

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
«Жесткий диск».

Расчетно-пояснительная записка

Разработал студент А.В. Гречишникова
Подпись, дата 15.01.25 Инициалы, фамилия

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

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

Защищена 15.01.25 Оценка отлично
дата

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВА-
ТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖ-
СКИЙ 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 «Жесткий диск».

Студентка группы БТИИ-241 Гречишникова Алёна Васильевна
Фамилия, имя, отчество

База данных «Жесткий диск», Признак: производитель, объем, скорость вращения, интерфейс, объем буфера (кеша), скорость обмена данных
Вариант сортировки: производитель и объем

Технические условия Windows 10, Microsoft Visual Studio 2022, язык программирования C

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

Сроки выполнения этапов анализ и постановка задачи (10.9-5.10.24); разработка пошаговой детализации программы (6.10 -11.11.24); реализация программы (11.11-5.12.24); тестирование программы (6.12-11.12.24); оформление пояснительной записки (11.12-14.12.24).


Срок защиты курсового проекта

Руководитель


Подпись, дата

Н. В. Акамсина
Инициалы, фамилия

Задание принял студент


Подпись, дата

А.В. Гречишникова
Инициалы, фамилия

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 ПОСТАНОВКА ЗАДАЧИ.....	5
2 РЕАЛИЗАЦИЯ ПРОГРАММЫ.....	8
3 ТЕСТИРОВАНИЕ ПРОГРАММЫ.....	15
ЗАКЛЮЧЕНИЕ	20
СПИСОК ЛИТЕРАТУРЫ	21
ПРИЛОЖЕНИЕ А	22
ПРИЛОЖЕНИЕ В	23

ВВЕДЕНИЕ

Файловая база – это файл, в котором хранятся упорядоченные записи данных, описывающих заданную предметную область. Для работы с ними необходимо реализовать программу, позволяющую создавать новые записи, изменять существующие, выполнять поиск по заданным критериям.

Целью данной работы является разработка программы для работы с файловой базой данных «Жесткий диск».

Для достижения поставленной цели необходимо решить следующие задачи:

1. Обосновать выбор структуры данных для хранения отдельных записей в файле, а также формат файла.
2. Реализовать простой и понятный интерфейс для взаимодействия пользователя с программой, который будет работать пока пользователь не захочет выйти из программы.
3. Привести блок-схемы используемых алгоритмов для понимания работы данной программы.
4. Обеспечить выполнение программой функции создания записи.
5. Обеспечить выполнение программой функции добавления произвольного количества записей в текущую базу данных.
6. Обеспечить возможность выполнения сортировки текущей базы данных по различным полям в порядке возрастания и убывания.
7. Обезопасить выполнение программы от ошибок при вводе данных пользователем.

1 ПОСТАНОВКА ЗАДАЧИ

Задача курсового проекта – написание программы для работы с файловой базой данных «Жесткий диск», содержащей записи из полей различного типа данных. Данная программа должна предусматривать выполнение хранения, поиска, добавления и сортировки данных.

Записи в базе данных представляют собой структуру «Жесткий диск», содержащую данные о производителе, объеме, скорости вращения, интерфейсе, объеме буфера (кеша), скорости обмена данных:

1. Производитель – данные этого поля будут храниться в массиве `przv[10]` типа `char`. Массив может содержать строку размером до 9 символов включительно, что является достаточным для многих наименований.
2. Объем – данные этого поля будут храниться в переменной `space` типа `float`, что позволит отображать в записи более точный объем.
3. Скорость вращения – переменная `sv` типа `int`; данный тип наиболее предпочтителен, поскольку скорость часто представляется целым значением оборотов в минуту.
4. Интерфейс – для хранения данных используется массив `intrfs[10]` типа `char`. Поскольку большинство названий для интерфейса являются аббревиатурами (USB, ATA, SATA, SCSI), то 9 символов будет достаточно.
5. Объем буфера – данные этого поля хранятся в переменной `sp_b` типа `float`. Данный тип позволит отображать объем более точно.
6. Скорость обмена данных – для хранения будет использоваться переменная `s_data` типа `int` (количество оборотов в минуту).

Для хранения и использования вышеперечисленных полей, имеющих различные типы, удобно объединить их в структуру данных. В файле `massiv1.txt` будет располагаться исходная база данных, в которой каждая строка представляет собой совокупность перечисленных полей для каждого жесткого диска. Пример записи в базе данных: «Производитель: Seagate

Объем: 10 Скорость вращения: 7200 Интерфейс: USB Объем буфера: 8
Скорость обмена данных: 700».

Текстовый файл massiv1.txt содержит 10 исходных записей, что является достаточным для выполнения сортировки и поиска, а также проверки корректности работы созданной программы:

Производитель: Seagate Объем: 10,000000 Скорость вращения: 7200
Интерфейс: USB Объем буфера: 8,000000 Скорость обмена данных: 700

Производитель: Toshiba Объем: 7,000000 Скорость вращения: 4200
Интерфейс: USB Объем буфера: 16,000000 Скорость обмена данных: 300

Производитель: Sony Объем: 3,000000 Скорость вращения: 5400
Интерфейс: ATA Объем буфера: 2,000000 Скорость обмена данных: 200

Производитель: Samsung Объем: 2,000000 Скорость вращения: 4200
Интерфейс: SAS Объем буфера: 32,000000 Скорость обмена данных: 300

Производитель: Adata Объем: 5,000000 Скорость вращения: 4200
Интерфейс: IDE Объем буфера: 2,000000 Скорость обмена данных: 600

Производитель: WD Объем: 4,000000 Скорость вращения: 5400
Интерфейс: USB Объем буфера: 8,0 Скорость обмена данных: 200

Производитель: Hitachi Объем: 1,000000 Скорость вращения: 7200
Интерфейс: eSATA Объем буфера: 16,000000 Скорость обмена данных: 600

Производитель: Quantum Объем: 1,000000 Скорость вращения: 5400
Интерфейс: PATA Объем буфера: 8,000000 Скорость обмена данных: 300

Производитель: Intel Объем: 3,000000 Скорость вращения: 4200
Интерфейс: SCSI Объем буфера: 16,000000 Скорость обмена данных: 500

Производитель: Lenovo Объем: 4,000000 Скорость вращения: 4200
Интерфейс: USB Объем буфера: 2,000000 Скорость обмена данных: 300

Разработанная программа должна выполнять следующие операции с базой данных:

1. Создание исходной базы данных (ввод данных пользователем или использование готового массива);
2. Запись базы данных в текстовый файл

3. Открытие файла, содержащего готовую базу данных;
4. Редактирование записи;
5. Добавление записей в готовую базу;
6. Поиск записи пользователем (должен осуществляться по полям «Объем» и/или «Интерфейс»);
7. Сортировка базы данных (выполняется в любом порядке по одному или нескольким критериям – производитель и/или объем;
8. Печать базы данных.

Для выполнения вышеописанных операций с базой данных необходимо реализовать взаимодействие с пользователем; его алгоритм описан блок-схемой (рисунок 1).

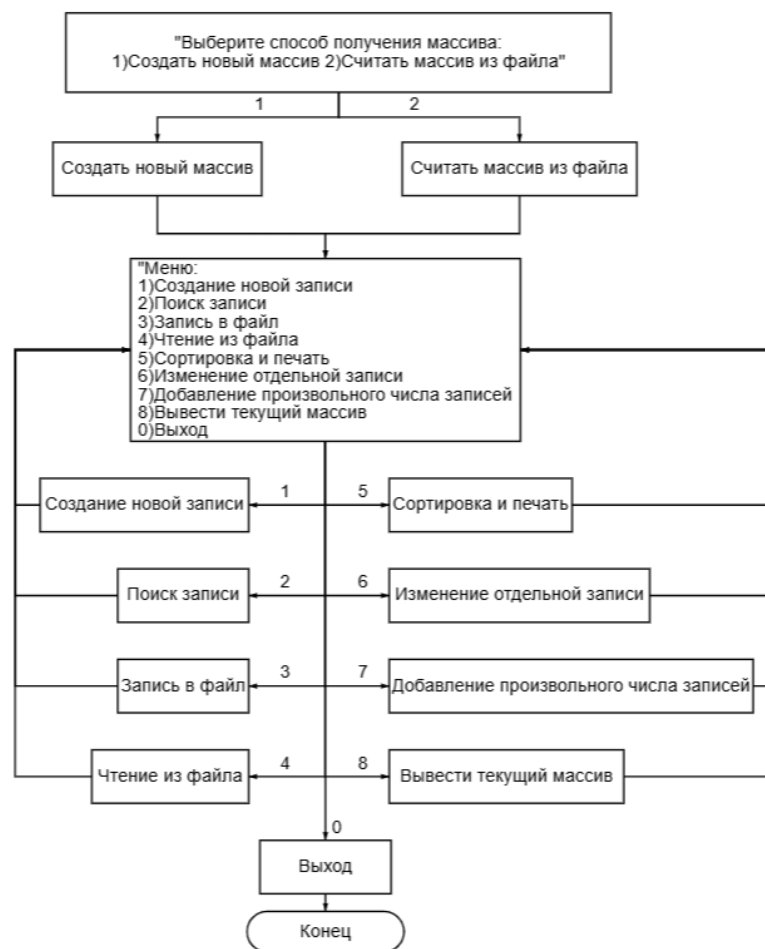


Рисунок 1 – Блок-схема взаимодействия с пользователем

2 РЕАЛИЗАЦИЯ ПРОГРАММЫ

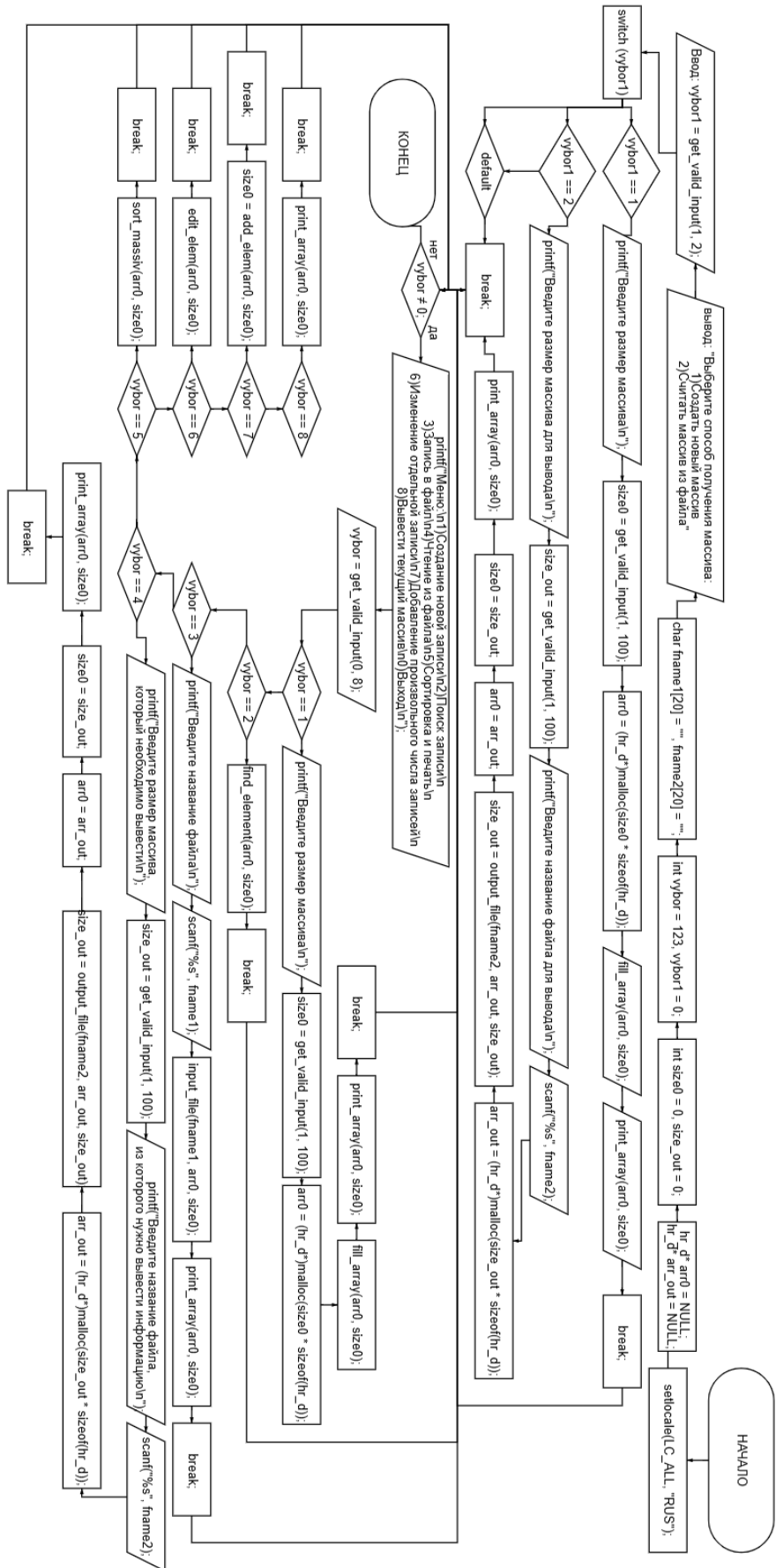


Рисунок 2 – Блок-схема функции main

На рисунке 2 приведена блок-схема функции main, которая включает общую информацию о программе и вызов функций.

Функция `int fill_array(hr_d* arr, int size)` предназначена для заполнения массива значениями, которые вводит пользователь с клавиатуры. Функция принимает указатель на данный массив структурного типа и целочисленный размер массива, возвращает размер массива. Блок-схема функции `fill_array` приведена на рисунке 3.

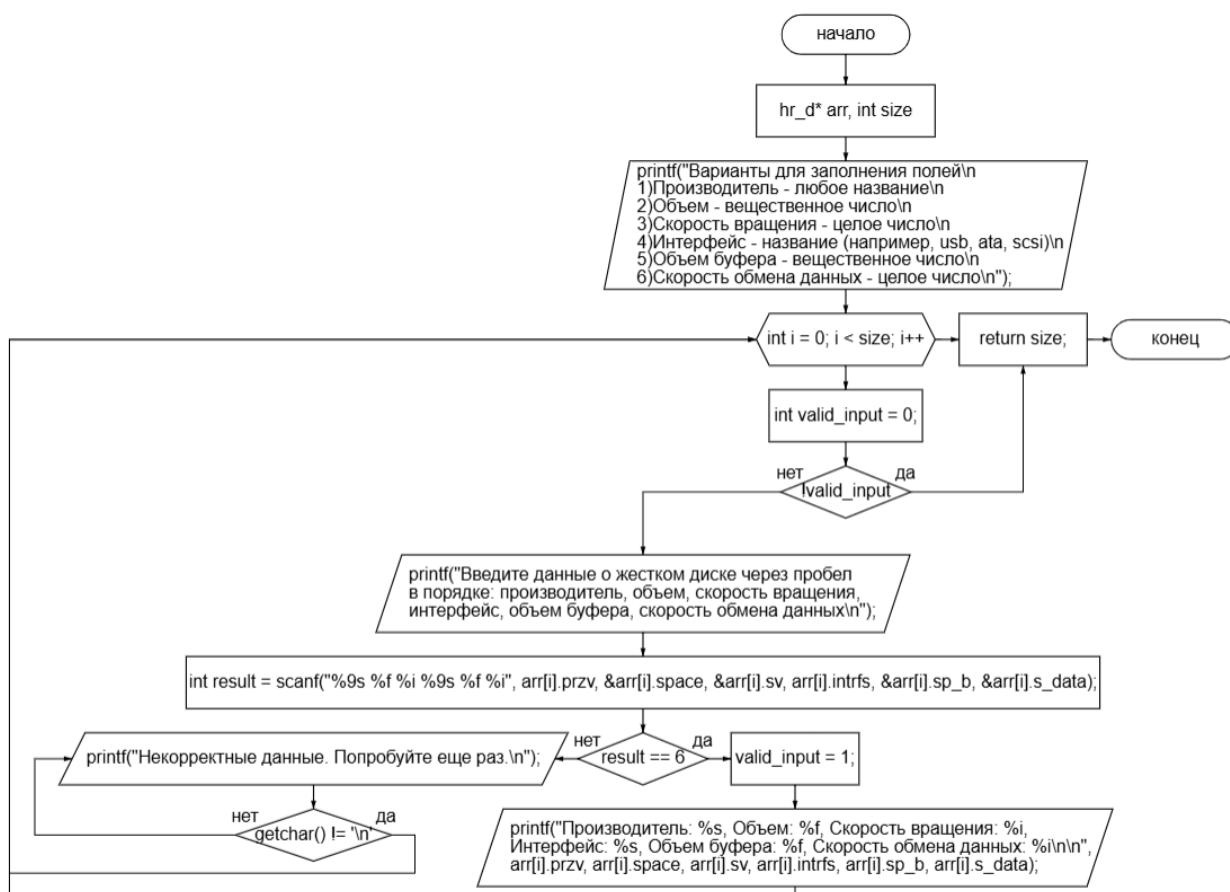


Рисунок 3 – Блок-схема функции `int fill_array(hr_d* arr, int size)`

Функция `int print_array(hr_d* arr, int size)` предназначена для вывода базы данных на экран. Функция принимает указатель структурного типа на массив и размер массива (целое число). Возвращает размер массива. Функция приведена на рисунке 4.

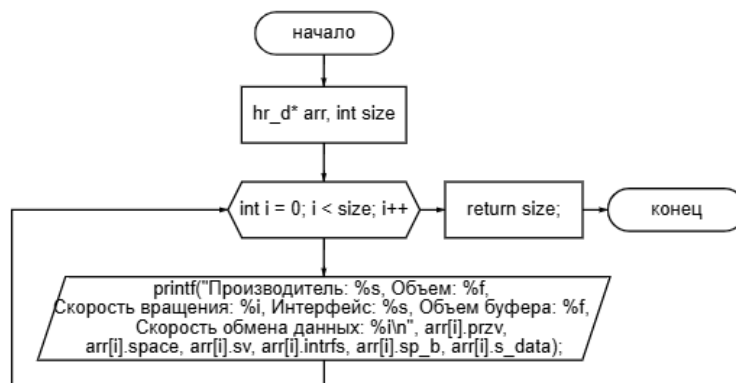


Рисунок 4 – Блок-схема функции int print_array(hr_d* arr, int size)

Функция int input_file(char* fname, hr_d* arr, int size) предназначена для записи в файл созданной базы данных. Функция принимает указатель на название файла для записи, указатель структурного типа на массив и размер массива, возвращает размер. Блок-схема функции приведена на рисунке 5.

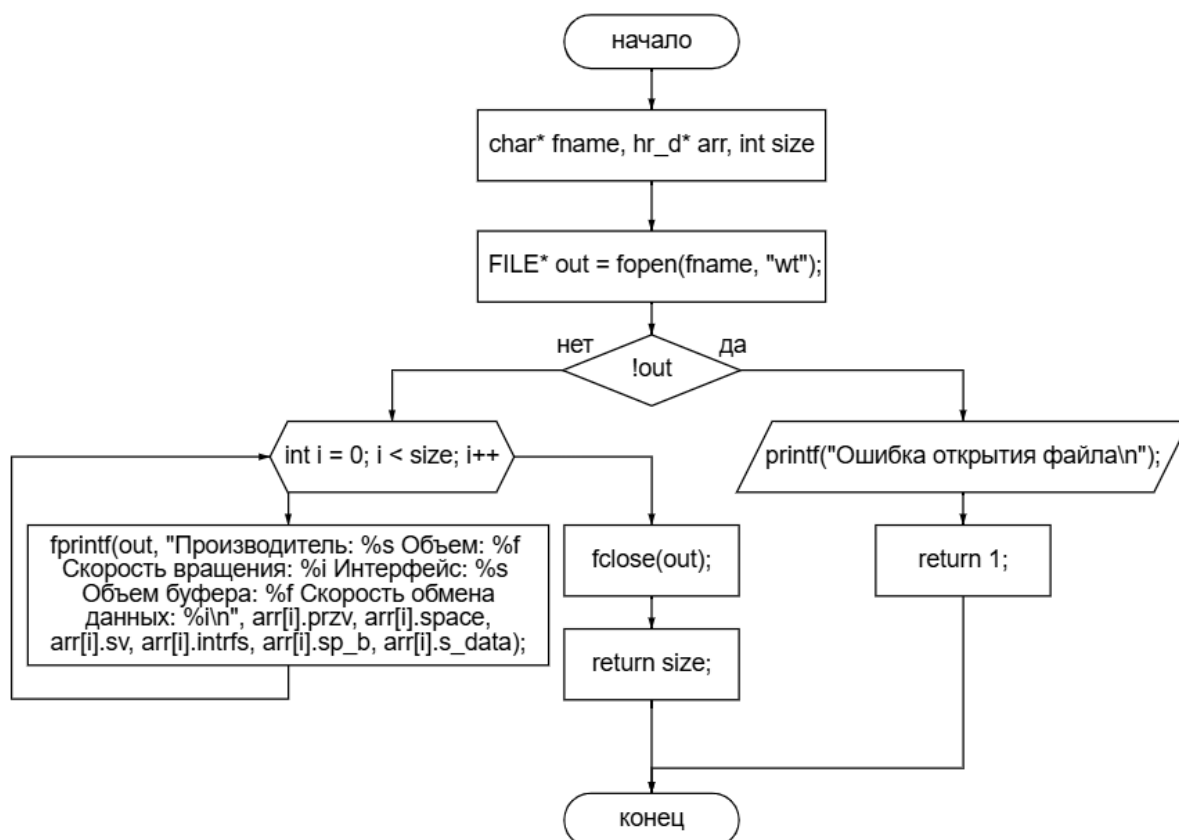


Рисунок 5 – Блок-схема функции int input_file(char* fname, hr_d* arr, int size)

Функция `int output_file(char* fname, hr_d* arr, int size)` предназначена для чтения базы данных из файла. Принимает параметры: указатель на название файла, указатель структурного типа на массив, в который записываются данные, целочисленный размер данного массива. Возвращает количество считанных структур. Блок-схема функции представлена на рисунке 6.

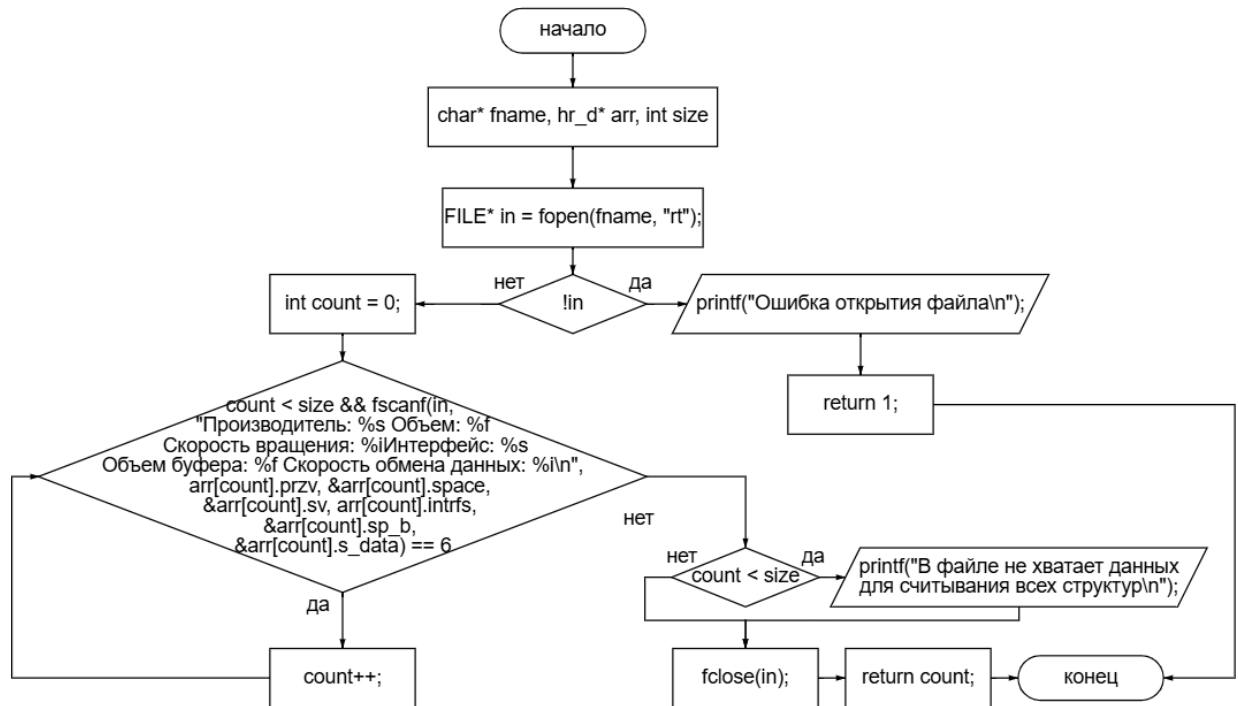


Рисунок 6 – Блок-схема функции `int output_file(char* fname, hr_d* arr, int size)`

Функция `int find_element(hr_d* arr, int size)` предназначена для поиска записи в базе данных по одному или двум полям. Функция принимает указатель структурного типа на исходный массив и его размер (целое число). Возвращает размер. Блок-схема функции представлена на рисунке 7.

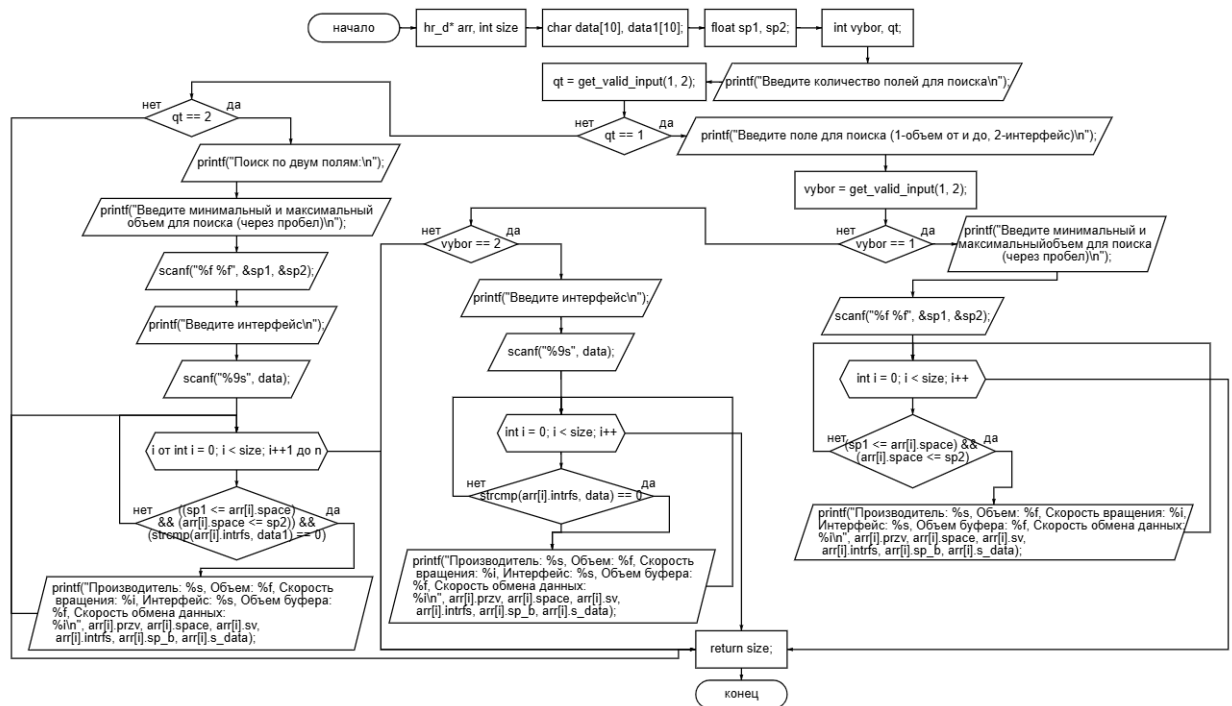


Рисунок 7 – Блок-схема функции int find_element(hr_d* arr, int size)

Функция int sort_massiv(hr_d* arr, int size) предназначена для сортировки базы данных по одному из полей. Функция принимает указатель структурного типа на исходный массив и размер массива, возвращает размер массива. Блок-схема функции представлена на рисунке 8.

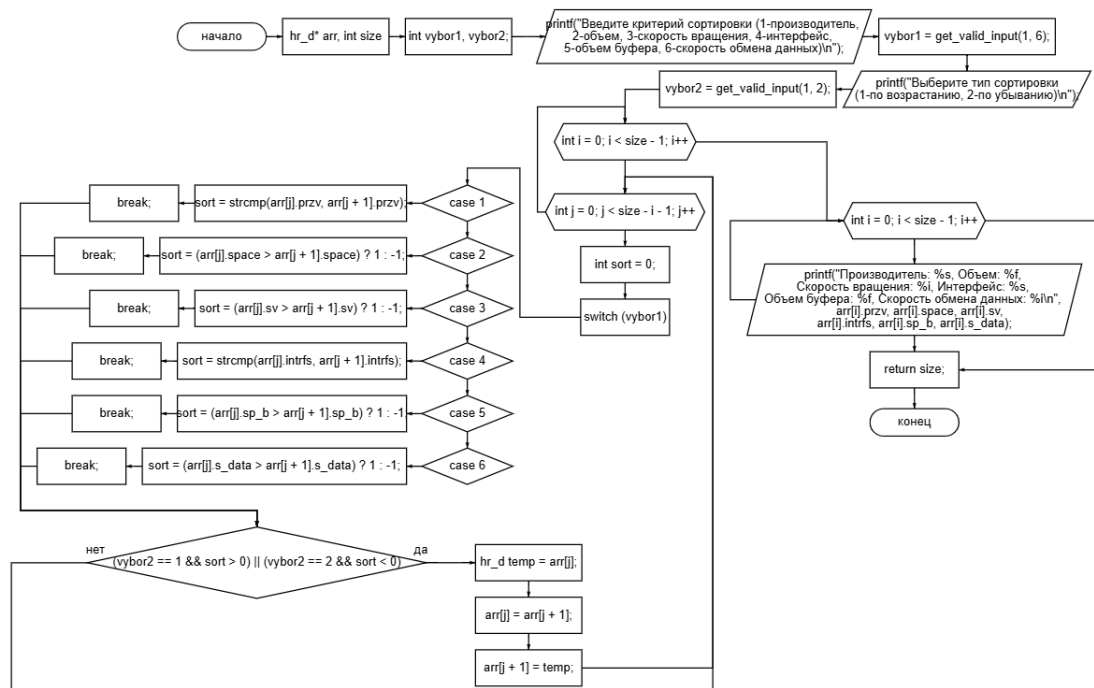


Рисунок 8 – Блок-схема функции int sort_massiv(hr_d* arr, int size)

Функция `int add_elem(hr_d* arr, int size)` предназначена для добавления в базу данных произвольного количества элементов. Принимает указатель структурного типа на исходный массив и целочисленный размер массива; возвращает новый размер массива. Блок-схема функции представлена на рисунке 9.

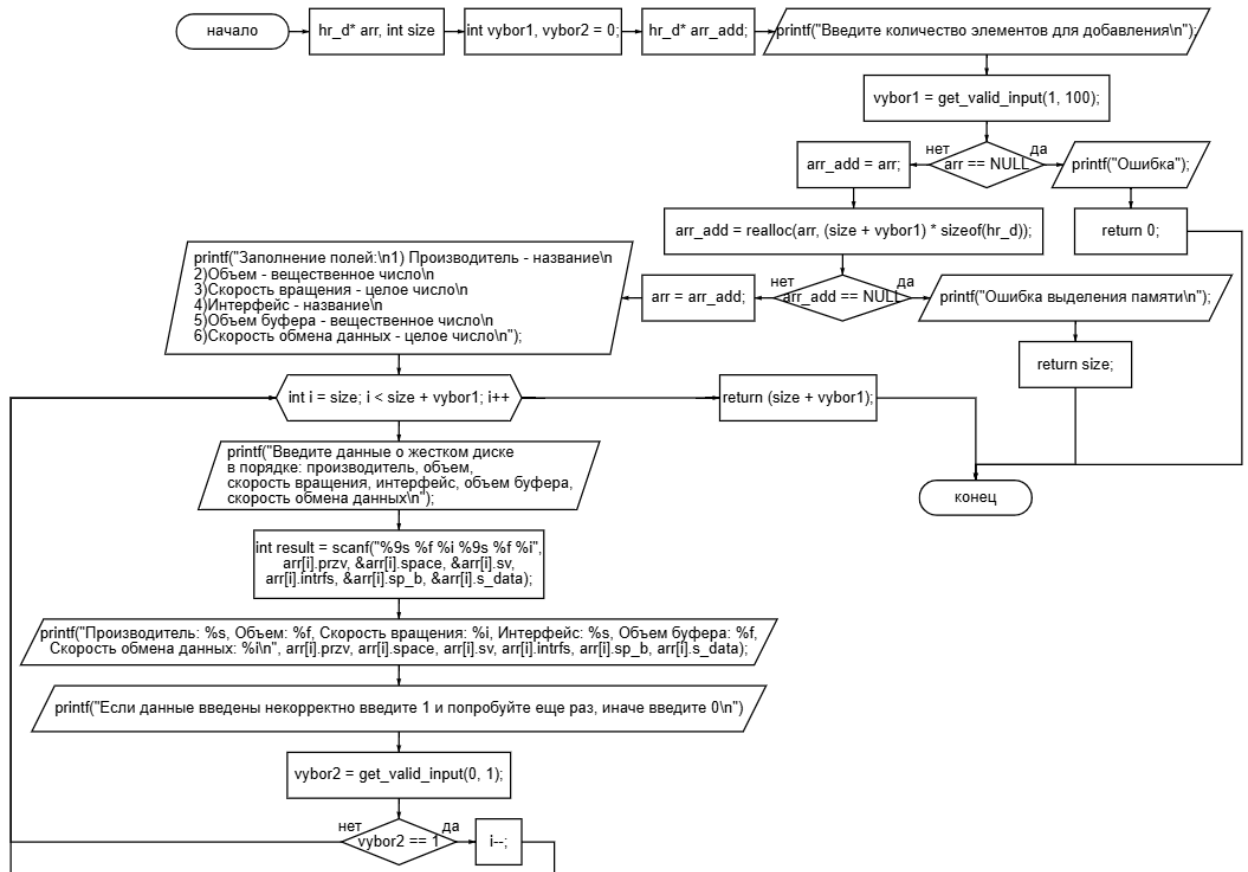


Рисунок 9 – Блок-схема функции `int add_elem(hr_d* arr, int size)`

Функция `int edit_elem(hr_d* arr, int size)` предназначена для изменения записей в базе данных. Принимает указатель структурного типа на массив и размер массива, возвращает размер. Блок-схема функции представлена на рисунке 10.

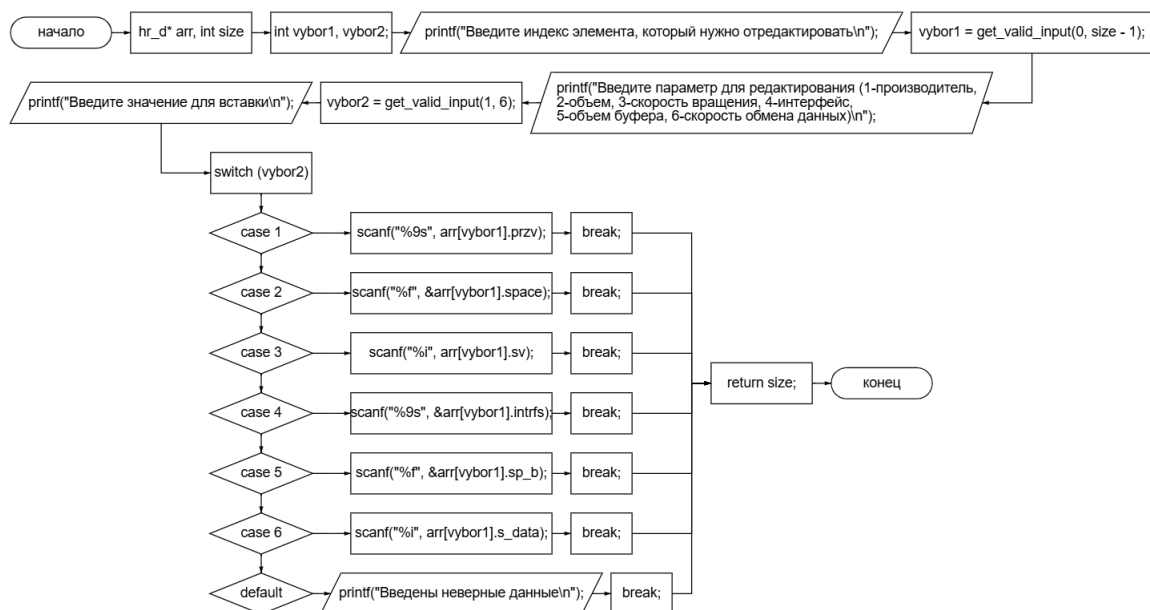


Рисунок 10 – Блок-схема функции int edit_elem(hr_d* arr, int size)

Все собственные функции, используемые для работы с базой данных, приведены в таблице 1 с описанием:

Функции	Назначение
int fill_array(hr_d* arr, int size)	Заполнение массива данными, введенными пользователем
int print_array(hr_d* arr, int size)	Вывод на экран всех записей
int input_file(char* fname, hr_d* arr, int size)	Запись текущего массива в файл с именем, введенным пользователем
int output_file(char* fname, hr_d* arr, int size)	Считывание данных из файла, указанного пользователем
int find_element(hr_d* arr, int size)	Поиск записи в массиве по одному или двум критериям
int sort_massiv(hr_d* arr, int size)	Сортировка текущего массива по указанному критерию
int add_elem(hr_d* arr, int size)	Добавление в текущий массив произвольного числа записей
int edit_elem(hr_d* arr, int size)	Изменение записи на введенное пользователем значение

3 ТЕСТИРОВАНИЕ ПРОГРАММЫ

После запуска на экран будет выведено меню с выбором источника изначальной базы данных (рисунок 11).

```
Выберите способ получения массива:  
1)Создать новый массив  
2)Считать массив из файла
```

Рисунок 11 – Получение массива

При выборе создания нового массива программа запросит размер массива, после чего выведет памятку о заполнении полей в базе данных (рисунок 12). При вводе некорректных данных программа выведет ошибку (рисунок 13).

```
Выберите способ получения массива:  
1)Создать новый массив  
2)Считать массив из файла  
1  
Введите размер массива  
5  
Варианты для заполнения полей  
1)Производитель - любое название  
2)Объем - вещественное число  
3)Скорость вращения - целое число  
4)Интерфейс - название (например, usb, ata, scsi)  
5)Объем буфера - вещественное число  
6)Скорость обмена данных - целое число  
Введите данные о жестком диске через пробел в порядке: производитель, объем, скорость вращения, интерфейс, объем буфера, скорость обмена данных  
Panasonic 3,2 400 ata 2,4 500  
Производитель: Panasonic, Объем: 3,200000, Скорость вращения: 400, Интерфейс: ata, Объем буфера: 2,400000, Скорость обмена данных: 500  
Введите данные о жестком диске через пробел в порядке: производитель, объем, скорость вращения, интерфейс, объем буфера, скорость обмена данных
```

Рисунок 12 – Заполнение нового массива данными

```
Введите данные о жестком диске через пробел в порядке: производитель, объем, скорость вращения, интерфейс, объем буфера, скорость обмена данных  
ф ф ф ф ф  
Некорректные данные. Попробуйте еще раз.  
Введите данные о жестком диске через пробел в порядке: производитель, объем, скорость вращения, интерфейс, объем буфера, скорость обмена данных
```

Рисунок 13 – Предупреждение о неправильном вводе данных

При выборе считывания массива из файла потребуется ввести количество записей и название файла (рисунок 14). Если требуемое количество записей меньше, чем количество записей в файле, будет считано нужное количество. Если требуемое количество превышает имеющееся, будет выведено предупреждение (рисунок 15).

```

Выберите способ получения массива:
1)Создать новый массив
2)Считать массив из файла
2
Введите размер массива для вывода
5
Введите название файла для вывода
massiv1.txt
Производитель: Seagate, Объем: 10,000000, Скорость вращения: 7200, Интерфейс: USB, Объем буфера: 8,000000, Скорость обмена данных: 700
Производитель: Toshiba, Объем: 7,000000, Скорость вращения: 4200, Интерфейс: USB, Объем буфера: 16,000000, Скорость обмена данных: 300
Производитель: Sony, Объем: 3,000000, Скорость вращения: 5400, Интерфейс: ATA, Объем буфера: 2,000000, Скорость обмена данных: 200
Производитель: Samsung, Объем: 2,000000, Скорость вращения: 4200, Интерфейс: SAS, Объем буфера: 32,000000, Скорость обмена данных: 300
Производитель: Adata, Объем: 5,000000, Скорость вращения: 4200, Интерфейс: IDE, Объем буфера: 2,000000, Скорость обмена данных: 600

```

Рисунок 14 – Считывание данных из файла

```

Введите размер массива для вывода
20
Введите название файла для вывода
massiv.txt
В файле не хватает данных для считывания всех структур

```

Рисунок 15 – Предупреждение о нехватке данных в файле

Содержание файла приведем на рисунке 16.

```

Производитель: Seagate Объем: 10,000000 Скорость вращения: 7200 Интерфейс: USB Объем буфера: 8,000000 Скорость обмена данных: 700
Производитель: Toshiba Объем: 7,000000 Скорость вращения: 4200 Интерфейс: USB Объем буфера: 16,000000 Скорость обмена данных: 300
Производитель: Sony Объем: 3,000000 Скорость вращения: 5400 Интерфейс: ATA Объем буфера: 2,000000 Скорость обмена данных: 200
Производитель: Samsung Объем: 2,000000 Скорость вращения: 4200 Интерфейс: SAS Объем буфера: 32,000000 Скорость обмена данных: 300
Производитель: Adata Объем: 5,000000 Скорость вращения: 4200 Интерфейс: IDE Объем буфера: 2,000000 Скорость обмена данных: 600
Производитель: WD Объем: 4,000000 Скорость вращения: 5400 Интерфейс: USB Объем буфера: 8,0 Скорость обмена данных: 200
Производитель: Hitachi Объем: 1,000000 Скорость вращения: 7200 Интерфейс: eSATA Объем буфера: 16,000000 Скорость обмена данных: 600
Производитель: Quantum Объем: 1,000000 Скорость вращения: 5400 Интерфейс: PATA Объем буфера: 8,000000 Скорость обмена данных: 300
Производитель: Intel Объем: 3,000000 Скорость вращения: 4200 Интерфейс: SCSI Объем буфера: 16,000000 Скорость обмена данных: 500
Производитель: Lenovo Объем: 4,000000 Скорость вращения: 4200 Интерфейс: USB Объем буфера: 2,000000 Скорость обмена данных: 300

```

Рисунок 16 – Содержание файла massiv1.txt

После получения массива будет выведено меню действий с базой данных (рисунок 17).

```

Производитель: 1, Объем: 1,000000, Скорость вращения: 1, Интерфейс: 1, Объем буфера: 1,000000, Скорость обмена данных: 1
Производитель: 2, Объем: 2,000000, Скорость вращения: 2, Интерфейс: 2, Объем буфера: 2,000000, Скорость обмена данных: 2
Производитель: 3, Объем: 3,000000, Скорость вращения: 3, Интерфейс: 3, Объем буфера: 3,000000, Скорость обмена данных: 3
Производитель: 4, Объем: 4,000000, Скорость вращения: 4, Интерфейс: 4, Объем буфера: 4,000000, Скорость обмена данных: 4
Производитель: 5, Объем: 5,000000, Скорость вращения: 5, Интерфейс: 5, Объем буфера: 5,000000, Скорость обмена данных: 5
Меню:
1)Создание новой записи
2)Поиск записи
3)Запись в файл
4)Чтение из файла
5)Сортировка и печать
6)Изменение отдельной записи
7)Добавление произвольного числа записей
8)Вывести текущий массив
0)Выход

```

Рисунок 17 – Меню действий с массивом

При выборе поиска записи будет предложено ввести количество критериев поиска и информацию для поиска (рисунок 18, 19).

```

Производитель: 5, Объем: 5,000000, Скорость вращения: 5, Интерфейс: 5, Объем буфера: 5,000000, Скорость обмена данных: 5
Производитель: 1, Объем: 1,000000, Скорость вращения: 1, Интерфейс: 1, Объем буфера: 1,000000, Скорость обмена данных: 1
Производитель: 2, Объем: 2,000000, Скорость вращения: 2, Интерфейс: 2, Объем буфера: 2,000000, Скорость обмена данных: 2
Производитель: 3, Объем: 3,000000, Скорость вращения: 3, Интерфейс: 3, Объем буфера: 3,000000, Скорость обмена данных: 3
Производитель: 4, Объем: 4,000000, Скорость вращения: 4, Интерфейс: 4, Объем буфера: 4,000000, Скорость обмена данных: 4
Производитель: 5, Объем: 5,000000, Скорость вращения: 5, Интерфейс: 5, Объем буфера: 5,000000, Скорость обмена данных: 5
Меню:
1)Создание новой записи
2)Поиск записи
3)Запись в файл
4)Чтение из файла
5)Сортировка и печать
6)Изменение отдельной записи
7)Добавление произвольного числа записей
8)Вывести текущий массив
9)Выход
Некорректный ввод. Пожалуйста, введите число от 0 до 8: 2
Введите количество полей для поиска
1
Введите поле для поиска (1-объем от и до, 2-интерфейс)
1
Введите минимальный и максимальный объем для поиска (через пробел)
3 10
Производитель: 3, Объем: 3,000000, Скорость вращения: 3, Интерфейс: 3, Объем буфера: 3,000000, Скорость обмена данных: 3
Производитель: 4, Объем: 4,000000, Скорость вращения: 4, Интерфейс: 4, Объем буфера: 4,000000, Скорость обмена данных: 4
Производитель: 5, Объем: 5,000000, Скорость вращения: 5, Интерфейс: 5, Объем буфера: 5,000000, Скорость обмена данных: 5

```

Рисунок 18 – Поиск по одному полю

```

Введите количество полей для поиска
2
Поиск по двум полям:
Введите минимальный и максимальный объем (через пробел)
2 6
Введите интерфейс
4
Производитель: 4, Объем: 4,000000, Скорость вращения: 4, Интерфейс: 4, Объем буфера: 4,000000, Скорость обмена данных: 4

```

Рисунок 19 – Поиск по двум полям

При выборе записи в файл (действие 3) будет предложено ввести название файла. При успешной записи в файл будет выведена информация из него (рисунок 20). При выборе чтения из файла (действие 4) созданный массив будет заменен считанным из файла (рисунок 14).

```

Некорректный ввод. Пожалуйста, введите число от 0 до 8: 3
Введите название файла
arr2.txt
Производитель: 1, Объем: 1,000000, Скорость вращения: 1, Интерфейс: 1, Объем буфера: 1,000000, Скорость обмена данных: 1
Производитель: 2, Объем: 2,000000, Скорость вращения: 2, Интерфейс: 2, Объем буфера: 2,000000, Скорость обмена данных: 2
Производитель: 3, Объем: 3,000000, Скорость вращения: 3, Интерфейс: 3, Объем буфера: 3,000000, Скорость обмена данных: 3
Производитель: 4, Объем: 4,000000, Скорость вращения: 4, Интерфейс: 4, Объем буфера: 4,000000, Скорость обмена данных: 4
Производитель: 5, Объем: 5,000000, Скорость вращения: 5, Интерфейс: 5, Объем буфера: 5,000000, Скорость обмена данных: 5

```

Рисунок 20 – Запись массива в файл

При выборе сортировки и печати (действие 5) программа отсортирует текущий массив по введенному пользователем критерию в введенном порядке и выведет его на экран (рисунок 21).

```
Некорректный ввод. Пожалуйста, введите число от 0 до 8: 5
Введите критерий сортировки (1-производитель, 2-объем, 3-скорость вращения, 4-интерфейс, 5-объем буфера, 6-скорость обмена данных)
3
Выберите тип сортировки (1-по возрастанию, 2-по убыванию)
2
Производитель: 5, Объем: 5,000000, Скорость вращения: 5, Интерфейс: 5, Объем буфера: 5,000000, Скорость обмена данных: 5
Производитель: 4, Объем: 4,000000, Скорость вращения: 4, Интерфейс: 4, Объем буфера: 4,000000, Скорость обмена данных: 4
Производитель: 3, Объем: 3,000000, Скорость вращения: 3, Интерфейс: 3, Объем буфера: 3,000000, Скорость обмена данных: 3
Производитель: 2, Объем: 2,000000, Скорость вращения: 2, Интерфейс: 2, Объем буфера: 2,000000, Скорость обмена данных: 2
Производитель: 1, Объем: 1,000000, Скорость вращения: 1, Интерфейс: 1, Объем буфера: 1,000000, Скорость обмена данных: 1
```

Рисунок 21 – Сортировка массива по убыванию

При выборе изменения отдельной записи (действие 6) будет предложено ввести индекс редактируемой записи и номер изменяемого поля (рисунок 22).

```
6
Введите индекс элемента, который нужно отредактировать
2
Введите параметр для редактирования (1-производитель, 2-объем, 3-скорость вращения, 4-интерфейс, 5-объем буфера, 6-скорость обмена данных)
1
Введите значение для вставки
аааааа
```

Рисунок 22 – Изменение отдельной записи

Чтобы увидеть изменения, можно выбрать печать текущего массива (действие 8). Приведем результат на рисунке 23.

```
Некорректный ввод. Пожалуйста, введите число от 0 до 8: 8
Производитель: 5, Объем: 5,000000, Скорость вращения: 5, Интерфейс: 5, Объем буфера: 5,000000, Скорость обмена данных: 5
Производитель: 4, Объем: 4,000000, Скорость вращения: 4, Интерфейс: 4, Объем буфера: 4,000000, Скорость обмена данных: 4
Производитель: аааааа, Объем: 3,000000, Скорость вращения: 3, Интерфейс: 3, Объем буфера: 3,000000, Скорость обмена данных: 3
Производитель: 2, Объем: 2,000000, Скорость вращения: 2, Интерфейс: 2, Объем буфера: 2,000000, Скорость обмена данных: 2
Производитель: 1, Объем: 1,000000, Скорость вращения: 1, Интерфейс: 1, Объем буфера: 1,000000, Скорость обмена данных: 1
```

Рисунок 23 – Печать текущего массива

При выборе добавления произвольного количества записей (действие 7) будет предложено ввести количество добавляемых элементов. Далее пользователь должен вводить новые записи по одной, после каждой записи

будет предложено отредактировать введенную информацию в случае неверного ввода (рисунки 24, 25) Чтобы увидеть новый массив, необходимо выбрать действие 8 (рисунок 26).

```
7
Введите количество элементов для добавления
2
```

Рисунок 24 – Ввод количества новых записей

```
Введите данные о жестком диске в порядке: производитель, объем, скорость вращения, интерфейс, объем буфера, скорость обмена данных
sony 3 400 ata 2 200
Производитель: sony, Объем: 3,000000, Скорость вращения: 400, Интерфейс: ata, Объем буфера: 2,000000, Скорость обмена данных: 200
Если данные введены некорректно введите 1 и попробуйте еще раз, иначе введите 0
```

Рисунок 25 – Проверка корректности введения записи

```
8
Производитель: 5, Объем: 5,000000, Скорость вращения: 5, Интерфейс: 5, Объем буфера: 5,000000, Скорость обмена данных: 5
Производитель: 4, Объем: 4,000000, Скорость вращения: 4, Интерфейс: 4, Объем буфера: 4,000000, Скорость обмена данных: 4
Производитель: аааааа, Объем: 3,000000, Скорость вращения: 3, Интерфейс: 3, Объем буфера: 3,000000, Скорость обмена данных: 3
Производитель: 2, Объем: 2,000000, Скорость вращения: 2, Интерфейс: 2, Объем буфера: 2,000000, Скорость обмена данных: 2
Производитель: 1, Объем: 1,000000, Скорость вращения: 1, Интерфейс: 1, Объем буфера: 1,000000, Скорость обмена данных: 1
Производитель: 1, Объем: 1,000000, Скорость вращения: 1, Интерфейс: 1, Объем буфера: 1,000000, Скорость обмена данных: 1
Производитель: 2, Объем: 2,000000, Скорость вращения: 2, Интерфейс: 2, Объем буфера: 2,000000, Скорость обмена данных: 2
```

Рисунок 26 – Добавление записей в массив

При выборе выхода (действие 0) программа завершается (рисунок 27).

```
Меню:
1)Создание новой записи
2)Поиск записи
3)Запись в файл
4)Чтение из файла
5)Сортировка и печать
6)Изменение отдельной записи
7)Добавление произвольного числа записей
8)Вывести текущий массив
0)Выход
0
C:\Users\User\source\repos\course_project\x64\D
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 27 – Выход из программы

ЗАКЛЮЧЕНИЕ

При выполнении курсового проекта была разработана программа для работы с файловыми базами данных. Были выполнены все поставленные задачи. Была создана структура для записей, реализован интерфейс для взаимодействия с пользователем. В программе предусмотрены: поиск записи по одному или двум критериям; сортировка массива по любому полю в порядке убывания или возрастания; добавление произвольного количества новых записей; редактирование произвольного поля произвольной записи; запись базы данных в файл и чтение из файла. Данная программа также включает проверки на корректность введения данных пользователем и на существование создаваемых файлов и массивов, что позволяет обезопасить ввод и хранение данных. Таким образом, в ходе выполнения данной курсовой работы была создана программа для хранения данных о жестких дисках, которая может быть использована для различных задач.

Ссылка на гитхаб: https://github.com/Alena892/Corse_project_MAIN.git

СПИСОК ЛИТЕРАТУРЫ

1. Кнут, Д. Искусство программирования. В 3 т. — М.: Вильямс, 2012.
2. Павловская, Т. А. С/C++. Программирование на языке высокого уровня. — СПб., 2003. — 461 с.
3. Подбельский В. В., Фомин С. С. Программирование на языке Си: учеб. пособие. 2-е доп. изд. — М.: Финансы и статистика, 2004. — 600 с.
4. Гукин, Д. Программирование на С для чайников. — М.: Диалектика, 2019. — 384 с.
5. Гергель, В. П. Современные языки и технологии параллельного программирования: Учебник. — М.: МГУ, 2016. — 408 с.
6. Кузин, А. В. Базы данных. Учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. - 5-е изд., испр. - М. : Академия, 2012. - 316 с. : рис., табл. - (Высшее профессиональное образование. Бакалавриат).
7. Пирогов В.Ю. Информационные системы и базы данных: организация и проектирование: учебное пособие – Издательство: БХВ-Петербург, 2009 г. (<http://www.knigafund.ru>).
8. Ашарина, И.В. Основы программирования на языках С и С++: Курс лекций для высших учебных заведений / И.В. Ашарина. — М.: Гор. линия-Телеком, 2018. — 208 с.
9. Керниган, Б. Язык программирования С. 2-е изд. / Б. Керниган, Д.М. Ритчи. — М.: Вильямс, 2016. — 288 с.
10. Дорогов, В.Г. Основы программирования на языке С: Учебное пособие / В.Г. Дорогов, Е.Г. Дорогова; Под общ. ред. проф. Л.Г. Гагарина. — М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2017. — 224 с.

ПРИЛОЖЕНИЕ А

ОПИСАНИЕ СТРУКТУРНОГО ТИПА ДАННЫХ

```
struct hard_drive {  
    char przv[10]; // производитель  
    float space; // объем  
    int sv; // скорость вращения  
    char intrfs[10]; // интерфейс  
    float sp_b; // объем буфера  
    int s_data; // скорость обмена данными  
};  
typedef struct hard_drive hr_d;
```

ПРИЛОЖЕНИЕ В

ЛИСТИНГ ПРОГРАММЫ

```
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
struct hard_drive {
    char przv[10]; // производитель
    float space; // объем
    int sv; // скорость вращения
    char intrfs[10]; // интерфейс
    float sp_b; // объем буфера
    int s_data; // скорость обмена данных
};
typedef struct hard_drive hr_d;

//-----функции-----
// заполнение массива
int fill_array(hr_d* arr, int size) {
    printf("Варианты для заполнения полей\n1)Производитель - любое название\n2)Объем - вещественное число\n3)Скорость вращения - целое число\n4)Интерфейс - название (например, usb, ata, scsi)\n5)Объем буфера - вещественное число\n6)Скорость обмена данных - целое число\n");
    for (int i = 0; i < size; i++) {
        int valid_input = 0; // корректность ввода

        while (!valid_input) {
            printf("Введите данные о жестком диске через пробел в порядке: производитель, объем, скорость вращения, интерфейс, объем буфера, скорость обмена данных\n");
            int result = scanf("%9s %f %i %9s %f %i", arr[i].przv, &arr[i].space, &arr[i].sv, arr[i].intrfs, &arr[i].sp_b, &arr[i].s_data);

            if (result == 6) {
                valid_input = 1;
                printf("Производитель: %s, Объем: %f, Скорость вращения: %i, Интерфейс: %s, Объем буфера: %f, Скорость обмена данных: %i\n", arr[i].przv, arr[i].space, arr[i].sv, arr[i].intrfs, arr[i].sp_b, arr[i].s_data);
            }
            else {
                printf("Некорректные данные. Попробуйте еще раз.\n");
                while (getchar() != '\n'); // очистка буфера ввода
            }
        }
    }
    return size;
}

// печать массива
int print_array(hr_d* arr, int size) {
    for (int i = 0; i < size; i++) {
        printf("Производитель: %s, Объем: %f, Скорость вращения: %i, Интерфейс: %s, Объем буфера: %f, Скорость обмена данных: %i\n", arr[i].przv, arr[i].space, arr[i].sv, arr[i].intrfs, arr[i].sp_b, arr[i].s_data);
    }
    return size;
}

// запись в файл
int input_file(char* fname, hr_d* arr, int size) {
    FILE* out = fopen(fname, "wt"); // файл для записи массива
    if (!out) {
        printf("Ошибка открытия файла\n");
        return 1;
    }
    for (int i = 0; i < size; i++) {
        fprintf(out, "Производитель: %s Объем: %f Скорость вращения: %i Интерфейс: %s Объем буфера: %f Скорость обмена данных: %i\n", arr[i].przv, arr[i].space, arr[i].sv, arr[i].intrfs, arr[i].sp_b, arr[i].s_data);
    }
    fclose(out);
    return size;
}

// чтение из файла
int output_file(char* fname, hr_d* arr, int size) {
    FILE* in = fopen(fname, "rt"); // файл, откуда считывается массив
    if (!in) {
```

```

        printf("Ошибка открытия файла\n");
        return 1;
    }
    int count = 0; // количество считанных структур
    while (count < size && fscanf(in, "Производитель: %s Объем: %f Скорость вращения: %i Интерфейс: %s Объем буфера: %f Скорость
обмена данных: %i\n", arr[count].przv, &arr[count].space, &arr[count].sv, arr[count].intrfs, &arr[count].sp_b, &arr[count].s_data) == 6) {
        count++;
    }
    if (count < size) {
        printf("В файле не хватает данных для считывания всех структур\n");
    }
    fclose(in);
    return count;
}
// поиск элемента
int find_element(hr_d* arr, int size) {
    char data[10], data1[10]; // данные об интерфейсе
    float sp1, sp2; // данные об объеме: sp1 - минимальный объем; sp2 - максимальный
    int vybor, qt; // vybor - выбор поля, qt - количество искомых полей

    printf("Введите количество полей для поиска\n");
    qt = get_valid_input(1, 2);
    if (qt == 1) {
        printf("Введите поле для поиска (1-объем от и до, 2-интерфейс)\n");
        vybor = get_valid_input(1, 2);

        if (vybor == 1) {
            printf("Введите минимальный и максимальный объем для поиска (через пробел)\n");
            scanf("%f %f", &sp1, &sp2);

            for (int i = 0; i < size; i++) {
                if ((sp1 <= arr[i].space) && (arr[i].space <= sp2)) printf("Производитель: %s, Объем: %f, Скорость вращения: %i, Интерфейс: %s,
Объем буфера: %f, Скорость обмена данных: %i\n", arr[i].przv, arr[i].space, arr[i].sv, arr[i].intrfs, arr[i].sp_b, arr[i].s_data);
            }
        }
        else if (vybor == 2) {
            printf("Введите интерфейс\n");
            scanf("%9s", data);
            for (int i = 0; i < size; i++) {
                if (strcmp(arr[i].intrfs, data) == 0)
                    printf("Производитель: %s, Объем: %f, Скорость вращения: %i, Интерфейс: %s, Объем буфера: %f, Скорость обмена данных:
%i\n",
                        arr[i].przv, arr[i].space, arr[i].sv, arr[i].intrfs, arr[i].sp_b, arr[i].s_data);
            }
        }
    }
    else if (qt == 2) {
        printf("Поиск по двум полям:\n");
        printf("Введите минимальный и максимальный объем (через пробел)\n");
        scanf("%f %f", &sp1, &sp2);
        printf("Введите интерфейс\n");
        scanf("%9s", data1);
        for (int i = 0; i < size; i++) {
            if (((sp1 <= arr[i].space) && (arr[i].space <= sp2)) && (strcmp(arr[i].intrfs, data1) == 0)) printf("Производитель: %s, Объем: %f,
Скорость вращения: %i, Интерфейс: %s, Объем буфера: %f, Скорость обмена данных: %i\n", arr[i].przv, arr[i].space, arr[i].sv, arr[i].intrfs,
arr[i].sp_b, arr[i].s_data);
        }
    }
    return size;
}
// сортировка массива
int sort_massiv(hr_d* arr, int size) {
    int vybor1, vybor2; // выбор пользователя
    printf("Введите критерий сортировки (1-производитель, 2-объем, 3-скорость вращения, 4-интерфейс, 5-объем буфера, 6-скорость
обмена данных)\n");
    vybor1 = get_valid_input(1, 6);
    printf("Выберите тип сортировки (1-по возрастанию, 2-по убыванию)\n");
    vybor2 = get_valid_input(1, 2);

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

            switch (vybor1) {
                case 1: sort = strcmp(arr[j].przv, arr[j + 1].przv); break;
                case 2: sort = (arr[j].space > arr[j + 1].space) ? 1 : -1; break;
                case 3: sort = (arr[j].sv > arr[j + 1].sv) ? 1 : -1; break;
                case 4: sort = strcmp(arr[j].intrfs, arr[j + 1].intrfs); break;
                case 5: sort = (arr[j].sp_b > arr[j + 1].sp_b) ? 1 : -1; break;
            }
        }
    }
}

```



```

        case 6: sort = (arr[j].s_data > arr[j + 1].s_data) ? 1 : -1; break;
    }

    if ((vybor2 == 1 && sort > 0) || (vybor2 == 2 && sort < 0)) {
        hr_d temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
    }
}

for (int i = 0; i < size; i++) {
    printf("Производитель: %s, Объем: %f, Скорость вращения: %i, Интерфейс: %s, Объем буфера: %f, Скорость обмена данных: %i\n",
        arr[i].przv, arr[i].space, arr[i].sv, arr[i].intrfs, arr[i].sp_b, arr[i].s_data);
}
return size;
}

// добавление произвольного количества элементов
int add_elem(hr_d* arr, int size) {
    int vybor1, vybor2 = 0; // выбор пользователя
    hr_d* arr_add; // измененный массив
    printf("Введите количество элементов для добавления\n");
    vybor1 = get_valid_input(1, 100); // ограничение количества добавляемых элементов
    if (arr == NULL) {
        printf("Ошибка");
        return 0;
    }
    arr_add = arr;
    arr_add = realloc(arr, (size + vybor1) * sizeof(hr_d));
    if (arr_add == NULL) {
        printf("Ошибка выделения памяти\n");
        return size;
    }
    arr = arr_add;
    printf("Заполнение полей:\n1) Производитель - название\n2) Объем - вещественное число\n3) Скорость вращения - целое
число\n4) Интерфейс - название\n5) Объем буфера - вещественное число\n6) Скорость обмена данных - целое число\n");
    for (int i = size; i < size + vybor1; i++) {
        printf("Введите данные о жестком диске в порядке: производитель, объем, скорость вращения, интерфейс, объем буфера, скорость
обмена данных\n");
        int result = scanf("%9s %f %i %9s %f %i", arr[i].przv, &arr[i].space, &arr[i].sv, arr[i].intrfs, &arr[i].sp_b, &arr[i].s_data);
        if (result != 6) {
            printf("Некорректные данные. Попробуйте еще раз.\n");
            while (getchar() != '\n'); // очистка буфера ввода
            i--;
            continue;
        }
        printf("Производитель: %s, Объем: %f, Скорость вращения: %i, Интерфейс: %s, Объем буфера: %f, Скорость обмена данных: %i\n",
            arr[i].przv, arr[i].space, arr[i].sv, arr[i].intrfs, arr[i].sp_b, arr[i].s_data);
        printf("Если данные введены некорректно введите 1 и попробуйте еще раз, иначе введите 0\n");
        vybor2 = get_valid_input(0, 1);
        if (vybor2 == 1) i--;
    }
    return (size + vybor1);
}

// изменение элемента
int edit_elem(hr_d* arr, int size) {
    int vybor1, vybor2; // выбор пользователя
    printf("Введите индекс элемента, который нужно отредактировать\n");
    vybor1 = get_valid_input(0, size - 1);
    printf("Введите параметр для редактирования (1-производитель, 2-объем, 3-скорость вращения, 4-интерфейс, 5-объем буфера, 6-
скорость обмена данных)\n");
    vybor2 = get_valid_input(1, 6);
    printf("Введите значение для вставки\n");
    switch (vybor2) {
        case 1: scanf("%9s", arr[vybor1].przv); break;
        case 2: scanf("%f", &arr[vybor1].space); break;
        case 3: scanf("%i", &arr[vybor1].sv); break;
        case 4: scanf("%9s", &arr[vybor1].intrfs); break;
        case 5: scanf("%f", &arr[vybor1].sp_b); break;
        case 6: scanf("%i", &arr[vybor1].s_data); break;
        default:
            printf("Введены неверные данные\n");
            break;
    }
    return size;
}

// функция проверки корректности ввода числа
get_valid_input(int min, int max) {
    int input;

```

```

char buffer[256];
while (1) {
    if (fgets(buffer, sizeof(buffer), stdin) != NULL) {
        if (sscanf(buffer, "%d", &input) == 1 && input >= min && input <= max) {
            return input;
        }
        else {
            printf("Некорректный ввод. Пожалуйста, введите число от %d до %d: ", min, max);
        }
    }
}
return 0;
}
// функция для проверки существования файла
file_have(const char* filename) {
    FILE* file = fopen(filename, "r");
    if (file) {
        fclose(file);
        return 1;
    }
    return 0;
}

//-----основная программа с интерфейсом-----
int main() {
    setlocale(LC_ALL, "RUS");
    hr_d* arr0 = NULL; // исходный массив
    hr_d* arr_out = NULL; // массив для вывода из файла
    int size0 = 0, size_out = 0; // размер массивов
    int vybor = 123, vybor1 = 0; // выбор вариантов
    char fname1[20] = "", fname2[20] = ""; // названия файлов

    // получение массива для работы
    printf("Выберите способ получения массива:\n1)Создать новый массив\n2)Считать массив из файла\n");
    vybor1 = get_valid_input(1, 2);
    switch (vybor1) {
        case 1:
            printf("Введите размер массива\n");
            size0 = get_valid_input(1, 100); // ограничение размера массива
            arr0 = (hr_d*)malloc(size0 * sizeof(hr_d));
            fill_array(arr0, size0);
            print_array(arr0, size0);
            break;
        case 2:
            printf("Введите размер массива для вывода\n");
            size_out = get_valid_input(1, 100); // ограничим размер массива до 100
            printf("Введите название файла для вывода\n");
            scanf("%s", fname2);
            if (!file_have(fname2)) {
                printf("Файл с таким именем не существует. Введите корректное имя файла.\n");
                break;
            }
            arr_out = (hr_d*)malloc(size_out * sizeof(hr_d));
            size_out = output_file(fname2, arr_out, size_out);
            arr0 = arr_out;
            size0 = size_out;
            print_array(arr0, size0);
            break;
        default:
            break;
    }

    //-----основной цикл-----
    while (vybor != 0) {
        printf("Меню:\n1)Создание новой записи\n2)Поиск записи\n3)Запись в файл\n4)Чтение из файла\n5)Сортировка и печать\n6)Изменение отдельной записи\n7)Добавление произвольного числа записей\n8)Вывести текущий массив\n0)Выход\n");
        vybor = get_valid_input(0, 8);
        switch (vybor) {
            case 1:// создание новой записи
                printf("Введите размер массива\n");
                size0 = get_valid_input(1, 100); // ограничение размера массива
                arr0 = (hr_d*)malloc(size0 * sizeof(hr_d));
                fill_array(arr0, size0);
                print_array(arr0, size0);
                break;
            case 2:// поиск элемента
                find_element(arr0, size0);
                break;
            case 3:// запись в файл

```

```

    printf("Введите название файла\n");
    scanf("%s", fname1);
    input_file(fname1, arr0, size0);
    print_array(arr0, size0);
    break;
case 4:// чтение из файла
    printf("Введите размер массива, который необходимо вывести\n");
    size_out = get_valid_input(1, 100); // ограничение размера массива
    printf("Введите название файла, из которого нужно вывести информацию\n");
    scanf("%s", fname2);
    if (!file_have(fname2)) {
        printf("Файл с таким именем не существует. Пожалуйста, введите корректное имя файла.\n");
        break;
    }
    arr_out = (hr_d*)malloc(size_out * sizeof(hr_d));
    size_out = output_file(fname2, arr_out, size_out);
    arr0 = arr_out;
    size0 = size_out;
    print_array(arr0, size0);
    break;
case 5:// сортировка массива
    sort_massiv(arr0, size0);
    break;
case 6:// изменение записи
    edit_elem(arr0, size0);
    break;
case 7:// добавление произвольного количества элементов
    size0 = add_elem(arr0, size0);
    break;
case 8:// вывод массива
    print_array(arr0, size0);
    break;
default:
    break;
}
}
free(arr0); // освобождение памяти
free(arr_out); // освобождение памяти
}

```