

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  
«Серверные платформы»

Расчетно-пояснительная записка

Разработал студент

15.01.2025 Н.С. Полянский  
Подпись, дата Инициалы, фамилия

Руководитель

15.01.25 Н.В. Акамсина  
Подпись, дата Инициалы, фамилия

Нормоконтролер

\_\_\_\_\_  
Подпись, дата Инициалы, фамилия

Защищена 15.01.25 Оценка уровень верн.

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  
«Серверные платформы»»

Студент БТИИ-241 Полянский Николай Сергеевич  
Группа, фамилия, имя, отчество

База данных «Серверные платформы», Признак поиска: сокет, максимальное количество устанавливаемых CPU, Вариант сортировки: производитель, форм-фактор сервера.

Технические условия Windows 11, CLion, язык программирования C

Содержание и объем проекта (графические работы, расчеты и прочее):  
35 стр, 24 рисунков, 1 таб, 1 приложений

Сроки выполнения этапов анализ и постановка задачи 10.09.24 - 05.10.24


Разработка пошаговой детализации программы 06.11.24 - 11.11.24 ;


Реализация программы 10.09.24 - 05.12.24 ;

Тестирование программы 06.12.24 - 11.12.24 ;

Оформление пояснительной записки 11.12.24 - 14.12.24 .

Срок защиты курсового проекта: 15.01.25

Руководитель  Н.В.Акамсина  
Подпись, дата Инициалы, фамилия

Задание принял студент  09.09.24 Н.С. Полянский  
Подпись, дата Инициалы, фамилия

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
1 ПОСТАНОВКА ЗАДАЧИ .....	5
2 КОНСТРУИРОВАНИЕ ПРОГРАММЫ .....	7
3 ТЕСТИРОВАНИЕ ПРОГРАММЫ .....	15
ЗАКЛЮЧЕНИЕ .....	24
СПИСОК ЛИТЕРАТУРЫ .....	25
ПРИЛОЖЕНИЕ .....	26

## **ВВЕДЕНИЕ**

База данных - это совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных. Это может быть представлено как актуальное состояние некоторой предметной области и использоваться для удовлетворения информационных потребностей пользователей. Базы данных могут содержать различные типы данных, включая слова, цифры, изображения, видео, аудио и текстовые файлы. Управление базой данных подразумевает собой хранение данных в вычислительной системе, логическую структуру данных и возможность их поиска и обработки.

Цель – разработка программы для работы с записями данных различного типа оборудования серверных платформ.

Для реализации поставленной цели, необходимо разбить задачу на подзадачи:

1. Продумать, как пользователь может взаимодействовать с БД
2. Реализовать необходимые функции, предварительно создав алгоритмы к ним.
3. Для удобства пользователя реализовать интерфейс взаимодействия с программой
4. Обеспечить выполнение тех или иных функций в зависимости от выбора действия пользователем.
5. Реализовать защиту от некорректных действий пользователя при взаимодействии с программой

# 1 ПОСТАНОВКА ЗАДАЧИ

Задачей курсового проекта является создание файловой базы данных «Серверные платформы». Программа должна предоставлять пользователю функциональные возможности для создания, поиска по ключу, записи и чтения, вывода сортированных записей, включая завершение программы. Тематика базы данных подразумевает следующие поля ввода:

Производитель – строка, содержащая название производителя (не более 20 символов). Это поле предназначено для выполнения сортировки.

Форм-фактор сервера – строка, содержащая тип и габариты серверного оборудования (не более 20 символов). Это поле предназначено для выполнения сортировки.

Сокет – строка, содержащая тип разъема на материнской плате (не более 20 символов). Это поле предназначено для поиска записи по типу сокета.

Максимальное количество устанавливаемых CPU – это целое число, отвечающее за то, сколько можно установить CPU. Это поле предназначено для поиска записи по его количеству.

Объем оперативной памяти – это вещественное число, отвечающее за максимальную вместимость оперативной памяти.

Для организации управления записями необходимо организовать меню, в зависимости от выбора пункта пользователем, которое будет выполнять следующие операции:

1. Создание новой записи.
2. Поиск записи (по сокету и (или) максимальному количеству устанавливаемых CPU).
3. Запись и чтение данных.
4. Сортировка записей (по производителю и (или) форм-фактору сервера).

Для того чтобы были реализованы действия по выбору определенного пункта меню, необходимо реализовать следующие функции:

- Чтение файла
- Заполнение структуры
- Запись в файл
- Выполнение поиска
- Вывод информации о серверной платформе
- Сравнение строк (по производителю и форм-фактору сервера)
- Сортировка
- Добавление платформы
- Удаление платформы

## 2 КОНСТРУИРОВАНИЕ ПРОГРАММЫ

Разобьем задачу по подзадачи. В программе управление всеми действиями над базой данных выполняет функция `main()`. В ней происходят необходимые вызовы функций, для выполнения задачи от пользователя (рисунок 1).

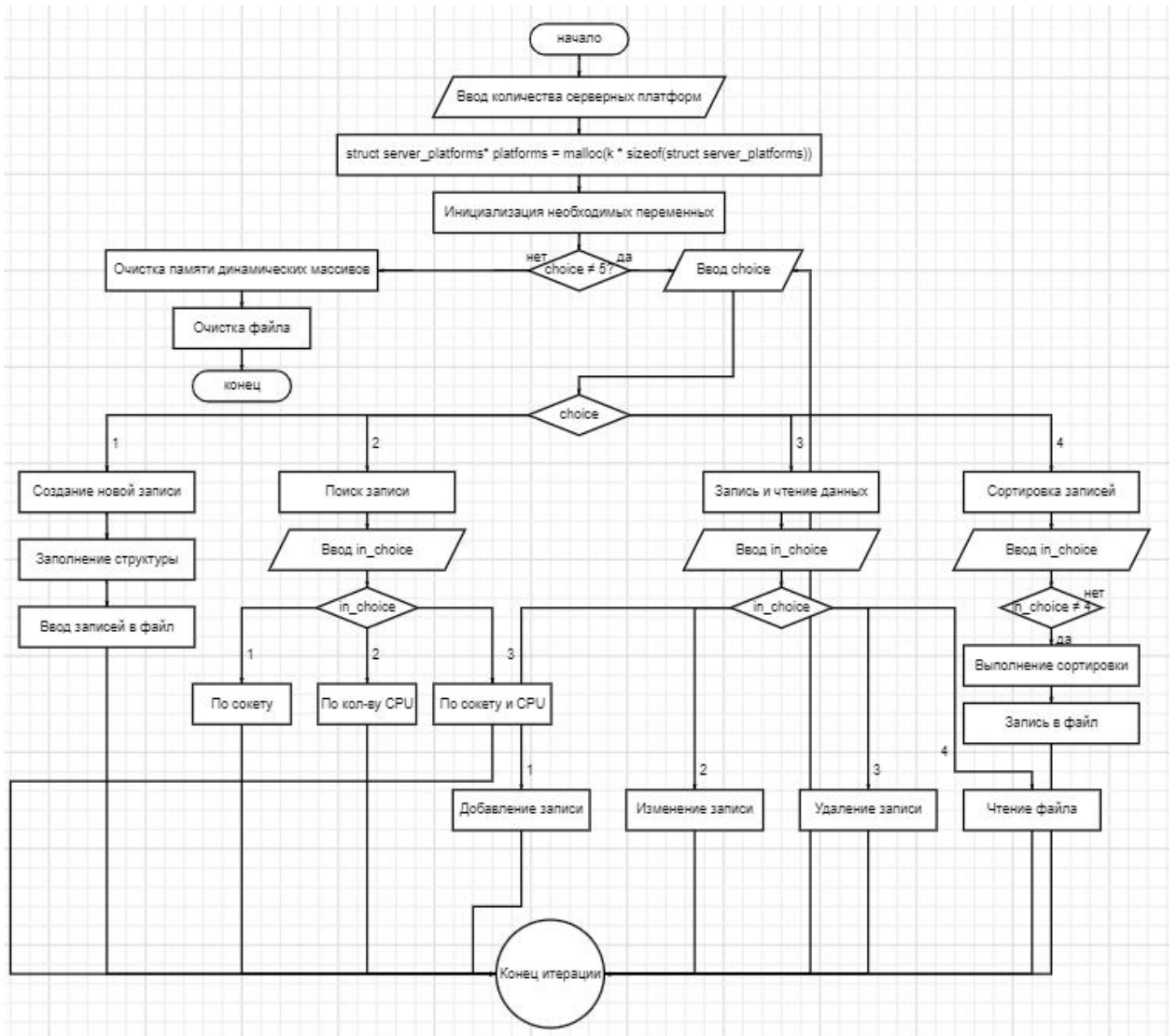


Рисунок 1 – блок-схема функции `main()`

Описание собственных функций предоставлено в таблице 1.



Таблица 1 – описание собственных функций

Функция	Описание
int read_file()	Открывает файл, проверяя его на отсутствие ошибок.
void fill_platforms(struct server_platforms* platforms, int k)	Позволяет пользователю вводить информацию о серверной платформе.
int write_file(struct server_platforms* platforms, int k)	Записывает в файл внесенную пользователем информацию в структуру.
int* do_search(struct server_platforms* platforms, int k, char* socket, int max_CPU)	Позволяет выполнить поиск по введенному пользователем значению в файле.
void print_platform(struct server_platforms platform)	Выводит информацию о серверной платформе.
int compare_manufacturer(const void* var_1, const void* var_2)	Сравнивает производителей (строки).
int compare_form_factor(const void* var_1, const void* var_2)	Сравнивает форм-факторы сервера (строки).
void sort_platforms(struct server_platforms* platforms, int k, int choice)	Сортирует каждую платформу по возрастанию в зависимости от выбранного критерия.
void add_platform(struct server_platforms* platforms, int i)	Дает возможность добавления новой платформы, с последующим занесением в файл.
void del_platform(struct server_platforms* platforms, int k, int i)	Позволяет удалить определенную платформу, с последующим изменением файла.

1. Функция int read\_file() открывает файл на чтение и проверяет на отсутствие ошибок. Если чтение файла не удалось, функция возвращает «1»,



далее пользователь получает информацию об ошибке чтения файла. Блок-схема функции на рисунке 2.

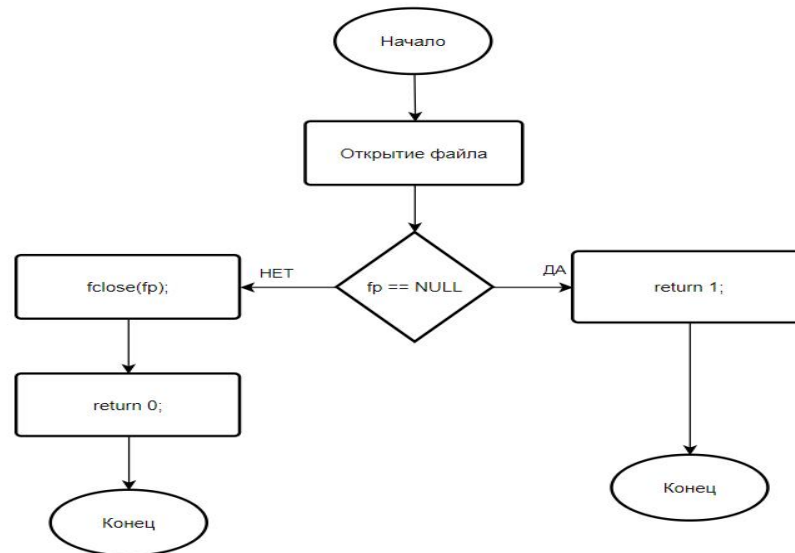


Рисунок 2 – блок-схема функции read\_file()

2. Функция void fill\_platforms(struct server\_platforms\* platforms, int k) добавляет в указатель на структуру данные о платформе. Информацию о каждом свойстве вводит пользователь. Блок-схема функции на рисунке 3.

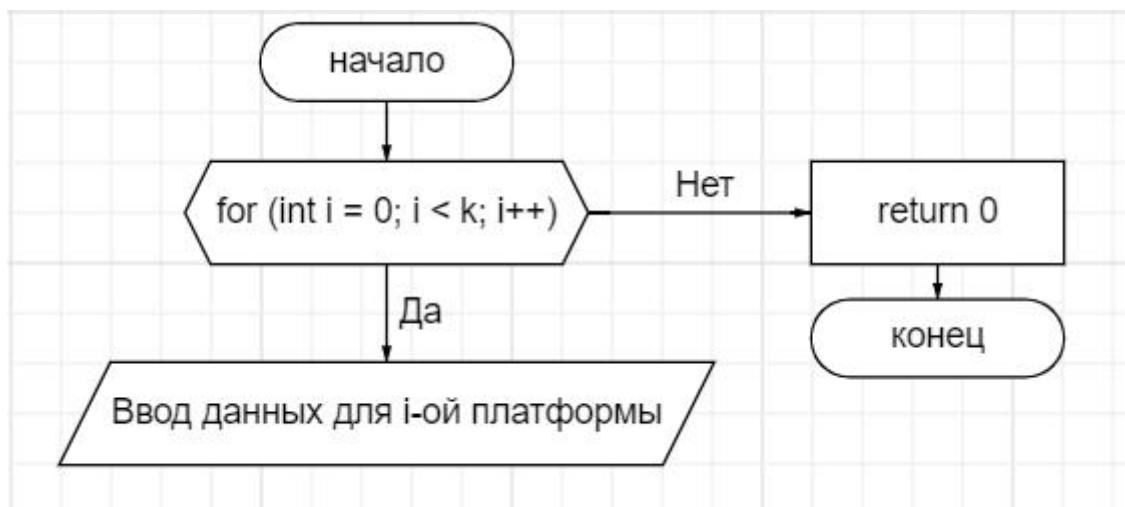


Рисунок 3 – блок-схема функции fill\_platforms()

3. Функция `int write_file(struct server_platforms* platforms, int k)` открывает файл на запись, с удалением существующего прежде (если файла до этого не было, то он создается). Далее идет копирование данных из платформы в файл. Блок-схема функции на рисунке 4.

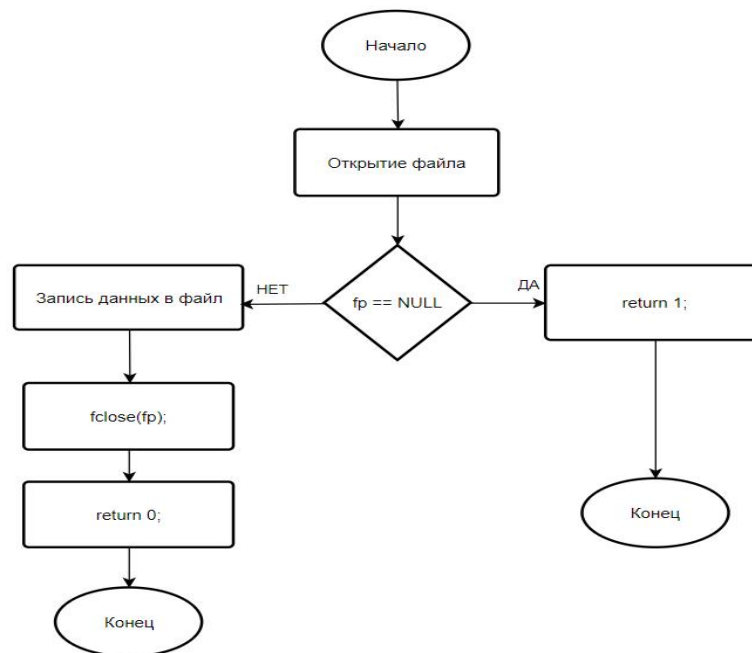


Рисунок 4 – блок-схема функции `write_file()`

4. Функция `int* do_search(struct server_platforms* platforms, int k, char* socket, int max_CPU)`. Сравнивает строки, введенные пользователем, для осуществления поиска платформы по определенному критерию, который пользователь запросил. В случае совпадения строк, в динамический целочисленный массив происходит ввод индекса строки. Блок-схема функции на рисунке 5.

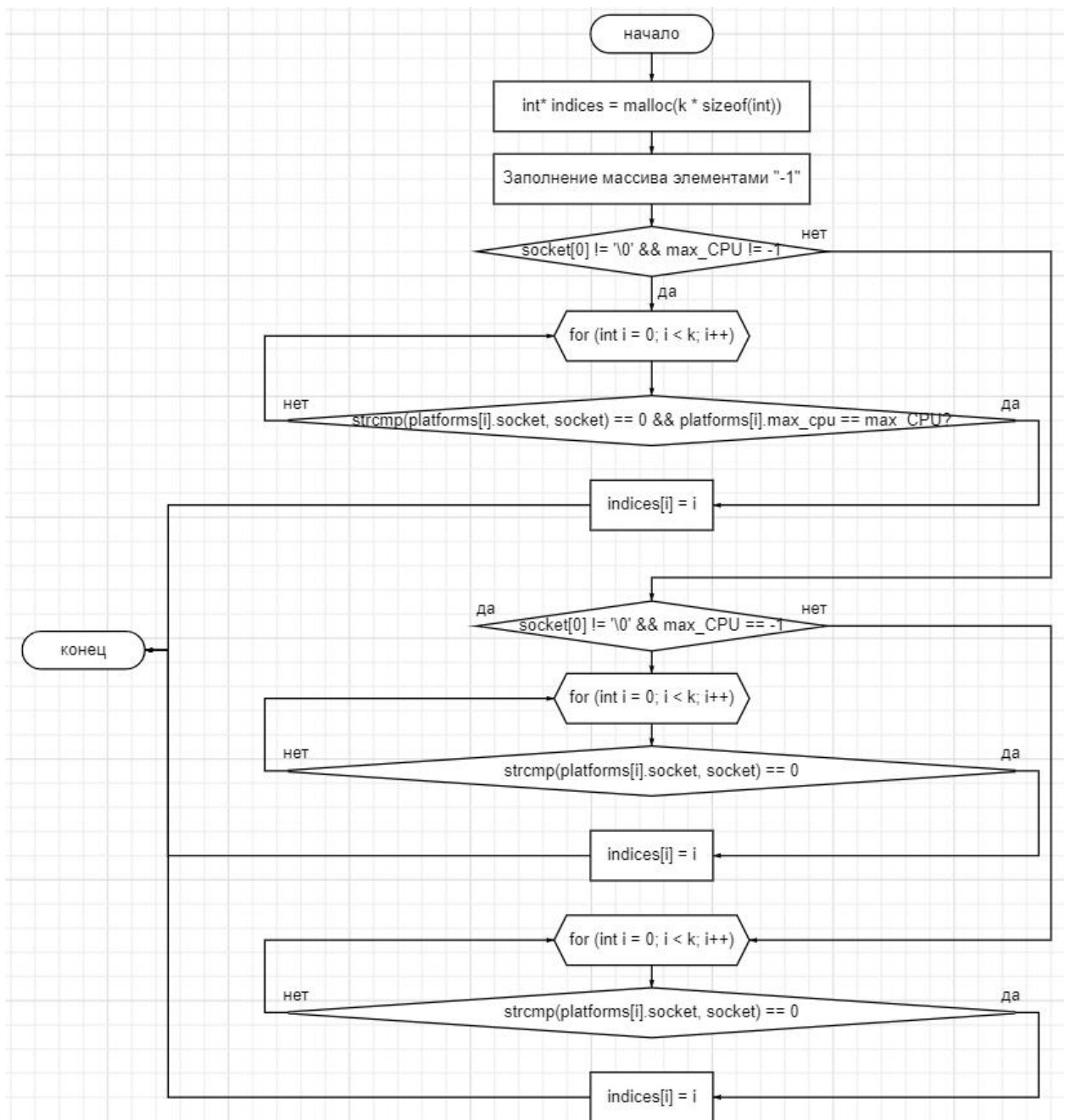


Рисунок 5 – блок-схема функции do\_search()

5. Функция void print\_platform(struct server\_platforms platform) выводит информацию о данных определенной платформы. Блок-схема функции на рисунке 6.

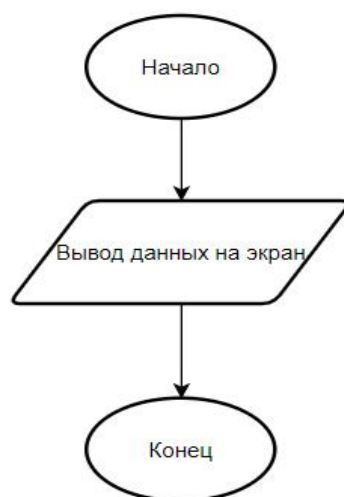


Рисунок 6 – блок-схема функции `print_platform()`

6. Функция `int compare_manufacturer(const void* var_1, const void* var_2)` сравнивает строки по производителю и возвращает определенное значение, которое вернет функция `strcmp()`, для дальнейшей сортировки. Блок-схема функции на рисунке 7.

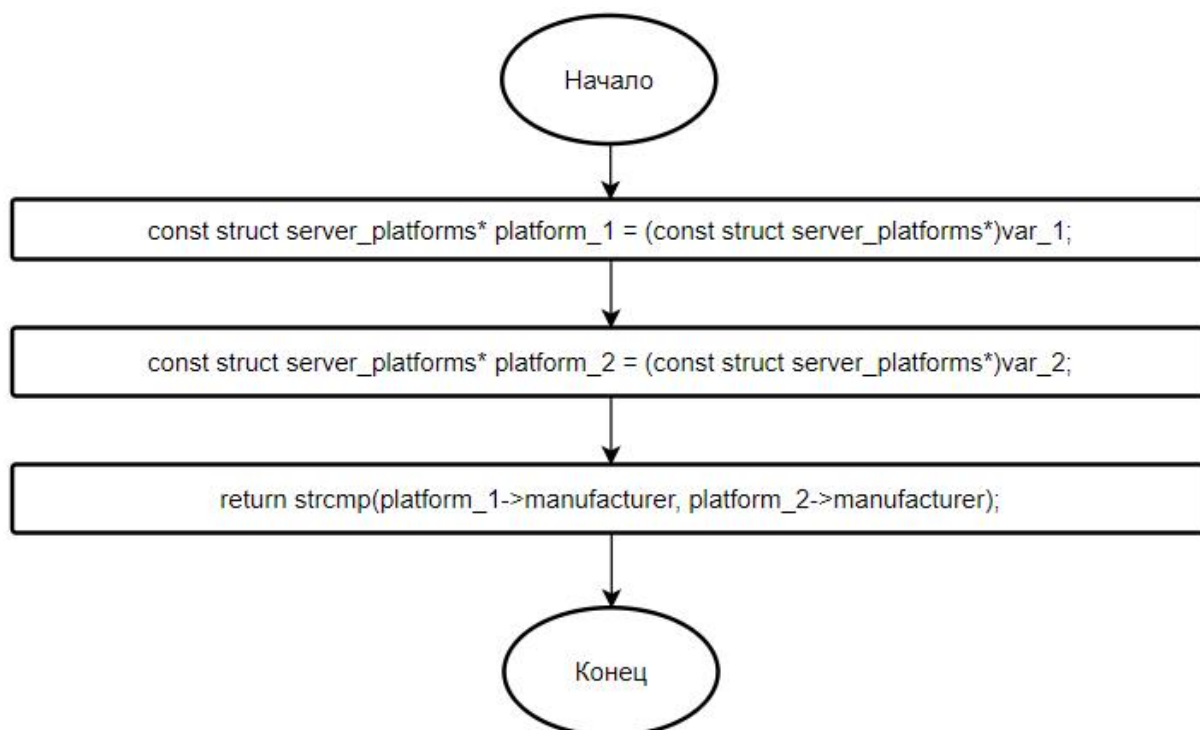


Рисунок 7 – блок-схема функции `compare_manufacturer()`

7. Функция `int compare_form_factor(const void* var_1, const void* var_2)` сравнивает строки по форм-фактору сервера и возвращает определенное значение, которое вернет функция `strcmp()`, для дальнейшей сортировки. Блок-схема функции на рисунке 8.

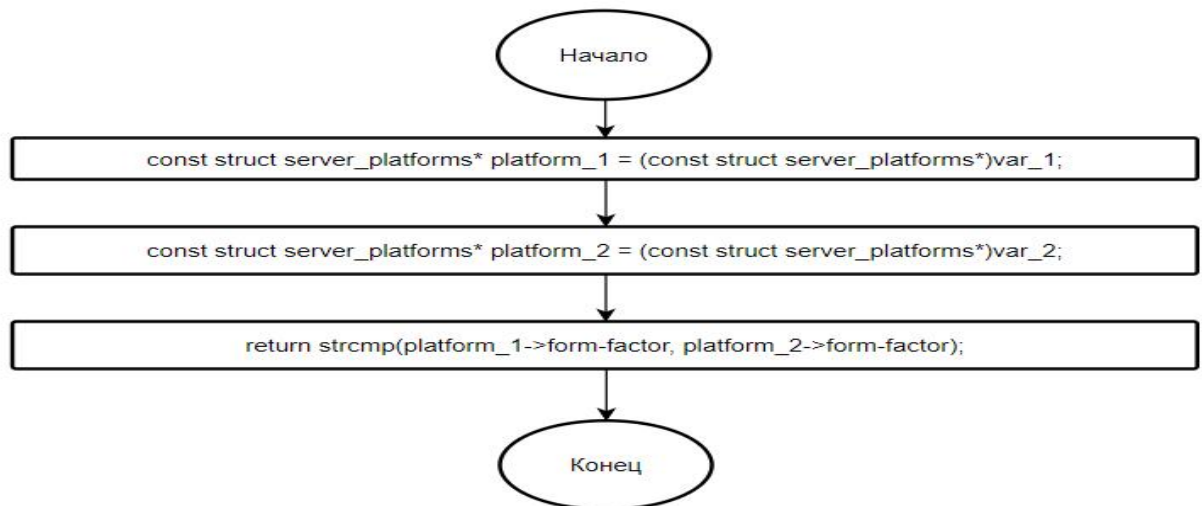


Рисунок 8 – блок-схема функции `compare_form_factor()`

8. Функция `void sort_platforms(struct server_platforms* platforms, int k, int choice)` сортирует платформы по одному или двум критериям (в зависимости от выбора пользователя). Блок-схема функции на рисунке 9.

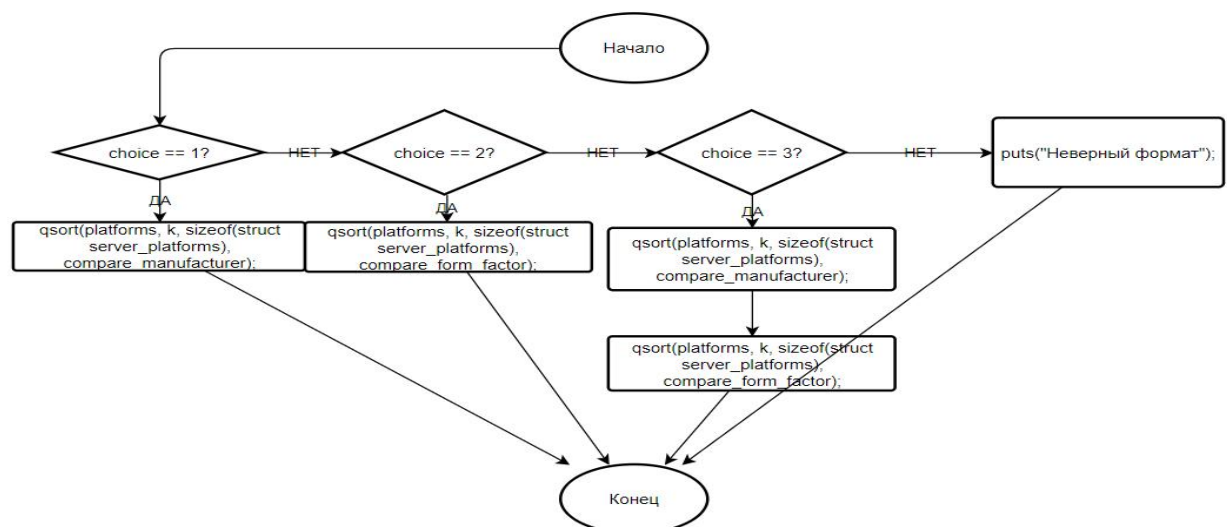


Рисунок 9 – блок-схема функции `sort_platforms()`

9. Функция `void add_platform(struct server_platforms* platforms, int i)` добавляет в структуру информацию о новой платформе с дальнейшим переносом ее в файл. Блок-схема функции на рисунке 10.



Рисунок 10 – блок-схема функции `add_platform()`

10. Функция `void del_platform(struct server_platforms* platforms, int k, int i)` удаляет платформу из структуры, с дальнейшим изменением файла. Блок-схема функции на рисунке 11.

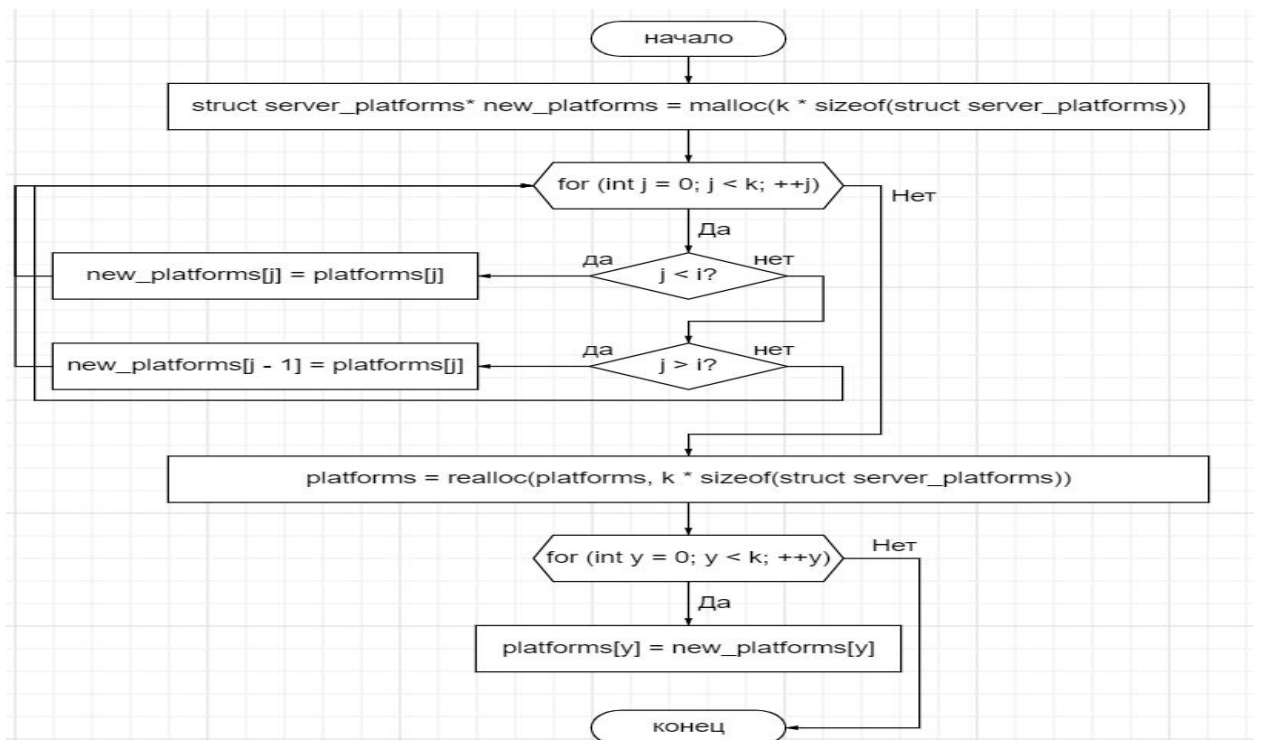


Рисунок 11 – блок-схема функции `del_platform()`

### 3 ТЕСТИРОВАНИЕ ПРОГРАММЫ

При запуске программы пользователю предлагается ввод количества платформ, которые будет содержать файл. Далее он может выбрать необходимое действие из предложенного меню. Для этого ему необходимо ввести число (1-5) и нажать enter. Предложенное меню показано на рисунке 12.

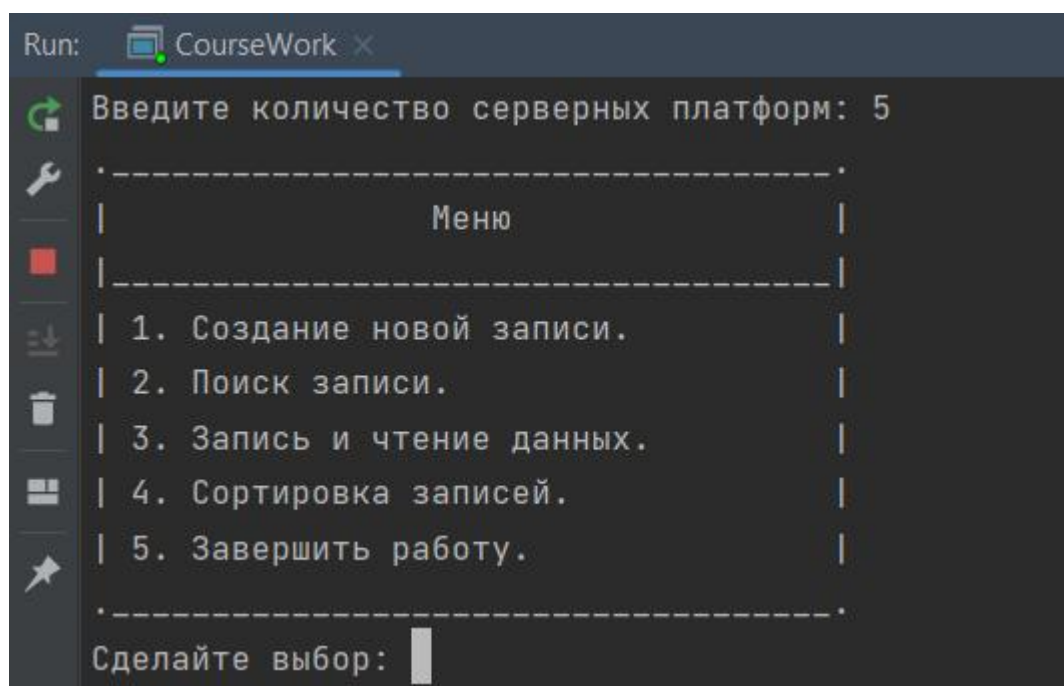


Рисунок 12 – меню при запуске программы

При выборе пункта «1», пользователю предлагается начать заполнять информацию о каждой платформе. После того, как пользователь введет информацию, идет автоматическое занесение информации в файл. Пример заполнения информации о платформах на рисунке 13 и результат занесения информации в файл на рисунке 14.



```
Сделайте выбор: 1
Производитель: DELL
Форм-фактор сервера: Tower
Сокет: LGA
Максимальное кол-во CPU: 6
Объем оперативной памяти: 24

Производитель: HP
Форм-фактор сервера: Rack
Сокет: PGA
Максимальное кол-во CPU: 7
Объем оперативной памяти: 32

Производитель: BROCADE
Форм-фактор сервера: Blade
Сокет: SP3
Максимальное кол-во CPU: 8
Объем оперативной памяти: 28

Производитель: HDINIX
Форм-фактор сервера: Steker
Сокет: HTG
Максимальное кол-во CPU: 4
Объем оперативной памяти: 12

Производитель: BROWLER
Форм-фактор сервера: Lager
Сокет: NTX
Максимальное кол-во CPU: 3
Объем оперативной памяти: 16

Создание записи прошло успешно
```

Рисунок 13 – Заполнение информации о платформах

```
1  Производитель: DELL
2  Форм-фактор сервера: Tower
3  Сокет: LGA
4  Максимальное кол-во CPU: 6
5  Объем оперативной памяти: 24.000000
6
7  Производитель: HP
8  Форм-фактор сервера: Rack
9  Сокет: PGA
10 Максимальное кол-во CPU: 7
11 Объем оперативной памяти: 32.000000
12
13 Производитель: BROCADE
14 Форм-фактор сервера: Blade
15 Сокет: SP3
16 Максимальное кол-во CPU: 8
17 Объем оперативной памяти: 28.000000
18
19 Производитель: HDINIX
20 Форм-фактор сервера: Steker
21 Сокет: HTG
22 Максимальное кол-во CPU: 4
23 Объем оперативной памяти: 12.000000
24
25 Производитель: BROWLER
26 Форм-фактор сервера: Lager
27 Сокет: NTX
28 Максимальное кол-во CPU: 3
29 Объем оперативной памяти: 16.000000
30
```

Рисунок 14 – содержание файла после занесения в него информации

После того, как данные были занесены в файл, пользователю снова всплывает меню выбора (рисунок 12). Теперь выберем пункт 2. После этого

пользователю всплывает меню для выбора по какой информации будет происходить поиск. Например, введем «1» (По сокету). Далее введем название сокета и в консоль выводится информация о платформе по найденному сокету. Выполнение этих действий на рисунке 15.

```
Сделайте выбор: 2
.------.
|                Поиск записи                |
|-----|
| 1. По сокету.                               |
| 2. По максимальному кол-ву CPU.            |
| 3. По сокету и максимальному кол-ву CPU.   |
| 4. Вернуться в меню.                       |
|-----|
Сделайте выбор: 1
Введите название сокета: SP3
Производитель: BROCADE
Форм-фактор сервера: Blade
Сокет: SP3
Максимальное кол-во CPU: 8
Объем оперативной памяти: 28.000000
```

Рисунок 15 – результат поиска информации о платформе по сокету

Теперь попробуем найти информацию о платформе по максимальному количеству CPU. Результат поиска на рисунке 16.

```
Сделайте выбор: 2
Введите максимальное кол-во CPU: 7
Производитель: HP
Форм-фактор сервера: Rack
Сокет: PGA
Максимальное кол-во CPU: 7
Объем оперативной памяти: 32.000000
```

Рисунок 16 – результат поиска информации по количеству CPU

Теперь протестируем пункт «3». После выбора пункта в консоль высвечивается новое окно с выбором определенных действий. Например, выберем пункт «добавить запись», для этого введем в консоль «1». Результат работы описанных действий на рисунке 17, результат занесения информации о новой платформе на рисунке 18.

```
Сделайте выбор: 3
.------.
|      Запись и чтение данных      |
|-----|
| 1. Добавить запись.              |
| 2. Изменить записи.              |
| 3. Удалить записи.               |
| 4. Чтение файла.                 |
| 5. Вернуться в меню.             |
|-----|
Сделайте выбор: 1
Введите кол-во добавления записей: 1
Производитель: HRIX
Форм-фактор сервера: Sme1a
Сокет: FGT
Максимальное кол-во CPU: 12
Объем оперативной памяти: 48

Записи успешно добавлены
```

Рисунок 17 – результат занесения информации о новой платформе



1	<u>Производитель: DELL</u>
2	<u>Форм-фактор сервера: Tower</u>
3	<u>Сокет: LGA</u>
4	<u>Максимальное кол-во CPU: 6</u>
5	<u>Объем оперативной памяти: 24.000000</u>
6	
7	<u>Производитель: HP</u>
8	<u>Форм-фактор сервера: Rack</u>
9	<u>Сокет: PGA</u>
10	<u>Максимальное кол-во CPU: 7</u>
11	<u>Объем оперативной памяти: 32.000000</u>
12	
13	<u>Производитель: BROCADE</u>
14	<u>Форм-фактор сервера: Blade</u>
15	<u>Сокет: SP3</u>
16	<u>Максимальное кол-во CPU: 8</u>
17	<u>Объем оперативной памяти: 28.000000</u>
18	
19	<u>Производитель: HDINIX</u>
20	<u>Форм-фактор сервера: Steker</u>
21	<u>Сокет: HTG</u>
22	<u>Максимальное кол-во CPU: 4</u>
23	<u>Объем оперативной памяти: 12.000000</u>
24	
25	<u>Производитель: BROWLER</u>
26	<u>Форм-фактор сервера: Lager</u>
27	<u>Сокет: NTX</u>
28	<u>Максимальное кол-во CPU: 3</u>
29	<u>Объем оперативной памяти: 16.000000</u>
30	
31	<u>Производитель: HRIX</u>
32	<u>Форм-фактор сервера: Smela</u>
33	<u>Сокет: FGT</u>
34	<u>Максимальное кол-во CPU: 12</u>
35	<u>Объем оперативной памяти: 48.000000</u>
36	

Рисунок 18 – результат занесения информации о файле

Теперь удалим добавленную платформу, тем самым выполнив пункт «3» (Удалить запись). Результат выполнения работы программы на рисунке 19 и результат удаления данных о платформе на рисунке 20.

```

*-----*
|      Запись и чтение данных      |
|-----|
| 1. Добавить запись.              |
| 2. Изменить записи.              |
| 3. Удалить записи.               |
| 4. Чтение файла.                 |
| 5. Вернуться в меню.             |
|-----|
Сделайте выбор: 3
Введите номер серверной платформы, которую вы хотите удалить: 6
Платформа успешно удалена

```

Рисунок 19 – результат работы программы при удалении записи

```

1  Производитель: DELL
2  Форм-фактор сервера: Tower
3  Сокет: LGA
4  Максимальное кол-во CPU: 6
5  Объем оперативной памяти: 24.000000
6
7  Производитель: HP
8  Форм-фактор сервера: Rack
9  Сокет: PGA
10 Максимальное кол-во CPU: 7
11 Объем оперативной памяти: 32.000000
12
13 Производитель: BROCADE
14 Форм-фактор сервера: Blade
15 Сокет: SP3
16 Максимальное кол-во CPU: 8
17 Объем оперативной памяти: 28.000000
18
19 Производитель: HDINIX
20 Форм-фактор сервера: Steker
21 Сокет: HTG
22 Максимальное кол-во CPU: 4
23 Объем оперативной памяти: 12.000000
24
25 Производитель: BROWLER
26 Форм-фактор сервера: Lager
27 Сокет: NTX
28 Максимальное кол-во CPU: 3
29 Объем оперативной памяти: 16.000000
30
31

```

Рисунок 20 – результат удаления записи из файла

Теперь протестируем сортировку файла. Выбираем пункт «4», и далее высвечивается окно с выбором сортировки. Отсортируем по производителю. Для этого введем в консоль «1» (рисунок 21). После этого файл автоматически отсортировался (рисунок 22).

```
Сделайте выбор: 4
|-----|
|          Сортировка записей          |
|-----|
| 1. По производителю.                 |
| 2. По форм-фактору сервера.          |
| 3. По производителю и форм-фактору сервера. |
| 4. Вернуться в меню.                 |
|-----|
Сделайте выбор: 1
Файл успешно отсортирован
```

Рисунок 21 – окно с выбором сортировки

```
1  Производитель: BROCADE
2  Форм-фактор сервера: Blade
3  Сокет: SP3
4  Максимальное кол-во CPU: 8
5  Объем оперативной памяти: 28.000000
6
7  Производитель: BROWLER
8  Форм-фактор сервера: Lager
9  Сокет: NTX
10 Максимальное кол-во CPU: 3
11 Объем оперативной памяти: 16.000000
12
13 Производитель: DELL
14 Форм-фактор сервера: Tower
15 Сокет: LGA
16 Максимальное кол-во CPU: 6
17 Объем оперативной памяти: 24.000000
18
19 Производитель: HDINIX
20 Форм-фактор сервера: Steker
21 Сокет: HTG
22 Максимальное кол-во CPU: 4
23 Объем оперативной памяти: 12.000000
24
25 Производитель: HP
26 Форм-фактор сервера: Rack
27 Сокет: PGA
28 Максимальное кол-во CPU: 7
29 Объем оперативной памяти: 32.000000
```

Рисунок 22 – результат сортировки файла по производителю



Теперь отсортируем по форм-фактору. Результат сортировки файла на рисунке 23.

```
1  Производитель: BROCADE
2  Форм-фактор сервера: Blade
3  Сокет: SP3
4  Максимальное кол-во CPU: 8
5  Объем оперативной памяти: 28.000000
6
7  Производитель: BROWLER
8  Форм-фактор сервера: Lager
9  Сокет: NTX
10 Максимальное кол-во CPU: 3
11 Объем оперативной памяти: 16.000000
12
13 Производитель: HP
14 Форм-фактор сервера: Rack
15 Сокет: PGA
16 Максимальное кол-во CPU: 7
17 Объем оперативной памяти: 32.000000
18
19 Производитель: HDINIX
20 Форм-фактор сервера: Steker
21 Сокет: HTG
22 Максимальное кол-во CPU: 4
23 Объем оперативной памяти: 12.000000
24
25 Производитель: DELL
26 Форм-фактор сервера: Tower
27 Сокет: LGA
28 Максимальное кол-во CPU: 6
29 Объем оперативной памяти: 24.000000
30
31 |
```

Рисунок 23 – результат сортировки файла по форм-фактору

Далее выберем пункт «5», тем самым завершив программу (рисунок 24).

```
Сделайте выбор: 5
Завершение работы...

Process finished with exit code 0
```

Рисунок 24 – результат завершения работы

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана программа для реализации файловой базы данных, которая успешно решает поставленные задачи. Программа представляет собой простую систему хранения данных с минимальной долей избыточности, что соответствует основным принципам современных баз данных. В рамках проекта была выбрана структура данных для хранения записей, определен формат файла и реализован удобный интерфейс для взаимодействия пользователя с системой.

Программа поддерживает создание записей в базе данных, что является важной функциональной возможностью. Кроме того, в ходе работы была внедрена сортировка записей по названию, что значительно упрощает навигацию и поиск данных в базе. Таким образом была создана удобная и быстрая система для хранения данных.

Ссылка на гитхаб - [https://github.com/KraSeq/course\\_work\\_C](https://github.com/KraSeq/course_work_C)

## СПИСОК ЛИТЕРАТУРЫ

1. Добрый, добрый C/C++ с Сергеем Балакиревым: электронный ресурс / <https://stepik.org/course/193691/syllabus>
2. Практикум по Си: электронный ресурс / <https://sites.google.com/view/course-of-study1-c>
3. Руководство по языку программирования Си: электронный ресурс / <https://metanit.com/c/tutorial/>
4. Основы Си: с нуля изучаем язык программирования: электронный ресурс / <https://skillbox.ru/media/code/nachinaem-izuchat-yazyk-programmirovaniya-s-cs50-na-russkom-lektsiya-11/>
5. C | Введение: электронный ресурс / <https://metanit.com/c/tutorial/1.1.php>
6. Введение в Си. Послание из прошлого столетия: электронный ресурс / <https://habr.com/ru/articles/464075/>
7. Полный справочник по Си: справочник / Шилдт (Herbert Schildt)
8. Солдатенко И.С. Основы программирования на языке Си: учебное пособие / Тверской государственный университет, 2017 – 159 с.
9. К. Дж. Дейт «Введение в системы баз данных» – 7-е изд. – М. Вильямс, 2001. – 43с.
10. Керниган Б. Язык программирования Си [Текст] / Пер. с англ., 3-е изд., испр. / Керниган Б., Ритчи Д. – СПб.: «Невский Диалект», 2001. – 352 с.

## ПРИЛОЖЕНИЕ

### Листинг программы

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <windows.h>

struct server_platforms {
    char manufacturer[20];
    char form_factor[20];
    char socket[20];
    int max_cpu;
    double max_ram;
};

int read_file()
{
    FILE* fp = fopen("coursework.txt", "r");
    if (fp == NULL) {
        return 1;
    }

    fclose(fp);

    return 0;
}

int fill_platforms(struct server_platforms* platforms, int k)
{
    for (int i = 0; i < k; i++) {
        printf("Производитель: ");
        if (scanf("%19s", platforms[i].manufacturer) != 1)
            return 1;

        printf("Форм-фактор сервера: ");
        if (scanf("%19s", platforms[i].form_factor) != 1)
            return 1;

        printf("Сокет: ");
        if (scanf("%19s", platforms[i].socket) != 1)
            return 1;

        printf("Максимальное кол-во CPU: ");
```

```

        if (scanf("%d", &platforms[i].max_cpu) != 1)
            return 1;

        printf("Объем оперативной памяти: ");
        if (scanf("%lf", &platforms[i].max_ram) != 1)
            return 1;

        printf("\n");
    }

    return 0;
}

int write_file(struct server_platforms* platforms, int k)
{
    FILE* fp = fopen("coursework.txt", "w");
    if (fp == NULL) {
        puts("Error func write_file");
        return 1;
    }

    for (int i = 0; i < k; i++) {
        if (fprintf(fp, "Производитель: %s\n", platforms[i].manufacturer) < 0 ||
            fprintf(fp, "Форм-фактор сервера: %s\n", platforms[i].form_factor) < 0 ||
            fprintf(fp, "Сокет: %s\n", platforms[i].socket) < 0 ||
            fprintf(fp, "Максимальное кол-во CPU: %d\n", platforms[i].max_cpu) < 0 ||
            fprintf(fp, "Объем оперативной памяти: %f\n\n", platforms[i].max_ram) < 0) {
            fclose(fp);

            return 1;
        }
    }

    fclose(fp);

    return 0;
}

int* do_search(struct server_platforms* platforms, int k, char* socket, int max_CPU)
{
    int* indices = malloc(k * sizeof(int));
    for (int i = 0; i < k; i++) {
        indices[i] = -1;
    }

    if (socket[0] != '\0' && max_CPU != -1) {

```

```

    for (int i = 0; i < k; i++) {
        if (strcmp(platforms[i].socket, socket) == 0 && platforms[i].max_cpu == max_CPU)
            indices[i] = i;
    }
}
else if (socket[0] != '\0' && max_CPU == -1) {
    for (int i = 0; i < k; i++) {
        if (strcmp(platforms[i].socket, socket) == 0)
            indices[i] = i;
    }
}
else {
    for (int i = 0; i < k; i++) {
        if (platforms[i].max_cpu == max_CPU)
            indices[i] = i;
    }
}

return indices;
}

void print_platform(struct server_platforms platform)
{
    printf("Производитель: %s\n", platform.manufacturer);
    printf("Форм-фактор сервера: %s\n", platform.form_factor);
    printf("Сокет: %s\n", platform.socket);
    printf("Максимальное кол-во CPU: %d\n", platform.max_cpu);
    printf("Объем оперативной памяти: %f\n", platform.max_ram);
    printf("\n");
}

int compare_manufacturer(const void* var_1, const void* var_2)
{
    const struct server_platforms* platform_1 = (const struct server_platforms*)var_1;
    const struct server_platforms* platform_2 = (const struct server_platforms*)var_2;

    return strcmp(platform_1->manufacturer, platform_2->manufacturer);
}

int compare_form_factor(const void* var_1, const void* var_2)
{
    const struct server_platforms* platform_1 = (const struct server_platforms*)var_1;
    const struct server_platforms* platform_2 = (const struct server_platforms*)var_2;

    return strcmp(platform_1->form_factor, platform_2->form_factor);
}

```

```

void sort_platforms(struct server_platforms* platforms, int k, int choice)
{
    if (choice == 1)
        qsort(platforms, k, sizeof(struct server_platforms), compare_manufacturer);
    else if (choice == 2) {
        qsort(platforms, k, sizeof(struct server_platforms), compare_form_factor);
    }
    else if (choice == 3) {
        qsort(platforms, k, sizeof(struct server_platforms), compare_manufacturer);
        qsort(platforms, k, sizeof(struct server_platforms), compare_form_factor);
    }
    else
        puts("Неверный выбор");
}

void add_platform(struct server_platforms* platforms, int i)
{
    printf("Производитель: ");
    if (scanf("%19s", platforms[i].manufacturer) != 1)
        puts("Ошибка формата");

    printf("Форм-фактор сервера: ");
    if (scanf("%19s", platforms[i].form_factor) != 1)
        puts("Ошибка формата");

    printf("Сокет: ");
    if (scanf("%19s", platforms[i].socket) != 1)
        puts("Ошибка формата");

    printf("Максимальное кол-во CPU: ");
    if (scanf("%d", &platforms[i].max_cpu) != 1)
        puts("Ошибка формата");

    printf("Объем оперативной памяти: ");
    if (scanf("%lf", &platforms[i].max_ram) != 1)
        puts("Ошибка формата");

    printf("\n");
}

void del_platform(struct server_platforms* platforms, int k, int i)
{
    struct server_platforms* new_platforms = malloc(k * sizeof(struct server_platforms));

    for (int j = 0; j < k; ++j) {

```



```

        if (j < i)
            new_platforms[j] = platforms[j];
        else if (j > i)
            new_platforms[j - 1] = platforms[j];
    }

    platforms = realloc(platforms, k * sizeof(struct server_platforms));
    for (int y = 0; y < k; ++y)
        platforms[y] = new_platforms[y];
}

int main(void)
{
    SetConsoleOutputCP(CP_UTF8);

    int k = 0;
    printf("Введите количество серверных платформ: ");
    if (scanf("%d", &k) != 1) {
        puts("Error");
        return 1;
    }

    struct server_platforms* platforms = malloc(k * sizeof(struct server_platforms));

    int error; // Для проверки на создание новой записи
    char socket[20] = {"\0"}; // Для выполнения функции поиска
    int max_CPU = -1; // Для выполнения функции поиска
    int* indices = malloc(k * sizeof(int));
    int count;
    struct server_platforms* new_platforms;

    int choice = 0;
    int in_choice; // Для вложенных выборов

    while (choice != 5) {
        printf("_____.\n");
        printf("|           Меню           |\n");
        printf("|_____|\n");
        printf("| 1. Создание новой записи.   |\n");
        printf("| 2. Поиск записи.           |\n");
        printf("| 3. Запись и чтение данных.  |\n");
        printf("| 4. Сортировка записей.      |\n");
        printf("| 5. Завершить работу.       |\n");
        printf("_____.\n");

        printf("Сделайте выбор: ");
    }

```

```

if (scanf("%d", &choice) != 1) {
    puts("Error");
    return 1;
}

switch (choice) {
    case 1:
        error = fill_platforms(platforms, k);
        if (error) {
            puts("Ошибка формата");
            return 1;
        }
        error = write_file(platforms, k);

        if (error)
            puts("Ошибка создания записи\n");
        else
            puts("Создание записи прошло успешно\n");

        break;

    case 2:
        printf("_____.\n");
        printf("|          Поиск записи          |\n");
        printf("|_____|\n");
        printf("| 1. По сокету.                  |\n");
        printf("| 2. По максимальному кол-ву CPU.    |\n");
        printf("| 3. По сокету и максимальному кол-ву CPU. |\n");
        printf("| 4. Вернуться в меню.              |\n");
        printf("_____.\n");

        printf("Сделайте выбор: ");
        if (scanf("%d", &in_choice) != 1) {
            puts("Некорректный формат");
            return 1;
        }

        switch (in_choice) {
            case 1:
                printf("Введите название сокета: ");
                getchar();
                fgets(socket, sizeof(socket), stdin);
                socket[strcspn(socket, "\n")] = '\0';
                max_CPU = -1;
                break;
            case 2:

```

```

    printf("Введите максимальное кол-во CPU: ");
    scanf("%d", &max_CPU);
    socket[0] = '\0';
    break;
case 3:
    printf("Введите название сокета: ");
    getchar();
    fgets(socket, sizeof(socket), stdin);
    socket[strcspn(socket, "\n")] = '\0';
    printf("Введите максимальное кол-во CPU: ");
    scanf("%d", &max_CPU);
    break;
default:
    break;
}
indices = do_search(platforms, k, socket, max_CPU);
for (int i = 0; i < k; ++i) {
    if (indices[i] != -1)
        print_platform(platforms[i]);
}

break;

```

```

case 3:
    printf("_____.\n");
    printf("|   Запись и чтение данных   |\n");
    printf("|_____|\n");
    printf("| 1. Добавить запись.      |\n");
    printf("| 2. Изменить записи.      |\n");
    printf("| 3. Удалить записи.       |\n");
    printf("| 4. Чтение файла.         |\n");
    printf("| 5. Вернуться в меню.     |\n");
    printf("_____.\n");

```

```

printf("Сделайте выбор: ");
if (scanf("%d", &in_choice) != 1) {
    puts("Некорректный формат");
    return 1;
}

```

```

switch (in_choice) {
case 1:
    error = read_file();
    if (error == 1) {
        puts("Ошибка чтения файла");
        break;
    }
}

```

```

    }

    printf("Введите кол-во добавления записей: ");
    if (scanf("%d", &count) != 1) {
        puts("Некорректный формат данных");
        break;
    }

    k += count;

    new_platforms = realloc(platforms, k * sizeof(struct server_platforms));
    for (int i = 0; i < k - count; ++i)
        new_platforms[i] = platforms[i];

    for (int i = k - count; i < k; ++i)
        add_platform(new_platforms, i);

    platforms = malloc((k + count) * sizeof(struct server_platforms));
    for (int i = 0; i < k + count; ++i)
        platforms[i] = new_platforms[i];

    error = write_file(platforms, k);
    if (error)
        puts("Ошибка добавления новых записей");
    else
        puts("Записи успешно добавлены");

    break;

case 2:
    printf("Введите номер серверной платформы, которую вы хотите изменить: ");
    if (scanf("%d", &in_choice) != 1) {
        puts("Некорректный формат данных");
        break;
    }

    add_platform(platforms, in_choice-1);

    error = write_file(platforms, k);
    if (error)
        puts("Ошибка изменения платформы");
    else
        puts("Данные платформы изменены");

    break;

```

```

case 3:
    printf("Введите номер серверной платформы, которую вы хотите удалить: ");
    if (scanf("%d", &in_choice) != 1) {
        puts("Некорректный формат данных");
        break;
    }

    if (in_choice < 1 || in_choice > k) {
        printf("Платформы под таким номером не существует");
        break;
    }

    k -= 1;
    in_choice -= 1;

    del_platform(platforms, k, in_choice);

    error = write_file(platforms, k);
    if (error)
        puts("Ошибка удаления платформы");
    else
        puts("Платформа успешно удалена");

    break;

case 4:
    error = read_file();
    if (error) {
        puts("Ошибка чтения файла");
        break;
    }

    for (int i = 0; i < k; ++i)
        print_platform(platforms[i]);
    puts("Файл успешно прочитан");

    break;

case 5:
    break;
default:
    puts("Неверный выбор");
}
break;

```

case 4:

```

printf("_____.\n");
printf("|          Сортировка записей          |\n");
printf("|_____|\n");
printf("| 1. По производителю.                  |\n");
printf("| 2. По форм-фактору сервера.          |\n");
printf("| 3. По производителю и форм-фактору сервера. |\n");
printf("| 4. Вернуться в меню.                  |\n");
printf("_____.\n");

printf("Сделайте выбор: ");
if (scanf("%d", &in_choice) != 1) {
    puts("Некорректный формат");
    break;
}

if (in_choice != 4) {
    sort_platforms(platforms, k, in_choice);
    write_file(platforms, k);
}
puts("Файл успешно отсортирован");

break;

case 5:
    puts("Завершение работы...");
    break;
default:
    puts("Неверный выбор");
}
}

free(platforms);
free(indices);

// Очистка файла
FILE* fp = fopen("coursework.txt", "w");
fclose(fp);

return 0;
}

```