Липецкий государственный технический университет

Кафедра АСУ

ЛАБОРАТОРНАЯ РАБОТА

по программированию

Обработка массивов значений

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

наименование темы

**(Наименование темы работ указывают без слов «тема», «на тему»)**

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Курганов С. В.

подпись, дата

Группа МА-16-1

Руководитель

к.т.н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Журавлева М.Г.

подпись, дата

Липецк, 2020

Оглавление

[1 Цель работы 3](#_Toc60239254)

[2 Задание 3](#_Toc60239255)

[3 Текст программы 4](#_Toc60239256)

[4 Пример выполнения 8](#_Toc60239257)

# Цель работы

Освоить работу с массивами, элементами которых являются переменные структурного типа данных языка C.

# Задание

Написать программу, выполняющую следующие действия:

1. Создание и ввод элементов массива структур.
2. Установка/получение значений полей элемента массива структур по заданному индексу в массиве (индекс вводится пользователем).
3. Поиск элемента массива структур, в котором значение выбранного поля совпадает со значением поля, введенным пользователем (номер поля, по которому осуществляется поиск, вводится пользователем).
4. Обмен значений соответствующих полей двух заданных элементов массива структур (индексы элементов вводятся пользователем).
5. Упорядочение элементов массива структур по возрастанию/убыванию значений заданного поля.
6. Вывод одного элемента массива структур или диапазона элементов (индекс или диапазон индексов вводится пользователем).

В качестве элементов массива необходимо использовать структуру, представляющую собой театральный билет и содержащую 3 поля – номер билета, название концерта и стоимость билета.

# Текст программы

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#define MAX\_NAME\_SIZE 256

#define MENU\_OUTPUT 1

#define MENU\_ACTION 2

#define MENU\_SEARCH 3

#define MENU\_SWAP 4

#define MENU\_EXIT 5

typedef struct {

unsigned num;

char name[MAX\_NAME\_SIZE];

float price;

} ticket\_t;

void input\_ticket\_num(ticket\_t\* ticket) {

printf("Введите номер билета: ");

scanf("%u", &ticket->num);

}

void input\_concert\_name(ticket\_t\* ticket) {

printf("Введите название спектакля: ");

scanf ("%255s", ticket->name);

}

void input\_ticket\_price(ticket\_t\* ticket) {

printf("Введите цену билета: ");

scanf("%f", &ticket->price);

}

void input\_ticket(ticket\_t\* ticket) {

input\_ticket\_num(ticket);

input\_concert\_name(ticket);

input\_ticket\_price(ticket);

}

ticket\_t\* create\_tickets(unsigned count) {

ticket\_t\* tickets = (ticket\_t\*)malloc(count \* sizeof(ticket\_t));

for(unsigned i = 0; i < count; i++) {

input\_ticket(&tickets[i]);

}

return tickets;

}

void destroy\_tickets(ticket\_t\* tickets) {

free(tickets);

}

void print\_ticket(const ticket\_t\* ticket) {

if(ticket) {

printf("%u\t\"%s\" (%.2f рублей)\n", ticket->num, ticket->name, ticket->price);

} else {

printf("Билет не найден\n");

}

}

void print\_tickets(ticket\_t\* tickets, unsigned from, unsigned to) {

for(unsigned i = from; i < to; i++) {

print\_ticket(&tickets[i]);

}

}

ticket\_t\* find\_ticket\_by\_num(ticket\_t\* tickets, unsigned count, unsigned num) {

for(unsigned i = 0; i < count; i++) {

if(tickets[i].num == num) {

return &tickets[i];

}

}

return NULL;

}

ticket\_t\* find\_ticket\_by\_name(ticket\_t\* tickets, unsigned count, const char\* name) {

for(unsigned i = 0; i < count; i++) {

if(strcmp(tickets[i].name, name) == 0) {

return &tickets[i];

}

}

return NULL;

}

ticket\_t\* find\_ticket\_by\_price(ticket\_t\* tickets, unsigned count, float price) {

for(unsigned i = 0; i < count; i++) {

if(tickets[i].price == price) {

return &tickets[i];

}

}

return NULL;

}

void swap\_tickets(ticket\_t\* a, ticket\_t\* b) {

ticket\_t c;

memcpy(&c, a, sizeof(ticket\_t));

memcpy(a, b, sizeof(ticket\_t));

memcpy(b, &c, sizeof(ticket\_t));

}

int check\_index(unsigned count, unsigned index) {

if(count == 0)

return 0;

if(index >= count) {

printf("Индекс может быть от 0 до %u\n", count - 1);

return 0;

}

return 1;

}

// точка входа программы

int main( int argc, char\* argv[] )

{

unsigned i, j, n, menu, action;

ticket\_t ticket;

ticket\_t\* ptr = NULL;

printf("Введите количество билетов: ");

scanf("%u", &n);

if(n == 0)

return 0;

ticket\_t\* tickets = create\_tickets(n);

do {

printf("[%d]\tВывод данных\n", MENU\_OUTPUT);

printf("[%d]\tРабота с данными выбранного билета\n", MENU\_ACTION);

printf("[%d]\tПоиск\n", MENU\_SEARCH);

printf("[%d]\tОбмен двух значений\n", MENU\_SWAP);

printf("[%d]\tВыход\n", MENU\_EXIT);

printf("Выберите действие: ");

scanf("%u", &menu);

switch(menu) {

case MENU\_OUTPUT:

printf("Введите индекс начала диапазона: ");

scanf("%u", &i);

if(!check\_index(n, i))

break;

printf("Введите индекс конца диапазона: ");

scanf("%u", &j);

if(!check\_index(n, j))

break;

if(i >= j) {

printf("Ошибка: начало диапазона должно быть меньше конца диапазона!\n");

break;

}

print\_tickets(tickets, i, j);

break;

case MENU\_ACTION:

printf("Введите индекс элемента: ");

scanf("%u", &i);

if(!check\_index(n, i))

break;

printf("Введите номер операции, 1 – установка, 2 – получение: ");

scanf("%u", &action);

switch(action) {

case 1:

input\_ticket(&tickets[i]);

break;

case 2:

print\_ticket(&tickets[i]);

break;

default:

printf("Неверное значение\n");

}

break;

case MENU\_SEARCH:

printf("Введите индекс поля, по которому будет выполняться поиск (0 - номер, 1 - название, 2 - цена): ");

scanf("%u", &i);

switch(i) {

case 0:

input\_ticket\_num(&ticket);

print\_ticket(find\_ticket\_by\_num(tickets, n, ticket.num));

break;

case 1:

input\_concert\_name(&ticket);

print\_ticket(find\_ticket\_by\_name(tickets, n, ticket.name));

break;

case 2:

input\_ticket\_price(&ticket);

print\_ticket(find\_ticket\_by\_price(tickets, n, ticket.price));

break;

default:

printf("Номер может быть от 0 до 2\n");

}

break;

case MENU\_SWAP:

printf("Введите индекс первого элемента: ");

scanf("%u", &i);

if(!check\_index(n, i))

break;

printf("Введите индекс второго элемента: ");

scanf("%u", &j);

if(!check\_index(n, j))

break;

if(i == j) {

printf("Ошибка: индексы не должны совпадать!\n");

break;

}

swap\_tickets(&tickets[i], &tickets[j]);

print\_tickets(tickets, 0, n);

break;

case MENU\_EXIT:

printf("Завершение работы\n");

break;

default:

printf("Неверное значение\n");

}

printf("---------------------------------------------------------------------\n");

} while(menu != MENU\_EXIT);

destroy\_tickets(tickets);

return 0;

}

# Пример выполнения

Протестируем большинство действий пользователя. Сразу после запуска, программа запрашивает у пользователя количество билетов и затем поочерёдно запрашивает ввод параметров каждого из них. Далее на экране в виде нумерованного списка отображаются все возможные действия пользователя.

Ввод данных и вывод на экран подмножества билетов (действие 1) показаны на рисунке 1.

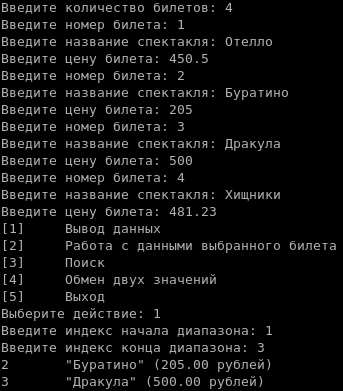


Рисунок 1 – вывод данных.

Следующее по порядку действие – работа с выбранным элементом массива, то есть с данным конкретного билета. При выборе этого действия, программа запросит ввести индекс элемента массива и вид действия – изменение (ввод) данных или их отображение (вывод). Установка новых параметров билета с индексом 0 показана на рисунке 2.

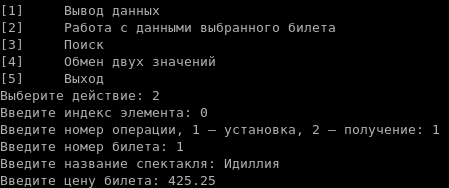


Рисунок 2 – установка параметров выбранного билета.

На рисунке 3 показана проверка только что установленных параметров – вывод на экран билета с индексом 0.

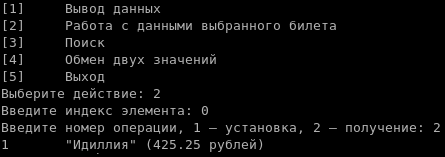


Рисунок 3 – получение параметров выбранного билета.

Действие 3 предназначено для поиска билета по одному из полей. После выбора этого действия программа запрашивает ввод индекса, означающего поле данных, по которому будет производиться поиск:

0 – номер билета;

1 – название концерта;

2 – цена билета.

После этого программа запрашивает ввод искомого значения, осуществляет поиск по всему массиву и выводит результат на экран.

Все варианты поиска изображены на рисунке 4 – сперва мы ищем билет по его номеру, затем по названию концерта и, наконец, по цене.

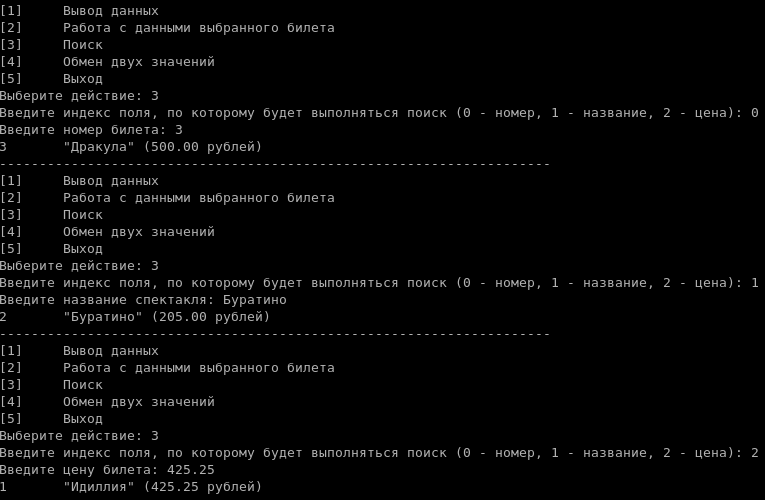


Рисунок 4 – поиск билета.

Следующее действие – обмен данными двух билетов. Для его выполнения требуется ввести несовпадающие индексы двух билетов, данные которых нужно обменять местами. Программа производит стандартную проверку значения каждого их индексов, а также их равенство между собой. Выполнение действия «Обмен двух значений», включая попытку ввода недопустимых значений, показано на рисунке 5.

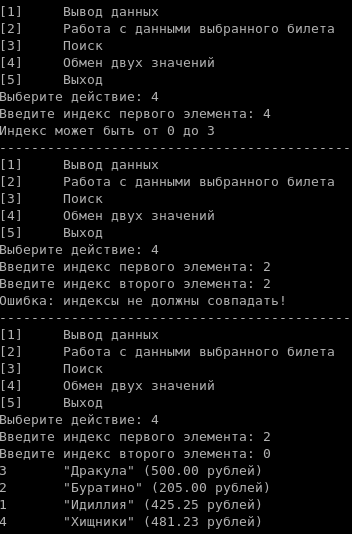


Рисунок 5 – обмен двух значений.

Последнее по порядку действие – выход из программы, как показано на рисунке 6.

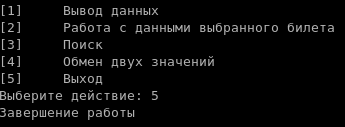


Рисунок 6 – завершение работы.