МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра информационно-аналитических систем безопасности

имени Л.С. Берштейна

**ОТЧЕТ**

**О лабораторной работе**

**по дисциплине**

**Основы алгоритмизации и программирования**

**на тему**

**«Работа со структурами»**

Выполнил:

студент гр. КТбо1-8

Грицено Д. А.

Вариант 9

«\_\_\_» 2017 г.

Проверил:

Профессор кафедры ИАСБ ИКТИБ

Беляков С.Л.

Таганрог – 2017

**Оглавление**

[Алгоритм работы 4](file:///C:\Users\Woland\Downloads\Otchet_OAiP_Dnestranskaya_5lab%20(1).docx#_Toc500968107)

[Пример входных и выходных данных 5](file:///C:\Users\Woland\Downloads\Otchet_OAiP_Dnestranskaya_5lab%20(1).docx#_Toc500968108)

[Программный код 6](file:///C:\Users\Woland\Downloads\Otchet_OAiP_Dnestranskaya_5lab%20(1).docx#_Toc500968109)

[Демонстрация работы программы 11](file:///C:\Users\Woland\Downloads\Otchet_OAiP_Dnestranskaya_5lab%20(1).docx#_Toc500968110)

[Заключение 12](file:///C:\Users\Woland\Downloads\Otchet_OAiP_Dnestranskaya_5lab%20(1).docx#_Toc500968111)

**Цели работы**

**1**. Целью лабораторной работы является освоение способов задания и методов обработки данных структурного типа. Изучение инициализации и присваивания структур, понятия массива структур, указателей на структуру и операции над указателями.

**2.** Выполнить задание: написать программу, которая записывает с клавиатуры файл структуру согласно выданному варианту задания.

В программе реализовать: дополнение существующего массива структур новыми структурами, поиск структуры с заданным значением выбранного элемента, вывод на экран содержимого массива структур, упорядочение массива структур по заданному полю

Вариант задания 9: «Владелец автомобиля»: имя, номер автомобиля, номер техпаспорта, дата рождения, телефон, отделение регистрации ГИБДД.

## **Алгоритм работы**

1. Подключаем необходимые нам библиотеки
2. Задаем максимальные длины строк данных
3. Создаем структуру, которая будет содержать поля(элементы) с информацией о владельце автомобиля
4. Создаем функцию поиска по элементам структур, в которой пользователь выбирает определенный элемент. Далее в соответствии с выбранным элементом происходит сравнение массивов и вывод владельца(владельцев) по заданному запросу
5. Создаем функцию сортировки, в которой сравниваются элементы, и если они различны, меняются местами и выводятся на экран
6. Создаем функцию задания элементов страны при помощи операторов ввода/вывода
7. Задаётся функция вывода автовладельцев в соответствии с порядковым номером
8. Создаем дополнительные функции для запросов
9. Организовываем проверку корректности вводимых данных
10. Результат записывается в файл

## **Пример входных и выходных данных**

|  |  |
| --- | --- |
| ***Входные данные*** | ***Выходные данные*** |
| 1  1  Vlad  A550mp77  93gh92201  08 01 1998  89993242854  2  1  2  2  1  Vlad  2 | *Выберите тип запроса: 1) Внесение структуры в файл 2) Поиск структуры в файле 3) Вывод структур на экран 4) Сортировка данных в файле 5) Завершение работы программы*  *Введите количество структур*  *Введите структуру в форме:*   1. *Имя* 2. *Номер автомобиля* 3. *Номер техпаспорта* 4. *Дата рождения* 5. *Телефон* 6. *Отд. Рег. ГИБДД*   *Будет еще запрос? (1 – да; 2 – нет)*  *Ввод:*  *Выберите тип запроса: 1) Внесение структуры в файл 2) Поиск структуры в файле 3) Вывод структур на экран 4) Сортировка данных в файле 5) Завершение работы программы*  *Будут вноситься структуры в файл перед поиском? (1 – да, 2 – нет) Ввод:*  *Выберите критерий поиска:*   1. *Имя* 2. *Номер авто* 3. *Номер техпаспорта* 4. *Дата рождения* 5. *Телефон* 6. *Отд. Рег. ГИБДД*   *Введите текст для поиска:*  *Vlad*  *A550mp77*  *93gh92201*  *08 01 1998*  *89993242854*  *2*  *Будет еще запрос? (1 – да, 2 – нет)*  *Ввод:* |

**Программный код**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <locale>

/\*Константы длины\*/

#define MAX\_LENGTH\_NAME 50

#define MAX\_LENGTH\_NUMBER\_CUR 20

#define MAX\_LENGTH\_NUMBER\_TECHNICALCERTIFICATION 40

#define LENGTH\_DATE 12

#define MAX\_LENGTH\_PHONE 20

#define MAX\_LENGTH\_OFFICEGIBDD 60

/\*Путь к файлу для работы\*/

#define WAY\_FILE "StructFile.txt"

struct car {

char name[MAX\_LENGTH\_NAME];

char NumberCur[MAX\_LENGTH\_NUMBER\_CUR];

char NumberTechnicalCertificate[MAX\_LENGTH\_NUMBER\_TECHNICALCERTIFICATION];

char date[LENGTH\_DATE];

char phone[MAX\_LENGTH\_PHONE];

char OfficeGIBDD[MAX\_LENGTH\_OFFICEGIBDD];

};

/\*код\*/

void main() {

setlocale(0, "rus");

int request = 0;

printf("Выбирете тип запроса:\n");

EnterRequest();

}

/\*функции\*/

/\*общие\*/

void EntryFile(FILE \*file, struct car Auto);

void EnterNumberOrSymbol(char \*word, int max\_length);

void RepeatRequest(int exist\_file);

void ChoiceFunction(int request);

void EnterOfficeGIBDD(char \*word);

void EnterPhone(char \*word);

void EnterName(char \*word);

void EnterDate(char \*word);

void FormStructText();

void EnterRequest();

void FormRequest();

void FormStruct();

int StringLength(char \*StringText);

int PositionTub(char \*word);

int СhoiceRequest(int max);

/\*1 запрос\*/

void EnterWord(char \*word, int max\_length, int \*length);

void EnterStruct(struct car \*Auto);

void EnterStructs();

int EnterCountStructs();

/\*2 запрос\*/

void PassingFile(int request, char \*search\_text);

void StringCopy(char \*new\_text, char \*text);

void SearchText();

int SearchPunct(char \*text\_file, int request, char \*search\_text);

/\*3 запрос\*/

void OutputStruct();

/\*4 запрос\*/

void SwapStructs(struct car &Auto1, struct car &Auto2);

void AddStruct(struct car \*Auto, char \*text);

void ClearStruct(struct car \*Auto);

void AddStructs(FILE \*StructFile);

void SortingFile();

int StringLength(char \*StringText) {

int i;

for (i = 0; StringText[i]; i++);

return i;

}

void FormRequest() {

printf("1)Внесение структуры в файл\n");

printf("2)Поиск структуры в файле\n");

printf("3)Вывод стуктур на экран\n");

printf("4)Сортировка данных в файле\n");

printf("5)Завершение работы программы\n");

}

int СhoiceRequest(int max) {

int request = 0;

bool error = true;

do {

scanf("%d", &request);

if (getchar() != '\n') {

printf("Ошибка ввода. Повторите попытку: ");

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

}

else (request < 1 || request > max) ? printf("Ошибка. Данный запрос отсутствует. Повторите попытку: ") : error = false;

} while (error);

return request;

}

void ChoiceFunction(int request) {

switch (request) {

case 1:

EnterStructs();

break;

case 2:

SearchText();

break;

case 3:

OutputStruct();

break;

case 4:

SortingFile();

break;

default:

exit(0);

}

}

void EnterStructs() {

int count = 0;

printf("Введите количество структур: ");

count = EnterCountStructs();

struct car \*Auto = (struct car \*)malloc(sizeof(struct car \*) \* count);

FormStruct();

FILE \*StructFile = fopen(WAY\_FILE, "a");

if (StructFile == NULL) {

fclose(StructFile);

StructFile = fopen(WAY\_FILE, "w");

}

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

printf("Ввод %d структуры:\n", i + 1);

EnterStruct(&Auto[i]);

EntryFile(StructFile, Auto[i]);

}

fclose(StructFile);

}

void RepeatRequest(int exist\_file) {

exist\_file ? printf("Будет еще запрос?(1-да; 2-нет)\n") : printf("Файл не найден. Будут производиться другие запросы?(1-да; 2-нет)\n");

printf("Ввод: ");

int request = СhoiceRequest(2);

request == 1 ? EnterRequest() : "";

}

void EnterRequest() {

FormRequest();

printf("Ввод: ");

int request = СhoiceRequest(5);

ChoiceFunction(request);

RepeatRequest(1);

}

void EntryFile(FILE \*file, struct car Auto) {

fprintf(file, "%s\t", Auto.name);

fprintf(file, "%s\t", Auto.NumberCur);

fprintf(file, "%s\t", Auto.NumberTechnicalCertificate);

fprintf(file, "%s\t", Auto.date);

fprintf(file, "%s\t", Auto.phone);

fprintf(file, "%s\n\0", Auto.OfficeGIBDD);

fclose(file);

}

int EnterCountStructs() {

int count = 0;

bool error = true;

do {

scanf("%d", &count);

if (getchar() != '\n') {

printf("Ошибка ввода. Повторите попытку: ");

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

}

else (count < 1) ? printf("Ошибка. Количество элементов положительно. Повторите попытку: ") : error = false;

} while (error);

return count;

}

void FormStruct() {

printf("Введите структуру в форме:\n");

FormStructText();

}

void FormStructText() {

printf("\t1)имя(без инициалов)\n");

printf("\t2)номер автомобиля\n");

printf("\t3)номер техпаспорта\n");

printf("\t4)дата рождения(в виде дд:мм:гггг)\n");

printf("\t5)телефон\n");

printf("\t6)отделение регистрации ГИБДД\n");

}

void EnterStruct(struct car \*Auto) {

printf("\tВведите имя: ");

EnterName(Auto->name);

printf("\tВведите номер автомобиля: ");

EnterNumberOrSymbol(Auto->NumberCur, MAX\_LENGTH\_NUMBER\_CUR);

printf("\tВведите номер техпаспорта: ");

EnterNumberOrSymbol(Auto->NumberTechnicalCertificate, MAX\_LENGTH\_NUMBER\_TECHNICALCERTIFICATION);

printf("\tВведите дату рождения: ");

EnterDate(Auto->date);

printf("\tВведите номер телефона: ");

EnterPhone(Auto->phone);

printf("\tВведите отделение регистрации ГИБДД: ");

EnterOfficeGIBDD(Auto->OfficeGIBDD);

}

void EnterName(char \*word) {

int length = 0;

EnterWord(word, MAX\_LENGTH\_NAME, &length);

for (int i = 0; i < length; i++) {

if (!(CHECK\_NAME)) {

printf("\tОшибка ввода имени. Повторите попытку: ");

EnterName(word);

break;

}

}

}

void EnterNumberOrSymbol(char \*word, int max\_length) {

int length = 0;

EnterWord(word, max\_length, &length);

for (int i = 0; i < length; i++) {

if (!(CODE\_NUMBER\_OR\_SYMBOL)) {

printf("\tОшибка ввода. Повторите попытку: ");

EnterNumberOrSymbol(word, max\_length);

break;

}

}

}

void EnterDate(char \*word) {

int length = 0;

EnterWord(word, LENGTH\_DATE, &length);

if (!(CHECK\_CODE\_DATE)) {

printf("\tОшибка ввода даты. Повторите попытку: ");

EnterDate(word);

}

}

void EnterPhone(char \*word) {

int length = 0,

brace\_left = 0,

brace\_right = 0;

EnterWord(word, MAX\_LENGTH\_PHONE, &length);

if (!(word[0] == '+' || ((int)word[0] >= 48 && (int)word[0] <= 57))) {

printf("\tОшибка ввода номера телефона. Повторите попытку: ");

EnterPhone(word);

}

for (int i = 1; i < length; i++) {

if (word[i] == '(') brace\_left++;

else if (word[i] == ')')brace\_right++;

else if (!(CODE\_NUMBER\_SYMBOL || word[i] == '(' || word[i] == ')' || word[i] == ' ' || FINISH\_SYMBOL) ||

(word[i] == ')' && brace\_left == 0)) {

printf("\tОшибка ввода номера телефона. Повторите попытку: ");

EnterPhone(word);

break;

}

if (brace\_left > 1 || brace\_right > 1) {

printf("\tОшибка ввода номера телефона. Повторите попытку: ");

EnterPhone(word);

break;

}

}

}

void EnterOfficeGIBDD(char \*word) {

int length = 0;

EnterWord(word, MAX\_LENGTH\_OFFICEGIBDD, &length);

for (int i = 0; i < length; i++) {

if (!(CHECK\_OFFICE\_GIBDD)) {

printf("\tОшибка ввода отделения регистрации ГИБДД. Повторите попытку: ");

EnterOfficeGIBDD(word);

break;

}

}

}

void EnterWord(char \*word, int max\_length, int \*length) {

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

word[i] = '\0';

fgets(word, max\_length, stdin);

\*length = StringLength(word);

word[\*length - 1] = '\0';

}

/\*2 запрос\*/

void SearchText() {

printf("Будут вноситься структуры в файл перед поиском?(1-да 2-нет)\n");

printf("Ввод: ");

int request = СhoiceRequest(2);

if (request == 1)

EnterStructs();

printf("Выберете критерий поиска:\n");

FormStructText();

printf("Ввод: ");

request = СhoiceRequest(6);

printf("Введите текст для поиска: ");

char search\_text[MAX\_LENGTH\_OFFICEGIBDD] = "";

switch (request) {

case 1:

EnterName(search\_text);

break;

case 2:

EnterNumberOrSymbol(search\_text, MAX\_LENGTH\_NUMBER\_CUR);

break;

case 3:

EnterNumberOrSymbol(search\_text, MAX\_LENGTH\_NUMBER\_TECHNICALCERTIFICATION);

break;

case 4:

EnterDate(search\_text);

break;

case 5:

EnterPhone(search\_text);

break;

case 6:

EnterOfficeGIBDD(search\_text);

break;

}

PassingFile(request, search\_text);

}

void PassingFile(int request, char \*search\_text) {

FILE \*StructFile = fopen(WAY\_FILE, "r");

if (StructFile == NULL) {

fclose(StructFile);

RepeatRequest(0);

}

else {

char text\_file[MAX\_LENGTH\_STRING] = "",

buf\_text\_file[MAX\_LENGTH\_STRING] = "";

int coincidence = 0,

count = 0;

while (fgets(text\_file, MAX\_LENGTH\_STRING, StructFile) != NULL) {

text\_file[StringLength(text\_file) - 1] = '\0';

StringCopy(buf\_text\_file, text\_file);

coincidence = SearchPunct(text\_file, request, search\_text);

if (coincidence) {

printf("%s\n", buf\_text\_file);

count++;

}

}

if (!count) printf("Совпадений не найдено!\n");

}

fclose(StructFile);

}

void StringCopy(char \*new\_text, char \*text) {

for (int i = 0; i < MAX\_LENGTH\_STRING; i++)

new\_text[i] = '\0';

int length = StringLength(text);

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

new\_text[i] = text[i];

}

int SearchPunct(char \*text\_file, int request, char \*search\_text) {

int length = StringLength(text\_file),

position\_tub = 0,

coincidence = 1;

for (int i = 0; i < (request - 1); i++) {

position\_tub = PositionTub(text\_file);

text\_file = text\_file + position\_tub + 1;

}

if (StringLength(search\_text) == PositionTub(text\_file))

for (int i = 0; text\_file[i] != '\n' || text\_file[i] != '\0' || text\_file[i] != '\t'; i++) {

if ((text\_file[i] == '\n' || text\_file[i] == '\0' || text\_file[i] == '\t') && i > 0) {

break;

}

if (text\_file[i] != search\_text[i] ||

((search\_text[i] == '\n' || search\_text[i] == '\0') && i < StringLength(search\_text))) {

coincidence = 0;

break;

}

}

else coincidence = 0;

if (coincidence) return 1; else return 0;

}

int PositionTub(char \*word) {

int length = StringLength(word);

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

if (word[i] == '\t')

return i;

return NULL;

}

/\*3 запрос\*/

void OutputStruct() {

FILE \*StructFile = fopen(WAY\_FILE, "r");

if (StructFile == NULL) {

RepeatRequest(0);

}

else {

char text\_file[MAX\_LENGTH\_STRING] = "";

while (fgets(text\_file, MAX\_LENGTH\_STRING, StructFile) != NULL)

printf("%s\n", text\_file);

fclose(StructFile);

}

fclose(StructFile);

}

/\*4 запрос\*/

void SortingFile() {

FILE \*StructFile = fopen(WAY\_FILE, "r");

if (StructFile == NULL) {

fclose(StructFile);

RepeatRequest(0);

}

else {

AddStructs(StructFile);

//fclose(StructFile);

}

fclose(StructFile);

}

void AddStructs(FILE \*StructFile) {

char text\_file[MAX\_LENGTH\_STRING] = "";

int count = 0,

max\_length\_name = 0;

while (fgets(text\_file, MAX\_LENGTH\_STRING, StructFile) != NULL) count++;

struct car \*Auto = (struct car \*)calloc(count + 1, sizeof(struct car \*));

rewind(StructFile);

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

fgets(text\_file, MAX\_LENGTH\_STRING, StructFile);

AddStruct(&Auto[i], text\_file);

}

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

if (StringLength(Auto[i].name) > max\_length\_name) max\_length\_name = StringLength(Auto[i].name);

for (int k = 0; k < max\_length\_name; k++)

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

for (int j = i + 1; j < count; j++) {

if (Auto[i].name[k] && Auto[j].name[k]) {

if (!k)

if (((int)Auto[i].name[0] >(int)Auto[j].name[0]))

SwapStructs(Auto[i], Auto[j]);

if (((int)Auto[i].name[k] >(int)Auto[j].name[k]) &&

((int)Auto[i].name[k - 1] == (int)Auto[j].name[k - 1]))

SwapStructs(Auto[i], Auto[j]);

}

}

}

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

EntryFile(StructFile, Auto[i]);

fclose(StructFile);

}

void SwapStructs(struct car &Auto1, struct car &Auto2) {

struct car tmp;

tmp = Auto2;

Auto2 = Auto1;

Auto1 = tmp;

}

void AddStruct(struct car \*Auto, char \*text) {

int position\_tub = 0;

ClearStruct(Auto);

for (int i = 0; i < 6; i++) {

position\_tub = PositionTub(text);

switch (i) {

case 0:

for (int i = 0; i < position\_tub; i++)

Auto->name[i] = text[i];

break;

case 1:

for (int i = 0; i < position\_tub; i++)

Auto->NumberCur[i] = text[i];

break;

case 2:

for (int i = 0; i < position\_tub; i++)

Auto->NumberTechnicalCertificate[i] = text[i];

break;

case 3:

for (int i = 0; i < position\_tub; i++)

Auto->date[i] = text[i];

break;

case 4:

for (int i = 0; i < position\_tub; i++)

Auto->phone[i] = text[i];

break;

case 5:

for (int i = 0; i < position\_tub; i++)

Auto->OfficeGIBDD[i] = text[i];

break;

}

text = text + position\_tub + 1;

}

}

void ClearStruct(struct car \*Auto) {

for (int i = 0; i < MAX\_LENGTH\_NAME; i++)

Auto->name[i] = '\0';

for (int i = 0; i < MAX\_LENGTH\_NUMBER\_CUR; i++)

Auto->NumberCur[i] = '\0';

for (int i = 0; i < MAX\_LENGTH\_NUMBER\_TECHNICALCERTIFICATION; i++)

Auto->NumberTechnicalCertificate[i] = '\0';

for (int i = 0; i < LENGTH\_DATE; i++)

Auto->date[i] = '\0';

for (int i = 0; i < MAX\_LENGTH\_PHONE; i++)

Auto->phone[i] = '\0';

for (int i = 0; i < MAX\_LENGTH\_OFFICEGIBDD; i++)

Auto->OfficeGIBDD[i] = '\0';

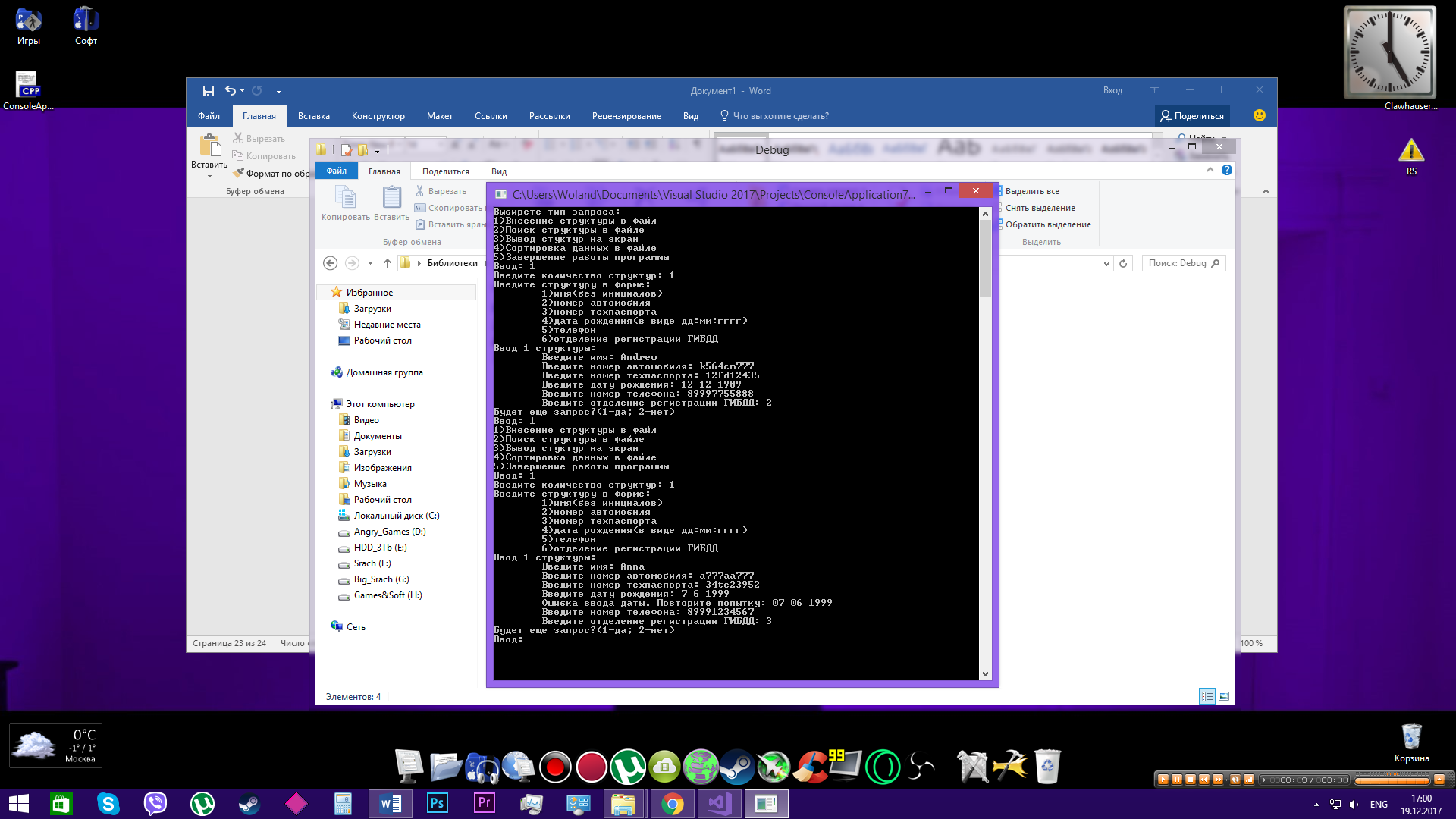
}**Демонстрация работы программы**

Рис.1 – Добавление новых владельцев в картотеку

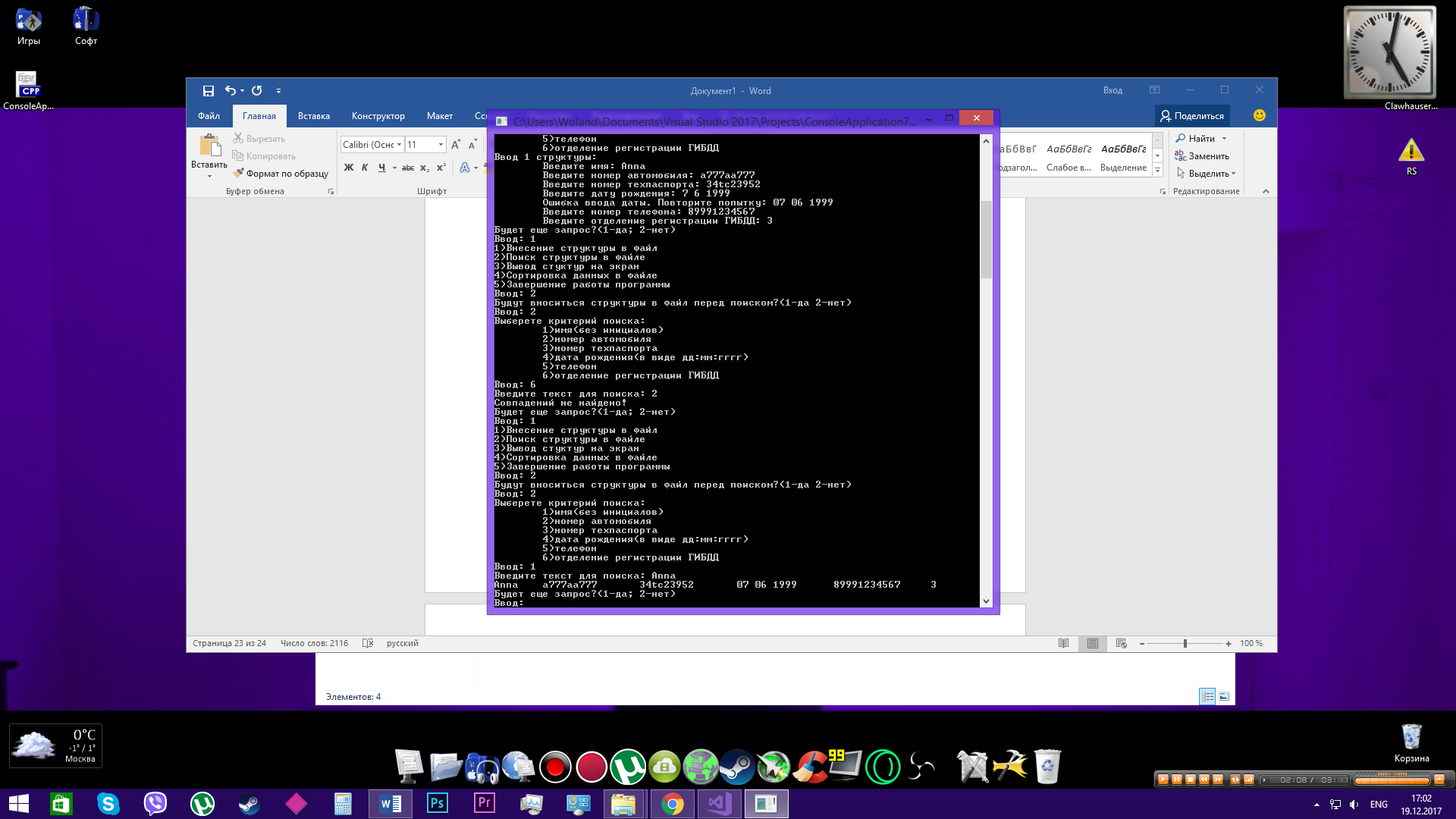


Рис.2 – Поиска владельца по информации и вывод его на экран

## **Заключение**

В данной лабораторной работе были освоены способы задания структур и методы обработки данных структурного типа. Так же было изучены: инициализация и присваивание структур, понятия массива структур, указателей на структуру и операции над указателями.