МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ФГБОУ ВО «ВГТУ»)

Факультет информационных технологий и компьютерной безопасности

Кафедра Системы автоматизированного проектирования и информационных систем

КУРСОВОЙ ПРОЕКТ

по дисциплине Основы программирования и алгоритмизации

Тема: Разработка программы для работы с файловой базой данных «Техническая поддержка интернет-провайдера».

**Расчетно-пояснительная записка**

Разработал студент А. Д. Мальков

Подпись, дата Инициалы, фамилия

Руководитель О. В. Минакова Подпись, дата Инициалы, фамилия

Нормоконтролер О. В. Минакова

Подпись, дата Инициалы, фамилия

Защищена Оценка

дата

Воронеж 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ФГБОУ ВО «ВГТУ»)

Факультет информационных технологий и компьютерной безопасности

Кафедра Системы автоматизированного проектирования и информационных систем

ЗАДАНИЕ

на курсовой проект

по дисциплине Основы программирования и алгоритмизации

Тема: Разработка программы для работы с файловой базой данных «Техническая поддержка интернет-провайдера».

Студент группы бИСТ-221 Мальков Александр Дмитриевич

Группа, фамилия, имя, отчество

База данных «Техническая поддержка интернет-провайдера», Признак: дата обращения, время обращения, адрес абонента, характер неисправности, состояние заявки, Вариант сортировки: Дата и время обращения

Технические условия Windows 10, Microsoft VisualStudio2019, язык программирования C

Содержание и объем проекта (графические работы, расчеты и прочее):

стр., иллюстр., табл., приложение

Сроки выполнения этапов анализ и постановка задачи (10.9-5.10.22); разработка пошаговой детализации программы (6.10 -11.11.22); реализация программы (11.11-5.12.22); тестирование программы (6.12-11.12.22); оформление пояснительной записки (11.12-14.12.22).

Срок защиты курсового проекта

Руководитель О. В. Минакова

Подпись, дата Инициалы, фамилия

Задание принял студент А. Д. Мальков

Подпись, дата Инициалы, фамилия

**Содержание**

Оглавление

[ВВЕДЕНИЕ 4](#_Toc122995159)

[1. Постановка задачи 5](#_Toc122995160)

[2. Конструирование программы 8](#_Toc122995161)

[3. Тестирование программы 16](#_Toc122995162)

[ЗАКЛЮЧЕНИЕ 22](#_Toc122995163)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc122995164)

[Приложение 24](#_Toc122995165)

# ВВЕДЕНИЕ

Целью данного курсового проекта является разработка программного продукта с заданным функциональным назначением, включающая весь процесс создания программы от описания и понимания поставленной задачи и ее анализа, многократных ее реализациях на языке программирования до появления отлаженного законченного программного продукта.

Цель работы: для заданной базы данных написать программу, которая с помощью различных пунктов меню будет реализовать доступ к следующим функциям: заполнение базы данных пользователем с клавиатуры, запись и чтение данных из файла, поиск записи по заданным пользователем значениям полей, вывод данных из базы данных на экран, сортировка данных, поиск данных, добавление новых записей в файл.

Шаги для реализации поставленной задачи:

1. Провести анализ поставленной задачи и составить алгоритм решения.

2. Реализовать возможность выбора пользователем заполнения массива структур самостоятельно.

3. Создание функции поиска записи по заданным пользователем значениям полей - «статус заявки» и «адрес абонента».

4. Создание функции упорядочивания записей по дате и времени обращения в порядке возрастания.

5. Обеспечить добавление новых записей, как в файл, так и в базу данных.

Программа разрабатывается на языке C в среде Microsoft Visual Studio 2022.

# Постановка задачи

Цель работы курсового проекта состоит в разработке файловой базы данных по теме «Техническая поддержка интернет-провайдера». Тематика базы данных подразумевает следующие поля ввода:

* *Дата обращения (число, месяц, год) – это строка, содержащая целые числа и использующая точку в качестве разделителя;*
* *Время обращения (число, время) – это строка, содержащая целые числа и двоеточие в качестве разделителя;*
* *Адрес абонента (улица, дом) – строка, содержащая целые числа и символы (к примеру \_ или - );*
* *Характер поломки – строка, содержащая буквы;*
* *Статус заявки (рассматривается, устранено, ведутся работы и т.д) – это строка, содержащая буквы.*

1. Реализовать диалог взаимодействия с пользователем: Программа должна предоставлять пользователю возможность выполнения следующих действий:

1) ввод данных по заданному пользователем количеству записей;

2) вывод результатов упорядочивания в заданный файл и/или на экран;

3) вывод диагностических сообщений в ходе проверки корректности ввода данных;

4) пользовательский выбор полей записи для упорядочивания массива структур;

5) задание пользователем значения одного из двух поисковых полей.

В задании требуется выполнить поиск состоянию заявки и адресу абонента. Будет выполняться он следующим образом: ввод данных → сопоставление с данными из файла, при нахождении соответствия производится вывод данных.

Согласно заданию необходимо реализовать сортировку массива по дате и времени обращения.

В данной программе будет задействована быстрая сортировка, как самая эффективная среди прочих сортировок.

Для работы с файлом, необходимо определить его структуру. Текстовый файл содержит множество последовательных записей следующего вида:

Дата: *10.11.2022*

Время обращения: 20:30

Адрес абонента: *Московский\_проспект\_14*

Характер поломки: *нет\_сети*

Состояние заявки: *на\_рассмотрении/не\_рассмотрена/ведутся\_работы*

В программе будут реализованы следующие функции работы с файловой БД:

1. *Ввод данных для дальнейшей записи в файл/БД.*
2. *Чтение данных из файла.*
3. *Вывод данных на экран*
4. *Поиск по адресу.*
5. *Поиск по статусу заявки.*
6. *Сортировка данных по дате. Пользователь сможет увидеть список массивов, расположенных в порядке возрастания.*
7. *Сортировка данных по времени.*
8. *Запись в файл.*
9. *Дозапись в файл/БД.*

База данных (БД) - это совокупность массивов и файлов данных, организованная по определённым правилам и отражающая состояние объектов какой-то конкретной предметной области и отношения между этими объектами.

Базы данных представляют собой некоторый список, который состоит из набора однотипных записей, расположенных друг за другом. Эти записи формируют таблицу со строками, над которыми можно выполнять определенные действия: добавление, удаление и изменение.

Задачей курсового проекта является создание базы данных, выбор действия над которой будет осуществляться с помощью различных пунктов меню.

Выбор пунктов меню реализуется с помощью оператора многоальтернативного выбора switch(). При выборе определенного case происходит вызов соответствующего названия пункта меню, номер которого совпадает с номером определенного case.

Для управления этими функциями в программе следует реализовать меню из следующих пунктов:

1. *выход из программы;*
2. *ввод данных;*
3. *вывод данных;*
4. *запись данных;*
5. *чтение данных;*
6. *поиск по состоянию заявки;*
7. *поиск по адресу абонента;*
8. *сортировка по дате;*
9. *времени обращения;*
10. *дозапись данных.*

Пользователь выбирает опцию, после чего программа выводит сообщение о необходимом для выполнения данной функции действии, например о необходимости ввода определенных данных.

В ходе выполнения функций на экран выводятся диагностические сообщения, информирующие о том, корректно ли выполнены действия.

# Конструирование программы

Функция – это именованная часть программы, которая может быть многократно вызвана из другого участка программы (в области видимости). Функция может принимать фиксированное или переменное число аргументов, но может и не иметь аргументов. Также функция может возвращать значение, но еще может быть пустой и ничего не возвращать.

В языке С можно использовать как стандартные функция, которые уже прописаны в библиотеках, так и собственные функции.

Сoбственные функции требуют объявления прототипа. Он дает основную информацию о структуре функции: он сообщает компилятору, какое значение функция возвращает, как функция будет вызываться, а также какие аргументы функции могут быть переданы.

Разбиение программы на функции дает следующие преимущества:

* функцию можно вызвать из различных мест программы, что позволяет избежать повторения программного кода;
* одну и ту же функцию можно использовать в разных программах;
* функции повышают уровень модульности программы и облегчают ее проектирование;
* использование функций облегчает чтение и понимание программы и ускоряет поиск и исправление ошибок.

По окончании выполнения вызываемой функции осуществляется возврат значения в точку ее вызова. Это значение присваивается переменной, тип которой должен соответствовать типу возвращаемого значения функции.

Пользовательские функции, содержащиеся в программе:

1. int main()- главная функция.

Вначале функция вызывает функцию system("chcp 1251") для того, чтобы поменять кодовую страницу в консоли на кириллицу. Далее программа запрашивает у пользователя количество записей в базе данных, после чего выделяет под нее память и вызывает меню. Далее пользователь вводит желаемое действие. С помощью оператора многоальтернативного выбора switch() вызываются функции, отвечающие за выбранное пользователем действие. На рисунке 2 представлена блок-схема функции.

|  |
| --- |
|  |
| Рисунок 1 – блок-схема функции main()   |  |  |  | | --- | --- | --- | | Описание собственных функций | | | | наименование | назначение | примечание | | init | Заполнение массива данных вводимыми пользователем значениями | Принимает массив структур и количество требуемых записей | | read\_file | Открытие файла | Принимает массив символов, содержащих название файла | | put | Вывод данных на экран | Принимает массив структур и количество введенных записей | | write\_file | Сохраняет введенные данные в файл | Принимает массив структур, имя файла для записи и количество записей | | add | Добавляет новые данные в файл | Принимает массив структур, имя файла для записи и количество записей | | searchByAdress | Организует поиск по адресу | Принимает массив данных, количество записей и параметр поиска | | searchByStatus | Организует поиск по статусу | Принимает массив данных, количество записей и параметр поиска | | sortByDate | Организует сортировку по дате | Принимает массив данных и количество записей | | sortByTime | Организует сортировку по времени | Принимает массив и количество записей | |

1. Функция int init(user\_data\* point, int size, int n) заполняет массив, на который указывает параметр user\_data\* данных вводимыми пользователем значениями. Size отвечает за количество записей. С помощью цикла с фиксированным числом повторений for(), функции вывода printf() и ввода scanf() происходит заполнения массива данных. Блок-схема функции представлена на рисунке 2.

|  |
| --- |
|  |
| Рисунок 2 – блок-схема функции init() |

1. Функция user\_data\* read\_file(char\* filename, int\* size) при успешном открытии файла считывает каждую строку и выводит результат на экран, в случае, если открыть файл не удалось, выводится сообщение об ошибке. За передачу названия файла в функцию отвечает параметр filename, параметр size указывает на количество записей. В конце работы функция с помощью fclose() закрывает файл. Блок-схема функции представлена на рисунке 3.

|  |
| --- |
|  |
| Рисунок 3 – блок-схема функции readfile() |

1. Функция void put(user\_data\* point, int size) выводит массив данных, на который указывает параметр user\_data\*. Параметр size указывает на количество записей. С помощью цикла с фиксированным числом повторений for() функции вывода printf() происходит вывод массива данных. Блок-схема функции представлена на рисунке 4.

|  |
| --- |
|  |
| Рисунок 4 – блок-схема функции put() |

1. Функция int write\_file(user\_data\* point, char\* filename, int size) открывает файл для записи, в случае успешного открытия записывает данные из массива структур в файл. В конце работы выводит информацию об успешной записи и закрывает файл. Принимает массив структур, за который отвечает параметр user\_data. Параметр filename отвечает за имя файла для записи, size за количество записей. Блок-схема функции представлена на рисунке 5.

|  |
| --- |
|  |
| Рисунок 5 – блок-схема функции writefile() |

1. Функция int add(user\_data\* point, char\* filename, int size, int n) открывает файл для добавления новой записи. Принимает массив структур, за который отвечает параметр user\_data. Параметр filename отвечает за имя файла для записи, size за количество записей. Блок-схема функции представлена на рисунке 6.

|  |
| --- |
|  |
| Рисунок 6 – блок-схема функции add() |

1. Функция int searchByAdress(user\_data\* point, char\* adress, int size) находит пользователя по адресу и выводит данные в главную функцию. За массив структур отвечает параметр user\_data, параметр adress отвечает за адрес абонента, size за количество записей. С помощью цикла с фиксированным числом повторений for(), оператора условного перехода if() и функции сравнения строк strcmp() находим адрес пользователя. Блок-схема функции представлена на рисунке 7.

|  |
| --- |
|  |
| Рисунок 7 – блок-схема функции searchByAdress() |

1. Функция int searchByStatus(user\_data\* point, char\* status, int size) находит пользователя по статусу заявки и выводит данные в главную функцию. За массив структур отвечает параметр user\_data, параметр status отвечает за адрес абонента, size за количество записей. С помощью цикла с фиксированным числом повторений for(), оператора условного перехода if() и функции сравнения строк strcmp() находим адрес пользователя. Блок-схема функции представлена на рисунке 8.

|  |
| --- |
|  |
| Рисунок 8 – блок-схема функции searchByStatus() |

1. Функция void sortByDate(user\_data\* point, int size) сортирует массив пользователей point размером size в порядке возрастания и возвращает в главную функцию main() массив с отсортированными записями по дате. Блок-схема функций представлена на рисунках 9, 10 и 11.

|  |
| --- |
|  |
| Рисунок 9 – блок-схема вспомогательной функции dateComparator() |

1. Функция sortByTime(user\_data\* point, int size) сортирует массив пользователей point размером size в порядке возрастания и возвращает в главную функцию main() массив с отсортированными записями по времени. Блок-схема функций представлена на рисунках 12, 13 и 14.

|  |
| --- |
|  |
| Рисунок 12 – блок-схема функции timeComparator() |

# Тестирование программы

При запуске программы на экран выводится меню с возможностью ввода желаемого действия (см. Рисунок 15).

|  |
| --- |
|  |
| Рисунок 15 – интерфейс программы |

При нажатии на клавишу «1» появляется возможность вводить данные (см. Рисунок 16).

|  |
| --- |
|  |
| Рисунок 16 – результат выполнения команды «Ввод данных» |

При нажатии на клавишу «2» появляется возможность считать данные из файла (см. Рисунок 17).

|  |
| --- |
|  |
| Рисунок 17 – результат выполнения команды «Чтение данных из файла» |

При нажатии на клавишу «3» можно вывести введенные ранее с помощью клавиши «1» данные (см. Рисунок 18).

|  |
| --- |
|  |
| Рисунок 18 – результат выполнения команды «Вывод данных на экран» |

Для того, чтобы осуществить поиск по адресу, необходимо прочитать файл, нажав клавишу «2». После чего нажать клавишу «4» для выполнения поиска по адресу (см. Рисунок 19).

|  |
| --- |
|  |
| Рисунок 19 – результат выполнения команды «Поиск по адресу» |

«Поиск по статусу заявки» осуществляется тем же методом, что и «поиск по адресу», сначала читаем файл, после нажимаем клавишу «5» и получаем результат (см. Рисунок 20).

|  |
| --- |
|  |
| Рисунок 20 – результат выполнения команды «Поиск по статусу заявки» |

Для того, чтобы сортировать данные по дате, необходимо прочитать файл, после чего нажать клавишу «6» и после нажать клавишу «3», чтобы вывести результат на экран (см. Рисунки 21 и 22).

|  |
| --- |
|  |
| Рисунок 21 – сначала читаем файл, после чего нажимаем клавишу «6» |

|  |
| --- |
|  |
| Рисунок 22 – результат сортировки по дате |

«Сортировка по времени» выполняется точно также, как и «сортировка по дате». Сначала читаем файл, нажимаем клавишу «7» для того, чтобы программа отсортировала данные по времени, и выводим результат на экран клавишей «3» (см. Рисунки 23 и 24).

|  |
| --- |
|  |
| Рисунок 23 – читаем файл, после чего нажимаем клавишу «7» |

|  |
| --- |
|  |
| Рисунок 24 – результат сортировки по времени |

Для осуществления записи каких-либо данных в файл их надо сначала ввести, нажав клавишу «1». После чего, нажав на клавишу «8», программа запишет введенные данные в файл (см. Рисунок 25).

|  |
| --- |
|  |
|  |
| Рисунок 25 – результат выполнения команды «запись в файл» |

При нажатии клавиши «9» программа предложит дозаписать необходимое количество пользователей либо в базу данных, либо в файл (см. Рисунок 26).

|  |
| --- |
|  |
| Рисунок 26 – процесс выполнения команды «Дозапись в файл» |

Нажав клавишу «0» можно остановить работу программы.

# ЗАКЛЮЧЕНИЕ

Проблема структуризации и сбора информации в базах будет постепенно увеличиваться из-за постоянно растущего количества информации в постиндустриальном обществе, а значит будет необходимо формировать все больше баз данных, которые будут хранить данные по какому-либо направлению. Ее решению будет содействовать развитие информационных технологий, которое заключается как в совершенствовании методов организации информационных процессов, так и их реализации с помощью конкретных инструментов - сред и языков программирования.

Правильно оформленная база данных в разы облегчит поиск информации для людей разных профессий, и сфер занятости, позволит проще вести бизнес и заниматься самообразованием.

Итогом работы можно считать созданную файловую базу данных «Техническая поддержка интернет-провайдера».

При решении поставленной задачи были реализованы следующие функции:

1. Ввод данных для дальнейшей записи в файл/БД.
2. Чтение данных из файла.
3. Вывод данных на экран
4. Поиск по адресу.
5. Поиск по статусу заявки.
6. Сортировка данных по дате.
7. Сортировка данных по времени.
8. Запись в файл.
9. Дозапись в файл/БД.

Все требования к программе выполнены успешно. Реализован интерфейс программы, позволяющий пользователю выбирать необходимые для его задач действия.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шишкин А.Д. Программирование на языке Си [Электронный ресурс]: учебное пособие/ Шишкин А.Д. – Электрон. текстовые данные. – СПб.: Российский государственный гидрометеорологический университет, 2003 – 104 c.
2. Д. Кольцов: Си на примерах. Практика, практика и только практика [Электронный ресурс]
3. Курипта О.В. Основы программирования и алгоритмизации: практикум / О.В. Курипта, О.В. Минакова, Д.К. Проскурин; Воронежский ГАСУ. – Воронеж, 2015 – 132 с.
4. Подбельский В.В., Фомин С.С. Программирование на языке Си. М.: ФиС, 1999 – 600 с.
5. Всероссийский киберфорум <http://www.cyberforum.ru/>.
6. Хирьянов Тимофей YouTube-канал о программировании, МФТИ <https://www.youtube.com/user/tkhirianov/>.
7. YouTube-канал Begin End о программировании <https://www.youtube.com/@beginend95>
8. YouTube-канал о программировании

<https://www.youtube.com/@gosha_dudar/featured>

1. YouTube-канал о программировании <https://www.youtube.com/@AlekOS>
2. YouTube-канал о программировании, изучение языка Си для начинающих <https://www.youtube.com/watch?v=9IBXUWYqcoM>

# Приложение

Листинг программы:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h> // для функции ввода/вывода

#include <malloc.h> // библиотека для использования функции выделения памяти

#include <windows.h> // для задания языка консоли

#include <string.h> // для работы со строками

#include <stdlib.h> // для работы со стандартной библиотекой

typedef struct {

int data[3]; // день, месяц, год

int timeH; // часы

int timeMin; // минуты

char adress[50]; // адрес абонента

char failures[150]; // характер поломки

char status[50]; // состояние заявки

}user\_data;

int init(user\_data\* point, int size, int n);

user\_data\* read\_file(char\* filename, int\* size);

void put(user\_data\* point, int size);

int write\_file(user\_data\* point, char\* filename, int size);

int add(user\_data\* point, char\* filename, int size, int n);

int searchByAdress(user\_data\* point, char\* adress, int size);

int searchByStatus(user\_data\* point, char\* status, int size);

int dateComparator(const user\_data\* userA, const user\_data\* userB);

int timeComparator(const user\_data\* userA, const user\_data\* userB);

void sort\_(user\_data\* point, int size, int(\*comparator)(const void\*, const void\*));

void sortByDate(user\_data\* point, int size);

void sortByTime(user\_data\* point, int size);

int main() {

system("chcp 1251");

user\_data\* point = NULL; // объявление БД

int size, number, n, action;

char filename[120], adress[50], status[50]; // массивы для хранения названия файла, адреса абонента, состояния заявки

printf("База данных - Техническая поддержка интернет-провайдера\n");

while (1) {

printf("\tМеню\n");

printf("\nВыберите действие:\n");

printf("1) Ввод данных.....\n");

printf("2) Чтение данных из файла.....\n");

printf("3) Вывод данных на экран.....\n");

printf("4) Поиск по адресу.....\n");

printf("5) Поиск по статусу заявки.....\n");

printf("6) Сортировать по дате.....\n");

printf("7) Сортировать по времени.....\n");

printf("8) Запись в файл.....\n");

printf("9) Дозапись в файл.....\n");

printf("0) Выход из программы.....\n");

scanf("%d", &action);

switch (action) {

case 0:

return -1; // выход из программы

case 1:

system("cls"); // очистка консоли

printf("Введите кол-во записей: ");

scanf("%d", &size);

point = (user\_data\*)malloc(size \* sizeof(user\_data)); // выделение памяти

n = 0;

init(point, size, n);

break;

case 2:

printf("Введите имя файла, который необходимо прочитать: ");

scanf("%s", &filename);

point = read\_file(filename, &size);

printf("Считано %d строк\n", size);

put(point, size);

break;

case 3:

if (point == NULL) { // проверяется наличие БД

printf("База данных не найдена.\n");

break;

}

put(point, size);

break;

case 4:

system("cls");

if (point == NULL) { // проверяется наличие БД

printf("База данных не найдена.\n");

break;

}

printf("Укажите адрес абонента: ");

scanf("%s", &adress);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

int a = searchByAdress(point, adress, size);

if (a >= 0) {

printf("Дата обращения: %d.%d.%d\n", point[a].data[0], point[a].data[1], point[a].data[2]);

printf("Время обращения: %d:%d\n", point[a].timeH, point[a].timeMin);

printf("Адрес абонента: %s\n", &point[a].adress);

printf("Характер поломки: %s\n", &point[a].failures);

printf("Статус заявки: %s\n", &point[a].status);

}

else {

printf("Абонент не найден.\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

break;

case 5:

system("cls");

if (point == NULL) { // проверяется наличие БД

printf("База данных не найдена.\n");

break;

}

printf("Укажите состояние заявки: ");

scanf("%s", &status);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

int b = searchByStatus(point, status, size);

if (b >= 0) {

printf("Дата обращения: %d.%d.%d\n", point[b].data[0], point[b].data[1], point[b].data[2]);

printf("Время обращения: %d:%d\n", point[b].timeH, point[b].timeMin);

printf("Адрес абонента: %s\n", &point[b].adress);

printf("Характер поломки: %s\n", &point[b].failures);

printf("Статус заявки: %s\n", &point[b].status);

}

else {

printf("Абонент не найден.\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

break;

case 6:

if (point == NULL) {

printf("База данных не найдена.\n");

break;

}

sortByDate(point, size);

break;

case 7:

if (point == NULL) {

printf("База данных не найдена.\n");

break;

}

sortByTime(point, size);

break;

case 8:

system("cls");

if (point == NULL) {

printf("База данных не найдена\n");

break;

}

printf("Введите имя файла для записи: ");

scanf("%s", &filename);

write\_file(point, filename, size);

break;

case 9:

if (point == NULL) {

printf("База данных не найдена\n");

break;

}

printf("Введите кол-во пользователей, которых необходимо добавить: ");

scanf("%d", &number);

n = size;

size += number;

point = (user\_data\*)realloc(point, size \* sizeof(user\_data)); //перераспределение памяти

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Добавить пользователя\n");

printf("...в файл\n");

printf("...в БД\n");

scanf("%d", &action);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

if (action == 1) { // добавление в файл

printf("Введите имя файла для добавления абонента:");

scanf("%s", &filename);

add(point, filename, size, n);

}

else if (action == 2) { //добавление в базу данных

init(point, size, n);

}

break;

default:

printf("Выполнить невозможно.\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

return 0;

}

//функция для заполнения БД

int init(user\_data\* point, int size, int n) {

for (int i = n; i < size; i++) {

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Введите дату обращения: ");

scanf("%d.%d.%d", &point[i].data[0], &point[i].data[1], &point[i].data[2]);

printf("Введите время обращения: ");

scanf("%d:%d", &point[i].timeH, &point[i].timeMin);

printf("Введите адрес абонента: ");

scanf("%s", &point[i].adress);

printf("Введите характер поломки: ");

scanf("%s", &point[i].failures);

printf("Введите состояние заявки: ");

scanf("%s", &point[i].status);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

return point;

}

//функция чтения из файла

user\_data\* read\_file(char\* filename, int\* size) {

FILE\* f; // объявляем переменную

int number\_field = 0; // номер поля за записи

int ind = 0; // номер строки

user\_data\* list = (user\_data\*)malloc(1000 \* sizeof(user\_data\*));

char buffer[200]; // массив для хранения строки из файла

f = fopen(filename, "r");

if (f != NULL) {

printf("\nФайл открыт.\n");

while (!feof(f)) {

fgets(buffer, 200, f);

if (buffer[0] == '\n')

continue;

sscanf(buffer, "%d.%d.%d %d:%d %s %s %s",

&list[ind].data[0],

&list[ind].data[1],

&list[ind].data[2],

&list[ind].timeH,

&list[ind].timeMin,

list[ind].adress,

list[ind].failures,

list[ind].status);

ind++;

}

\*size = ind; // возвращаем число записей

return list;

}

else {

printf("\nОткрыть файл не удалось.\n");

return NULL;

}

fclose(f); // закрытие файла

return NULL;

}

// функция вывода на экран

void put(user\_data\* point, int size) {

printf("\tДата обращения\t\tВремя обращения\t\tАдрес абонента\t\t\tХарактер поломки\t\tСтатус заявки");

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

printf("\n\n%d. ", i + 1);

printf("\n\t%d.%d.%d\t", point[i].data[0], point[i].data[1], point[i].data[2]); //дата обращения

printf("\t%d:%d\t", point[i].timeH, point[i].timeMin); //время обращения

printf("\t\t%s\t", point[i].adress); //адрес абонента

printf("\t\t%s\t", point[i].failures); //характер поломки

printf("\t%s\t", point[i].status); //статус заявки

printf("\n");

}

printf("\t");

}

// функция записи в файл

int write\_file(user\_data\* point, char\* filename, int size) {

FILE\* f;

if ((f = fopen(filename, "w")) == NULL) { //проверяем возможность открыть файл

printf("Невозможно открыть файл для записи. \n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

system("pause");

}

else {

printf("Файл успешно открыт.\n");

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

fprintf(f, "%d.%d.%d ", point[i].data[0], point[i].data[1], point[i].data[2]);

fprintf(f, "%d:%d ", point[i].timeH, point[i].timeMin);

fprintf(f, "%s ", point[i].adress);

fprintf(f, "%s ", point[i].failures);

fprintf(f, "%s", point[i].status);

system("pause");

}

fclose(f); // закрытие файла

printf("\n Данные записаны в файл.\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

system("pause");

}

return 0;

}

// функция дозаписи

int add(user\_data\* point, char\* filename, int size, int n) {

FILE\* f;

if ((f = fopen(filename, "a")) == NULL) {

fprintf(stderr, "Невозможно открыть файл для записи. \n");

return -1; //выход, если ошибка

}

else {

init(point, size, n);

for (int i = n; i < size; i++) {

fprintf(f, "\n");

fprintf(f, "%d.%d.%d ", point[i].data[0], point[i].data[1], point[i].data[2]); //дата обращения

fprintf(f, "%d:%d ", point[i].timeH, point[i].timeMin);

fprintf(f, "%s ", point[i].adress);

fprintf(f, "%s ", point[i].failures);

fprintf(f, "%s ", point[i].status);

}

}

fclose(f); // закрытие файла

return 1;

}

int searchByAdress(user\_data\* point, char\* adress, int size) {

int number = -1;

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

if (strcmp(adress, point[i].adress) == 0) { // проверяем строки на соответствие введенным данным

return i; //возвращаем номер

}

}

return number;

}

int searchByStatus(user\_data\* point, char\* status, int size) {

int number = -1;

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

if (strcmp(status, point[i].status) == 0) { // проверяем строки на соответствие введенным данным

return i; //возвращаем номер

}

}

return number;

}

int dateComparator(const user\_data\* userA, const user\_data\* userB) {

if (userA->data[2] != userB->data[2])

return userA->data[2] - userB->data[2];

if (userA->data[1] != userB->data[1])

return userA->data[1] - userB->data[1];

return userA->data[0] - userB->data[0];

}

int timeComparator(const user\_data\* userA, const user\_data\* userB) {

if (userA->timeH != userB->timeH)

return userA->timeH - userB->timeH;

return userA->timeMin - userB->timeMin;

}

void sort\_(user\_data\* point, int size, int (\*comparator)(const void\*, const void\*)) {

qsort(point, size, sizeof(point[0]), comparator);

}

// сортирует массив пользователей point размером size по дате регистрации

void sortByDate(user\_data\* point, int size) {

sort\_(point, size, dateComparator);

}

// сортирует массив пользователей point размером size по времени регистрации

void sortByTime(user\_data\* point, int size) {

sort\_(point, size, timeComparator);

}