过程中，最重要的挑战是如何实现多种编辑操作，如插入、删除、撤销等。我们需要用指针正确地指向文本中的位置，并进行各种操作的处理，结合简单行编辑的需求，我们需要在文字处理上深入思考和实践。通过这个实验，我深刻意识到程序设计的重要性，认识到程序设计的不仅仅是代码的实现，它的要点还在于算法设计的优化、数据结构的设计、注重程序的可读性、可维护性和可重用性等细节优化。同时，这个实验也让我更加深入地理解到文本编辑器的基本实现方法，同时也提高了我的代码设计能力。

附录：源程序文件清单

各程序源代码文件随本实验报告电子版一起打包，存放在文件夹简单行编辑程序中。

文件清单如下：

简单行编辑程序.c…………………………………………简单行编辑程序的实现

文章结构链表.h………………………………………文章结构链表及其基本操作的头文件