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 «Корпус ПК»

**Расчетно-пояснительная записка**

Разработал студент А.А.Гаврилов

Подпись, дата Инициалы, фамилия

Руководитель Н.В.Акамсина

Подпись, дата Инициалы, фамилия

Нормоконтролер

Подпись, дата Инициалы, фамилия

Защищена Оценка

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 «Корпус ПК»»

Студент бТИИ-241 Гаврилов Александр Александрович

Группа, фамилия, имя, отчество

База данных «Корпус ПК», Признак поиска: производитель, размер, Вариант сортировки: цвет.

Технические условия Windows 10, Microsoft Visual Studio Code, 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 А.А. Гаврилов

Подпись, дата Инициалы, фамилия

**Содержание**

[Введение 3](#_Toc187817553)

[1. Постановка задачи 4](#_Toc187817554)

[2. Реализация программы 6](#_Toc187817555)

[3. Тестирование программы 12](#_Toc187817556)

[Заключение 19](#_Toc187817557)

[Список источников 20](#_Toc187817558)

[Приложение 21](#_Toc187817559)

## Введение

Файловая база данных – это файл, в котором записи структурированы таким образом, чтобы обеспечить удобство обработки и поиска информации. Для работы с информацией необходимо разработать программу, которая позволяет создавать новые строки, записывать строки в файл, выполнять поиск, осуществлять чтение и сортировку файла. А также выводить диагностические сообщения и уведомлять пользователя об ошибках.

Цель данной работы – разработка программы для работы с записями данных различного типа в предметной области «Корпус ПК». Записи должны содержать в себе информацию о производителе, форм-факторе платы, типоразмере корпуса, высоте и цвете.

Для успешной реализации намеченной цели необходимо поэтапно решить ряд важных задач, каждая из которых важна надежной работы программы:

1. Выбрать наиболее подходящие структуры данных и формат файла.
2. Спроектировать алгоритм действий программы, представив его в виде блок-схемы для наглядности и простоты понимания.
3. Разработать интуитивно понятный интерфейс, обеспечивающий комфортную работу пользователя и предоставляющий ему полный контроль над программой до момента ее завершения.
4. Разработать функции для создания записей, сохранения данных в файле в виде структур, а также для чтения определенного пользователем количества структур из файла.
5. Реализовать функционал для сортировки базы данных и эффективного поиска необходимых данных в ней.
6. Заложить надежные механизмы обработки исключительных ситуаций и предоставления пользователю понятных сообщений об ошибках.

## Постановка задачи

Программа предназначена для работы с записями данных различного типа заданной предметной области «Корпус ПК», содержащимися в файле.

Программа должна предусматривать выполнение следующих функций:

1. создание новой записи (т.е. реализовать возможность заполнения одномерного массива записей (структур) значениями из стандартного потока (клавиатуры);
2. поиск записи по заданным пользователем значениям полей (т.е. реализовать собственные функции для поиска по ключу в соответствии с вариантом по фиксированному полю массива структур)
3. запись и чтение всех данных из файла;
4. печать всех записей, упорядоченных по определённому критерию (указан в столбце «вариант сортировки») на экран;

Интерфейс программы должен обеспечивать следующие возможности:

1. выбор одной из функций программы;
2. ввод значений полей для новой или редактируемой записи;
3. вывод результатов выполнения функции программы.
4. вывод диагностических сообщений в ходе проверки корректности ввода данных.
5. Пользовательский выбор полей записи для упорядочивания массива структур (предусмотрено упорядочивание по всем полям);
6. задание пользователем значения одного из двух поисковых полей или обоих сразу.

Каждая запись в базе данных будет являться структурой «Корпус ПК». Она будет содержать информацию о производителе, форм-факторе платы, типоразмере корпуса, высоте и цвете. Структура содержит в себе следующие данные:

1. Производитель – строка типа char, в которой хранится до 15 символов и содержится информация о производителе корпуса. Данное поле нужно для выполнения поиска.
2. Форм-фактор платы – строка типа char, в которой хранится до 10 символов и содержится информация о типе и размере платы.
3. Типоразмер корпуса – число типа int, которое содержит информацию о типе и размере корпуса. Данное поле нужно для выполнения поиска.
4. Высота – вещественное число типа float, которое содержит информацию о высоте корпуса.
5. Цвет – строка типа char, в которой хранится до 10 символов и содержится информация о цвете корпуса. Данное поле нужно для выполнения сортировки.

## Реализация программы

Для корректного построения программы разработаем алгоритмы каждой функции в виде блок-схем. Функция main() будет координировать все операции с базой данных. В первую очередь создадим блок-схему для main() (рисунок 1).

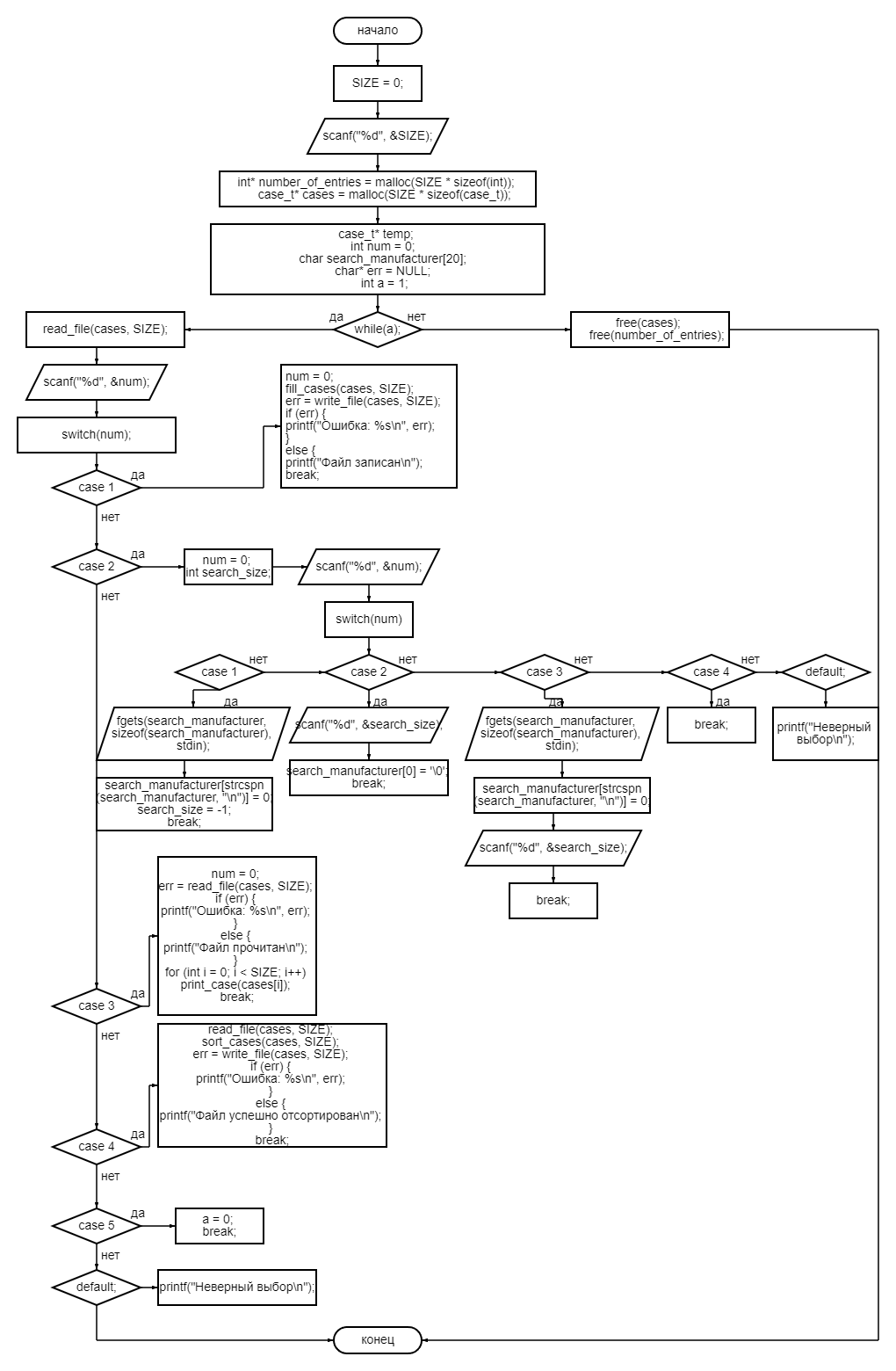


Рисунок 1 – блок-схема функции main()

Функция case\_t\* fill\_cases(case\_t\* cases, int size) отвечает за заполнение структуры, на которую указывает параметр cases, данными. Параметр size определяет количество элементов структуры (рисунок 2).

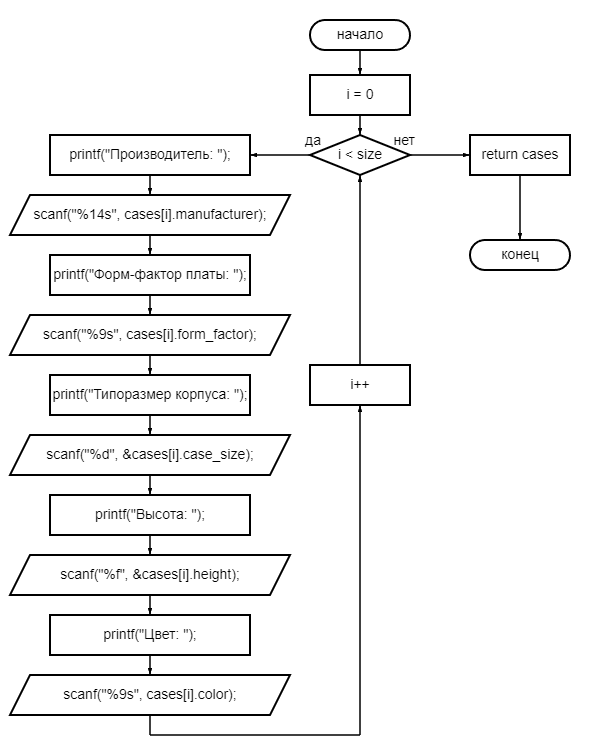


Рисунок 2 – блок-схема функции fill\_cases

Функция void print\_case(case\_t cases) выводит содержимое структуры, переданной в параметре cases, в терминал (рисунок 3).

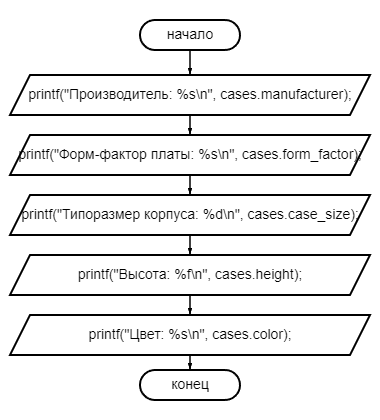


Рисунок 3 – блок-схема функции print\_case

Функция int\* search\_case(case\_t\* cases, int SIZE, char\* manufacturer, int case\_size) выполняет поиск корпусов в массиве структур case\_t на основе критериев: производитель (manufacturer) и размер корпуса (case\_size). Она возвращает массив индексов найденных записей (рисунок 4).

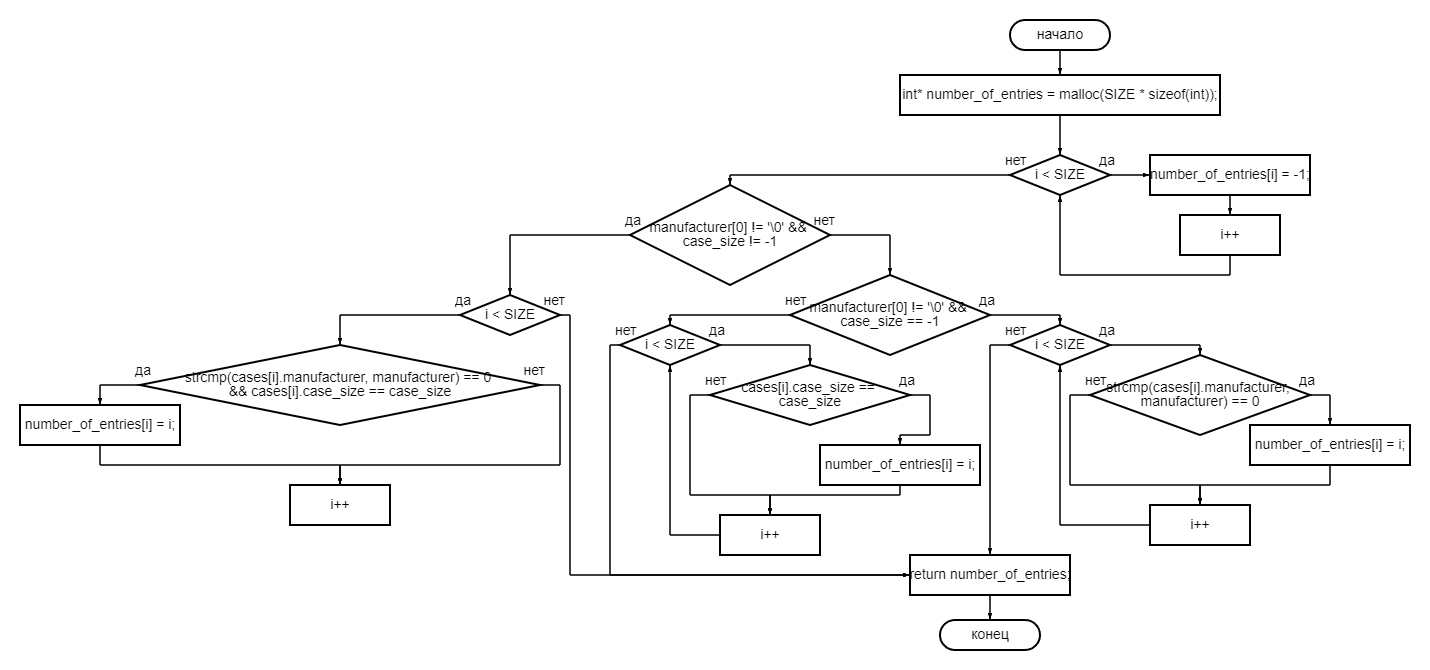


Рисунок 4 – блок-схема функции search\_case

Функция int compare\_color(const void\* a, const void\* b) используется для сравнения двух элементов массива структур case\_t по полю color. Она нужна для того, чтобы передать её в стандартную функцию сортировки qsort, которая использует функции сравнения для упорядочивания элементов (рисунок 5).

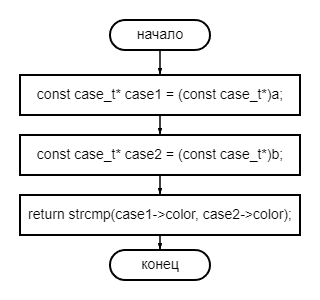


Рисунок 5 – блок-схема функции compare\_color

Функция case\_t\* sort\_cases(case\_t\* cases, int size) сортирует массив структур case\_t по цвету с использованием стандартной функции qsort, которая требует функции сравнения (рисунок 6).

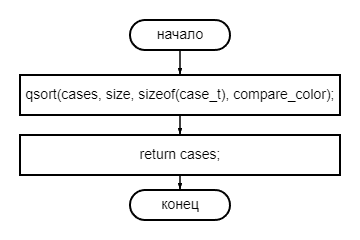


Рисунок 6 – блок-схема функции sort\_cases

Функция char\* write\_file(case\_t\* cases, int size) записывает данные из массива структур case\_t в файл с именем coursework.txt. В случае успешной записи функция возвращает NULL, а при ошибке — строку с описанием проблемы. Если файл не удается открыть, функция возвращает сообщение об ошибке. (рисунок 7).

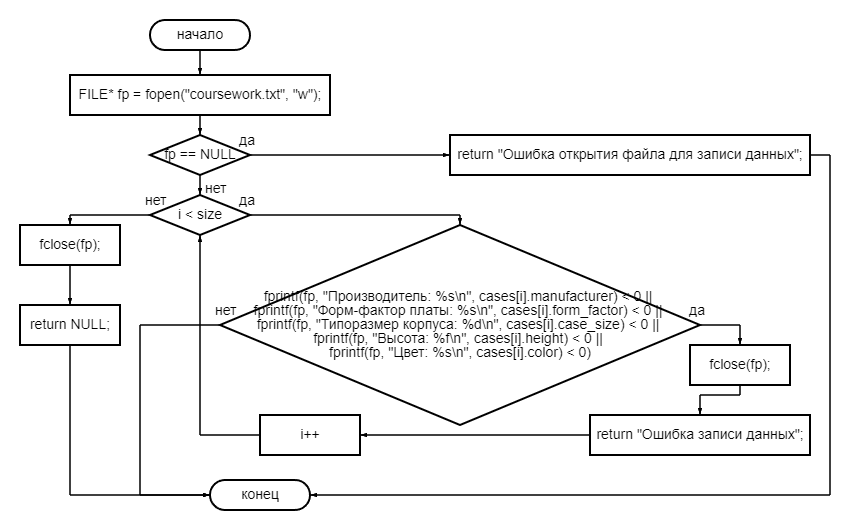


Рисунок 7 – блок-схема функции write\_file

Функция char\* read\_file(case\_t\* cases, int size) читает данные из файла coursework.txt и заполняет массив структур case\_t соответствующими значениями. Если файл не удается открыть функция возвращает ошибку. (рисунок 8).

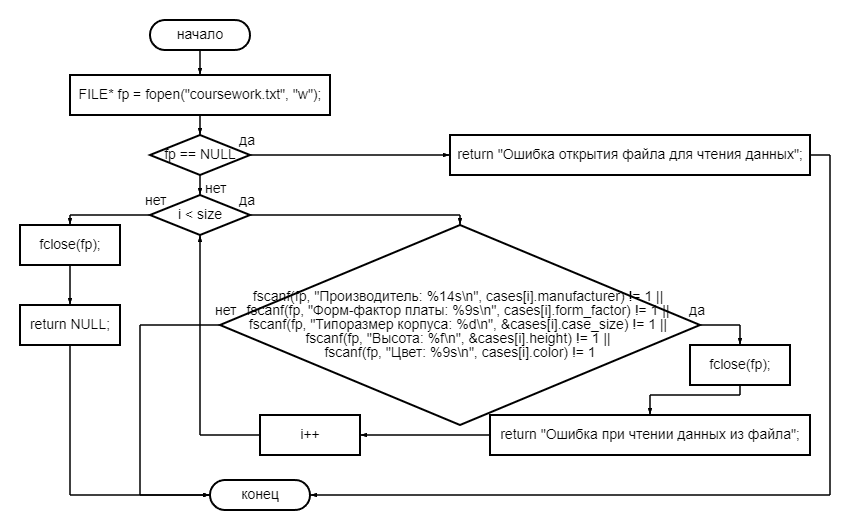


Рисунок 8 – блок-схема функции read\_file

В таблице 1 представлены разработанные функции для работы с массивом, а также их описание.

Таблица 1 – описание собственных функций

|  |  |
| --- | --- |
| Функция | Описание |
| case\_t\* fill\_cases(case\_t\* cases, int size) | Заполняет массив данными. |
| void print\_case(case\_t cases) | Выводит данные в терминал. |
| int\* search\_case(case\_t\* cases, int SIZE, char\* manufacturer, int case\_size) | Осуществляет поиск по указанному параметру в массиве. |
| int compare\_color(const void\* a, const void\* b) | Сравнивает строки с информацией о цвете. |
| case\_t\* sort\_cases(case\_t\* cases, int size) | Выполняет сортировку массива. |
| char\* write\_file(case\_t\* cases, int size) | Записывает данные в файл |
| char\* read\_file(case\_t\* cases, int size) | Выполняет чтение из файла |

## Тестирование программы

Приступаем к тестированию программы. Запустив код, необходимо ввести количество записей (рисунок 9).

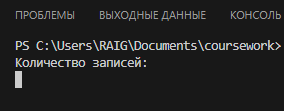


Рисунок 9 – ввод количество записей

Затем в терминале появится меню для выбора функции (рисунок 10).

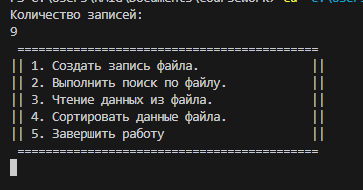


Рисунок 10 – меню для выбора функции

Выберем пункт 1 для записи данных в файл с помощью клавиатуры (рисунок 11).

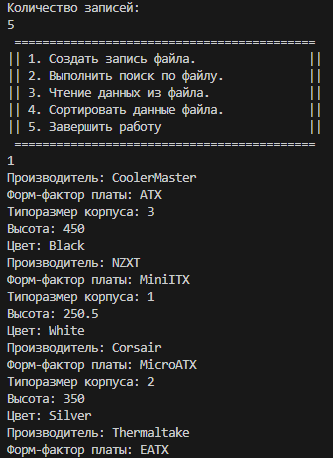


Рисунок 11 – заполнение файла данными

После записи данных появляется диалоговое сообщение о том, что файл записан (рисунок 12).



Рисунок 12 – диалоговое сообщение

Затем перейдем в файл coursework.txt чтобы увидеть результат заполнения (рисунок 13).

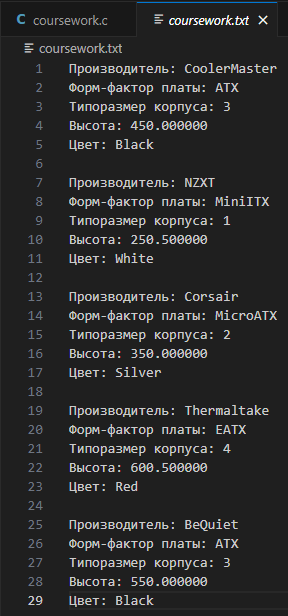


Рисунок 13 – результат заполнения файла

Выбираем пункт 2, чтобы выполнить поиск по файлу. В терминале появляется меню поиска (рисунок 14).

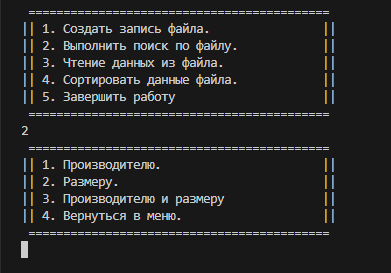


Рисунок 14 – меню поиска

Выполним поиск по производителю. Укажем название производителя и в терминал выводится информация о корпусе (рисунок 15).

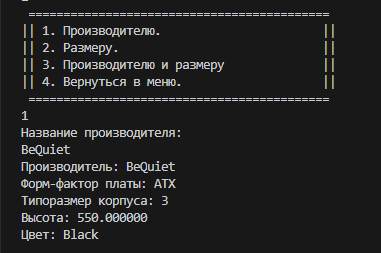


Рисунок 15 – результат поиска по производителю

Выполним поиск по размеру. Укажем типоразмер корпуса и в терминал выводится найденная информация (рисунок 16).

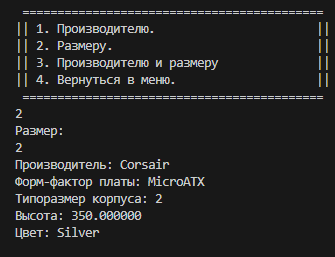


Рисунок 16 – результат поиска по размеру

Выполним поиск по производителю и размеру (рисунок 17).

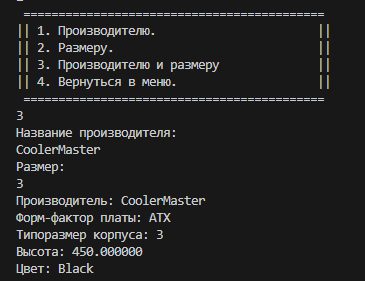


Рисунок 17 – результат поиска по производителю и размеру

Если неверно указать пункт, то появится сообщение об ошибке (рисунок 18).

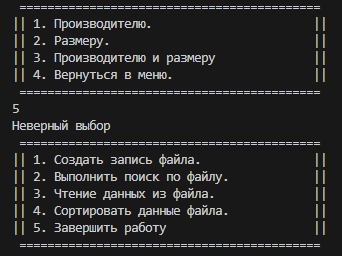


Рисунок 18 – результат неверного выбора пункта

Выполним чтение данных из файла. Для этого выберем пункт 3. В терминале появится диалоговое сообщение о том, что файл прочитан, а затем данные, которые находились в файле (рисунок 19).

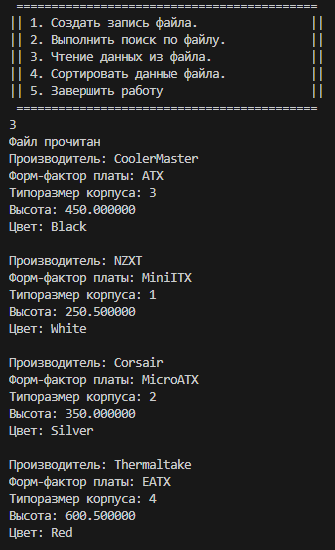


Рисунок 19 – результат чтения данных из файла

Выполним сортировку данных файла. Для этого выберем пункт 4. В терминале появится сообщение о том, что сортировка была выполнена (рисунок 20). Затем выберем пункт 3, чтобы увидеть результат сортировки (рисунок 21).

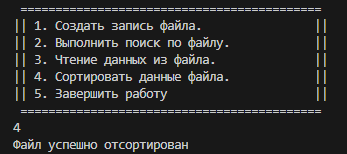


Рисунок 20 – сообщение о сортировке

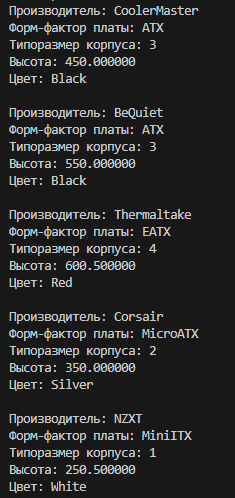


Рисунок 21 – результат сортировки данных

При возникновении ошибки открытия файла в терминале отобразится соответствующее сообщение, которое информирует об ошибке (рисунок 22).

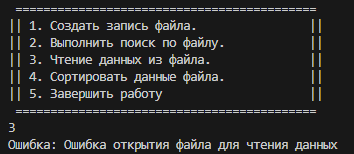


Рисунок 23 – сообщение при ошибке открытия файла

В случае, если файл будет поврежден, появится сообщение об ошибке (рисунок 24).

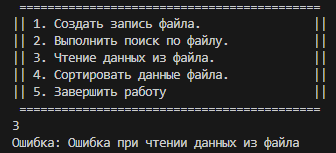


Рисунок 24 – сообщение при ошибке чтения данных

## Заключение

В ходе выполнения курсовой работы была разработана программа, предназначенная для работы с данными предметной области «Корпус ПК». Программа успешно реализует функции создания и хранения записей, обеспечивая эффективное управление файловой базой данных.

Основной функционал программы включает:

* Создание новых записей: пользователь может добавлять новые записи, заполняя все необходимые поля.
* Поиск записей: предусмотрен поиск по значениям полей производителя и размера корпуса, что обеспечивает быстрый доступ к нужной информации.
* Хранение данных: реализованы функции записи данных в файл и их последующего чтения.
* Просмотр данных: вывод всех записей на экран позволяет пользователю просматривать и анализировать данные, в том числе с возможностью сортировки по цвету.

Интерфейс программы разработан с акцентом на удобство использования. Он предлагает интуитивно понятный выбор функций, ввод данных для новых и редактируемых записей, просмотр результатов и информирование пользователя о возможных ошибках ввода.

В результате выполнения курсовой работы создано простое в использовании и функциональное приложение, эффективно решающее поставленные задачи. Программа соответствует заявленным требованиям и может быть применена для работы с данными в заданной предметной области.

Ссылка:

## Список источников

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. Введение в Си. Послание из прошлого столетия: электронный ресурс / <https://habr.com/ru/articles/464075/>
3. Солдатенко И.С. Основы программирования на языке Си: учебное пособие / Тверской государственный университет, 2017 – 159 с.
4. Microsoft Learn. Справочник по языку С: операторы (С): оператор switch (С) [Электронный ресурс] / Microsoft Learn: официальный сайт. - 2022. - Режим доступа: <https://learn.microsoft.com>.
5. Руководство по языку программирования Си: электронный ресурс / <https://metanit.com/c/tutorial/>
6. Керниган Б., Ритчи Д. Язык программирования Си [Текст] / Пер. с англ., 3-е изд., испр. – СПб.: «Невский Диалект», 2001. – 352 с.
7. Дейт К.Дж. Введение в системы баз данных: 7-е изд. / К.Дж. Дейт. - М.: Вильямс, 2001. - 43 с.
8. 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>.
9. Полный справочник по Си: справочник / Шилдт (Herbert Schildt)
10. Неземский В.И. Процедуры и функции: методические указания / В.И. Неземский, О.А. Орешкина. - М.: Московский государственный технический университет имени Н.Э. Баумана, 2009. - 28 с.

## Приложение

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct pc\_case {

    char manufacturer[15];

    char form\_factor[10];

    int case\_size;

    float height;

    char color[10];

} case\_t;

case\_t\* fill\_cases(case\_t\* cases, int size) {

    for (int i = 0; i < size; i++) {

        printf("Производитель: ");

        scanf("%14s", cases[i].manufacturer);

        getchar();

        printf("Форм-фактор платы: ");

        scanf("%9s", cases[i].form\_factor);

        getchar();

        printf("Типоразмер корпуса: ");

        scanf("%d", &cases[i].case\_size);

        getchar();

        printf("Высота: ");

        scanf("%f", &cases[i].height);

        getchar();

        printf("Цвет: ");

        scanf("%9s", cases[i].color);

        getchar();

    }

    return cases;

}

void print\_case(case\_t cases) {

    printf("Производитель: %s\n", cases.manufacturer);

    printf("Форм-фактор платы: %s\n", cases.form\_factor);

    printf("Типоразмер корпуса: %d\n", cases.case\_size);

    printf("Высота: %f\n", cases.height);

    printf("Цвет: %s\n", cases.color);

    printf("\n");

}

int\* search\_case(case\_t\* cases, int SIZE, char\* manufacturer, int case\_size) {

    int\* number\_of\_entries = malloc(SIZE \* sizeof(int));

    for (int i = 0; i < SIZE; i++) {

        number\_of\_entries[i] = -1;

    }

    if (manufacturer[0] != '\0' && case\_size != -1) {

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

            if (strcmp(cases[i].manufacturer, manufacturer) == 0 && cases[i].case\_size == case\_size) number\_of\_entries[i] = i;

        }

    }

    else if (manufacturer[0] != '\0' && case\_size == -1) {

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

            if (strcmp(cases[i].manufacturer, manufacturer) == 0) number\_of\_entries[i] = i;

        }

    }

    else {

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

            if (cases[i].case\_size == case\_size) number\_of\_entries[i] = i;

        }

    }

    return number\_of\_entries;

}

int compare\_color(const void\* a, const void\* b) {

    const case\_t\* case1 = (const case\_t\*)a;

    const case\_t\* case2 = (const case\_t\*)b;

    return strcmp(case1->color, case2->color);

}

case\_t\* sort\_cases(case\_t\* cases, int size) {

    qsort(cases, size, sizeof(case\_t), compare\_color);

    return cases;

}

char\* write\_file(case\_t\* cases, int size) {

    FILE\* fp = fopen("coursework.txt", "w");

    if (fp == NULL) {

        return "Ошибка открытия файла для записи данных";

    }

    for (int i = 0; i < size; i++) {

        if (fprintf(fp, "Производитель: %s\n", cases[i].manufacturer) < 0 ||

            fprintf(fp, "Форм-фактор платы: %s\n", cases[i].form\_factor) < 0 ||

            fprintf(fp, "Типоразмер корпуса: %d\n", cases[i].case\_size) < 0 ||

            fprintf(fp, "Высота: %f\n", cases[i].height) < 0 ||

            fprintf(fp, "Цвет: %s\n", cases[i].color) < 0) {

            fclose(fp);

            return "Ошибка записи данных";

        }

    }

    fclose(fp);

    return NULL;

}

char\* read\_file(case\_t\* cases, int size) {

    FILE\* fp = fopen("coursework.txt", "r");

    if (fp == NULL) {

        return "Ошибка открытия файла для чтения данных";

    }

    for (int i = 0; i < size; i++) {

        if (fscanf(fp, "Производитель: %14s\n", cases[i].manufacturer) != 1 ||

            fscanf(fp, "Форм-фактор платы: %9s\n", cases[i].form\_factor) != 1 ||

            fscanf(fp, "Типоразмер корпуса: %d\n", &cases[i].case\_size) != 1 ||

            fscanf(fp, "Высота: %f\n", &cases[i].height) != 1 ||

            fscanf(fp, "Цвет: %9s\n", cases[i].color) != 1) {

            fclose(fp);

            return "Ошибка при чтении данных из файла";

        }

    }

    fclose(fp);

    return NULL;

}

int main() {

    int SIZE = 0;

    printf("Количество записей: \n");

    scanf("%d", &SIZE);

    int\* number\_of\_entries = malloc(SIZE \* sizeof(int));

    case\_t\* cases = malloc(SIZE \* sizeof(case\_t));

    case\_t\* temp;

    int num = 0;

    char search\_manufacturer[20];

    char\* err = NULL;

    int a = 1;

    while(a) {

        read\_file(cases, SIZE);

        printf(" ===========================================\n");

        printf("|| 1. Создать запись файла.                ||\n");

        printf("|| 2. Выполнить поиск по файлу.            ||\n");

        printf("|| 3. Чтение данных из файла.              ||\n");

        printf("|| 4. Сортировать данные файла.            ||\n");

        printf("|| 5. Завершить работу                     ||\n");

        printf(" ===========================================\n");

        scanf("%d", &num);

        switch(num) {

            case 1:

                num = 0;

                fill\_cases(cases, SIZE);

                err = write\_file(cases, SIZE);

                if (err) {

                    printf("Ошибка: %s\n", err);

                }

                else {

                    printf("Файл записан\n");

                }

                break;

            case 2:

                num = 0;

                int search\_size;

                printf(" ===========================================\n");

                printf("|| 1. Производителю.                       ||\n");

                printf("|| 2. Размеру.                             ||\n");

                printf("|| 3. Производителю и размеру              ||\n");

                printf("|| 4. Вернуться в меню.                    ||\n");

                printf(" ===========================================\n");

                scanf("%d", &num);

                switch(num) {

                    case 1:

                        printf("Название производителя: \n");

                        getchar();

                        fgets(search\_manufacturer, sizeof(search\_manufacturer), stdin);

                        search\_manufacturer[strcspn(search\_manufacturer, "\n")] = 0; //Удаляем символ новой строки

                        search\_size = -1;

                        break;

                    case 2:

                        printf("Размер: \n");

                        scanf("%d", &search\_size);

                        search\_manufacturer[0] = '\0';

                        break;

                    case 3:

                        getchar();

                        printf("Название производителя: \n");

                        fgets(search\_manufacturer, sizeof(search\_manufacturer), stdin);

                        search\_manufacturer[strcspn(search\_manufacturer, "\n")] = 0; //Удаляем символ новой строки

                        printf("Размер: \n");

                        scanf("%d", &search\_size);

                        break;

                    case 4:

                        break;

                    default:

                        printf("Неверный выбор\n");

                }

                number\_of\_entries = search\_case(cases, SIZE, search\_manufacturer, search\_size);

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

                    if (number\_of\_entries[i] != -1) print\_case(cases[i]);

                }

                break;

            case 3:

                num = 0;

                err = read\_file(cases, SIZE);

                if (err) {

                    printf("Ошибка: %s\n", err);

                }

                else {

                    printf("Файл прочитан\n");

                }

                for (int i = 0; i < SIZE; i++)

                    print\_case(cases[i]);

                break;

            case 4:

                read\_file(cases, SIZE);

                sort\_cases(cases, SIZE);

                err = write\_file(cases, SIZE);

                if (err) {

                    printf("Ошибка: %s\n", err);

                }

                else {

                    printf("Файл успешно отсортирован\n");

                }

                break;

                case 5:

                a = 0;

                break;

            default:

                printf("Неверный выбор\n");

        }

    }

    free(cases);

    free(number\_of\_entries);

}