МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра вычислительной техники

Отчет по лабораторной работе №9
по дисциплине «Программирование»
Тема: Указатели на структуры и функции

Студент гр. 9305 _____ Китаев И.А.

Преподаватель _____ Перязева Ю.В.

Санкт-Петербург
2020

Содержание:

Введение

Данная лабораторная работа выполнена с целью приобретения практических навыков в разработке алгоритма и написании программы с использованием указателей на структуры и функции на языке Си.

Задание

Для выбранной предметной области создать динамический массив структур, содержащих характеристики объектов предметной области.

Обязательный набор полей:

динамический массив символов, включая пробелы (name)

произвольный динамический массив символов

числовые поля типов int и float (не менее двух полей каждого типа)

поле с числовым массивом.

Написать программу, обеспечивающую начальное формирование массива структур при чтении из файла (текст с разделителями — CSV) с последующим возможным дополнением элементов массива при вводе с клавиатуры. Следует использовать указатели на структуры и указатели на функции обработки массива в соответствии с вариантом задания.

Во всех случаях, когда при поиске записей результат отсутствует, следует вывести сообщение.

Выбор записей по значению любого символьного поля (выбор из меню), сортировка результата по убыванию значений последнего числового поля.

Постановка задачи и описание решения

Для начала объявим структуру state, которая выглядит следующим образом:

```
typedef struct state {
```

```
    char *name; //Название государства
    char *inter_org; //Название международной организации
    int terr; //Территория страны
    float pop; //Население страны в миллионах
    float pop_cap; // Население столицы в миллионах
    int year_of_entry; //Год вступления в организацию
    int data_of_app[3]; //Дата основания государства
    }states;
```

Затем внутри главной функции main создадим указатель на массив states0, который состоит из произвольного количества структур. Откроем csv-файл, сделаем проверку на отсутствие ошибок. Посчитаем количество строк в файле, спросим у пользователя, хотел бы он дополнить структуру и выделим память под массив структур. Далее заполним структуру, используя для этого такие функции, как simple_split и struct_fill, а также, если надо, дополним структуру введенными с клавиатуры данными при помощи функции add_records. Вызовем функцию sort_course, которая предназначена для сортировки структуры по убыванию значения последнего числового поля и спросим у пользователя, по значению какого поля он хочет осуществить сортировку, после чего вызовем функцию new_gets для считывания строки, по которой будет произведен выбор записей и уже при помощи общей функции OutputKind вызовем необходимую нам функцию и выведем ответ. В конце обратимся к функции ClearStructure для очистки структуры.

Описание структуры

Имя поля	Тип	Назначение
name	char	Название государства
inter_org	char	Название международной организации
terr	int	Территория страны
pop	float	Население страны в миллионах
pop_cap	float	Население столицы в миллионах
year_of_entry	int	Год вступления в организацию
data_of_app	int	Дата основания государства

Структура вызовов функций

1. Функция main

Описание: Является точкой входа в программу.

Прототип: int main()

Пример вызова: main()

Описание переменных:

Имя переменной	Тип	Назначение
**states0	states	Массив указателей на структуры
(**kind)(states**, int)	char	Массив указателей на функции
**s2	char	Массив строк
s1[maxlen]	char	Строка из файла
sep	char	Разделитель
*Nam	char	Указатель на строку, которую ввел пользователь
n	int	Счетчик количества строк в файле
i	int	Счетчик (индекс элемента массива)
slen	int	Длина строки Nam
option	int	Переменная выбора
printed	int	Содержит значение, возвращаемое функцией OutputKind
inp	int	Переменная выбора
add	int	Число структур, которые пользователь хочет добавить
m	int	Содержит значение количества функций
add1	int	Число структур, которые пользователь хочет добавить
*df	FILE	Указатель на файл

2. Функция **simple_split

Описание: Функция для разбиения строки из файла на подстроки по заданному символу. Сами подстроки записываются в строки двумерного массива. Полученный двумерный массив возвращается в функцию main.

Прототип: char **simple_split(char *str, int length, char sep)

Пример вызова: s2 = simple_split(s1, strlen(s1), sep);

Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	*str	char	Строка из файла
формальный аргумент	length	int	Длина строки
формальный аргумент	sep	char	Символ разделитель
локальная	**str_array	char	Массив строк
локальная	i	int	Индекс массива
локальная	j	int	Номер столбца массива
локальная	k	int	Номер столбца массива
локальная	m	int	Номер строки массива
локальная	key	int	Успешно ли выделилась память
локальная	count	int	Счетчик успешно выделенной памяти

Возвращаемое значение: массив строк.

3. Функция ClearStringArray

Описание: Алгоритм, реализующий очистку памяти строк.

Прототип: void ClearStringArray(char **str, int n)

Пример вызова: ClearStringArray(str_array, count);

Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	*str	char	Строка из файла
формальный аргумент	n	int	Длина строки
локальная	i	int	Номер строки

Возвращаемое значение: нет

4. Функция print_header

Описание: Вывод заголовка таблицы.

Прототип: void print_header()

Пример вызова: print_header();

Описание переменных: нет

Возвращаемое значение: нет

5. Функция *struct_fill

Описание: Функция для заполнения структур.

Прототип: states *struct_fill(char **str)

Пример вызова: states0[i] = struct_fill(s2);

Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	**str	char	Строка из файла
локальная	*str0	states	Указатель на структуру

Возвращаемое значение: массив структур

6. Функция struct_out

Описание: Функция для вывода массива структур.

Прототип: void struct_out(states *str0)

Пример вызова: struct_out(states0[i]);

Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	*str0	states	Структура для вывода

Возвращаемое значение: нет

7. Функция new_gets

Описание: Функция считывания строки, введенную пользователем.

Прототип: void new_gets(char *s, int lim)

Пример вызова: new_gets(Nam, maxlen);

Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	*s	char	Строка
формальный аргумент	lim	int	Максимальная длина строки
локальная	c	char	Одиночный символ
локальная	i	int	Счетчик символов

Возвращаемое значение: нет

8. Функция sort_course

Описание: Функция, осуществляющая сортировку структур по убыванию значений последнего числового поля, используя пузырьковый метод.

Прототип: void sort_course(int n, states **str0)

Пример вызова: sort_course(n, states0);

Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	n	int	Число строк в массиве структур
формальный аргумент	**str0	states	Массив структур
локальная	*tmp_struct	states	Указатель на структуру

локальная	i	int	Индекс структуры в массиве структур
локальная	j	int	Индекс структуры

Возвращаемое значение: нет

9. Функция OutputKind

Описание: Вывод отсортированной структуры в зависимости от выбора пользователя.

Прототип: int OutputKind(int n, states **str0, int (*funcName)(states**, int), char *str1, int len)

Пример вызова: printed = OutputKind(n, states0, kind[option - 1], Nam, slen);

Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	n	int	Число строк в массиве структур
формальный аргумент	**str0	states	Массив структур
формальный аргумент	(*funcName)(states**, int)	int	Указатель на функцию для сортировки
формальный аргумент	*str1	char	Строка, введенная пользователем
формальный аргумент	len	int	Длина строки
локальная	count	int	Счетчик выводимых структур
локальная	flag	int	Флаг

локальная	*string	char	Переменная, хранящая строку (название страны или название международной организации)
локальная	i	int	Индекс структуры
локальная	k	int	Счетчик длины строки

Возвращаемое значение: число выводимых структур

10. Функция state_name

Описание: Возвращает значение поля структуры для сортировки.
Прототип: char state_name(states **str0, int i0)
Пример вызова: kind[0] = state_name;
Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	**str0	states	Массив структур
формальный аргумент	i0	int	Номер нужной структуры

Возвращаемое значение: строка внутри поля структуры

11. Функция international_organizations

Описание: Возвращает значение поля структуры для сортировки.
Прототип: char international_organizations(states **str0, int i0);
Пример вызова: kind[1] = international_organizations;
Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	**str0	states	Массив структур
формальный аргумент	i0	int	Номер нужной структуры

Возвращаемое значение: строка внутри поля структуры

12. Функция *add_records

Описание: Функция, при помощи которой добавляются структуры путем ввода значения их полей с клавиатуры.
Прототип: states *add_records()
Пример вызова: states0[i] = add_records();
Описание переменных:

Вид	Имя переменной	Тип	Назначение
локальная	*tab	states	Указатель на структуру
локальная	i	int	Счетчик символов
локальная	c	int	Хранит в себе введенный символ или пустое значение

Возвращаемое значение: введенную пользователем структуру

13. Функция ClearStructure

Описание: Функция для очистки памяти, в которой хранилась структура.

Прототип: void ClearStructure(states **str0, int n);

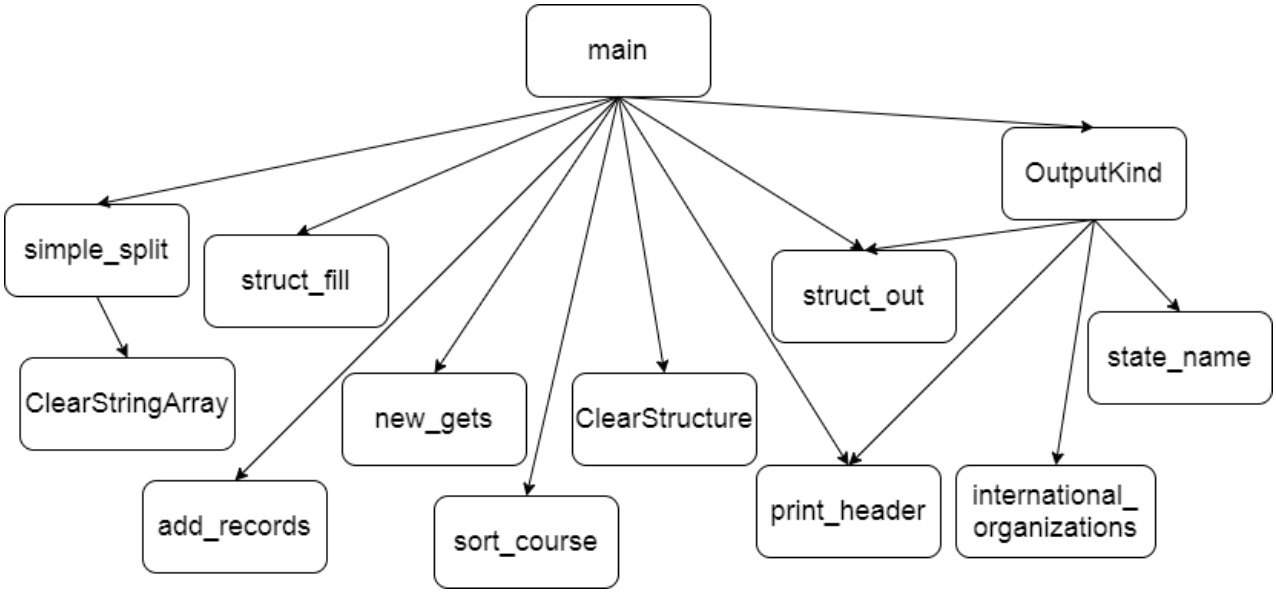
Пример вызова: ClearStructure(states0, n);

Описание переменных:

Вид	Имя переменной	Тип	Назначение
формальный аргумент	**str0	states	Массив структур
формальный аргумент	n	int	Число структур
локальная	i	int	Счетчик структур

Возвращаемое значение: нет

Схема вызовов функций



Контрольные примеры

Файловые данные:

Russia;UN;17125191;142.857;12.692;1945;881;12;25

Russia;WTO;17125191;142.857;12.692;1995;881;12;25

Japan;UN;377944;126.225;13.952;1956;660;2;11

China;UN;9598962;1404.329;21.705;1945;1911;10;1

Russia;OSCE;17125191;142.857;12.692;1975;881;12;25

Spain;UN;505990;46.715;3.166;1955;1479;11;7

German;UN;357408;83.019;3.645;1973;843;8;10

Russia;BRICS;17125191;142.857;12.692;2006;881;12;25

China;BRICS;9598962;1404.329;21.705;2006;1911;10;1

1) **Ввод:**

```
Hello! You have a structure. Do you want to add something?
1 - Yes
2 - No
2
Kinds of sort:
1 - state name
2 - international organizations
3 - show full table
Enter your choice: 1_
```

```
Enter name: Russia
```

Результат программы:

```
¦State name ¦Inter org ¦State terr ¦State pop ¦Cap pop ¦ Year ¦  State app¦
+-----------+----------+-----------+----------+--------+------+-----------+
¦    Russia ¦       UN ¦  17125191 ¦  142,000 ¦ 12,000 ¦ 1945 ¦  881 12 25¦
¦    Russia ¦     OSCE ¦  17125191 ¦  142,000 ¦ 12,000 ¦ 1975 ¦  881 12 25¦
¦    Russia ¦      WTO ¦  17125191 ¦  142,000 ¦ 12,000 ¦ 1995 ¦  881 12 25¦
¦    Russia ¦    BRICS ¦  17125191 ¦  142,000 ¦ 12,000 ¦ 2006 ¦  881 12 25¦
```

2) **Ввод:**

```
Hello! You have a structure. Do you want to add something?
1 - Yes
2 - No
1
How many states do you want to add: 1
Enter name: France
Enter name of international organization: UN
Enter territory of coutry: 121222
Enter population: 123.12
Enter population of capital: 12.12
Enter year of entry into organization: 1987
Enter data of appearance (XXXX XX XX): 1233 12 12_
```

```
Kinds of sort:
1 - state name
2 - international organizations
3 - show full table
Enter your choice: 2
```

```
Enter name: UN
```

Результат программы:

State name	Inter org	State terr	State pop	Cap pop	Year	State app
Russia	UN	17125191	142,000	12,000	1945	881 12 25
China	UN	9598962	1404,000	21,000	1945	1911 10 1
Spain	UN	505990	46,000	3,000	1955	1479 11 7
Japan	UN	377944	126,000	13,000	1956	660 2 11
German	UN	357408	83,000	3,000	1973	843 8 10
France	UN	121222	123,000	12,000	1987	1233 12 12

Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <locale.h>
typedef struct state {
    char *name; //Название государства
    char *inter_org; //Название международной
организации
    int terr; //Территория страны
    float pop; //Население страны в миллионах
    float pop_cap; // Население столицы в миллионах
    int year_of_entry; //Год вступления в организацию
    int data_of_app[3]; //Дата основания государства
}states;

int OutputKind(int n, states **str0, int
(*funcName)(states**, int), char *str1, int len);
void print_header();
char **simple_split(char *str, int length, char sep);
void ClearStringArray(char **str, int n);
states *struct_fill(char **str);
void struct_out(states *str0);
void ClearStructure(states **str0, int n);
char state_name(states **str0, int i0);
char international_organizations(states **str0, int
i0);
void new_gets(char *s, int lim);
void sort_course(int n, states **str0);
states *add_records();
int main()
{
```

```
    setlocale(LC_ALL, "RUS");
    enum {maxlen = 128};
    states **states0 = NULL;
    char (**kind)(states**, int) = NULL;
    char **s2 = NULL;
    char s1[maxlen];
    char sep = ';';
    char *Nam;
    int n, i, slen, option, printed, inp, add, m, add1;
    FILE *df;
    n = 0;
    add = 0;
    m = 2;
    fflush(stdin);
    if((df = fopen("text.csv","r")) != NULL){
        //Нахождение количества строк в файле
        while((fgets(s1, maxlen, df)) != NULL) n++;
        rewind(df);
        printf("Hello! You have a structure. Do you
want to add something?\n");
        do
        {
            printf("1 - Yes\n");
            printf("2 - No\n");
            scanf("%d", &inp);
            fflush(stdin);
        } while((inp < 1) || (inp > 2));
        if (inp == 1) {
            printf("How many states do you want to add:
");
            scanf("%d", &add1);
            fflush(stdin);
            add = add1;
```

```
        }
        n = n + add;
        //Выделение памяти под структуру
        if((states0 =
(states**)malloc(n*sizeof(states*))) != NULL) {
            //Заполнение структуры
            for (i = 0; i < n - add; i++) {
                fgets(s1, maxlen, df);
                s1[strlen(s1)] = '\0';
                s2 = simple_split(s1, strlen(s1), sep);
                if (s2 != NULL) {
                    states0[i] = struct_fill(s2);
                    if (states0[i] == NULL)
puts("Structure not allocated!");
                }
                else puts("Error at data reading!");
            }
            //Дополнение элементов массива при вводе с
клавиатуры
            if (inp == 1) {
                for (i = n - add; i < n; i++){
                    states0[i] = add_records();
                }
            }
        }
        else puts("No memory allocation!");
        fclose(df);
        kind = malloc(m *sizeof(int(*)(states**,
int)));
        //Выбор сортировки, либо вывод структуры
        if((states0 != NULL) && (kind != NULL)){
            kind[0] = state_name;
            kind[1] = international_organizations;
```

```
            do
            {
                printf("Kinds of sort:\n");
                printf("1 - state name\n");
                printf("2 - international
organizations\n");
                printf("3 - show full table\n");
                printf("Enter your choice: ");
                scanf("%d", &option);
                system("cls");
                if (option == 3) {
                    print_header();
                    for(i = 0; i < n; i++){
                        struct_out(states0[i]);
                    }
                }
            } while((option < 1) || (option > 2));
            //Сортировка структуры по введенной строки
            printf("Enter name: ");
            Nam = (char*)malloc(maxlen*sizeof(char));
            fflush(stdin);
            new_gets(Nam, maxlen);
            slen = strlen(Nam);
            sort_course(n, states0);
            printed = OutputKind(n, states0,
kind[option - 1], Nam, slen);
            if(printed == 0) printf("\nThere's no such
name\n");
            ClearStructure(states0, n);
        }
        else puts("Unable to make functions!");
        if(states0 != NULL){
            free(states0);
```

```
            states0 = NULL;
        }
        if(kind != NULL){
            free(kind);
            kind = NULL;
        }
    }
    else puts("File not found!");
    return 0;
}
//Вывод заголовка таблицы
void print_header()
{
    printf("
_____
_____ \n");
    printf("|%10s |%9s |%10s |%9s |%7s |%5s
|%11s|\n","State name","Inter org","State terr","State
pop","Cap pop","Year","State app");
    printf("+-----------+----------+-----------+-------
---+--------+------+-----------+\n");
}

char **simple_split(char *str, int length, char sep)
{
    char **str_array = NULL;
    int i, j, k, m, key, count;
    //Подсчет слов в строке
    for(j = 0, m = 0; j < length; j++){
        if(str[j] == sep) m++;
    }
    key = 0;
    //Выделение памяти под массив строк
    str_array = (char**)malloc((m + 1)*sizeof(char*));
```

```
    if(str_array != NULL) {
        for(i = 0, count = 0; i <= m; i++, count++) {
            //Выделение памяти под каждую отдельную
строку
            str_array[i] =
(char*)malloc(length*sizeof(char));
            if(str_array[i] != NULL) key = 1;
            else {
                key = 0;
                i = m;
            }
        }
        if(key) {
            k = 0;
            m = 0;
            for(j = 0; j < length; j++) {
                if(str[j] != sep) str_array[m][j - k] =
str[j];
                else {
                    str_array[m][j - k] = '\0';
                    k = j + 1;
                    m++;
                }
            }
        }
        else {
            puts("ERROR at string allocation!\n");
            ClearStringArray(str_array, count);
        }
     }
     return str_array;
}
//Функция очистки памяти, выделенной под двумерный
массив
```

```
void ClearStringArray(char **str, int n)
{
    int i;
    for(i = 0; i < n; i++) {
        free(str[i]);
        str[i] = NULL;
    }
    free(str);
    str = NULL;

}

states *struct_fill(char **str)
{
    states *str0 = NULL;
    str0 = (states*)malloc(sizeof(states));
    if(str0 != NULL) {
        str0 -> name = str[0];
        str0 -> inter_org = str[1];
        str0 -> terr = atoi(str[2]);
        str0 -> pop = atof(str[3]);
        str0 -> pop_cap = atof(str[4]);
        str0 -> year_of_entry = atoi(str[5]);
        str0 -> data_of_app[0] = atoi(str[6]);
        str0 -> data_of_app[1] = atoi(str[7]);
        str0 -> data_of_app[2] = atoi(str[8]);
    }
    return str0;
}
void ClearStructure(states **str0, int n)
{
    int i;
    for (i = 0; i < n; i++) {
```

```
        free(str0[i] -> name);
        str0[i] -> name = NULL;
        free(str0[i] -> inter_org);
        str0[i] -> inter_org = NULL;
    }
    free(str0);
}
void struct_out(states *str0)
{
    printf("|%10s |%9s |%10d |%9.3f |%7.3f |%5d |%5d
%2d %2d|\n",
        str0->name,str0->inter_org,str0->terr,str0-
>pop,str0->pop_cap,str0->year_of_entry,str0-
>data_of_app[0],str0->data_of_app[1],str0-
>data_of_app[2]);
}


char state_name(states **str0, int i0)
{
    return str0[i0] -> name;
}
//Функция, при помощи которой добавляются структуры
путем ввода их с клавиатуры
states *add_records()
{
    states *tab;
    int i, c;
    c = ' ';
    i = 0;
    if ((tab = malloc(sizeof(states))) != NULL) {
        tab -> name = malloc(sizeof(char) * 20);
        printf("Enter name: ");
```

```
        while (c != '\n') {
            c = getchar();
            tab -> name[i] = c;
            i++;
        }
        tab -> name[i - 1] = '\0';
        fflush(stdin);
        c = ' ';
        i = 0;
        tab -> inter_org = malloc(sizeof(char) * 20);
        printf("Enter name of international
organization: ");
        while (c != '\n'){
            c = getchar();
            tab -> inter_org[i] = c;
            i++;
        }
        tab -> inter_org[i - 1] = '\0';
        fflush(stdin);
        printf("Enter territory of coutry: ");
        scanf("%d", &tab -> terr);
        fflush(stdin);
        printf("Enter population: ");
        scanf("%f", &tab -> pop);
        fflush(stdin);
        printf("Enter population of capital: ");
        scanf("%f", &tab -> pop_cap);
        fflush(stdin);
        printf("Enter year of entry into organization:
");
        scanf("%d", &tab -> year_of_entry);
        fflush(stdin);
```

```
        printf("Enter data of appearance (XXXX XX XX):
");
        scanf("%d %d %d", &tab -> data_of_app[0], &tab
-> data_of_app[1], &tab -> data_of_app[2]);
        fflush(stdin);
        puts("Press any key...");
        getchar();
        system("cls");
    }
    return (tab);
}

char international_organizations(states **str0, int i0)
{
    return str0[i0] -> inter_org;
}
void new_gets(char *s, int lim)
{
    char c;
    int i;
    i = 0;
    while(((c = getchar()) != '\n') && (i < lim - 1))
    {
        s[i] = c;
        i++;
    }
    s[i] = '\0';
}
//Сортировка структуры по убыванию значений последнего
числового поля
void sort_course(int n, states **str0)
{
    states *tmp_struct;
```

```
    int i, j;
    for(i = 0; i < n; i = i + 1) {
        for(j = 0; j < n - i - 1; j = j + 1) {
            if(str0[j] -> year_of_entry > str0[j + 1] -
> year_of_entry) {
                tmp_struct = str0[j];
                str0[j] = str0[j + 1];
                str0[j + 1] = tmp_struct;
            }
        }
     }
}
//Вывод отсортированной структуры в зависимости от
выбора пользователя
int OutputKind(int n, states **str0, int
(*funcName)(states**, int), char *str1, int len)
{
    int i, count, k, flag;
    char *string = NULL;
    count = 0;
    flag = 0;
    print_header();
    for(i = 0; i < n; i++) {
        string = funcName(str0, i);
        for (k = 0; k < strlen(string); k++) {
            if (string[k] == str1[k]) flag++;
        }
        if (flag == strlen(string)) {
            struct_out(str0[i]);
            count++;
        }
        flag = 0;
        free(string);
```

```
        string = NULL;
    }
    return count;
}
```

Пример работы программы:

Файловые данные:

Russia;UN;17125191;142.857;12.692;1945;881;12;25

Russia;WTO;17125191;142.857;12.692;1995;881;12;25

Japan;UN;377944;126.225;13.952;1956;660;2;11

China;UN;9598962;1404.329;21.705;1945;1911;10;1

Russia;OSCE;17125191;142.857;12.692;1975;881;12;25

Spain;UN;505990;46.715;3.166;1955;1479;11;7

German;UN;357408;83.019;3.645;1973;843;8;10

Russia;BRICS;17125191;142.857;12.692;2006;881;12;25

China;BRICS;9598962;1404.329;21.705;2006;1911;10;1

Ввод:

```
Hello! You have a structure. Do you want to add something?
1 - Yes
2 - No
1
How many states do you want to add: 1
Enter name: France
Enter name of international organization: UN
Enter territory of coutry: 121211
Enter population: 121.12
Enter population of capital: 12.12
Enter year of entry into organization: 1917
Enter data of appearance (XXXX XX XX): 1212 12 12
Press any key...
```

```
Kinds of sort:
1 - state name
2 - international organizations
3 - show full table
Enter your choice: 3
```

```
 _____________________________________________________________
|State name |Inter org |State terr |State pop |Cap pop | Year |  State app|
+-----------+----------+-----------+----------+--------+------+-----------+
|    Russia |       UN |  17125191 |  142,000 | 12,000 | 1945 |  881 12 25|
|    Russia |      WTO |  17125191 |  142,000 | 12,000 | 1995 |  881 12 25|
|     Japan |       UN |    377944 |  126,000 | 13,000 | 1956 |  660  2 11|
|     China |       UN |   9598962 | 1404,000 | 21,000 | 1945 | 1911 10  1|
|    Russia |     OSCE |  17125191 |  142,000 | 12,000 | 1975 |  881 12 25|
|     Spain |       UN |    505990 |   46,000 |  3,000 | 1955 | 1479 11  7|
|    German |       UN |    357408 |   83,000 |  3,000 | 1973 |  843  8 10|
|    Russia |    BRICS |  17125191 |  142,000 | 12,000 | 2006 |  881 12 25|
|     China |    BRICS |   9598962 | 1404,000 | 21,000 | 2006 | 1911 10  1|
|    France |       UN |    121211 |  121,000 | 12,000 | 1917 | 1212 12 12|
Kinds of sort:
1 - state name
2 - international organizations
3 - show full table
Enter your choice: 2
```

```
Enter name: UN
```

Вывод программы:

State name	Inter org	State terr	State pop	Cap pop	Year	State app
France	UN	121211	121,000	12,000	1917	1212 12 12
Russia	UN	17125191	142,000	12,000	1945	881 12 25
China	UN	9598962	1404,000	21,000	1945	1911 10 1
Spain	UN	505990	46,000	3,000	1955	1479 11 7
Japan	UN	377944	126,000	13,000	1956	660 2 11
German	UN	357408	83,000	3,000	1973	843 8 10

Заключение

При выполнении лабораторной работы были получены практические навыки в разработке алгоритма и написании программы на языке Си. Были получены основные знания о синтаксисе языка Си, в частности, об указателях на структуры и функции.