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 Систем управления и информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине Основы программирования и алгоритмизации

Тема: Разработка программы для работы с файловой базой данных «Усилители беспроводного сигнала».

**Расчетно-пояснительная записка**

Разработал студент . А.А. Харебин.

Подпись, дата Инициалы, фамилия

Руководитель . Н. В. Акамсина.

Подпись, дата Инициалы, фамилия

Нормоконтролер . Н. В. Акамсина.

Подпись, дата Инициалы, фамилия

Защищена . . Оценка . .

Дата

Воронеж

2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ»)

Кафедра Систем управления и информационных технологий в строительстве

ЗАДАНИЕ

на курсовой проект

по дисциплине Основы программирования и алгоритмизации

Тема: Разработка программы для работы с файловой базой данных «Усилители беспроводного сигнала».

Студент группы бТИИ-241 Харебин Антон Александрович.

Фамилий, имя, отчество

База данных «Усилители беспроводного сигнала», Признак: Производитель, тип антенны, Вариант сортировки: Стандарт Wi-Fi, выходная мощность передатчика (dBm).

Технические условия Windows 10, Microsoft Visual Studio, язык программирования С

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

стр., рис., табл., приложение

Сроки выполнения этапов: анализ и постановка задачи (20.9.24-6.10.24); разработка пошаговой детализации программы (21.10.24-20.11.24); реализация программы (18.11.24-23.12.24); тестирование программы (3.12.24-17.12.24); оформление пояснительной записки (3.12.24-24.12.24).

Срок защиты курсового проекта . .

Руководитель . Н. В. Акамсина.

Подпись, дата Инициалы, фамилия

Задание принял студент . А.А. Харебин.

Подпись, дата Инициалы, фамилия

Замечания руководителя

**Постановка задачи**

Задачей курсового проекта является создание файловой базы данных на тему «Усилители беспроводного сигнала». Программа должна обеспечивать функциональные возможности для добавления, поиска, изменения, сортировки и вывода данных, их сохранения в файл и чтение из файла, удаления данных из файла, а также очистка загруженных данных. В базе данных присутствуют следующие поля для ввода:

• Производитель — строка из 50 символов, предназначенная для хранения названия производителя.

• Тип антенны — строка из 50 символов, предназначенная для хранения наименования типа антенны.

• Режимы работы — строка из 50 символов, предназначенная для хранения режимов работы усилителя.

• Частота работы передатчика (ГГц) — вещественное число, обозначающее частоту работы передатчика.

• Одновременная работа в двух диапазонах — логическое значение, указывающее на возможность работы в двух диапазонах.

• Количество портов Ethernet — целое число, обозначающее число портов Ethernet на устройстве.

• Поддержка MESH — логическое значение, указывающее на поддержку технологии MESH.

• Стандарт Wi-Fi — строка из 50 символов, предназначенная для хранения стандарта Wi-Fi.

• Скорость передачи данных Ethernet (Мбит/с) — целое число, показывающее скорость передачи данных по Ethernet.

• Класс Wi-Fi — строка из 50 символов, предназначенная для хранения класса Wi-Fi.

• Выходная мощность передатчика (dBm) — вещественное число, показывающее выходную мощность передатчика.

• Максимальная скорость беспроводного соединения (Мбит/с) — целое число, показывающее максимальную скорость беспроводного соединения.

Возможности программы: \uf0be Добавление одной или нескольких записей

1) Поиск записей по производителю, типу антенны, режимам работы, частоте работы передатчика (ГГц), одновременной работе в двух диапазонах, количеству портов Ethernet, поддержке MESH

2) Сохранение новых записей в файл

3) Чтение данных из файла

4) Печать всех данных с сортировкой по производителю, стандарту Wi-Fi или обоим параметрам

5) Изменение записи

6) Удаление всех данных из файла

7) Очистка загруженных из файла данных (очистка занимаемой ими памяти)

В соответствии с заданием необходимо реализовать поиск записи по производителю, типу антенны, режимам работы, частоте работы передатчика (ГГц) и другим параметрам. Поиск будет происходить следующим образом: во время прохода по массиву в цикле введенное пользователем значение будет сравниваться с необходимым полем структуры данных, и в случае, если данные совпадают, то будет выведена соответствующая запись. Также должна быть реализована сортировка записей. Используемый тип сортировки — пузырьковый. Так как признаками для сортировки являются строки, то для их сравнения будет использоваться функция strcmp, которая берет отношение одной строки к другой и в случае, если отношение больше 0, то меняет их местами. Программа предназначена для управления базой данных маршрутизаторов (routers). Она позволяет добавлять, редактировать, удалять и искать записи, а также сохранять и загружать данные из текстового файла. Программа предоставляет пользователю интерфейс меню для выполнения различных операций.

**Таблица 1 — Используемые в программе функции**

| **Функция** | **Описание функции** |
| --- | --- |
| **int addRouter(router\_t routers[ ], int\* count)** | Последовательно считывает вводимые пользователем данные и записывает их в поля структуры данных. |
| **int searchRouter(const router\_t routers[], int count, router\_t results[ ], char\* wifiClass, char\* wifiStandard, int searchOption)** | Запускает цикл прохода по массиву структур и сравнения введенных пользователем значений (класс Wi-Fi и стандарт Wi-Fi). В случае совпадения параметров, выводит найденные записи, а иначе выводит сообщение о том, что записи не найдены. |
| **int saveToTextFile(router\_t routers[ ], int count, const char\* filename)** | Открывает для записи файл (или создает его, если таковой отсутствует). Затем запускает цикл прохода по массиву структур данных и записывает поля структуры в файл, выделяя под запись каждой структуры одну строку в файле, а затем закрывает файл. |
| **int loadFromTextFile(router\_t routers[], int\* count, const char\* filename)** | Открывает для чтения файл. Затем запускает цикл записи данных из файла в поля массива структур данных, после чего закрывает файл. |
| **int bubbleSort(router\_t\* routers, int count, int criterion)** | Сортирует записи по одному из выбранных пользователем признаков или по обоим сразу. |
| **int editRouter(router\_t routers[], int count, const char\* manufacturer)** | Последовательно считывает вводимые пользователем значения и записывает их в поля структуры, производителя которой указал пользователь. |
| **int deleteRouter(router\_t routers[], int\* count, const char\* manufacturer)** | Удаляет запись из массива структур данных по указанному производителю. |

**Блок схемы.**

Для создания блок схем используем онлайн редактор.

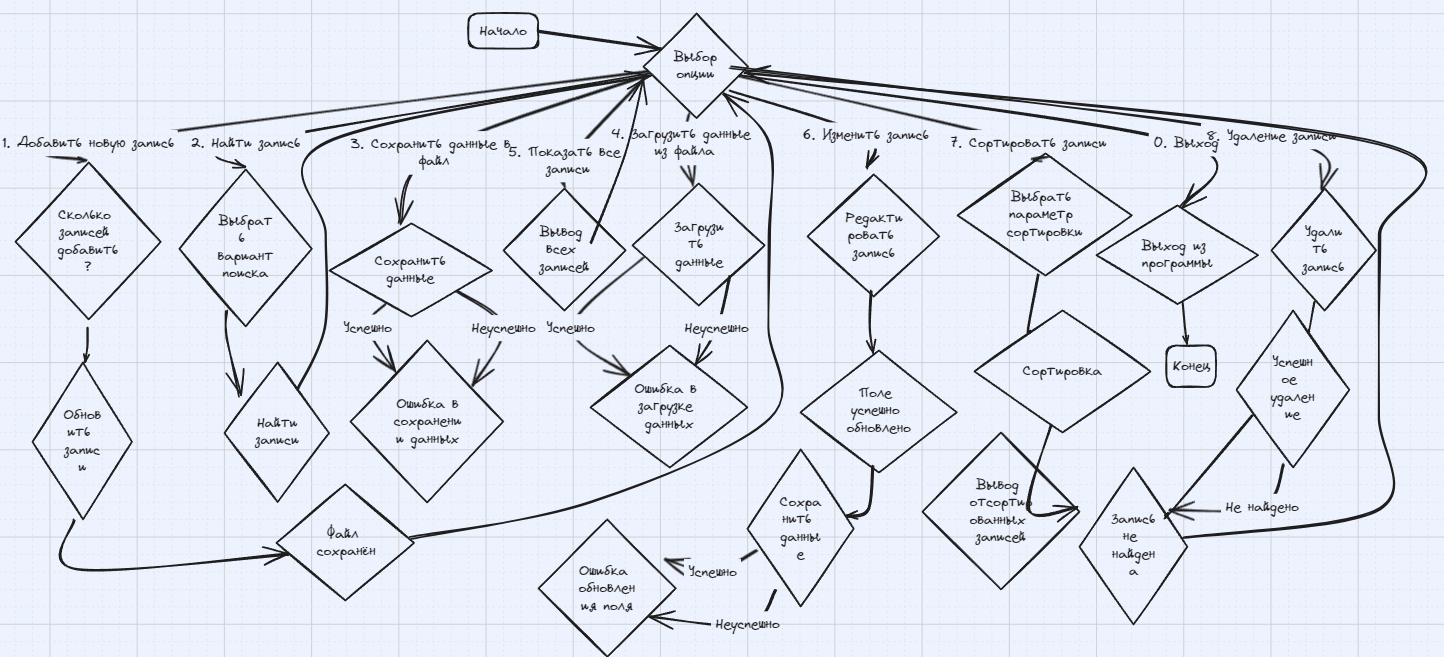
****

Рисунок 1 – main

На представленной схеме изображен алгоритм работы с данными о маршрутизаторах, включающий добавление, поиск, сортировку, редактирование и удаление записей.

**Основные блоки и их описание:**

1. Начало:
   * Это начальный блок, с которого начинается выполнение алгоритма.
2. Выбор опции:
   * Пользователь выбирает одну из следующих опций:
     1. Добавить новую запись
     2. Найти запись
     3. Сохранить данные в файл
     4. Загрузить данные из файла
     5. Показать все записи
     6. Изменить запись
     7. Сортировать записи
     8. Выход (завершение работы)

**Опция 1: Добавить новую запись**

1. Очистка данных, ввод полей:
   * Система очищает данные для новой записи и запрашивает у пользователя ввод данных для всех полей.
2. Обновление данных:
   * Введенные данные обновляются в массиве записей.
3. Возврат к выбору опции:
   * Пользователь возвращается к выбору опции.

**Опция 2: Найти запись**

1. Выбор параметров поиска:
   * Пользователь выбирает параметры для поиска записи (например, класс Wi-Fi, стандарт Wi-Fi и т.д.).
2. Поиск записи:
   * Система ищет записи, соответствующие введенным параметрам.
3. Ошибка или успех:
   * Если записи найдены, они отображаются пользователю.
   * Если записи не найдены, система выводит сообщение об ошибке.
4. Возврат к выбору опции:
   * Пользователь возвращается к выбору опции.

**Опция 3: Сохранить данные в файл**

1. Сохранение данных в файл:
   * Система сохраняет все данные из массива записей в файл.
2. Возврат к выбору опции:
   * Пользователь возвращается к выбору опции.

**Опция 4: Загрузить данные из файла**

1. Загрузка данных из файла:
   * Система загружает данные из файла в массив записей.
2. Возврат к выбору опции:
   * Пользователь возвращается к выбору опции.

**Опция 5: Показать все записи**

1. Показ всех записей:
   * Система отображает все записи из массива.
2. Возврат к выбору опции:
   * Пользователь возвращается к выбору опции.

**Опция 6: Изменить запись**

1. Выбор параметра, по которому будет производиться поиск:
   * Пользователь выбирает параметр для поиска записи, которую нужно изменить (например, производитель).
2. Поиск записи:
   * Система ищет запись по выбранному параметру.
3. Ошибка или успех:
   * Если запись найдена, пользователь может внести изменения.
   * Если запись не найдена, система выводит сообщение об ошибке.
4. Ввод новых данных:
   * Пользователь вводит новые данные для изменения.
5. Сохранение изменений:
   * Система сохраняет изменения в массиве записей.
6. Возврат к выбору опции:
   * Пользователь возвращается к выбору опции.

**Опция 7: Сортировать записи**

1. Выбор параметра сортировки:
   * Пользователь выбирает параметр, по которому будет производиться сортировка (например, одновременная работа в двух диапазонах, стандарт Wi-Fi и т.д.).
2. Сортировка записей:
   * Система сортирует записи в массиве по выбранному параметру.
3. Возврат к выбору опции:
   * Пользователь возвращается к выбору опции.

**Опция 0: Выход (завершение работы)**

1. Завершение работы:
   * Программа завершает выполнение.

**Пошаговое объяснение алгоритма:**

1. Начало:
   * Алгоритм начинает выполнение.
2. Выбор опции:
   * Пользователь выбирает одну из предложенных опций.
3. Выполнение выбранной опции:
   * В зависимости от выбранной опции выполняется соответствующий набор действий (добавление, поиск, сохранение, загрузка, отображение, редактирование, сортировка или завершение работы).
4. Возврат к выбору опции:
   * После выполнения выбранной опции пользователь возвращается к выбору следующей опции или завершает работу.

Этот алгоритм позволяет пользователю управлять базой данных маршрутизаторов, выполняя различные операции с данными, такие как добавление, поиск, сортировка, редактирование и удаление записей, а также сохранение и загрузка данных из файла.

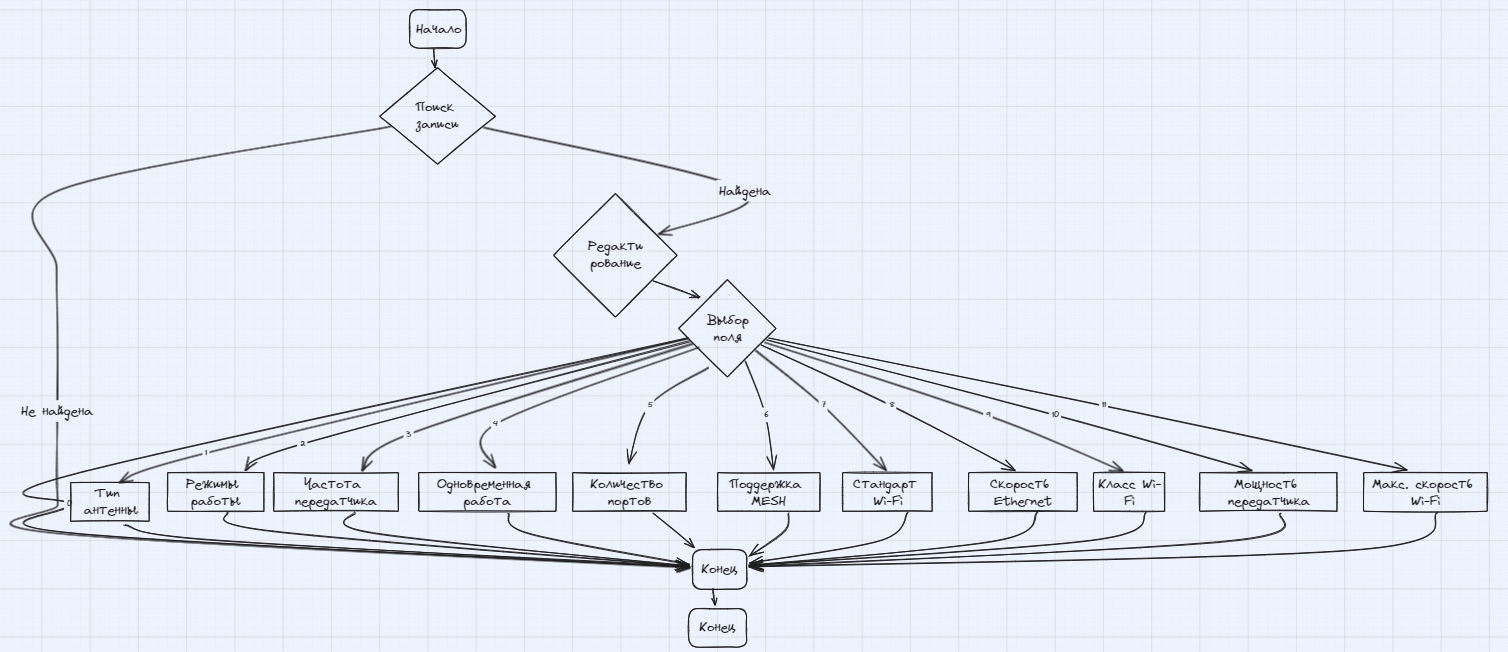


Рисунок 2 – editRouter

Структура программы для управления базой данных маршрутизаторов.

1. Начало:
   * Программа начинает свою работу с этого блока.
2. Ввод данных:
   * Пользователь вводит данные, которые будут использоваться для создания новой записи маршрутизатора.
3. Редактировать записи:
   * Пользователь может редактировать существующие записи в базе данных.
4. Выбор меню:
   * Пользователь выбирает одну из опций меню для выполнения различных операций с базой данных маршрутизаторов.
5. Опции меню:
   * Тип антенны: Пользователь может ввести или изменить тип антенны маршрутизатора.
   * Производитель роутера: Пользователь может ввести или изменить производителя маршрутизатора.
   * Частота передатчика: Пользователь может ввести или изменить частоту передатчика маршрутизатора.
   * Одновременная работа: Пользователь может ввести или изменить информацию о поддержке одновременной работы в двух диапазонах.
   * Количество портов: Пользователь может ввести или изменить количество портов Ethernet.
   * Поддержка MESH: Пользователь может ввести или изменить информацию о поддержке технологии MESH.
   * Стандарт Wi-Fi: Пользователь может ввести или изменить стандарт Wi-Fi.
   * Скорость Ethernet: Пользователь может ввести или изменить скорость передачи данных Ethernet.
   * Класс Wi-Fi: Пользователь может ввести или изменить класс Wi-Fi.
   * Мощность передатчика: Пользователь может ввести или изменить выходную мощность передатчика.
   * Макс. скорость Wi-Fi: Пользователь может ввести или изменить максимальную скорость беспроводного соединения.
6. Конец:
   * Программа завершает свою работу.

**Объяснение процесса**

1. Начало:
   * Программа запускается, и пользователь видит главное меню.
2. Ввод данных:
   * Пользователь выбирает опцию ввода данных для создания новой записи маршрутизатора.
   * Пользователь вводит все необходимые данные (тип антенны, производитель, частота передатчика и т.д.).
3. Редактировать записи:
   * Пользователь выбирает опцию редактирования записей.
   * Пользователь выбирает конкретную запись для редактирования и вносит необходимые изменения.
4. Выбор меню:
   * Пользователь выбирает одну из опций меню для выполнения различных операций:
     + Ввод данных для создания новой записи.
     + Редактирование существующих записей.
     + Выбор конкретного параметра для ввода или изменения (тип антенны, производитель, частота передатчика и т.д.).
5. Опции меню:
   * Пользователь может выбрать любой из параметров для ввода или изменения данных.
   * После ввода или изменения данных программа возвращается к главному меню.
6. Конец:
   * Пользователь выбирает опцию завершения работы программы.
   * Программа завершает свою работу.

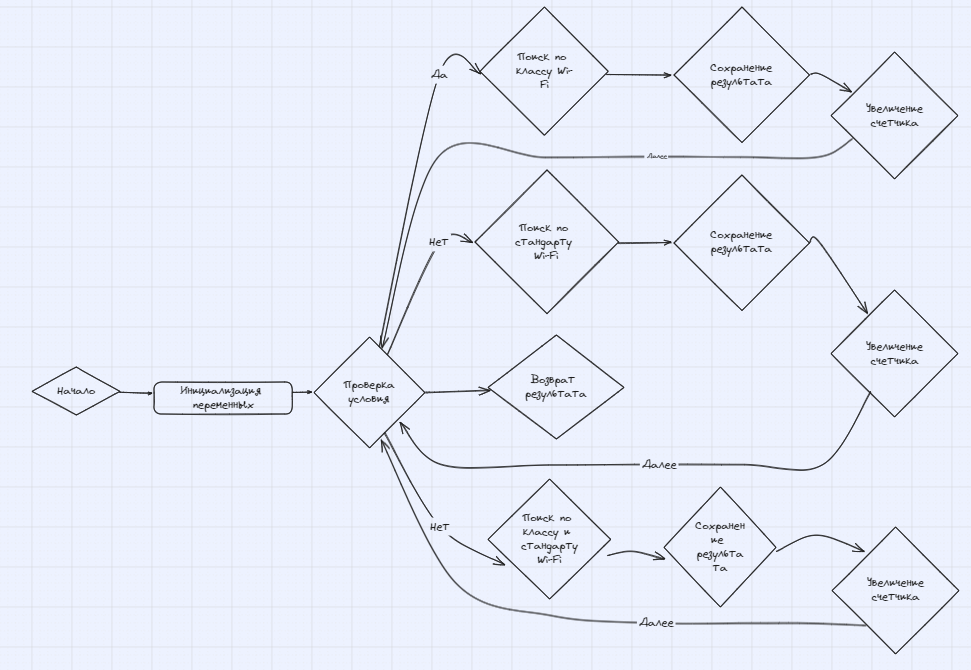


Рисунок 3 – searchRouter

На изображении представлена блок-схема алгоритма поиска записей по классу Wi-Fi и стандарту Wi-Fi.

1. Начало:
   * Программа начинает выполнение с этого блока.
2. Инициализация переменных:
   * В этом блоке происходит инициализация переменных, которые будут использоваться в алгоритме.
3. Проверка условий:
   * В этом блоке происходит проверка условий, которые определяют дальнейший ход выполнения программы.
4. Поиск по классу Wi-Fi:
   * Если условие для поиска по классу Wi-Fi выполнено (например, пользователь выбрал поиск по классу Wi-Fi), то выполнение переходит к следующему блоку.
5. Сохранение результата:
   * В этом блоке сохраняются результаты поиска по классу Wi-Fi.
6. Увеличение счетчика:
   * Увеличивается счетчик, который отслеживает количество найденных записей.
7. Поиск по стандарту Wi-Fi:
   * Если условие для поиска по стандарту Wi-Fi выполнено (например, пользователь выбрал поиск по стандарту Wi-Fi), то выполнение переходит к следующему блоку.
8. Сохранение результата:
   * В этом блоке сохраняются результаты поиска по стандарту Wi-Fi.
9. Увеличение счетчика:
   * Увеличивается счетчик, который отслеживает количество найденных записей.
10. Возврат результата:
    * В этом блоке происходит возврат результатов поиска.
11. Поиск по классу и стандарту Wi-Fi:
    * Если условие для поиска по классу и стандарту Wi-Fi выполнено (например, пользователь выбрал поиск по обоим критериям), то выполнение переходит к следующему блоку.
12. Сохранение не результата та:
    * В этом блоке сохраняются результаты поиска по обоим критериям.
13. Увеличение счетчика:
    * Увеличивается счетчик, который отслеживает количество найденных записей.
14. Нет:
    * Если ни одно из условий не выполнено, то выполнение переходит к следующему блоку.
15. Далее:
    * Этот блок указывает на продолжение выполнения программы.

Таким образом, блок-схема описывает алгоритм поиска записей по различным критериям (классу Wi-Fi, стандарту Wi-Fi или обоим критериям) и сохранение результатов поиска.

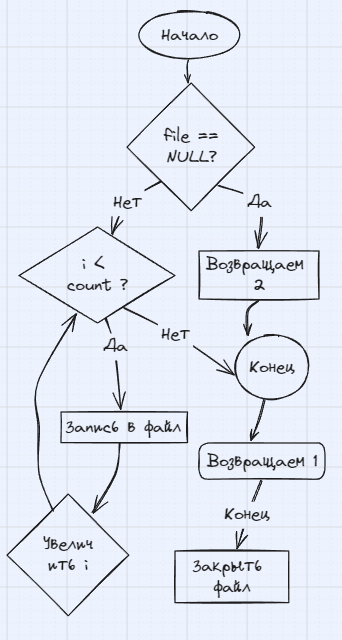


Рисунок 4 – saveToTextFile

На изображении представлена блок-схема алгоритма записи данных в файл. Начало:

* + Программа начинает выполнение с этого блока.

1. Проверка файла на NULL:
   * Проверяется, равен ли указатель на файл **NULL**. Если файл не был успешно открыт, указатель будет равен **NULL**.
   * Если файл равен **NULL** (нет), то выполнение переходит к блоку "Возвращаем 2".
   * Если файл не равен **NULL** (да), то выполнение переходит к следующему блоку.
2. Проверка индекса **i**:
   * Проверяется, меньше ли индекс **i** чем **count** (количество записей).
   * Если **i < count** (да), то выполнение переходит к блоку "Запись в файл".
   * Если **i >= count** (нет), то выполнение переходит к блоку "Возвращаем 1".
3. Запись в файл:
   * В этом блоке происходит запись данных в файл.
   * После записи данных индекс **i** увеличивается на 1.
4. Увеличение индекса **i**:
   * Индекс **i** увеличивается на 1.
   * Выполнение возвращается к блоку проверки индекса **i**.
5. Возвращаем 2:
   * Возвращается значение 2, что обычно означает ошибку (например, невозможно открыть файл для записи).
   * Выполнение программы завершается.
6. Возвращаем 1:
   * Возвращается значение 1, что обычно означает успешное выполнение (все записи успешно записаны в файл).
   * Выполнение программы завершается.
7. Закрыть файл:
   * Файл закрывается после завершения всех операций записи.

Таким образом, блок-схема описывает алгоритм записи данных в файл с проверкой на успешное открытие файла и последовательной записью всех записей из массива. Если файл не был успешно открыт, возвращается ошибка. Если все записи были успешно записаны, возвращается успешный статус, и файл закрывается.

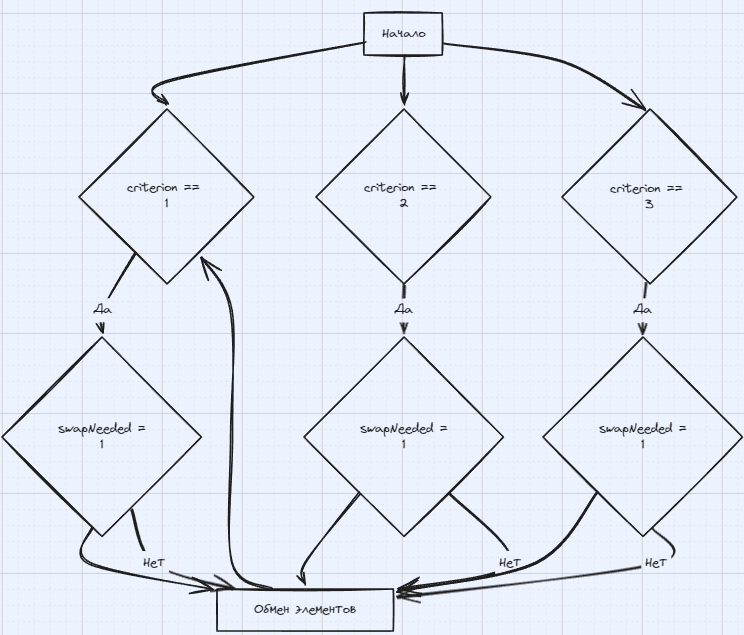


Рисунок 5 – bubbleSort

**Пошаговое объяснение алгоритма:**

1. Начало:
   * Программа начинает выполнение.
2. Проверка критерия сортировки:
   * Программа проверяет, какой критерий сортировки был выбран пользователем (**criterion == 1**, **criterion == 2** или **criterion == 3**).
3. Проверка необходимости обмена:
   * В зависимости от выбранного критерия, программа проверяет, нужно ли поменять местами текущие элементы массива (**swapNeeded == 1**).
4. Обмен элементов:
   * Если **swapNeeded == 1**, элементы меняются местами.
5. Повторение:
   * Процесс повторяется для всех элементов массива до тех пор, пока массив не будет отсортирован.
6. Завершение:
   * После завершения сортировки программа заканчивает выполнение.

Этот алгоритм позволяет сортировать массив структур **router\_t** по различным критериям, используя метод сортировки пузырьком.

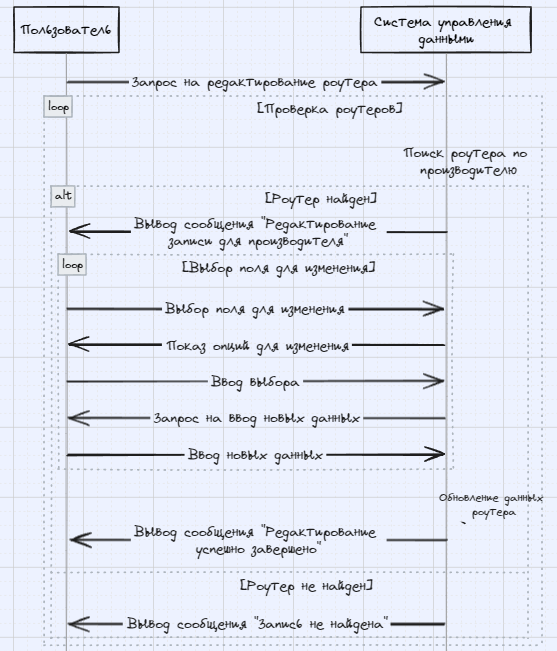


Рисунок 6 – editRouter

**Циклы и условия:**

* loop:
  + Цикл, который повторяет запрос на редактирование роутера до тех пор, пока пользователь не выберет другую опцию.
* alt:
  + Условный оператор, который проверяет, найден ли роутер в базе данных. В зависимости от результата проверки выполнение переходит либо к редактированию записи, либо к выводу сообщения о том, что запись не найдена.

**Пошаговое объяснение алгоритма:**

1. Начало:
   * Пользователь выбирает опцию редактирования роутера.
2. Запрос на редактирование роутера:
   * Система запрашивает у пользователя ввод названия производителя роутера.
3. Проверка наличия роутера:
   * Система проверяет, существует ли роутер с указанным производителем в базе данных.
4. Роутер найден:
   * Если роутер найден, система предлагает пользователю выбрать поле для редактирования.
5. Выбор поля для изменения:
   * Пользователь выбирает поле, которое хочет изменить.
6. Ввод нового значения:
   * Пользователь вводит новое значение для выбранного поля.
7. Подтверждение нового значения:
   * Пользователь подтверждает новое значение.
8. Обновление данных роутера:
   * Система обновляет данные роутера в базе данных.
9. Сообщение о успешном редактировании:
   * Система выводит сообщение о том, что редактирование успешно завершено.
10. Роутер не найден:
    * Если роутер не найден, система выводит сообщение о том, что запись не найдена.
11. Повторение:
    * Процесс повторяется для других роутеров или завершается, если пользователь выбирает другую опцию.

Этот алгоритм позволяет пользователю редактировать записи в базе данных маршрутизаторов, обеспечивая проверку наличия записи и корректность введенных данных.

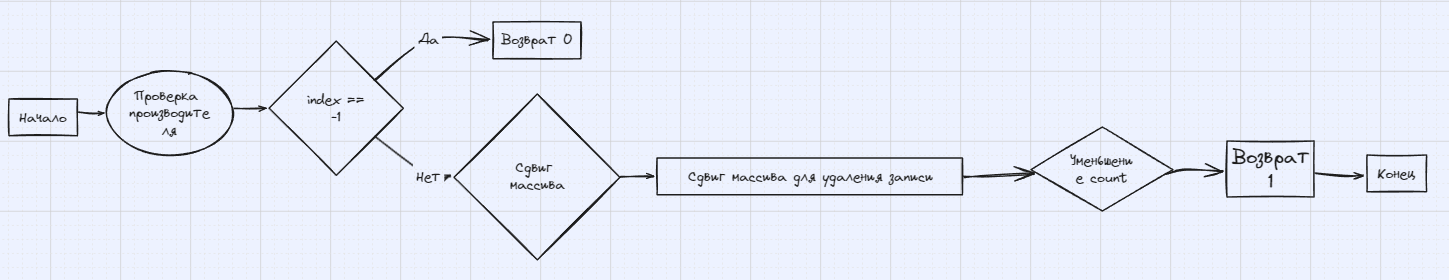


Рисунок 7 – deleteRouter

**Основные блоки и их описание:**

1. Начало:
   * Это начальный блок, с которого начинается выполнение алгоритма.
2. Проверка производителя:
   * Пользователь вводит производителя маршрутизатора, запись о котором нужно удалить.
3. index >= -1:
   * Это условие проверяет, найден ли индекс записи с указанным производителем в массиве. Если индекс найден (index >= 0), то выполнение переходит к следующему шагу. Если индекс не найден (index == -1), то выполнение переходит к блоку "Возврат 0".
4. Сдвиг массива:
   * Если запись найдена, выполняется сдвиг массива для удаления записи. Это означает, что все элементы массива, начиная с найденного индекса, сдвигаются на одну позицию влево, чтобы заполнить пустое место.
5. Сдвиг массива для удаления записи:
   * Этот блок выполняет фактический сдвиг элементов массива для удаления записи.
6. Уменьшение счетчика count:
   * После удаления записи счетчик count уменьшается на 1, так как количество записей в массиве уменьшилось.
7. Возврат 1:
   * Возврат значения 1 означает, что запись была успешно удалена.
8. Возврат 0:
   * Возврат значения 0 означает, что запись не была найдена и, следовательно, не была удалена.
9. Конец:
   * Это конечный блок, в который возвращается управление после завершения алгоритма.

**Пример использования программы**

Вход в меню:

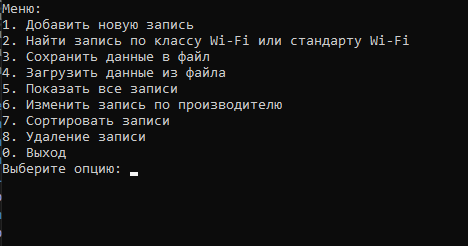


Рисунок 8 — Меню программы

Создадим 2 новые записи. Для этого выберем пункт 1 «Создать новую запись», после чего выведется подменю, в котором необходимо выбрать пункт 1 «Добавить новую запись», далее выбираем желаемое количество записей в нашем случае 2. Далее заполняем поля структуры для двух новых записей.

(Рисунок 9), (Рисунок 10).

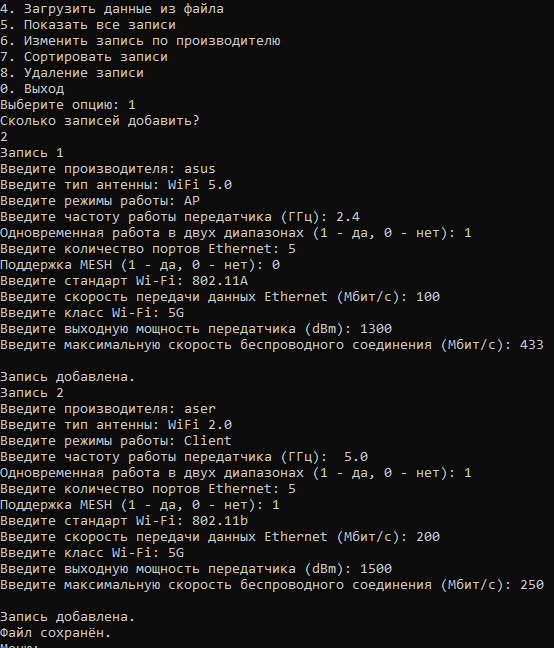


Рисунок 9 — Добавление двух новых записей



Рисунок 10 — Данные, записанные в файл

Далее выполним поиск записи по классу Wi-Fi или стандарту Wi-Fi. Для этого выбираем пункт «2. Найти запись по классу Wi-Fi или стандарту Wi-Fi», после чего в появившемся подменю выбираем пункт «2) По стандарту Wi-Fi», после чего вводим «5G». В результате нам выведется запись, разрешение которой совпадает с введенным (Рисунок 11).

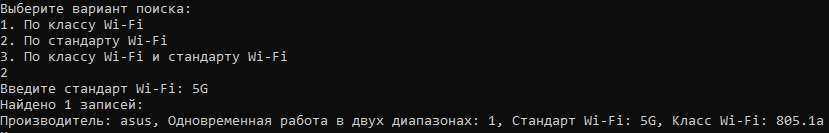


Рисунок 11 — Поиск записи

Так же выполним поиск по обоим параметрам. Результат представлен на рисунке. (Рисунок 12)

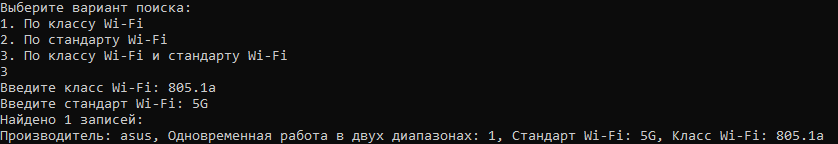


Рисунок 12 — Результат поиска по обоим параметрам

Пункты 3 и пункт 4 созданы для сохранения новых данных в файл для работы с ними позже и загрузки данных из файла соответственно (Рисунок 13)

(Рисунок 14).

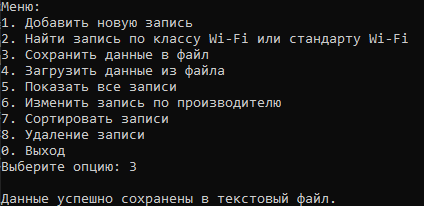


Рисунок 13 — Загрузка данных в файл.

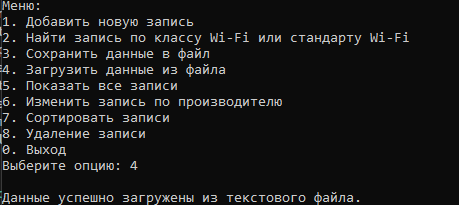


Рисунок 14 — Загрузка данных из файла.

Далее проведем удаление записи из файла для этого в меню программы выбираем «8», после этого вводим производителя для удаления (Рисунок 15)

(Рисунок 16).

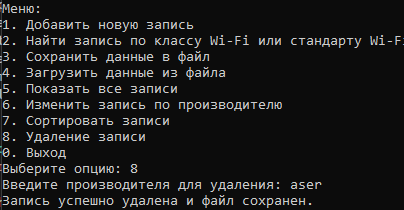


Рисунок 15 — Удаление данных из файла.



Рисунок 16 — Файл после удаления элемента.

Далее выведем записи с сортировкой по признакам:

1. Одновременная работа в двух диапазонах

2. Стандарт Wi-Fi

3. По обоим критериям.

Для этого выбираем в меню пункт «5. Показать все записи», после чего в появившемся подменю выбираем пункт по которому требуется провести сортировку «1) Одновременная работа в двух диапазонах», после чего записи будут отсортированы и выведены на экран (Рисунок 17) (Рисунок 18).

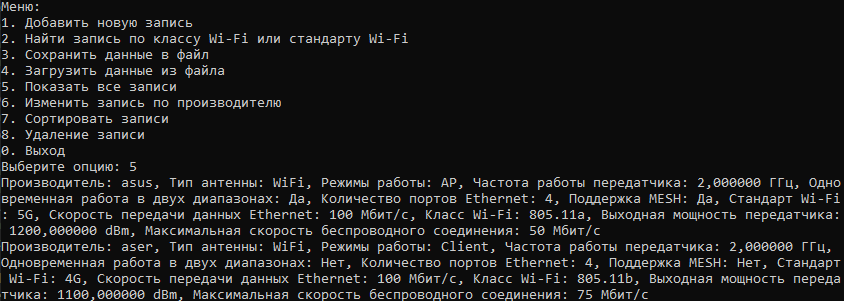


Рисунок17 — Изначальный файл.

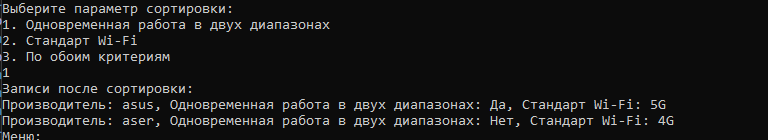


Рисунок 18 — Отсортированный файл.

Для редактирования уже существующего файла используем функцию редактирования по производителю. Для этого нажимаем в меню «6» а после вводим производителя чью запись мы хотим отредактировать. После этого мы можем выбрать что именно мы хотим изменить и изменяем это (Рисунок 19).

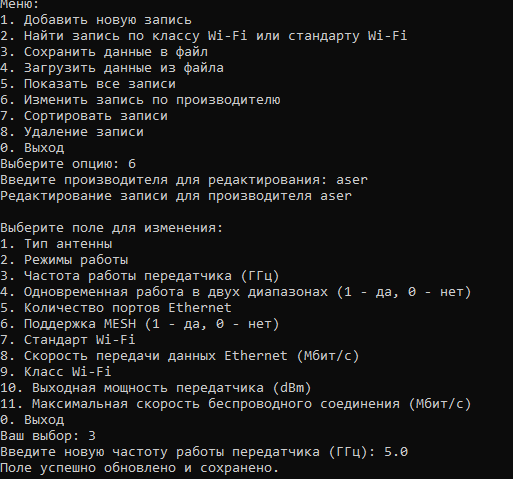
**

Рисунок 19 – Изменение файла.

**Заключение**

Функционал программы позволяет выполнять следующие действия:

1) Создание одной или нескольких записей

2) Поиск записи по признакам «Производитель», «Тип антенны», «Режимы работы», «Частота работы передатчика (ГГц)», «Одновременная работа в двух диапазонах», «Количество портов Ethernet», «Поддержка MESH»

3) Сохранение данных в файл. Также реализовано автоматическое сохранение при создании и изменении записей

4) Загрузка данных из файла

5) Сортировка записей по признакам «Стандарт Wi-Fi», «Скорость передачи данных Ethernet (Мбит/с)», «Класс Wi-Fi», «Выходная мощность передатчика (dBm)», «Максимальная скорость беспроводного соединения (Мбит/с)», «Тип антенны»

6) Изменение записи

7) Удаление данных из файла

8) Очистка занимаемой записями памяти.

Интерфейс программы разработан для обеспечения удобства использования. Он позволяет пользователю выбирать функции, вводить данные для новых и редактируемых записей, просматривать результаты работы, а также уведомляет о возможных ошибках ввода. Обеспечена гибкость при работе с записями: можно выбирать критерии сортировки и производить поиск по одному или нескольким полям одновременно. В результате работы было создано простое и функциональное приложение, эффективно решающее поставленные задачи. Программа соответствует заявленным требованиям и может быть использована для работы с данными в определенной предметной области.

**Список литературы**

# «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»

Авторы: Т.А. Жданова Ю.С. Бузыкова

«Практикум по си» - Основы программирования и алгоритмизации

Авторы курса: О.В. Минакова, Курипта О.В.

**Приложение**

#include <stdio.h>

#include <locale.h>

#include <string.h>

#include <stdlib.h>

#include <stdbool.h>

#include <ctype.h>

#define MAX\_ROUTERS 100

typedef struct {

char manufacturer[50];

char antennaType[50];

char operatingModes[50];

float transmitterFrequency;

bool dualBandOperation;

int ethernetPorts;

bool meshSupport;

char wifiStandard[50];

int ethernetSpeed;

char wifiClass[50];

float transmitterPower;

int maxWirelessSpeed;

} router\_t;

int addRouter(router\_t routers[], int\* count) {

printf("Введите производителя: ");

scanf("%s", routers[\*count].manufacturer);

while (getchar() != '\n'); // отчистка буфера ввода

printf("Введите тип антенны: ");

scanf("%s", routers[\*count].antennaType);

while (getchar() != '\n'); // отчистка буфера ввода

printf("Введите режимы работы: ");

scanf("%s", routers[\*count].operatingModes);

while (getchar() != '\n'); // отчистка буфера ввода

printf("Введите частоту работы передатчика (ГГц): ");

while (scanf("%f", &routers[\*count].transmitterFrequency) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

printf("Одновременная работа в двух диапазонах (1 - да, 0 - нет): ");

while (scanf("%d", &routers[\*count].dualBandOperation) != 1 || (routers[\*count].dualBandOperation != 0 && routers[\*count].dualBandOperation != 1)) {

printf("Ошибка: введите 1 - (да) или 0 - (нет).\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

printf("Введите количество портов Ethernet: ");

while (scanf("%d", &routers[\*count].ethernetPorts) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

printf("Поддержка MESH (1 - да, 0 - нет): ");

while (scanf("%d", &routers[\*count].meshSupport) != 1 || (routers[\*count].meshSupport != 0 && routers[\*count].meshSupport != 1)) {

printf("Ошибка: введите 1 - (да) или 0 - (нет).\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

printf("Введите стандарт Wi-Fi: ");

scanf("%s", routers[\*count].wifiStandard);

while (getchar() != '\n'); // отчистка буфера ввода

printf("Введите скорость передачи данных Ethernet (Мбит/с): ");

while (scanf("%d", &routers[\*count].ethernetSpeed) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

printf("Введите класс Wi-Fi: ");

scanf("%s", routers[\*count].wifiClass);

while (getchar() != '\n'); // отчистка буфера ввода

printf("Введите выходную мощность передатчика (dBm): ");

while (scanf("%f", &routers[\*count].transmitterPower) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

printf("Введите максимальную скорость беспроводного соединения (Мбит/с): ");

while (scanf("%d", &routers[\*count].maxWirelessSpeed) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

(\*count)++;

return 1;

}

bool isInteger(const char\* str) {

if (\*str == '\0') return false; // пустая строка — не число

// проверка каждого символа строки

while (\*str) {

if (!isdigit((unsigned char)\*str)) {

return false; // если символ не цифра, возвращаем false

}

str++;

}

return true; // все символы — цифры

}

int getValidInteger(const char\* prompt) {

char input[50];

int value;

while (true) {

printf("%s", prompt);

scanf("%49s", input); // считываем как строку

if (isInteger(input)) {

sscanf(input, "%d", &value); // преобразование строки в int

return value;

}

else {

printf("Ошибка: недопустимые символы. Введите число.\n");

}

}

}

int searchRouter(const router\_t routers[], int count, router\_t results[], char\* wifiClass, char\* wifiStandard, int searchOption) {

int resultCount = 0;

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

int mwifiClass = (searchOption == 1 || searchOption == 3) && strcmp(routers[i].wifiClass, wifiClass) == 0;

int mwifiStandard = (searchOption == 2 || searchOption == 3) && strcmp(routers[i].wifiStandard, wifiStandard) == 0;

if (searchOption == 1 && mwifiClass)

results[resultCount++] = routers[i]; // сохранение указателя на запись

else if (searchOption == 2 && mwifiStandard)

results[resultCount++] = routers[i];

else if (searchOption == 3 && mwifiClass && mwifiStandard)

results[resultCount++] = routers[i];

}

return resultCount;// возврат количества найденных записей через указатель

}

int saveToTextFile(router\_t routers[], int count, const char\* filename) {

FILE\* file = fopen(filename, "w");

if (file == NULL) {

return 2;

}

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

fprintf(file, "%s,%s,%s,%f,%d,%d,%d,%s,%d,%s,%f,%d\n",

routers[i].manufacturer,

routers[i].antennaType,

routers[i].operatingModes,

routers[i].transmitterFrequency,

routers[i].dualBandOperation,

routers[i].ethernetPorts,

routers[i].meshSupport,

routers[i].wifiStandard,

routers[i].ethernetSpeed,

routers[i].wifiClass,

routers[i].transmitterPower,

routers[i].maxWirelessSpeed);

}

fclose(file);

return 1;

}

int loadFromTextFile(router\_t routers[], int\* count, const char\* filename) {

FILE\* file = fopen(filename, "r");

\*count = 0;

while

(fscanf(file, "%49[^,],%49[^,],%49[^,],%f,%d,%d,%d,%49[^,],%d,%49[^,],%f,%d\n",

routers[\*count].manufacturer,

routers[\*count].antennaType,

routers[\*count].operatingModes,

&routers[\*count].transmitterFrequency,

&routers[\*count].dualBandOperation,

&routers[\*count].ethernetPorts,

&routers[\*count].meshSupport,

routers[\*count].wifiStandard,

&routers[\*count].ethernetSpeed,

routers[\*count].wifiClass,

&routers[\*count].transmitterPower,

&routers[\*count].maxWirelessSpeed) == 12) {

(\*count)++;

if (\*count >= MAX\_ROUTERS) {

return 2;

break;

}

}

fclose(file);

return 1;

}

int bubbleSort(router\_t\* routers, int count, int criterion) {

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

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

int swapNeeded = 0;

if (criterion == 1) {

// сортировка по одновременной работе в двух диапазонах

if (routers[j].dualBandOperation < routers[j + 1].dualBandOperation) {

swapNeeded = 1;

}

}

else if (criterion == 2) {

// сортировка по стандарту Wi-Fi

if (strcmp(routers[j].wifiStandard, routers[j + 1].wifiStandard) > 0) {

swapNeeded = 1;

}

}

else if (criterion == 3) {

// сортировка по обоим критериям

if (routers[j].dualBandOperation < routers[j + 1].dualBandOperation ||

(routers[j].dualBandOperation == routers[j + 1].dualBandOperation &&

strcmp(routers[j].wifiStandard, routers[j + 1].wifiStandard) > 0)) {

swapNeeded = 1;

}

}

// обмен элементов

if (swapNeeded) {

router\_t temp = routers[j];

routers[j] = routers[j + 1];

routers[j + 1] = temp;

}

}

}

return 1;

}

int editRouter(router\_t routers[], int count, const char\* manufacturer) {

int found = 0;

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

if (strcmp(routers[i].manufacturer, manufacturer) == 0) {

found = 1;

printf("Редактирование записи для производителя %s\n", manufacturer);

int choice;

do {

printf("\nВыберите поле для изменения:\n");

printf("1. Тип антенны\n");

printf("2. Режимы работы\n");

printf("3. Частота работы передатчика (ГГц)\n");

printf("4. Одновременная работа в двух диапазонах (1 - да, 0 - нет)\n");

printf("5. Количество портов Ethernet\n");

printf("6. Поддержка MESH (1 - да, 0 - нет)\n");

printf("7. Стандарт Wi-Fi\n");

printf("8. Скорость передачи данных Ethernet (Мбит/с)\n");

printf("9. Класс Wi-Fi\n");

printf("10. Выходная мощность передатчика (dBm)\n");

printf("11. Максимальная скорость беспроводного соединения (Мбит/с)\n");

printf("0. Выход\n");

printf("Ваш выбор: ");

scanf("%d", &choice);

while (getchar() != '\n'); // отчистка буфера ввода

switch (choice) {

case 1:

printf("Введите новый тип антенны: ");

scanf("%s", routers[i].antennaType);

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 2:

printf("Введите новые режимы работы: ");

scanf("%s", routers[i].operatingModes);

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 3:

printf("Введите новую частоту работы передатчика (ГГц): ");

while (scanf("%f", &routers[i].transmitterFrequency) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 4:

printf("Одновременная работа в двух диапазонах (1 - да, 0 - нет): ");

while (scanf("%d", &routers[i].dualBandOperation) != 1 || (routers[i].dualBandOperation != 0 && routers[i].dualBandOperation != 1)) {

printf("Ошибка: введите 1 - (да) или 0 - (нет).\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 5:

printf("Введите новое количество портов Ethernet: ");

while (scanf("%d", &routers[i].ethernetPorts) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 6:

printf("Поддержка MESH (1 - да, 0 - нет): ");

while (scanf("%d", &routers[i].meshSupport) != 1 || (routers[i].meshSupport != 0 && routers[i].meshSupport != 1)) {

printf("Ошибка: введите 1 - (да) или 0 - (нет).\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 7:

printf("Введите новый стандарт Wi-Fi: ");

scanf("%s", routers[i].wifiStandard);

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 8:

printf("Введите новую скорость передачи данных Ethernet (Мбит/с): ");

while (scanf("%d", &routers[i].ethernetSpeed) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 9:

printf("Введите новый класс Wi-Fi: ");

scanf("%s", routers[i].wifiClass);

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 10:

printf("Введите новую выходную мощность передатчика (dBm): ");

while (scanf("%f", &routers[i].transmitterPower) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 11:

printf("Введите новую максимальную скорость беспроводного соединения (Мбит/с): ");

while (scanf("%d", &routers[i].maxWirelessSpeed) != 1) {

printf("Ошибка: введите корректное значение.\n");

while (getchar() != '\n'); // отчистка буфера ввода

}

while (getchar() != '\n'); // отчистка буфера ввода

break;

case 0:

printf("Выход из режима редактирования.\n");

break;

default:

printf("Неверный выбор. Попробуйте снова.\n");

}

if (choice != 0) {

return 1;

}

} while (choice != 0);

return 0;

}

}

if (!found) {

printf("Запись не найдена.\n");

}

}

int deleteRouter(router\_t routers[], int\* count, const char\* manufacturer) {

int index = -1;

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

if (strcmp(routers[i].manufacturer, manufacturer) == 0) {

index = i;

break;

}

}

if (index == -1) {

return 0;

}

// сдвиг массива для удаления записи

for (int i = index; i < \*count - 1; i++) {

routers[i] = routers[i + 1];

}

(\*count)--;

return 1;

}

int main() {

setlocale(LC\_ALL, "ru");

router\_t\* routers = (router\_t\*)malloc(MAX\_ROUTERS \* sizeof(router\_t));

int count = 0;

int choice;

char filename[] = "data\_base.txt";

int sortCr = 0;

int cnt;

do {

printf("Меню:\n");

printf("1. Добавить новую запись\n");

printf("2. Найти запись по классу Wi-Fi или стандарту Wi-Fi\n");

printf("3. Сохранить данные в файл\n");

printf("4. Загрузить данные из файла\n");

printf("5. Показать все записи\n");

printf("6. Изменить запись по производителю\n");

printf("7. Сортировать записи\n");

printf("8. Удаление записи\n");

printf("0. Выход\n");

printf("Выберите опцию: ");

scanf("%d", &choice);

while (getchar() != '\n'); // отчистка буфера ввода

switch (choice) {

case 1: {

printf("Сколько записей добавить?\n");

scanf("%d", &cnt);

while (getchar() != '\n'); // отчистка буфера ввода

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

printf("Запись %d\n", i + 1);

int result1 = addRouter(routers, &count);

if (result1 == 1) {

printf("\nЗапись добавлена.\n");

}

else

printf("\nОшибка добавления записи\n");

}

printf("Файл сохранён.\n");

saveToTextFile(routers, count, filename);

break;

}

case 2: {

int searchOption;

printf("Выберите вариант поиска:\n1. По классу Wi-Fi\n2. По стандарту Wi-Fi\n3. По классу Wi-Fi и стандарту Wi-Fi\n");

scanf("%d", &searchOption);

while (getchar() != '\n'); // отчистка буфера ввода

char wifiClass[20];

char wifiStandard[20];

if (searchOption == 1 || searchOption == 3) {

printf("Введите класс Wi-Fi: ");

scanf("%19s", wifiClass);

while (getchar() != '\n'); // отчистка буфера ввода

}

if (searchOption == 2 || searchOption == 3) {

printf("Введите стандарт Wi-Fi: ");

scanf("%19s", wifiStandard);

while (getchar() != '\n'); // отчистка буфера ввода

}

router\_t results[MAX\_ROUTERS];

int foundCount = searchRouter(routers, count, results, wifiClass, wifiStandard, searchOption);

if (foundCount > 0) {

printf("Найдено %d записей:\n", foundCount);

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

printf("Производитель: %s, Одновременная работа в двух диапазонах: %d, Стандарт Wi-Fi: %s, Класс Wi-Fi: %s\n", results[i].manufacturer, results[i].dualBandOperation, results[i].wifiStandard, results[i].wifiClass);

}

}

else {

printf("Записи не найдены.\n");

}

break;

}

case 3: {

int result3 = saveToTextFile(routers, count, filename);

if (result3 == 1)

printf("\nДанные успешно сохранены в текстовый файл.\n");

else if (result3 == 2)

printf("Ошибка: невозможно открыть файл для записи.\n");

else

printf("\nОшибка сохранения данных\n");

break;

}

case 4: {

int result4 = loadFromTextFile(routers, &count, filename);

if (result4 == 1)

printf("\nДанные успешно загружены из текстового файла.\n");

else if (result4 == 2)

printf("\nДостигнуто максимальное количество записей.\n");

else

printf("\nОшибка считывания данных\n");

break;

}

case 5:

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

printf("Производитель: %s, Тип антенны: %s, Режимы работы: %s, Частота работы передатчика: %f ГГц, Одновременная работа в двух диапазонах: %s, Количество портов Ethernet: %d, Поддержка MESH: %s, Стандарт Wi-Fi: %s, Скорость передачи данных Ethernet: %d Мбит/с, Класс Wi-Fi: %s, Выходная мощность передатчика: %f dBm, Максимальная скорость беспроводного соединения: %d Мбит/с\n",

routers[i].manufacturer,

routers[i].antennaType,

routers[i].operatingModes,

routers[i].transmitterFrequency,

(routers[i].dualBandOperation ? "Да" : "Нет"),

routers[i].ethernetPorts,

(routers[i].meshSupport ? "Да" : "Нет"),

routers[i].wifiStandard,

routers[i].ethernetSpeed,

routers[i].wifiClass,

routers[i].transmitterPower,

routers[i].maxWirelessSpeed);

}

break;

case 6: {

char manufacturer[50];

printf("Введите производителя для редактирования: ");

scanf("%s", manufacturer);

while (getchar() != '\n'); // отчистка буфера ввода

int result6 = editRouter(routers, count, manufacturer);

if (result6 == 1) {

printf("Поле успешно обновлено и сохранено.\n");

saveToTextFile(routers, count, filename);

}

else

printf("\nОшибка обновления поля\n");

break;

}

case 7: {

printf("Выберите параметр сортировки:\n1. Одновременная работа в двух диапазонах \n2. Стандарт Wi-Fi \n3. По обоим критериям\n");

scanf("%d", &sortCr);

while (getchar() != '\n'); // отчистка буфера ввода

int result7 = bubbleSort(routers, count, sortCr);

if (result7 == 1) {

printf("Записи после сортировки:\n");

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

printf("Производитель: %s, Одновременная работа в двух диапазонах: %s, Стандарт Wi-Fi: %s\n", routers[i].manufacturer, (routers[i].dualBandOperation ? "Да" : "Нет"), routers[i].wifiStandard);

}

}

break;

}

case 8:

{

char manufacturer[50];

printf("Введите производителя для удаления: ");

scanf("%49s", manufacturer);

while (getchar() != '\n'); // отчистка буфера ввода

int result = deleteRouter(routers, &count, manufacturer);

if (result == 1) {

printf("Запись успешно удалена и файл сохранен.\n");

saveToTextFile(routers, count, filename);

}

else {

printf("Запись с указанным производителем не найдена.\n");

}

break;

}

case 0:

printf("Выход из программы.\n");

break;

default:

printf("Неверный выбор.\n");

}

} while (choice != 0);

}