МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ФГБОУ ВО «ВГТУ»)

Факультет экономики, менеджмента и информационных технологий

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 «Умные колонки»

**Расчетно-пояснительная записка**

Разработал студент Е.А. Комов

Подпись, дата Инициалы, фамилия

Руководитель Н.В.Акамсина

Подпись, дата Инициалы, фамилия

Нормоконтролер

Подпись, дата Инициалы, фамилия

Защищена Оценка

дата

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 «Умные колонки»

Студент бТИИ-241 Комов Егор Алексеевич

Группа, фамилия, имя, отчество

База данных «Умные колонки», Признак поиска: производитель, голосовой помощник; Вариант сортировки: производитель, мощность.

Технические условия Windows 10, MicrosoftVisualStudio2022, язык программирования C

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

стр, рисунков, таб, приложений

Сроки выполнения этапов анализ и постановка задачи ;

Разработка пошаговой детализации программы ;

Реализация программы ;

Тестирование программы ;

Оформление пояснительной записки .

Срок защиты курсового проекта:

Руководитель Н.В.Акамсина

Подпись, дата Инициалы, фамилия

Задание принял студент Е.А. Комов

Подпись, дата Инициалы, фамилия

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc187512230)

[1. Постановка задачи 5](#_Toc187512231)

[2. Реализация программы 6](#_Toc187512232)

[3. Тестирование программы 12](#_Toc187512233)

[Заключение 18](#_Toc187512234)

[Список литературы 19](#_Toc187512235)

[Приложение 20](#_Toc187512236)

# ВВЕДЕНИЕ

Файловая база представляет собой файл, содержащий упорядоченные записи данных, описывающие определённую предметную область. Для работы с такими записями необходимо разработать программу, которая позволяет создавать новые записи, изменять существующие, выполнять поиск по заданным критериям, сортировать структуры по заданным параметрам, сохранять информацию в файл и считывать её из файла, а также проверять ввод пользователя на ошибки, которые могут привести к сбою программы, и уведомлять пользователя о предотвращении таких ошибок.

Цель данной работы – разработка программы для управления записями данных различного типа в предметной области «Умные колонки», хранящимися в файле.

Для достижения этой цели необходимо решить ряд задач, каждая из которых важна для корректной работы программы:

1. Постараться использовать наиболее оптимальные структуры данных и формат файла.
2. Разработка алгоритма действий, а также создание блок-схемы для него.
3. Создать удобный и интуитивно понятный интерфейс, позволяющий работать с программой пока сам пользователь не захочет обратного.
4. Разработать функции создания записей; записи в файл структур; чтения структур в количестве, заданном пользователем.
5. Реализовать возможность сортировки базы данных и поиска в ней.
6. Предусмотреть все возможные ошибки и информировать пользователя при их появлении.

## 1. Постановка задачи

Главной задачей курсового проекта является создание базы данных на тему «Умные колонки». Для выполнения всех функций программа должна позволять добавлять данные в файл, удалять, производить поиск и сортировать их.

В базе должны содержаться поля:

* Производитель – строка, содержащая до 20 символов, записывающих название компании производителя. Функционально поле должно иметь возможность сортировки и поиска.
* Голосовой помощник – строка, в которой будет содержаться строка до 50 символов, несущая в себе информацию о возможных голосовых помощниках для данной модели. По этому полю должна быть возможность поиска.
* Мощность – поле, содержащие целое число, хранящее информацию о мощности колонки в ваттах. По этому полю предусматривается сортировка.

Необходимо реализовать возможность контакта с пользователем, по средству меню, в котором будут содержаться функции:

1. Создание новой записи.
2. Поиск записи по заданным пользователем значение полей, а именно «производитель» или «голосовой помощник».
3. Запись и чтение всех данных из файла.
4. Печать всех записей, упорядоченных по определенному критерию, то есть сортировка. Включает поля «производитель» и «мощность».

## 2. Реализация программы

Для того, чтобы максимально корректно сконструировать программу создадим алгоритм для каждой функции в виде блок схемы. Функция main() отвечает за выполнение всех действий с базой данных и использует другие функции. Составим блок-схему для нее в первую очередь. (рисунок 1)

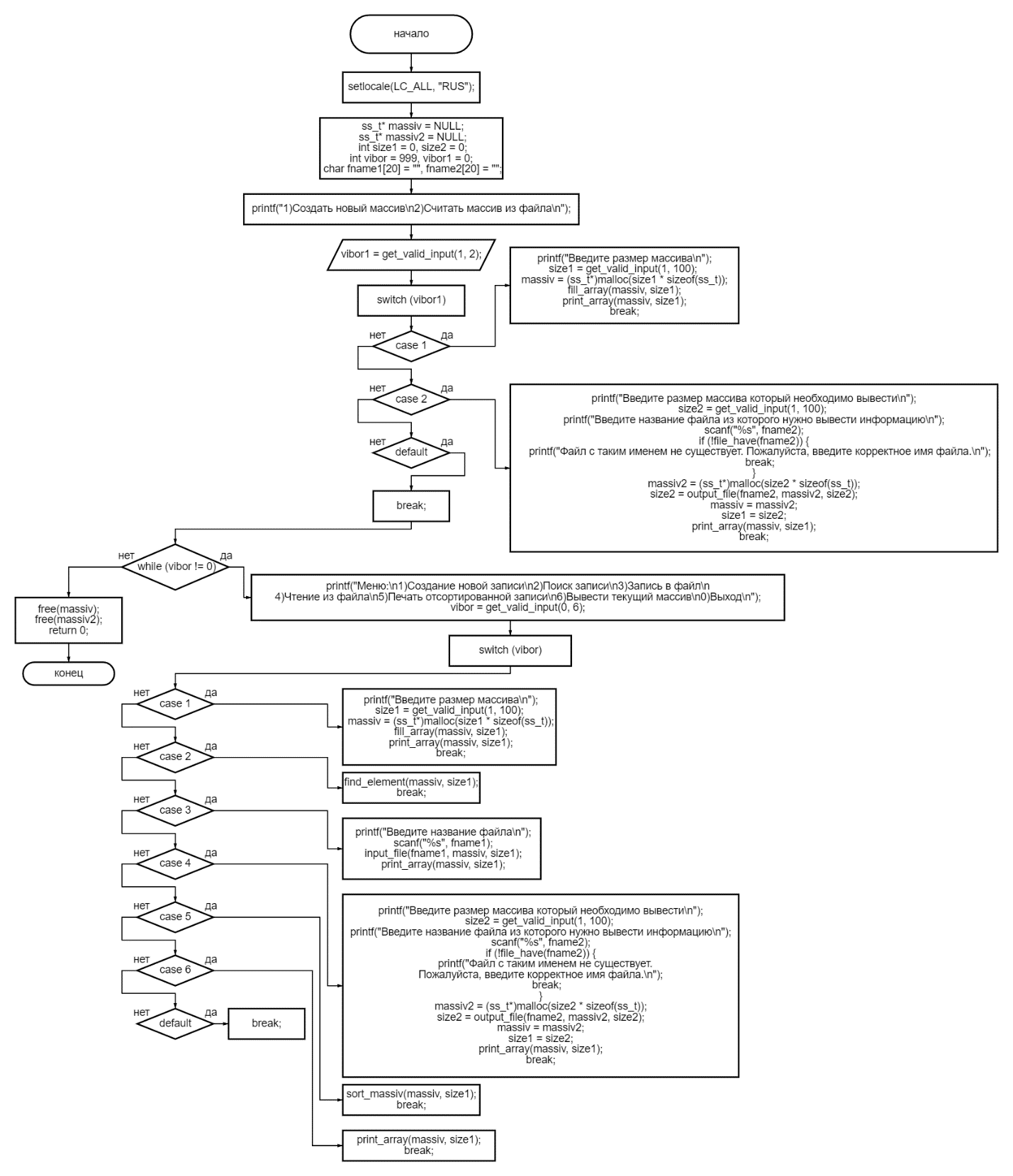


Рисунок 1 – блок-схема функции main().

Приступаем к созданию блок-схем для неосновных функций. print\_array(ss\_t\* massiv, int size) отвечает за вывод уже имеющегося массива на экран, с помощью построчного вывода в цикле каждого отдельного элемента массива(данные каждой умной колонки). (рисунок 2)

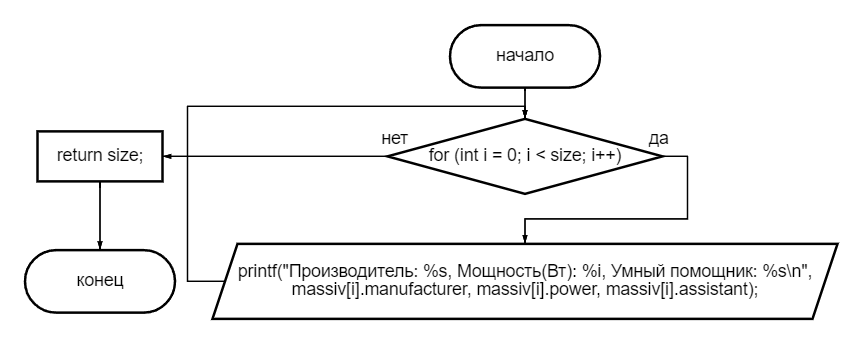


Рисунок 2 – блок-схема print\_array().

Функция input\_file(char\* fname, ss\_t\* massiv, int size) отвечает за запись массива, уже имеющегося в программе внутрь файла. В ней с помощью fopen() с клавиатуры задается файл, в который будут занесены данные, после чего проверяется имеется ли такой файл и если да, то с помощью цикла построчно заносятся данные в файл, после чего он закрывается. Создаем блок-схему для этой функции. (рисунок 3)

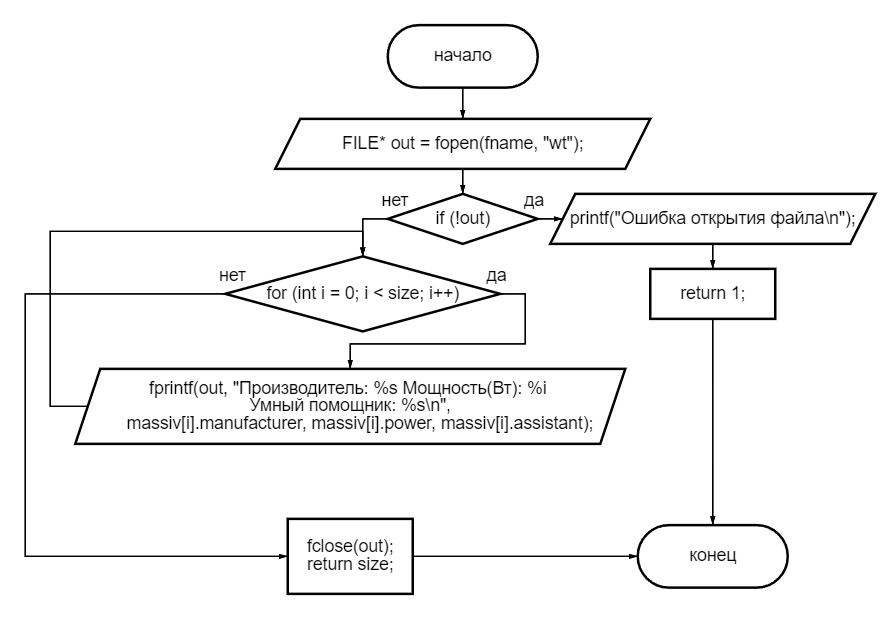


Рисунок 3 – блок-схема input\_file().

Далее рассмотрим функцию fill\_array(ss\_t\* massiv, int size), которая позволяет вручную вводить элементы массива с клавиатуры. Вводятся данные и проверяется корректность ввода, если все верно после первого элемента данные вводятся до конца заданного массива или пока не случится ошибка. Создаем блок-схему для этой функции. (рисунок 4)

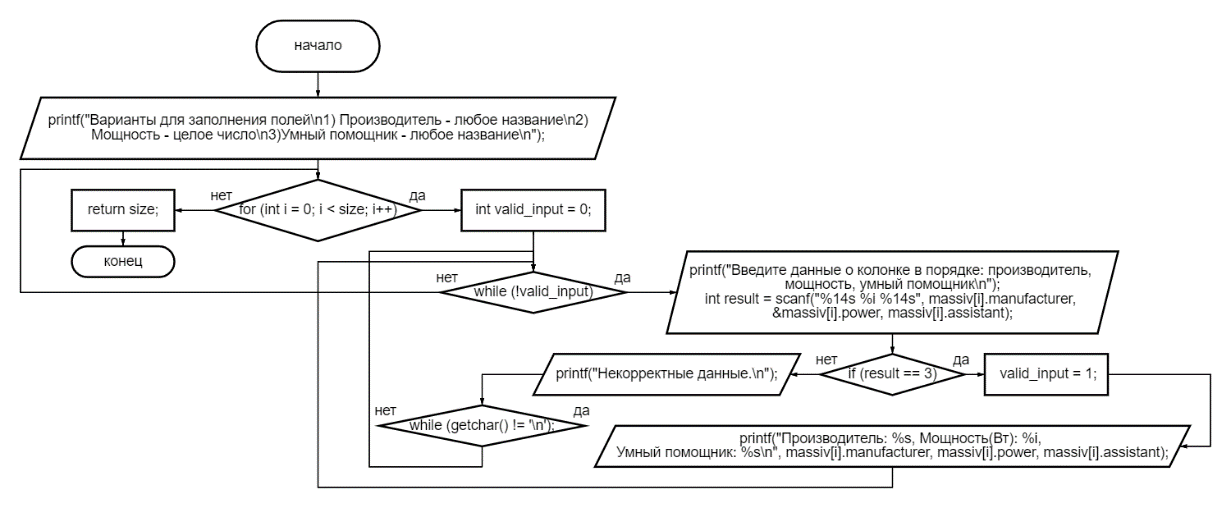


Рисунок 4 – блок-схема fill\_array().

Функция output\_file(char\* fname, ss\_t\* massiv, int size) используется для получения массива из файла. В начале в ней вводится название файла и проверяется существование введенного файла. Если файл удается удачно открыть, то из него считываются данные, до заданного момента или пока файл не закончится, в это время параллельно проверяется хватает ли элементов для заданного кол-ва и останавливает процесс если нет. В конце файл закрывается и возвращается кол-во элементов. Создадим блок-схему для этой функции. (рисунок 5)

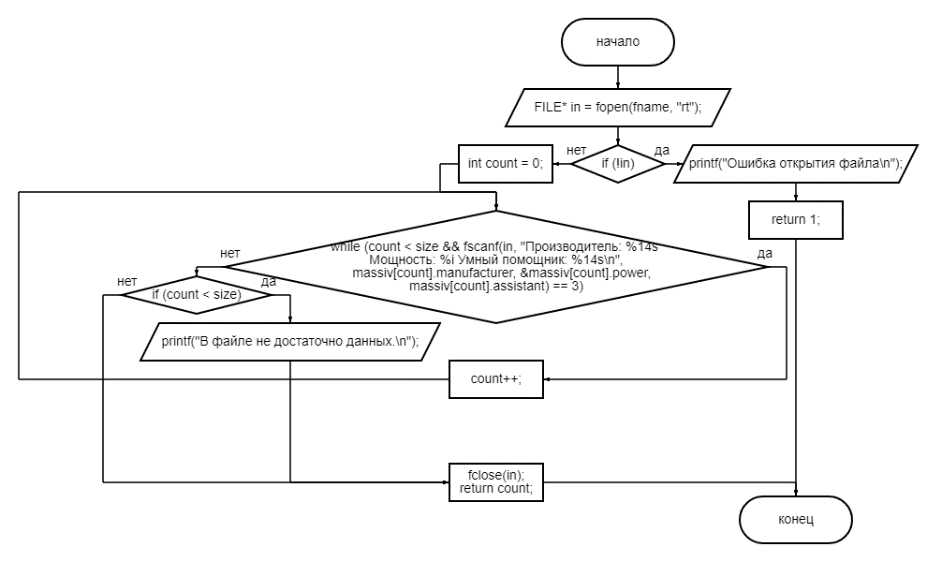


Рисунок 5 – блок-схема output\_file().

Приступаем к созданию блок-схемы для функции find\_element(ss\_t\* massiv, int size). Она отвечает за возможность поиска по массиву. Использует 2 переменные для выбора количества полей поиска и нужного поля для поиска. После этого проходится циклом по массиву и выбирает подходящие элементы. Создадим блок-схему для этой функции. (рисунок 6)

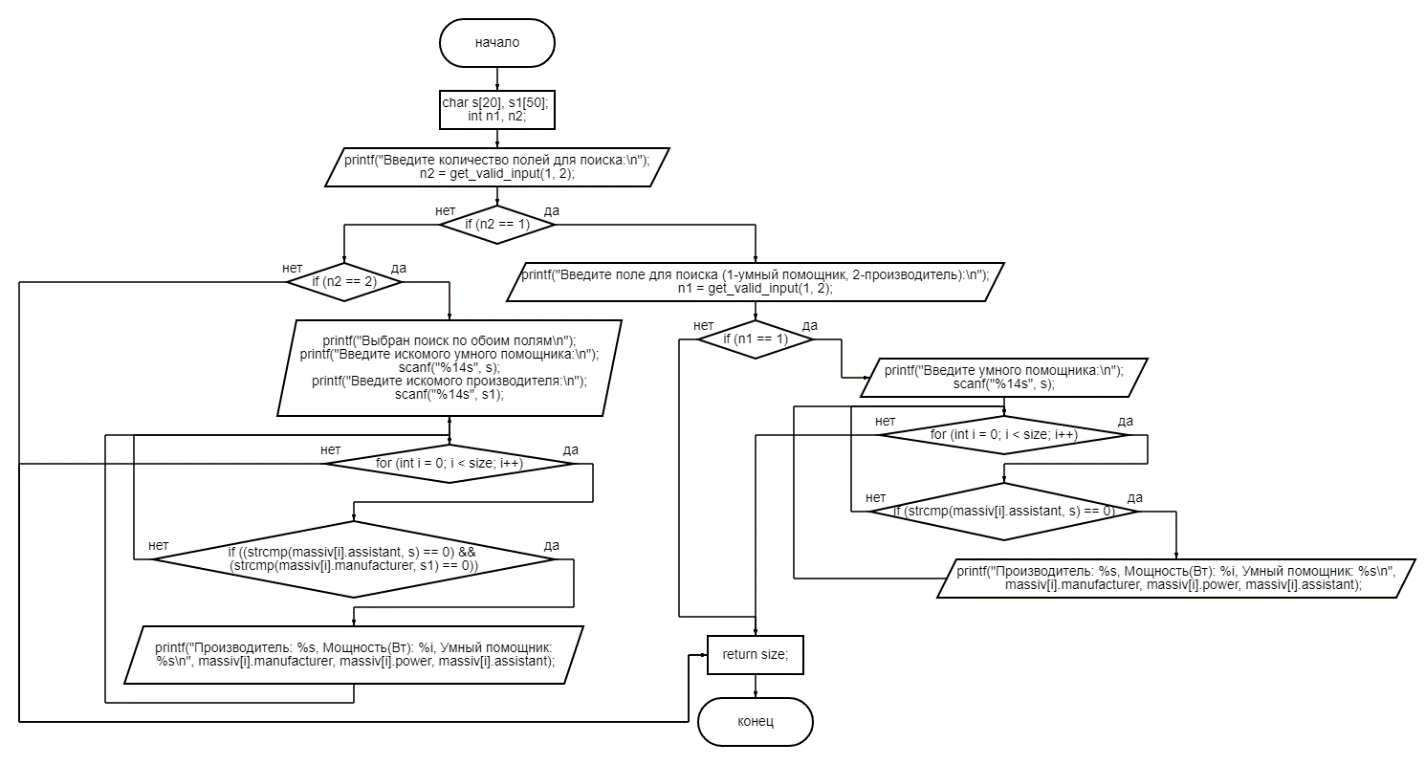


Рисунок 6 – блок-схема find\_element().

Функция сортировки sort\_massiv(ss\_t\* massiv, int size) похожа на функцию поиска, так как в обеих этих функциях используются переменные, в которые вводится значения для выбора поля сортировки и порядка сортировки. Цикл проходится по всему массиву выставляя элементы в заданном порядке, а в конце возвращает size. Создадим блок-схему для этой функции. (рисунок 7)

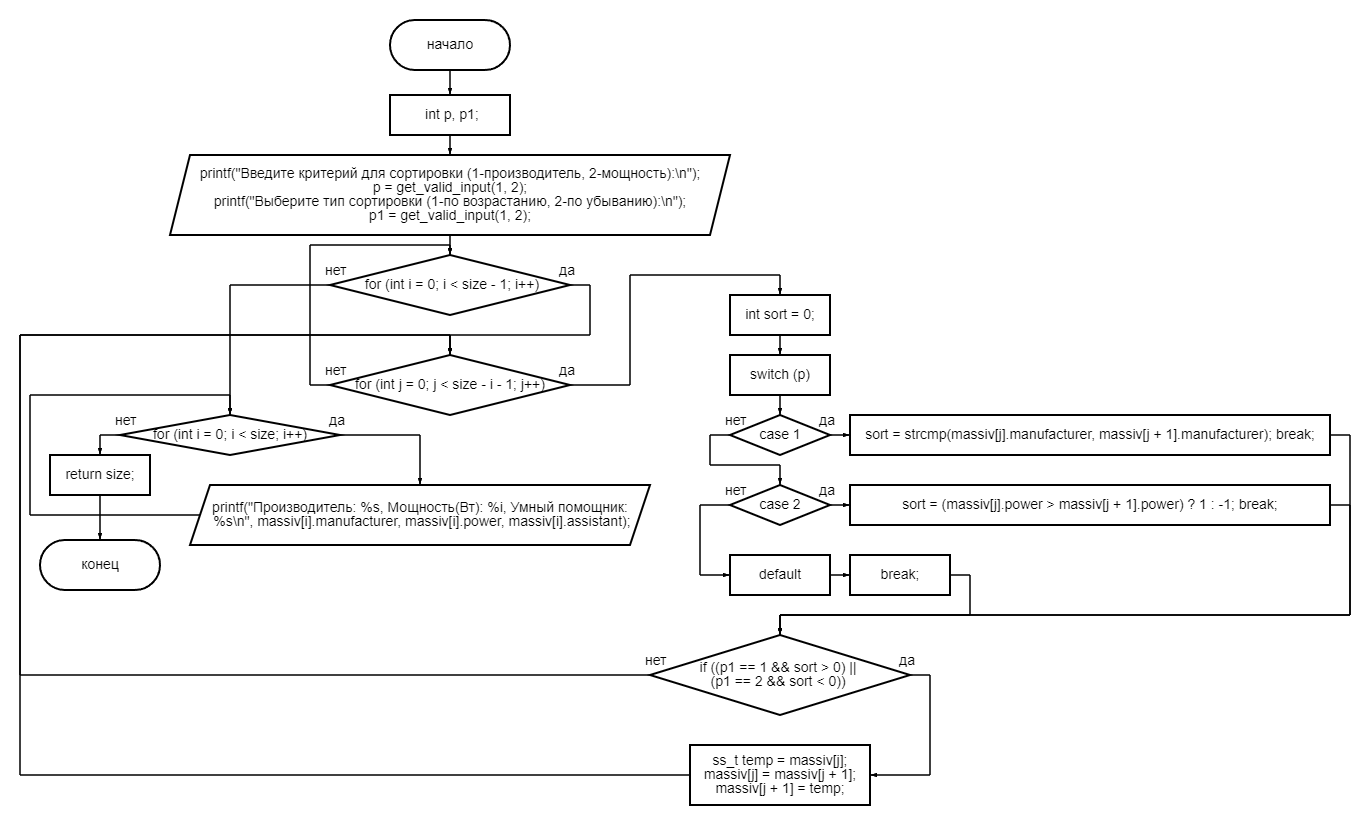


Рисунок 7 – блок-схемы sort\_massiv().

Функция get\_valid\_input(int min, int max) отвечает за получения размера массива с клавиатуры и проверяет правильность ввода данных. Создадим блок-схему для этой функции. (рисунок 8)

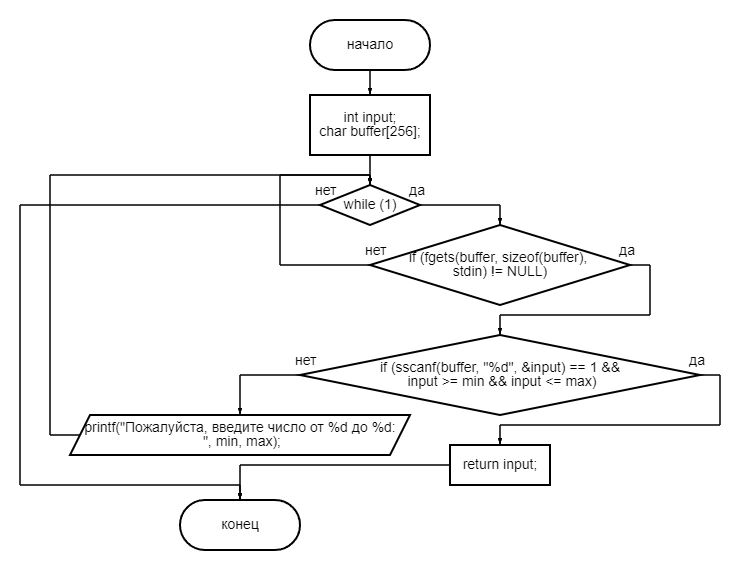


Рисунок 8 – блок-схема get\_valid\_input().

Последняя функция file\_have(const char\* filename) проверяет, существует ли введенный пользователем файл. Создадим блок-схему для это функции. (рисунок 9)

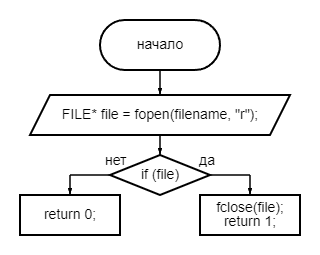


Рисунок 9 – блок-схема file\_have().

Теперь, когда мы расписали каждую функцию в блок-схемах, объединим имеющиеся данные о функциях в таблицу 1.

Таблица 1

|  |  |
| --- | --- |
| Функция | Использование |
| print\_array(ss\_t\* massiv, int size) | Вывод массива на экран. |
| input\_file(char\* fname, ss\_t\* massiv, int size) | Запись массива в файл. |
| fill\_array(ss\_t\* massiv, int size) | Ввод элементов массива пользователем |
| output\_file(char\* fname, ss\_t\* massiv, int size) | Считывание данных массива из файла |
| find\_element(ss\_t\* massiv, int size) | Поиск по полям массива |
| sort\_massiv(ss\_t\* massiv, int size) | Сортировка по полям массива |
| get\_valid\_input(int min, int max) | Ввод размер массива пользователем и проверка правильности ввода |
| file\_have(const char\* filename) | Проверка наличия введенного пользователем файла |

## 3. Тестирование программы

Приступаем к тестированию программы, запустив код, пользователь увидит в консоли меню для выбора, нужно ли создать новый массив или получить данные из файла. (рисунок 10)

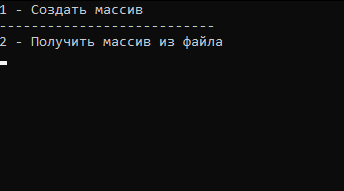


Рисунок 10 – меню начального выбора о способе получения массива.

При выборе любого из вариантов далее пользователю будет предложено ввести размер массива. Если вариант номер 1, то введенный размер массива будет равен количеству умных колонок, которые ему нужно будет описать вручную. При вводе слишком большого размера массива (рисунок 11), а также неверного формата данных (рисунок 12) при создании элементов массива будут выводиться ошибки или указания для верного взаимодействия с программой. Если все данные введены правильно, пользователь попадает в основное меню взаимодействия с массивом. (рисунок 13)

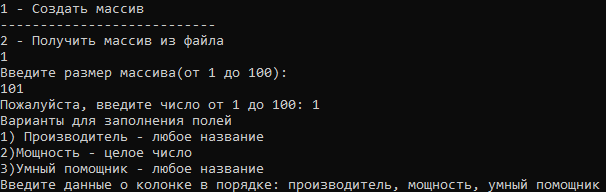


Рисунок 11 – ввод размера массива и возможные ошибки.

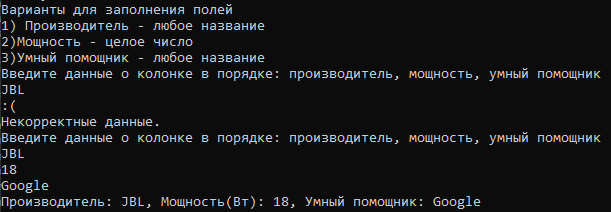


Рисунок 12 – меню заполнение массива элементами и возможная ошибка.

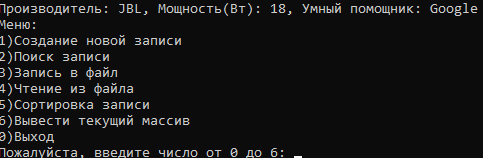


Рисунок 13 – меню управления массивом.

При выборе в начальном меню варианта 2, пользователю так же предложат ввести размер массива, но в этот раз это будет влиять на количество взятых из файла элементов. При вводе слишком большого числа происходит ошибка, аналогичная той, что происходит при выборе варианта 1. После ввода размера у пользователя запрашивается название файла, ошибка выводится при не нахождении данного файла, пользователь попадет в основное меню, (рисунок 13) но в этот момент не существует массива, с которым он мог бы взаимодействовать, поэтому ему либо нужно выбрать пункт 1 и создать новый, либо повторно попытаться считать файл, выбрав пункт 4. Если все данные введены верно, то пользователь попадает в основное меню управления массивом. (рисунок 14)

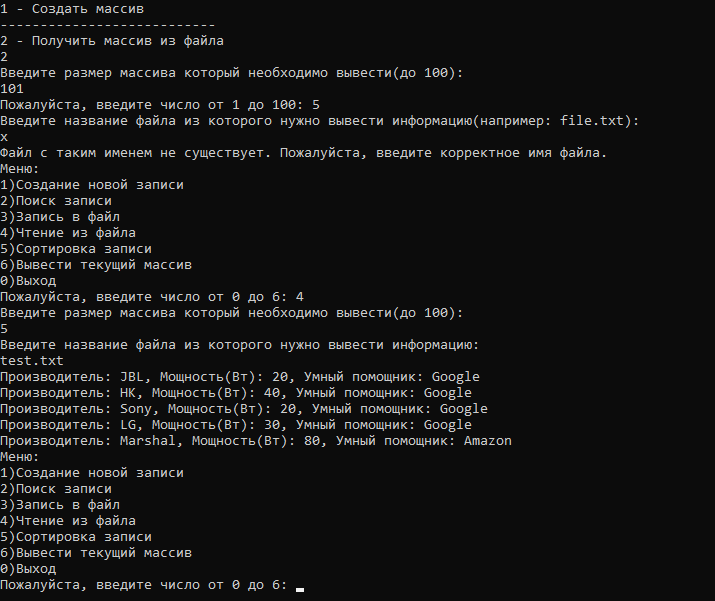


Рисунок 14 – возможные ошибки и правильный ввод при варианте 2.

Любые дальнейшие взаимодействия с массивом будут идентичны, в независимости был он создан вручную или считан из файла. Как уже говорилось ранее пункт 1 и 4 просто возвращают нас к пунктам 1 и 2 из начального меню соответственно. При выборе пункта 2 в основном меню у пользователя сначала спросят по скольким полям он хочет провести поиск, если он вводит 1, то после этого у него уточняется по какому именно полю он хочет провести поиск (1-умный помощник, 2-производитель). Выбрав любое число, ему будет предложено ввести слово, по которому будет проходить поиск. При выборе 2 полей для поиска пользователю будет предложено последовательно ввести искомое слово для каждого поля. При введении цифры больше 2 в предыдущих полях будет выведено сообщение для коррекции ввода пользователя. (рисунок 16) После окончания процесса поиска пользователь вновь попадает в основное меню. (рисунок 15)

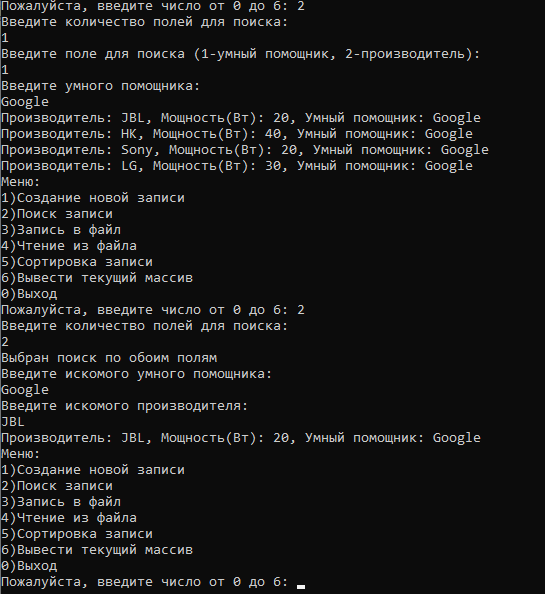


Рисунок 15 – возможные варианты ввода при выборе пункта 2.

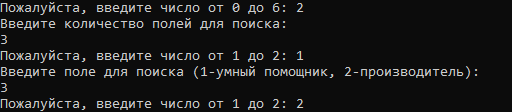


Рисунок 16 – возможные корректировки ввода пользователя.

При выборе пункта 3 пользователю будет предложено ввести называние файла, в который нужно записать уже имеющийся массив. После занесения массива в файл пользователю будет вновь выведен имеющийся массив, и он будет перенаправлен в основное меню. (рисунок 17)

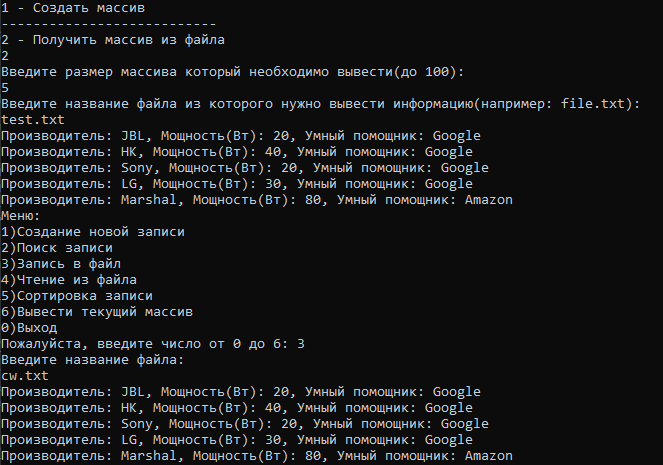


Рисунок 17 – функционал пункта 3.

Выбрав пункт 5, пользователь получит возможность сортировать массив путем, сначала, выбрав поле сортировки (1-производитель, 2-мощность), а потом порядок сортировки (1-по возрастанию, 2-по убыванию). После успешной сортировки будет выведен массив в новом порядке, а пользователь возвращен в меню. (рисунок 18)

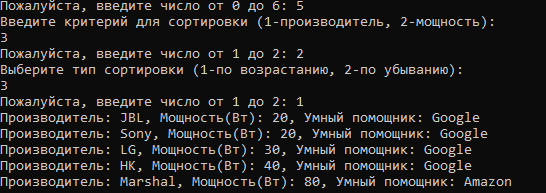


Рисунок 18 – процесс сортировки и возможные корректировки ввода.

При выборе пункта 6 на экран просто будет выведен массив, со всеми изменениями, внесенными в него ранее, (рисунок 19) а при выборе пункта 0 программа завершит работу.

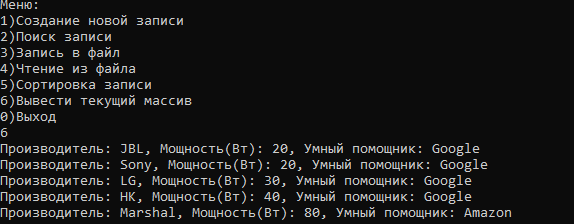


Рисунок 6 – вывод при выборе пункта 6.

# заключение

В ходе выполнения курсового проекта была разработана программа для управления файловой базой данных, которая эффективно решает поставленные задачи. В рамках проекта была выбрана структура данных для хранения записей и определен формат файла, а также создан удобный пользовательский интерфейс. Программа поддерживает сортировку по любому критерию и поиск по двум полям одновременно, что упрощает навигацию. Реализована возможность ручного создания файловой базы данных через программу, а также возможность взаимодействия с уже существующей. В программе учитываются возможные ошибки ввода пользователем, и она помогает их решить. В результате мы имеем систему, позволяющую с простотой взаимодействовать с файловой базой данных на тему «Умные колонки».

GitHub: https://github.com/Tokziccc/tokzic

# Список литературы

1. Курипта О.В. Основы программирования и алгоритмизации: практикум / О.В. Курипта, О.В. Минакова, Д.К. Проскурин. - Воронеж: Воронежский ГАСУ, 2015. - 132 с.
2. Баженова И.Ю. Введение в программирование [Электронный ресурс]: учебное пособие / И.Ю. Баженова, В.А. Сухомлин. - М.: БИНОМ. Лаборатория знаний, Интернет-Университет Информационных Технологий (ИНТУИТ), 2007. - 326 с. - Режим доступа: https://intuit.ru.
3. Microsoft Learn. Справочник по языку С: операторы (С): оператор switch (С) [Электронный ресурс] / Microsoft Learn: официальный сайт. - 2022.

- Режим доступа: https://learn.microsoft.com.

1. Солдатенко И.С. Основы программирования на языке Си: учеб. пособие. - Тверь: Твер. гос. ун-т, 2017. - 159 с.
2. Кнут, Д. Искусство программирования. В 3 т. — М.: Вильямс, 2012.
3. Касаткин, А. И. Профессиональное программирование на языке Си. Управление ресурсами: Справ. пособие. — Мн.: Выш. шк., 1992. — 432 с.
4. Дейт К.Дж. Введение в системы баз данных: 7-е изд. / К.Дж. Дейт. - М.: Вильямс, 2001. - 43 с.
5. Керниган Б., Ритчи Д. Язык программирования Си [Текст] / Пер. с англ., 3-е изд., испр. – СПб.: «Невский Диалект», 2001. – 352 с.
6. Полный справочник по Си: справочник / Шилдт (Herbert Schildt)
7. Павловская, Т. А. С/С++. Программирование на языке высокого уровня. — СПб., 2003. — 461 с.

# ПРИложение

#define \_CRT\_SECURE\_NO\_DEPRECATE

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

struct smart\_speaker {

char manufacturer[20]; // производитель

int power; // мощность

char assistant[50]; // умный помощник

};

typedef struct smart\_speaker ss\_t;

// Программа для работы интерфейса и объединнения всех функций

int main() {

setlocale(LC\_ALL, "RUS");

ss\_t\* massiv = NULL; // Создаем начальный массив

ss\_t\* massiv2 = NULL; // Создаем массив для записи данных из файла

int vibor = 999, vibor1 = 0; // Создаем переменные, использующиеся в выборе вариантов

int size1 = 0, size2 = 0; // Создаем переменные для размеров массивов

char fname1[30] = "", fname2[30] = ""; // Создаем переменные для названий файлов

// получение исходного массива для работы

printf("1 - Создать массив\n---------------------------\n2 - Получить массив из файла\n");

vibor1 = get\_valid\_input(1, 2);

switch (vibor1) {

case 1:

printf("Введите размер массива(от 1 до 100):\n");

size1 = get\_valid\_input(1, 100); // Ограничиваем размер файла

massiv = (ss\_t\*)malloc(size1 \* sizeof(ss\_t));

fill\_array(massiv, size1);

print\_array(massiv, size1);

break;

case 2:

printf("Введите размер массива который необходимо вывести(до 100):\n");

size2 = get\_valid\_input(1, 100);

printf("Введите название файла из которого нужно вывести информацию(например: file.txt):\n");

scanf("%s", fname2);

if (!file\_have(fname2)) {

printf("Файл с таким именем не существует. Пожалуйста, введите корректное имя файла.\n");

break;

}

massiv2 = (ss\_t\*)malloc(size2 \* sizeof(ss\_t));

size2 = output\_file(fname2, massiv2, size2);

massiv = massiv2;

size1 = size2;

print\_array(massiv, size1);

break;

default:

break;

}

// Цикл, содержащий варианты выбора меню

while (vibor != 0) {

printf("Меню:\n1)Создание новой записи\n2)Поиск записи\n3)Запись в файл\n4)Чтение из файла\n5)Сортировка записи\n6)Вывести текущий массив\n0)Выход\n");

vibor = get\_valid\_input(0, 6);

switch (vibor) {

case 1:

// Создание записи

printf("Введите размер массива(от 1 до 100):\n");

size1 = get\_valid\_input(1, 100);

massiv = (ss\_t\*)malloc(size1 \* sizeof(ss\_t));

fill\_array(massiv, size1);

print\_array(massiv, size1);

break;

case 2:

// Поиск по массиву

find\_element(massiv, size1);

break;

case 3:

// Записать имеющийся массив в файл

printf("Введите название файла:\n");

scanf("%s", fname1);

input\_file(fname1, massiv, size1);

print\_array(massiv, size1);

break;

case 4:

// Считать файл

printf("Введите размер массива который необходимо вывести(до 100):\n");

size2 = get\_valid\_input(1, 100);

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;

}

massiv2 = (ss\_t\*)malloc(size2 \* sizeof(ss\_t));

size2 = output\_file(fname2, massiv2, size2);

massiv = massiv2;

size1 = size2;

print\_array(massiv, size1);

break;

case 5:

// Сортировка

sort\_massiv(massiv, size1);

break;

case 6:

// Вывод массива

print\_array(massiv, size1);

break;

default:

break;

}

}

// Освобождение памяти

free(massiv);

free(massiv2);

return 0;

}

// Определение функций

// Вывод массива

int print\_array(ss\_t\* massiv, int size) {

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

printf("Производитель: %s, Мощность(Вт): %i, Умный помощник: %s\n", massiv[i].manufacturer, massiv[i].power, massiv[i].assistant);

}

return size;

}

// Сохранение массива в файл

int input\_file(char\* fname, ss\_t\* massiv, int size) {

FILE\* out = fopen(fname, "wt"); // Ввод названия файла, в который будет записсан массив

if (!out) {

printf("Ошибка открытия файла\n");

return 1;

}

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

fprintf(out, "Производитель: %s Мощность(Вт): %i Умный помощник: %s\n", massiv[i].manufacturer, massiv[i].power, massiv[i].assistant);

}

fclose(out);

return size;

}

// Заполнение массива

int fill\_array(ss\_t\* massiv, int size) {

printf("Варианты для заполнения полей\n1) Производитель - любое название\n2)Мощность - целое число\n3)Умный помощник - любое название\n");

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

int valid\_input = 0; // Проверка до какого момента вводить данные

while (!valid\_input) {

printf("Введите данные о колонке в порядке: производитель, мощность, умный помощник\n");

int result = scanf("%14s %i %14s", massiv[i].manufacturer, &massiv[i].power, massiv[i].assistant);

if (result == 3) {

valid\_input = 1;

printf("Производитель: %s, Мощность(Вт): %i, Умный помощник: %s\n", massiv[i].manufacturer, massiv[i].power, massiv[i].assistant);

}

else {

printf("Некорректные данные.\n");

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

}

}

}

return size;

}

// Считывание файла

int output\_file(char\* fname, ss\_t\* massiv, int size) {

FILE\* in = fopen(fname, "rt"); // Открытие файла, осуществляется через ввод названия

if (!in) {

printf("Ошибка открытия файла\n");

return 1;

}

int count = 0; // Счетчик кол-ва считанных структур

while (count < size && fscanf(in, "Производитель: %14s Мощность: %i Умный помощник: %14s\n", massiv[count].manufacturer, &massiv[count].power, massiv[count].assistant) == 3) {

count++;

}

if (count < size) {

printf("В файле не достаточно данных.\n");

}

fclose(in);

return count;

}

// Функция поиска

int find\_element(ss\_t\* massiv, int size) {

char s[20], s1[50]; // Для введения данных

int n1, n2; // n1 - нужное поле, n2 - кол-в полей

printf("Введите количество полей для поиска:\n");

n2 = get\_valid\_input(1, 2);

if (n2 == 1) {

printf("Введите поле для поиска (1-умный помощник, 2-производитель):\n");

n1 = get\_valid\_input(1, 2);

if (n1 == 1) {

printf("Введите умного помощника:\n");

scanf("%14s", s);

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

if (strcmp(massiv[i].assistant, s) == 0)

printf("Производитель: %s, Мощность(Вт): %i, Умный помощник: %s\n", massiv[i].manufacturer, massiv[i].power, massiv[i].assistant);

}

}

else if (n1 == 2) {

printf("Введите производителя:\n");

scanf("%14s", s);

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

if (strcmp(massiv[i].manufacturer, s) == 0)

printf("Производитель: %s, Мощность(Вт): %i, Умный помощник: %s\n", massiv[i].manufacturer, massiv[i].power, massiv[i].assistant);

}

}

}

else if (n2 == 2) {

printf("Выбран поиск по обоим полям\n");

printf("Введите искомого умного помощника:\n");

scanf("%14s", s);

printf("Введите искомого производителя:\n");

scanf("%14s", s1);

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

if ((strcmp(massiv[i].assistant, s) == 0) && (strcmp(massiv[i].manufacturer, s1) == 0))

printf("Производитель: %s, Мощность(Вт): %i, Умный помощник: %s\n", massiv[i].manufacturer, massiv[i].power, massiv[i].assistant);

}

}

return size;

}

// Функция сортировки

int sort\_massiv(ss\_t\* massiv, int size) {

int p, p1; // p - критерий сортировки, p1 - тип сортировки

printf("Введите критерий для сортировки (1-производитель, 2-мощность):\n");

p = get\_valid\_input(1, 2);

printf("Выберите тип сортировки (1-по возрастанию, 2-по убыванию):\n");

p1 = 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 (p) {

case 1: sort = strcmp(massiv[j].manufacturer, massiv[j + 1].manufacturer); break;

case 2: sort = (massiv[j].power > massiv[j + 1].power) ? 1 : -1; break;

default: break;

}

if ((p1 == 1 && sort > 0) || (p1 == 2 && sort < 0)) {

ss\_t temp = massiv[j];

massiv[j] = massiv[j + 1];

massiv[j + 1] = temp;

}

}

}

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

printf("Производитель: %s, Мощность(Вт): %i, Умный помощник: %s\n", massiv[i].manufacturer, massiv[i].power, massiv[i].assistant);

}

return size;

}

// Ввод лимитов массива и его проверка

int 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("Пожалуйста, введите число от %d до %d: ", min, max);

}

}

}

}

// Проверка наличия файла

int file\_have(const char\* filename) {

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

if (file) {

fclose(file);

return 1;

}

return 0;

}