ГУАП

КАФЕДРА №14

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Должность, уч. степень, звание подпись, дата инициалы, фамилия

**ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3**

тема: Работа с массивами структур

по курсу: ПРОГРАММИРОВАНИЕ НА ЯВУ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 1441 Лубинец М.И

подпись, дата инициалы, фамилия

Санкт-Петербург

2015

1. Постановка задачи

Ввести массив структур в соответствии с вариантом.   
Рассортировать массив в алфавитном порядке по первому полю, входящему в структуру. В программе реализовать меню:   
1) Ввод массива структур;   
2) Сортировка массива структур;   
3) Поиск в массиве структур по заданному параметру;   
4) Изменение заданной структуры;   
5) Удаление структуры из массива;   
6) Вывод на экран массива структур;   
7) Выход.

Структура «Информация»: носитель; объем; название; автор.

2. Листинг

types.h

#ifndef TYPES\_H

#define TYPES\_H

#include <stdlib.h>

**/\*** **Задание:**

**\*** **1)** **Ввод** **массива** **структур** **(SA\*)**

**\*** **2)** **Сортировка** **массива** **структур** **(SA\*)**

**\*** **3)** **Поиск** **по** **параметру** **(SA\*)**

**\*** **4)** **Изменение** **заданой** **структуры** **(SA\*)**

**\*** **5)** **Удаление** **структуры** **из** **массива** **(SA\*)**

**\*** **6)** **Вывод** **на** **экран** **массива** **структур** **(SA\*)**

**\*** **7)** **Выход** **(SA\*)**

**\*/**

#define BUFSIZE 64

**typedef** unsigned char ubyte;

**enum** MENU {

ENTER = 1,

SORT,

SEARCH,

EDIT,

DELETE,

PRINT,

EXIT

};

**typedef** **struct** data\_s {

char author[BUFSIZE];

char title [BUFSIZE];

char medium[BUFSIZE];

uint size;

} Data;

**typedef** **struct** struct\_array\_s {

Data \*array;

uint len;

} StructArray;

#endif **//** **TYPES\_H**

main.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <assert.h>

#include <stddef.h>

#include <string.h>

#include "types.h"

**/\*** **Задание:**

**\*** **1)** **Ввод** **массива** **структур** **(SA\*)**

**\*** **2)** **Сортировка** **массива** **структур** **(SA\*)**

**\*** **3)** **Поиск** **по** **параметру** **(SA\*)**

**\*** **4)** **Изменение** **заданой** **структуры** **(SA\*)**

**\*** **5)** **Удаление** **структуры** **из** **массива** **(SA\*)**

**\*** **6)** **Вывод** **на** **экран** **массива** **структур** **(SA\*)**

**\*** **7)** **Выход** **(SA\*)**

**\*/**

void enter (StructArray \*sa);

void sort (StructArray \*sa);

void search(StructArray \*sa);

void edit (StructArray \*sa);

void del (StructArray \*sa);

void print (StructArray \*sa);

void finish(StructArray \*sa);

**typedef** void (\*action\_fptr)(StructArray \*);

void print\_data(Data \*d);

void clear() {

system("clear");

}

void press\_key() {

system("read -p \"Press [Enter] key to continue...\"");

}

action\_fptr menu() {

**static** **const** char message[] = {

"1) Enter structures\n"

"2) Sort structures\n"

"3) Search by parameter\n"

"4) Edit structure\n"

"5) Delete structure\n"

"6) Print structures\n"

"7) Exit\n"

"\n"

"Your choise: "

};

int a = 0;

clear();

fflush(stdin);

printf("%s", message);

scanf("%i", &a);

**switch**(a) {

**case** ENTER: **return** enter;

**case** SORT: **return** sort;

**case** SEARCH: **return** search;

**case** EDIT: **return** edit;

**case** DELETE: **return** del;

**case** PRINT: **return** print;

**case** EXIT: **return** finish;

**default**:

printf("%i is undefinied, please enter %i-%i\n", (int)a, ENTER, EXIT);

press\_key();

**return** NULL;

}

}

**volatile** int done = 0;

int main(void) {

StructArray sa = {NULL, 0};

action\_fptr action;

**while**(!done) {

action = menu();

clear();

**if**(action) action(&sa);

}

**return** 0;

}

**inline** void getstr(char\* dst, int size, FILE\* stream) {

assert(dst != NULL && stream != NULL);

**do** {

fgets(dst, size, stream);

} **while**(dst[0] == '\n');

dst[strlen(dst)-1] = '\0';

}

int convert\_size (char\* size) {

assert(size != NULL);

uint len = strlen(size);

char\* end = size + len;

**while**(\*(end) == ' ') end--;

end -= 1;

int multiplier = 1;

**switch**(\*end) {

**case** 'G':

multiplier = 1024\*1024\*1024;

**break**;

**case** 'M':

multiplier = 1024\*1024;

**break**;

**case** 'K':

multiplier = 1024;

**break**;

**default**:

**break**;

}

**return** atoi(size) \* multiplier;

}

void enter(StructArray \*sa) {

assert(sa != NULL);

**if**(sa->array) free(sa->array);

printf("How many elements you want to enter? ");

scanf("%i", &sa->len);

clear();

sa->array = malloc(sa->len \* **sizeof**(Data));

Data\* current;

char sizebuf[BUFSIZE];

uint i;

**for**(i = 0; i < sa->len; i++) {

current = sa->array + i;

printf("Structure %i:\n", i+1);

printf("Author:\t\t\t");

getstr(current->author, BUFSIZE, stdin);

printf("Title:\t\t\t");

getstr(current->title, BUFSIZE, stdin);

printf("Medium:\t\t\t");

getstr(current->medium, BUFSIZE, stdin);

printf("Size(G/M/K):\t\t");

getstr(sizebuf, BUFSIZE, stdin);

current->size = convert\_size(sizebuf);

clear();

}

}

void struct\_menu(**const** char\* header, int \*offset, size\_t \*size) {

assert(header != NULL && offset != NULL && size != NULL);

int a, ofst = -1;

size\_t s;

**while**(ofst < 0) {

printf("%s:\n"

"1) Author\n"

"2) Title\n"

"3) Medium\n"

"4) Size\n"

"\n"

"Your choice: ", header);

scanf("%i", &a);

**switch**(a) {

**case** 1: ofst = offsetof(Data, author); s = BUFSIZE; **break**;

**case** 2: ofst = offsetof(Data, title); s = BUFSIZE; **break**;

**case** 3: ofst = offsetof(Data, medium); s = BUFSIZE; **break**;

**case** 4: ofst = offsetof(Data, size); s = **sizeof**(uint); **break**;

**default**:

printf("%i is undefinied, please enter %i-%i\n", (int)a, 1, 4);

press\_key();

}

clear();

}

\*offset = ofst;

\*size = s;

}

void sort(StructArray \*sa) {

assert(sa != NULL);

int offset;

size\_t size;

struct\_menu("Sort by", &offset, &size);

#define swap(a, b) \

Data swap\_temp = a; \

b = a; \

a = swap\_temp;

uint i, j;

**for**(i = 0; i < sa->len; i++) {

**for**(j = 0; j < sa->len-1; j++) {

**if**(offset != offsetof(Data, size)) {

**if**(strcmp((char\*)&sa->array[i] + offset,

(char\*)&sa->array[j] + offset) < 0) {

swap(sa->array[i], sa->array[j]);

}

} **else** {

**if**(sa->array[j].size < sa->array[i].size) {

swap(sa->array[i], sa->array[j]);

}

}

}

}

}

void search(StructArray \*sa) {

assert(sa != NULL);

int offset, counter;

size\_t size;

uint i;

struct\_menu("Search for", &offset, &size);

printf("Enter value to search for: ");

**if**(size == **sizeof**(uint)) {

uint target;

scanf("%u", &target);

**for**(i = 0; i < sa->len; i++) {

**if**(sa->array[i].size == target) {

printf("Structure %i match!\n", i);

print\_data(&sa->array[i]);

printf("\n");

counter++;

}

}

} **else** {

char target[BUFSIZE];

getstr(target, BUFSIZE, stdin);

**for**(i = 0; i < sa->len; i++) {

**if**(strcmp((char\*)&sa->array[i]+offset, target) == 0) {

printf("Structure %i match!\n", i);

print\_data(&sa->array[i]);

printf("\n");

counter++;

}

}

}

printf("%i structures matching\n", counter);

press\_key();

}

void edit(StructArray \*sa) {

assert(sa != NULL);

clear();

**if**(!sa->len || !sa->array) {

printf("There are no structures to edit");

} **else** {

int n = -1;

**while**(n < 0){

printf("What structure do you want to edit? ");

scanf("%i", &n);

**if**(n < 1 || n > sa->len+1) {

printf("Wrong value, number %i-%i is expected\n", 1, sa->len+1);

n = -1;

}

}

int offset;

size\_t size;

printf("Target structure:\n");

print\_data(&sa->array[n]);

struct\_menu("\nChoose field to edit:", &offset, &size);

printf("New value: ");

**if**(size == **sizeof**(uint)) {

uint val;

scanf("%u", &val);

sa->array[n].size = val;

} **else** {

getstr((char\*)&sa->array[n]+offset, size, stdin);

}

clear();

printf("Edited structure:\n");

print\_data(&sa->array[n]);

}

press\_key();

}

void del(StructArray \*sa) {

assert(sa != NULL);

clear();

**if**(!sa->len || !sa->array) {

printf("There are no structures to delete");

} **else** {

int n = -1;

**while**(n < 0){

printf("What structure do you want to delete? ");

scanf("%i", &n);

**if**(n < 1 || n > sa->len+1) {

printf("Wrong value, number %i-%i is expected\n", 1, sa->len+1);

n = -1;

}

}

sa->array[n] = sa->array[sa->len];

**if**(sa->len > 1) {

Data\* temp;

temp = realloc(sa->array, (--sa->len) \* **sizeof**(Data));

assert(temp != NULL);

sa->array = temp;

} **else** {

free(sa->array);

}

sa->len--;

}

press\_key();

}

void print\_data(Data \*d) {

assert(d != NULL);

printf("Author:\t%s\n", d->author);

printf("Title:\t%s\n", d->title);

printf("Meduim:\t%s\n", d->medium);

printf("Size:\t%u\n", d->size);

}

void print(StructArray \*sa) {

assert(sa != NULL);

uint i;

**for**(i = 0; i < sa->len; i++) {

printf("Structure %i:\n", i+1);

print\_data(&sa->array[i]);

printf("\n");

}

press\_key();

}

void finish(StructArray \*sa) {

assert(sa != NULL);

**if**(sa->array) free(sa->array);

done = 1;

}