#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#define MAX 50

#define MAXSS 5

struct list

{

char \* str;

struct list \* next;

};

//typedef struct list list;

typedef struct

{

char \*\* buf;

int top;

int size;

}stack;

void MakeFile(char \* name, int max);

void ReadingFromFILE(char \* name, stack \* p, int max);

struct list \* ListReadingFromFILE(char \* name, struct list \* p, int max);

int \* MaxWordLen(char \* str);

int intstrlen(int \* s);

char \* sget(char \* str, int max, FILE \* stream);

char \*\* init(char \*\* s, int max);

void pr(char \*\*s, int m);

void reverse(char \* str, int l, int r);

void DelSubstr(char \* str, int l, int r);

void korstrforDel(int \* str, int n, int len);;

void swap(char \* p1, char \* p2);

stack initStack(stack \* p, int maxss, int max);

void push(stack \* p, char \* str);

char \* pop(stack \* p);

int IsFull(stack \*p);

int IsEmpty(stack \*p);

void printBuf(stack \* p);

char \* peek(stack \* p);

void structFree(stack \* p);

char FirstCh(char \* str);

void SimpleeeSort(char \*\* ss, int h);

void swapSTR(char \*\* s1, char \*\* s2);

void swapALL(void \* a, void \* b, int size);

struct list \* ListAdd(struct list \* p, char \* str);

void ListDel(struct list \* p);

void ListPrint(struct list \* p);

struct list \* Listinit(struct list \* n, int size);

int Listlen(struct list \* p);

struct list \* SimpleeeSort(struct list \* p, int h);

int main(void)

{

printf("FROM FILE----->");

//MakeFile("file.txt");

/\*stack st;

initStack(&st, MAXSS, MAX);

ReadingFromFILE("file.txt", &st, MAX);

printBuf(&st);\*/

list \* y = NULL;

y = ListReadingFromFILE("file.txt", y, MAX);

ListPrint(y);

y = SimpleeeSort(y, Listlen(y));

ListPrint(y);

ListDel(y);

/\*SimpleeeSort(st.buf, st.top);

puts("\n\t\tAFTER ~SORT~");

printBuf(&st);

putchar('\n');

puts("\t\tFROM <STACK>\n");

for (int i = st.top; i > 0; --i)

printf(" string (%d) -> %s\n", i, pop(&st));

structFree(&st);\*/

/\*stack x;

initStack(&x, 10, 10);

push(&x, "RK5-12B");

printBuf(&x);

push(&x, "SOPR");

puts("\n");

printBuf(&x);

push(&x, "TERMEXXX");

puts("\n");

printBuf(&x);

structFree(&x);\*/

/\*struct list \* y = NULL;

y = ListAdd(y, "SOPR");

ListPrint(y);

printf("\n ---> %d", Listlen(y));

y = ListAdd(y, "RK5 - 12B");

puts("\n");

ListPrint(y);

printf("\n ---> %d", Listlen(y));

y = ListAdd(y, "TEEEERMMMMMM");

puts("\n");

ListPrint(y);

printf("\n ---> %d", Listlen(y));

printf("\n\nSORTIROVOCCCHHHHKAAA\n");

y = SimpleeeSort(y, Listlen(y));

ListPrint(y);

ListDel(y);\*/

/\*char \* str1 = "Hello";

char \* str2 = "Freind";

printf(" 1 = %s 2 = %s\n\n", str1, str2);

swapALL(&str1, &str2, sizeof(char \*));

printf(" 1 = %s 2 = %s\n\n", str1, str2);\*/

getchar();

return 0;

}

void swapALL(void \* a, void \* b, int size)

{

void \* temp = malloc(size);

memcpy(temp, a, size);

memcpy(a, b, size);

memcpy(b, temp, size);

free(temp);

}

void ReadingFromFILE(char \* name, stack \* p, int max)

{

FILE \* fp = fopen(name, "r");

char \* str = (char \*)malloc(sizeof(char) \* max);

while (!feof(fp) && (str = sget(str, max, fp)) != NULL)

{

int \* s = MaxWordLen(str); // REVERS MAX WORD(S)

for (int i = 1; i < intstrlen(s); ++i)

reverse(str, s[i], s[i] + s[0] - 1);

free(s);

//int \* s = MaxWordLen(str); // DELETEEE MAX WORD(S)

//for (int i = 1; i < intstrlen(s); ++i)

//{

// DelSubstr(str, s[i], s[0] + s[i]);

// korstrforDel(s, s[0], intstrlen(s));

//}

if (!IsFull(p))

{

push(p, str);

}

else

{

printf("\n\nATTENTION!! SIZE OF STACK == %d\n", MAXSS);

break;

}

}

free(str);

fclose(fp);

}

struct list \* ListReadingFromFILE(char \* name, struct list \* p, int max)

{

FILE \* fp = fopen(name, "r");

char \* str = (char \*)malloc(sizeof(char) \* max);

while (!feof(fp) && (str = sget(str, max, fp)) != NULL)

{

int \* s = MaxWordLen(str); // REVERS MAX WORD(S)

for (int i = 1; i < intstrlen(s); ++i)

reverse(str, s[i], s[i] + s[0] - 1);

free(s);

//int \* s = MaxWordLen(str); // DELETEEE MAX WORD(S)

//for (int i = 1; i < intstrlen(s); ++i)

//{

// DelSubstr(str, s[i], s[0] + s[i]);

// korstrforDel(s, s[0], intstrlen(s));

//}

p = ListAdd(p, str); //LIST

}

free(str);

fclose(fp);

return p;

}

void pr(char \*\*s, int m)

{

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

{

printf("%s\n", \*(s + i));

}

putchar('\n');

}

char \* sget(char \* str, int max, FILE \* stream)

{

char \* p;

int i = 0;

p = fgets(str, max, stream);

if (p)

{

while (str[i] != '\n' && str[i] != '\0')

i++;

if (str[i] == '\n')

str[i] = '\0';

else

while (getchar() != '\n');

}

if (strlen(str) == 0)

return NULL;

return p;

}

void MakeFile(char \* name, int max)

{

FILE \* fp;

fp = fopen(name, "w");

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

{

putc('-', fp);

}

fclose(fp);

}

char \*\* init(char \*\* s, int max)

{

s = (char \*\*)malloc(sizeof(char \*) \* 10);

for (int i = 0; i < 10; ++i)

\*(s + i) = (char \*)malloc(sizeof(char) \* max);

return s;

}

int \* MaxWordLen(char \* str) //key == 1 -inword == 0 -outside

{

int \* s = NULL;

s = (int \*)malloc(sizeof(int) \* 25);

char strW[] = "{}[]()1234567890-=/\*-+.,@#$%^& ";

int i = 0, a = 0, b = 0, key = 0, c = 0;

while (str[i])

{

if (!strchr(strW, str[i]))

{

if (!key)

{

b = i;

key = 1;

}

if (strchr(strW, str[i + 1]))

{

if (a < i + 1 - b)

{

a = i + 1 - b;

c = 0;

s[c++] = a;

s[c++] = b;

}

else if (a == i + 1 - b)

{

s[c] = b;

c++;

}

key = 0;

}

}

i++;

}

s[c] = -100;

return s;

}

int intstrlen(int \* s)

{

int i;

for (i = 0; s[i] != -100; ++i);

return i;

}

void reverse(char \* str, int l, int r)

{

for (; l < r; ++l, --r)

swap(str + l, str + r);

}

void DelSubstr(char \* str, int l, int r)

{

int i;

for (i = 0; r + i < strlen(str); ++i)

{

str[l + i] = str[r + i];

}

str[l + i] = '\0';

}

void swap(char\* pa, char\* pb)

{

char temp = \*pa;

\*pa = \*pb;

\*pb = temp;

}

void push(stack \* p, char \* str)

{

if (!IsFull(p))

{

\*(p->buf + p->top) = (char \*)memcpy(\*(p->buf + p->top), str, sizeof(char) \* (strlen(str) + 1));

p->top++;

}

}

stack initStack(stack \* p, int maxss, int max)

{

p->buf = (char \*\*)malloc(sizeof(char \*) \* maxss);

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

{

\*(p->buf + i) = (char \*)malloc(sizeof(char) \* max);

}

p->top = 0;

p->size = maxss;

return \*p;

}

void structFree(stack \* p)

{

free(p->buf);

p->top = 0;

p->size = 0;

}

char \* pop(stack \* p)

{

if (IsEmpty(p))

return NULL;

return p->buf[--p->top];

}

char \* peek(stack \* p)

{

if (IsEmpty(p))

return NULL;

return p->buf[p->top - 1];

}

int IsFull(stack \*p)

{

if (p->top == p->size)

return 1;

return 0;

}

int IsEmpty(stack \*p)

{

if (p->top == 0)

return 1;

return 0;

}

void printBuf(stack \* p)

{

putchar('\n');

for (int i = 0; i < p->top; ++i)

{

printf(" %s", p->buf[i]);

putchar('\n');

}

}

void SimpleeeSort(char \*\* ss, int h)

{

for (int i = h - 1; i > 1; --i)

{

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

{

if (strcmp(\*(ss + j), \*(ss + j + 1)) > 0)

{

swapSTR(ss + j, ss + j + 1);

}

}

}

}

struct list \* SimpleeeSort(struct list \* p, int h)

{

struct list \* n = p;

for (int i = h - 1; i > 0; --i)

{

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

{

if (strcmp(n->str, n->next->str) > 0)

{

swapSTR(&(n->str), &(n->next->str));

}

n = n->next;

}

n = p;

}

return p;

}

char FirstCh(char \* str)

{

int i = 0;

while (str[i] && !isalpha(str[i]))

++i;

return tolower(str[i]);

}

void swapSTR(char \*\* s1, char \*\* s2)

{

char \* temp = \*s1;

\*s1 = \*s2;

\*s2 = temp;

}

void korstrforDel(int \* str, int n, int len)

{

for (int i = 1; i < len; ++i)

{

str[i] -= n;

}

}

struct list \* ListAdd(struct list \* p, char \* str)

{

struct list \* n = NULL;

n = Listinit(n, strlen(str));

n->str = (char \*)memcpy(n->str, str, sizeof(char) \* (strlen(str) + 1));

n->next = p;

return n;

}

void ListDel(struct list \* p)

{

struct list \* temp = (struct list \*) malloc(sizeof(struct list));

while (p)

{

struct list \* temp = p;

p = p->next;

free(temp);

}

}

void ListPrint(struct list \* p)

{

putchar('\n');

while (p)

{

printf(" %s\n", p->str);

p = p->next;

}

}

struct list \* Listinit(struct list \* n, int size)

{

n = (struct list \*) malloc(sizeof(struct list));

n->str = (char \*)malloc(sizeof(char) \* size);

return n;

}

int Listlen(struct list \* p)

{

int i = 0;

struct list \* n = p;

while (n)

{

n = n->next;

i++;

}

return i;

}