Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

Кафедра системного анализа и телекоммуникаций

**ЛАБОРАТОРНАЯ РАБОТА № 3**

по дисциплине «Основы алгоритмизации и программирования»

Вариант 2

на тему:

«Работа со структурами»

Выполнил  
студент группы КТбо 1‑2 Акимов И. Н.

Принял

доцент кафедры САиТ Лапшин В. С.

Таганрог 2024

**Содержание**

Содержание 2

Техническое задание 3

Цель работы 3

Задача 3

Ход работы 4

Алгоритмы 4

Блок-схемы 6

Написание программы 7

Примеры работы программы 8

Вывод 12

Листинг (Приложение а) 13

**Техническое задание**

**Цель работы**

Цель данной лабораторной работы состоит в том, чтобы научиться работать со сложными типами данных, каковыми являются структуры данных.

**Задача**

Вариант 2.

Написать программу, которая записывает с клавиатуры в файл структуру согласно выданному варианту задания. В качестве разделителя полей структуры использовать символ табуляции.

В программе реализовать:

а) дополнение существующего массива структур новыми структурами;

б) поиск структуры с заданным значением выбранного элемента;

в) вывод на экран содержимого массива структур;

г) упорядочение массива структур по заданному полю (элементу), например государство по численности.

Содержание структуры:

«Покупатель»: фамилия, имя, отчество, город, улица, номер дома, номер квартиры, номер кредитной карточки, код покупки.

**Ход работы**

**Алгоритмы**

В программе применяется система меню с выбором реализации функций: запись новых пользователей, поиск пользователя по заданному параметру и значению, сортировка по параметру, вывод списка покупателей в консоль, вывод списка покупателей в файл, очистка данных, чтение существующих данных. Далее описываются алгоритмы функций.

Для каждой функции передаются параметры: customer\* list\_customers – массив структур с хранимыми значениями, unsigned count – счетчик количества записей.

Запись новых пользователей – int record\_struct:

Переменные: FILE\* input – указатель на файл для записи данных; Int new\_count – счетчик ввода новых пользователей.

1. Ввод числа новых пользователей, для ограничения кол-ва вызова функции заполнения структуры.
2. Циклическое заполнение элементов массива структур.
3. Вывод элементов в файл исходных данных для сохранения и последующего обращения к данным.

Возвращает значение счетчика количества исходных данных.

Поиск пользователя по заданному параметру и значению - void search:

Переменные:

char s\_goal[MAX\_INFO\_SIZE] - дополнительная переменная для сравнения строчных параметров, unsigned int int\_goal - дополнительная переменная для сравнения числовых параметров, unsigned int Param – значение выбранного пункта меню, unsigned int res – счетчик результата поиска.

1. Ввод пункта меню, выбор параметра для поиска.
2. Если параметр числовой, ввод числового значения для поиска, если нет – ввод символьного.
3. Циклический перебор массива данных.
4. Для каждой итерации счетчик результата поиска обновляется, в соответствии с выбранным параметром, идет сравнение введенного значения поиска с существующим значением элемента массива.
5. Если есть совпадение, счетчик результата обновляется, выводится элемент, имеющий совпадение. Иначе – сообщение об отсутствии совпадений.

Сортировка по параметру void sorting:

1. Ввод пункта меню, выбор параметра для поиска.
2. В зависимости от выбора, сортировка функцией qsort из библиотеки <stdlib.h>.
3. Вывод отсортированного массива данных.

Вывод списка покупателей в консоль void print:

1. Циклический вывод каждого элемента массива данных.

Вывод списка покупателей в файл void output\_file:

Параметр: FILE\* file – указатель на файл для вывода данных.

1. Циклический вывод каждого элемента массива данных.

Очистка данных clear\_files: Вызов открытия файлов с параметром перезаписи данных. Т.к. новые данные не записываются, файлы остаются пустыми.

Чтение существующих данных record\_struct:

Переменные: FILE\* output – указатель на файл с исходными данными, unsigned count – счетчик имеющихся данных, char stroke[256] – буфер считываемой строки.

1. Пока не закончатся считываемые значение, заполнение элементов массива данных из считываемой строки.

**Блок-схемы**

**Написание программы**

Для удобства были инициализированы глобальные переменные MAX\_INFO\_SIZE = 20 – максимальный размер строки данных в структуре и MAX\_LIST\_CUSTOMERS\_SIZE = 20 - максимальный размер списка покупателей

Для выполнения задачи, необходимо хранить значения, в данном случае, используется структура struct customer с параметрами: char famil[MAX\_INFO\_SIZE] – Фамилия, char name[MAX\_INFO\_SIZE] – Имя, char otch[MAX\_INFO\_SIZE] – Отчество, char city[MAX\_INFO\_SIZE] – Город, char street[MAX\_INFO\_SIZE] – Улица, unsigned int house\_num – Дом, unsigned int flat\_num – Квартира, unsigned int credit\_card – Кредитка, unsigned int code\_shop - Код покупки. Заполненные структуры хранятся в массиве структур с динамическим выделением customer\* list\_customers.

В начале выполнения программы, происходит считывание исходных данных с записью их количества в переменную count, чтобы иметь представление о длине списка заполненных в массиве данных элементов. В последующем, переменная count используется как длина массива.

Для реализации функционала меню выбора, используется команда switch, которая обрабатывает вводимое значение пункта меню, и в зависимости от него вызывает нужную функцию.

По завершению работы одной функции, есть необходимость заново вызывать главную функцию main(), чтобы иметь возможность вызвать нереализованную функцию.

Завершение программы происходит путем выбора соответствующего пункта меню, при этом происходит освобождение занимаемой массивом данных памяти.

**Примеры работы программы**

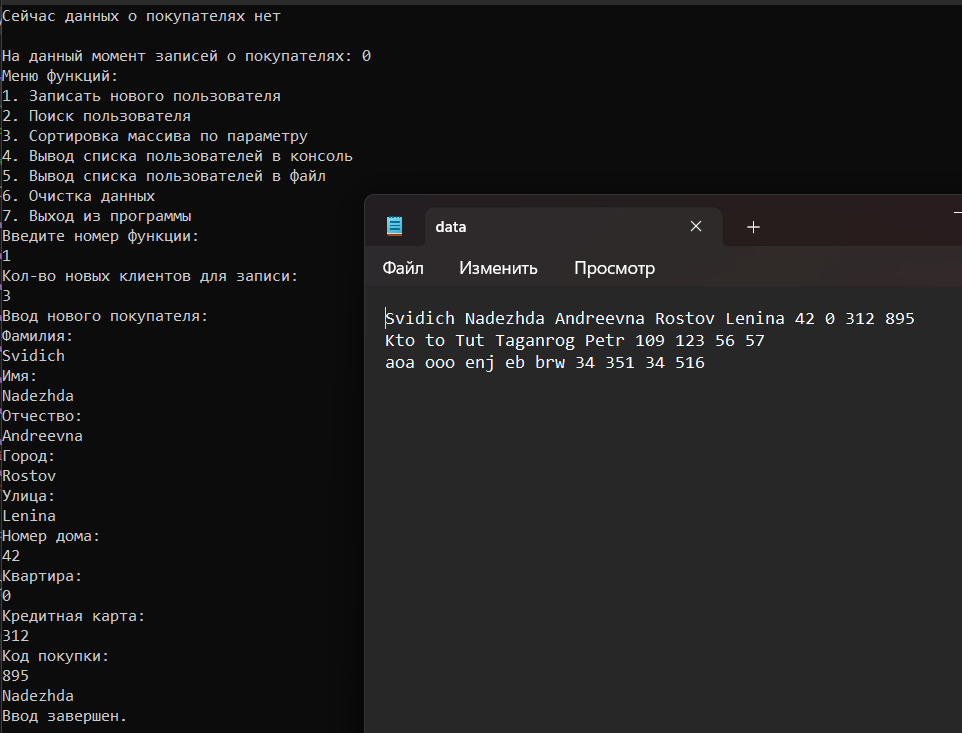


Рисунок 1 – Выполнение функции записи покупателя.

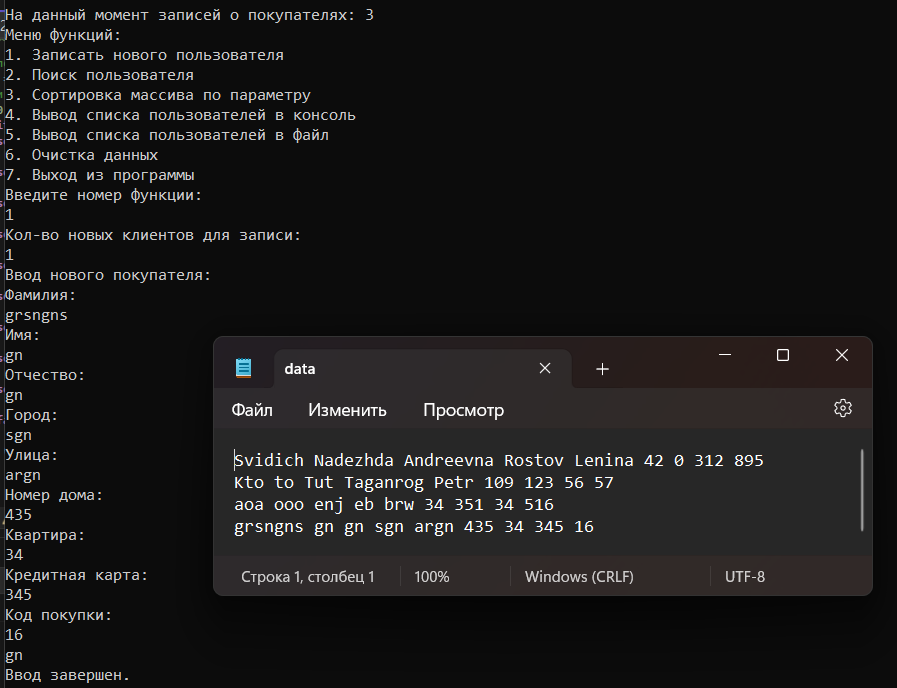


Рисунок 2 – Выполнение функции записи покупателя при уже существующих данных (добавление новых записей).

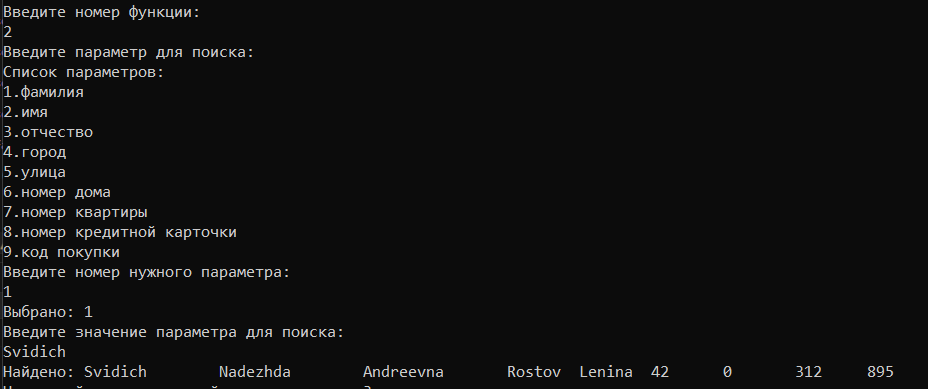


Рисунок 3 – Выполнение функции поиска по параметру и его значению.

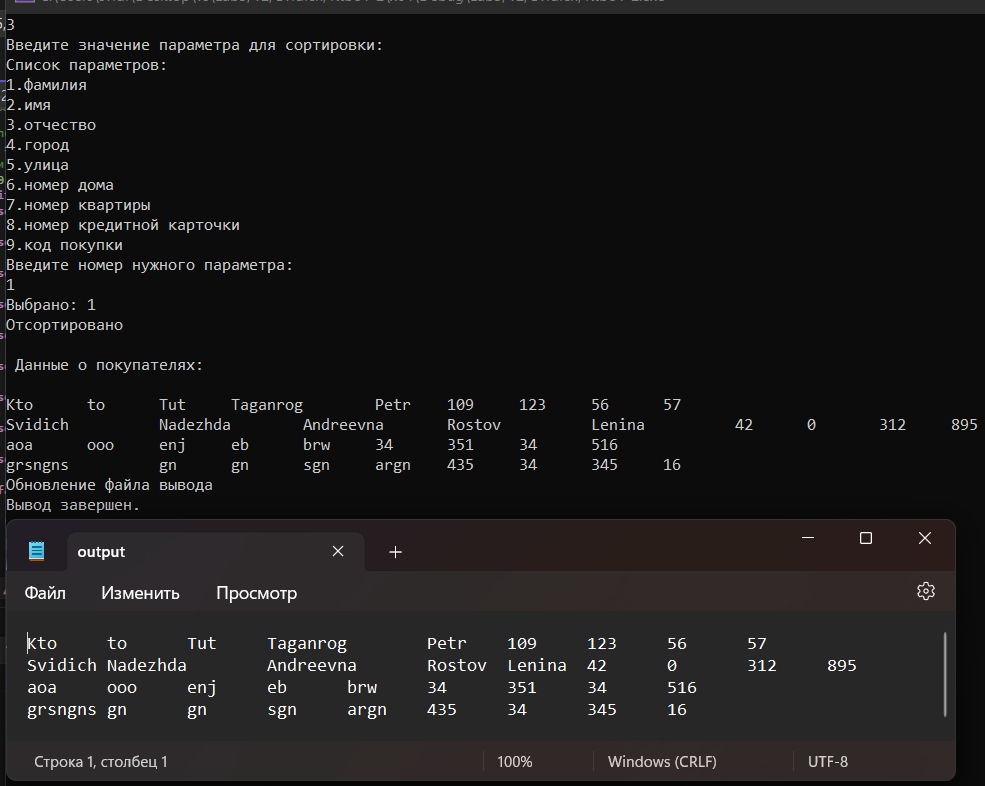


Рисунок 4 – Выполнение функции сортировки по параметру.

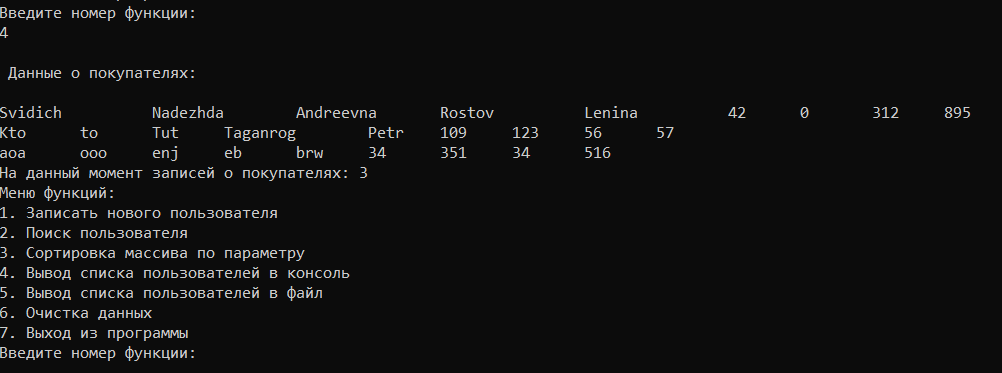


Рисунок 5 – Выполнение вывода данных в консоль.

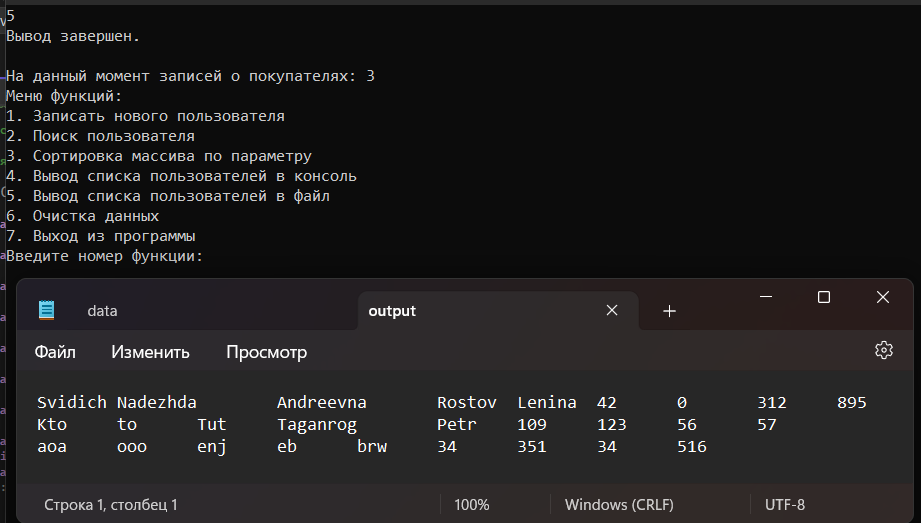


Рисунок 6 – Выполнение Вывода данных в файл.

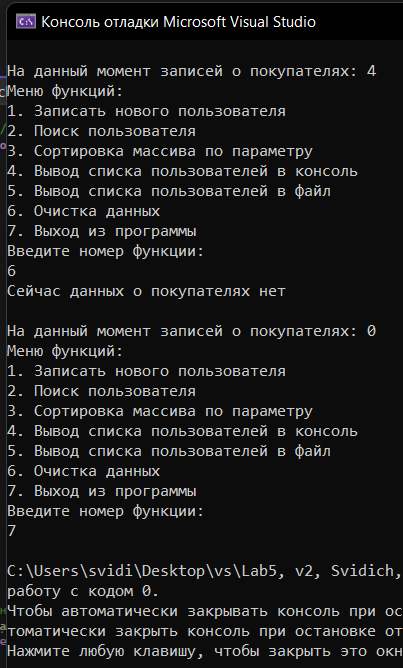


Рисунок 7 – Очищение данных и завершение программы.

**Вывод**

# Изучена работа со сложными типами данных, каковыми являются структур данных.

# **Листинг (Приложение а)**

// Copyright (c) 2023, Свидич Н.А., ЮФУ ИКТИБ, Ктбо1-2

//

// АиП Лабораторная работа №3. Вариант № 2. Работа со структурами.

//

// Последнее обновление: 22.12.2023

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#pragma warning(disable : 4996)

#define MAX\_INFO\_SIZE 20 // Макс размер строки данных в структуре

#define MAX\_LIST\_CUSTOMERS\_SIZE 20 // Макс размер списка покупателей

// ||

// Описание структуры хранения данных

struct customer {

char famil[MAX\_INFO\_SIZE]; // Фамилия

char name[MAX\_INFO\_SIZE]; // Имя

char otch[MAX\_INFO\_SIZE]; // Отчетсво

char city[MAX\_INFO\_SIZE]; // Город

char street[MAX\_INFO\_SIZE]; // Улица

unsigned int house\_num; // Дом

unsigned int flat\_num; // Квартира

unsigned int credit\_card; // Кредтка

unsigned int code\_shop; // Код покупки

};

// Вывод пунктов меню функций

int menu\_func() {

int choise;

puts("\nМеню функций: ");

puts("1. Записать нового пользователя \n2. Поиск пользователя \n3. Сортировка массива по параметру \n4. Вывод списка пользователей в консоль\n5. Вывод списка пользователей в файл \n6. Очистка данных \n7. Выход из программы");

puts("Введите номер функции: ");

scanf("%i", &choise);

return choise;

}

// Функция записи новой структуры. Работает корректно, проверено путем записи в файл

int record\_struct(customer\* customer, unsigned id) {

FILE\* input = fopen("..\\data.txt", "a");

int new\_count;

puts("Кол-во новых клиентов для записи: ");

scanf("%i", &new\_count);

if (input == NULL) perror("Ошибка открытия файла data.txt для записи и хранения данных о покупателях");

else {

for (int i = 0; i < new\_count; i++) {

puts("Ввод нового покупателя: ");

puts("Фамилия: ");

scanf("%s", &customer[id + i].famil);

puts("Имя: ");

scanf("%s", &customer[id + i].name);

puts("Отчество: ");

scanf("%s", &customer[id + i].otch);

puts("Город: ");

scanf("%s", &customer[id + i].city);

puts("Улица: ");

scanf("%s", &customer[id + i].street);

puts("Номер дома: ");

scanf("%u", &customer[id + i].house\_num);

puts("Квартира: ");

scanf("%u", &customer[id + i].flat\_num);

puts("Кредитная карта: ");

scanf("%u", &customer[id + i].credit\_card);

puts("Код покупки: ");

scanf("%u", &customer[id + i].code\_shop);

// Запись покупателя в файл исходных данных, чтобы не терять массив при новом запуске программы (используется каждый раз, после завершения выбранной в меню функции)

fprintf(input, "%s %s %s %s %s %u %u %u %u \n",

customer[id + i].famil, customer[id + i].name, customer[id + i].otch, customer[id + i].city, customer[id + i].street,

customer[id + i].house\_num, customer[id + i].flat\_num, customer[id + i].credit\_card, customer[id + i].code\_shop);

puts("Ввод завершен.\n");

}

}

fclose(input);

return id;

}

// Чтение уже существующих данных. Работает, проверено выводом кол-ва сущ записей при новом запуске программы

int read\_data(customer\* customr) {

FILE\* output = fopen("..\\data.txt", "r");

unsigned count = 0;

if (output == NULL) perror("Ошибка открытия файла data.txt для записи информации о покупателях в структуры");

else {

char stroke[256];

int i = 0;

while (fgets(stroke, sizeof(stroke), output) != NULL) {

sscanf(stroke, "%s %s %s %s %s %u %u %u %u \n", &customr[i].famil, &customr[i].name, &customr[i].otch, &customr[i].city, &customr[i].street, &customr[i].house\_num, &customr[i].flat\_num, &customr[i].credit\_card, &customr[i].code\_shop);

count++;

i++;

}

}

if (count == 0) puts("Сейчас данных о покупателях нет\n");

fclose(output);

return count;

}

// Функция вывода структур в консоль. Работает, проверено на практике - есть вывод

void print(customer\* customr, int count) {

if (customr == NULL) printf("Нет покупателей");

else {

puts("\n Данные о покупателях: \n");

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

printf("%s \t %s \t %s \t %s \t %s \t %d \t %d \t %d \t %d \n", &customr[i].famil, &customr[i].name, &customr[i].otch, &customr[i].city, &customr[i].street, customr[i].house\_num, customr[i].flat\_num, customr[i].credit\_card, customr[i].code\_shop);

}

}

}

// Функция вывода структур в файл. Работает, проверено путем записи в файл

void output\_file(customer\* customer, unsigned count) {

FILE\* file = fopen("..\\output.txt", "w");

if (file == NULL) perror("Ошибка открытия файла");

else {

if (customer == NULL) fprintf(file, "Нет покупателей");

else {

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

fprintf(file, "%s\t%s\t%s\t%s\t%s\t%u\t%u\t%u\t%u \n", customer[i].famil, customer[i].name, customer[i].otch, customer[i].city, customer[i].street, customer[i].house\_num, customer[i].flat\_num, customer[i].credit\_card, customer[i].code\_shop);

}

puts("Вывод завершен.\n");

}

}

fclose(file);

}

// Функция вывода выбора параметров структуры

unsigned int menu\_param() {

unsigned int choise;

puts("Список параметров: ");

puts("1.фамилия \n2.имя \n3.отчество \n4.город \n5.улица \n6.номер дома \n7.номер квартиры \n8.номер кредитной карточки \n9.код покупки");

puts("Введите номер нужного параметра: ");

do {

scanf("%u", &choise);

printf("Выбрано: %u\n", choise);

} while (choise < 1 || choise > 9);

return choise;

}

// Функция поиска по значению параметра. Работает, проверено на практике - есть вывод

void search(customer\* customer, int count) {

char s\_goal[MAX\_INFO\_SIZE]; // Доп переменная для сравнения по строкам

unsigned int int\_goal = 0, param, res; // Для числовых

puts("Введите параметр для поиска: ");

param = menu\_param();

puts("Введите значение параметра для поиска: ");

// По числу параметра определяем, сравнивать числа или строки

if (param >= 6)

scanf("%s", &int\_goal);

else scanf("%s", &s\_goal);

// проход по массиву данных

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

// Если хотя бы совпадение в параметре, вывод пользователя

res = 0;

switch (param) {

case 1: if (strcmp(customer[i].famil, s\_goal) == 0) res = 1;

break;

case 2: if (strcmp(customer[i].name, s\_goal) == 0) res = 1;

break;

case 3: if (strcmp(customer[i].otch, s\_goal) == 0) res = 1;

break;

case 4: if (strcmp(customer[i].city, s\_goal) == 0) res = 1;

break;

case 5: if (strcmp(customer[i].street, s\_goal) == 0) res = 1;

break;

case 6: if (customer[i].house\_num == int\_goal) res = 1;

break;

case 7: if (customer[i].flat\_num == int\_goal) res = 1;

break;

case 8: if (customer[i].credit\_card == int\_goal) res = 1;

break;

case 9:if (customer[i].code\_shop == int\_goal) res = 1;

break;

default: break;

}

if (res == 1) printf("Найдено: %s\t%s\t%s\t%s\t%s\t%u\t%u\t%u\t%u \n",

customer[i].famil, customer[i].name, customer[i].otch, customer[i].city, customer[i].street, customer[i].house\_num, customer[i].flat\_num, customer[i].credit\_card, customer[i].code\_shop);

}

if (res == 0) puts("Совпадений не найдено");

}

// Функции для указания в qsort, отдельно по параметрам для возможности выбора

int name\_cmp(const void\* a, const void\* b) {

return strcmp( ((customer\*)a)->name, ((customer\*)b)->name);

}

int famil\_cmp(const void\* a, const void\* b) {

return strcmp( ((customer\*)a)->famil, ((customer\*)b)->famil);

}

int otch\_cmp(const void\* a, const void\* b) {

return strcmp(((customer\*)a)->otch, ((customer\*)b)->otch);

}

int city\_cmp(const void\* a, const void\* b) {

return strcmp(((customer\*)a)->city, ((customer\*)b)->city);

}

int street\_cmp(const void\* a, const void\* b) {

return strcmp(((customer\*)a)->street, ((customer\*)b)->street);

}

int house\_cmp(const void\* a, const void\* b) {

return ((customer\*)a)->house\_num - ((customer\*)b)->house\_num;

}

int flat\_cmp(const void\* a, const void\* b) {

return ((customer\*)a)->flat\_num - ((customer\*)b)->flat\_num;

}

int credit\_cmp(const void\* a, const void\* b) {

return ((customer\*)a)->credit\_card - ((customer\*)b)->credit\_card;

}

int code\_cmp(const void\* a, const void\* b) {

return ((customer\*)a)->code\_shop - ((customer\*)b)->code\_shop;

}

// Конец функций - указателей

// Функция сортировки по выбранному параметру, сортирует строки по алфавиту и числовые данные по возрастанию. Работает - проверено выводом в косноль и файл

void sorting(customer\* customr, int count) {

puts("Введите значение параметра для сортировки: ");

//unsigned int param = menu\_param(); // Выбор параметра

// В зависимости от параметра, идет сортировка

switch (menu\_param()) {

case 1: qsort(customr, count, sizeof(customer), famil\_cmp);

break;

case 2: qsort(customr, count, sizeof(customer), name\_cmp);

break;

case 3: qsort(customr, count, sizeof(customer), otch\_cmp);

break;

case 4: qsort(customr, count, sizeof(customer), city\_cmp);

break;

case 5: qsort(customr, count, sizeof(customer), street\_cmp);

break;

case 6: qsort(customr, count, sizeof(customer), house\_cmp);

break;

case 7: qsort(customr, count, sizeof(customer), flat\_cmp);

break;

case 8: qsort(customr, count, sizeof(customer), credit\_cmp);

break;

case 9: qsort(customr, count, sizeof(customer), code\_cmp);

break;

default: break;

}

// Вывод отсортированного массива, файл исходных данных не трогается для сравнения результата

puts("Отсортировано");

print(customr, count);

puts("Обновление файла вывода");

output\_file(customr, count);

}

// Функция очищения файлов

void clear\_files() {

FILE\* data = fopen("..\\data.txt", "w");

FILE\* output = fopen("..\\output.txt", "w");

fclose(data);

fclose(output);

}

int main()

{

setlocale(LC\_ALL, "Russian");

customer\* list\_customers = (customer\*)malloc(sizeof(customer) \* MAX\_LIST\_CUSTOMERS\_SIZE); // Массив структур с динамическим выделением памяти, макс. кол-во записей - MAX\_LIST\_CUSTOMERS\_SIZE

customer\* customer = { 0 }; // Отдельный элемент массива структур

unsigned count = read\_data(list\_customers); // Подсчет записей в массиве структур

printf("На данный момент записей о покупателях: %u", count); // Сколько записей в исходных данных

// Меню функций

switch (menu\_func()) {

case 1: count = record\_struct(list\_customers, count); // Запись нового пользователя

break;

case 2:search(list\_customers, count); // Поиск записи по параметру

break;

case 3:sorting(list\_customers, count); // Сортировка массива структур

break;

case 4:print(list\_customers, count); // Вывод структуры в консоль

break;

case 5:output\_file(list\_customers, count); // Вывод структуры в файл

break;

case 6: clear\_files(); // Вызов очистки файлов

break;

case 7: // Если выбран выход из программы, очищаем память массива структур и прекращаем чтение кода

free(list\_customers);

return 0;

default: menu\_func(); // Если введен номер, которого нет в списке, выводим список снова

}

main(); // Рекурсия главной функции для нового вызова меню и реализации других возможностей программы

}