**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 «Ноутбуки»».

**Расчетно-пояснительная записка**

Разработал студент Е.А. Ледовской

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 «Ноутбуки»»

Студент бТИИ-241 Ледовской Егор Андреевич

Группа, фамилия, имя, отчество

База данных «Ноутбуки», Признак поиска: процессор, объем оперативной памяти, Вариант сортировки: производитель, операционная система.

Технические условия Windows 10, Microsoft VisualStudio 2020, язык про граммирования C

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

стр, рисунков, таб, приложение

Сроки выполнения этапов анализ и постановка задачи ;

Разработка пошаговой детализации программы ;

Реализация про граммы ;

Тестирование программы ;

Оформление пояснительной записки .

Срок защиты курсового проекта:

Руководитель Н.В.Акамсина

Подпись, дата Инициалы, фамилия

Задание принял студент Е.А.Ледовской

Подпись, дата Инициалы, фамилия

Замечания руководителя

**Содержание**

[**Введение** 5](#_Toc184928243)

[**Постановка задачи** 6](#_Toc184928244)

[**Реализация программы** 8](#_Toc184928245)

[**Тестирование** 10](#_Toc184928246)

[**Заключение** 19](#_Toc184928247)

[**Список используемых источников** 20](#_Toc184928248)

[**Приложение** 21](#_Toc184928249)

**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, изменят существующие, выполнять поиск по заданным критериям.

Целью данной работы является разработка программы для реализации файловой базы данных с заданным набором функций.

Для достижения поставленной цели, необходимо решить следующие за дачи:

1. Обосновать выбор структуры данных для хранения отдельных за писей 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. Обеспечить выполнение программой функции создания записей.

**Постановка задачи**

Программа предназначена для работы с записями данных различного типа заданной предметной области «Ноутбуки».

Каждая запись должна состоять из не менее пяти полей различных типов данных (но не менее 3).

Название полей по индивидуальному варианту:

* Производитель;
* Операционная система;
* Диагональ экрана;
* Процессор;
* Объем оперативной памяти.

Программа должна предусматривать выполнение следующих функций:

1. создание новой записи;
2. поиск записи по заданным пользователем значениям полей операционной системы и объема оперативной системы;
3. запись и чтение всех данных из файла;
4. печать всех записей, упорядоченных по производителю или процессору на экран;
5. изменение отдельных записей из ранее сохраненных в файле;
6. добавление произвольного числа новых записей в файл базы данных.

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;
6. задание пользователем значения одного из двух поисковых полей или обоих сразу.

**Реализация программы**

Функция notebook\_t\* fill\_notebooks(notebook\_t\* notebooks, int size) –

Заполняет структуру, на которую указывает параметр notebooks данным.

Параметр size указывает на количество заполняемых элементов структуры.

Функция void print\_notebook(notebook\_t notebooks) –

Выводит структуру, на которую указывает параметр notebooks в консоль.

Функция int\* search\_notebook(notebook\_t\* notebooks, int SIZE, char\* CPU, int RAM) – Выполняет поиск по структуре, на которую указывает параметр notebooks. Параметр SIZE указывает на количество элементов этой структуры. Параметры CPU и RAM указывают на значения процессора и объема оперативной памяти, по которым совершается поиск в структуре. Функция возвращает массив чисел, в котором нужные записи структуры помечены значением 0, остальные равны -1.

Функция int compare\_fabricator(const void\* a, const void\* b) – сравнивает элементы массива с ключом по производителю и возвращает число в зависимости от того, какой из производителей в алфавитном порядке стоит первый.

Функция int compare\_OS(const void\* a, const void\* b) – сравнивает элементы массива с ключом по операционной системе и возвращает число в зависимости от того, какая из операционных систем в алфавитном порядке стоит первой.

Функция notebook\_t\* sort\_notebooks(notebook \_t\* notebooks, int size, int criteria)– сортирует элементы структуры , на который указывает параметр notebooks , Параметр size указывает на количество элементов структуры, параметр criteria указывает на выбранный пользователем вид сортировки(Сортировка по производителю ,сортировка по операционной системе).

Функция char\* WriteFile(notebook \_t\* notebooks, int size)– выполняет запись массива структур из параметра notebooks в файл, параметр size указывает на количество записываемых в файл элементов. Возвращаемое значение задается массивом символов и при возникновении ошибки возвращает ее, иначе Null.

Функция char\* ReadFile(notebook\_t\* notebooks, int size)– выполняет чтение из файла текста в массив структур параметра notebooks, параметр size указывает на количество записываемых из файла элементов. Возвращаемое значение задается массивом символов и при возникновении ошибки возвращает ее, иначе Null.

Функция notebook\_t\* AddChangeNotes(notebook\_t\* notebooks, int num) – обновляет массив структур, на который указывает параметр notebooks , Параметр num указывает на индекс записи которой мы хотим обновить.

Функция notebook\_t\* DeleteNotes(notebook\_t\* notebooks, int num, int SIZE) – удаляет структуру из массива, на который указывает параметр notebooks , Параметр num указывает на индекс записи которую мы хотим удалить, параметр SIZE указывает на размерность массива структур, которая изменится при удалении.

**Тестирование**

При первом запуске приложения выберем сколько записей создать или считать из файла.

Выберем пункт 1 «Создать запись файла» и заполнили файл новыми записями.

Результаты представлены на рисунке 1 – 2.

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Рисунок 1 – Процесс заполнение записей через клавиатуру

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 2 – Результат заполнения записей через клавиатуру

Выполним добавление новой записи, для этого зайдем в пункт номер 3 «Запись и чтение файла» и выберем пункт 1 «Добавление записи». Выберем какое количество записей мы хотим добавить. Заполняем с клавиатуры данные о новых записях.

Результаты добавления новых записей представлен на рисунке 3-4

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Рисунок 3 – Процесс добавления новых записей

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рисунок 4 – Результат добавления новых записей в файл

Выполним поиск по названию операционной системы. Выбираем пункт 2 «Выполнить поиск по файлу». Выбираем пункт 1 «Процессору». Вводим название процессору и получаем результат.

Результат представлен на рисунке 5.

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Рисунок 5 – Процесс поиска по процессору.

Выполним аналогичным способом поиск по оперативной памяти и комбинированный поиск. Результаты представлены на рисунках 6–7

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 6 – Процесс поиска по оперативной памяти

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7 – Процесс комбинированного поиска

Выполним обновление записи. Выбираем пункт 3 «Запись и чтение файла», далее выбираем пункт 2 «Изменить запись», далее выбираем номер записи для изменения и заполняем новую запись. Результаты представлены на рисунках 8–9

Изображение выглядит как текст, снимок экрана, черно-белый, монохромный

Автоматически созданное описание

Рисунок 8 – Процесс обновления записи

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рисунок 9 – Результат обновления записи

Выполним удаление записи. Выбираем пункт 3 «Запись и чтение файла», далее выбираем пункт 3 «Удалить запись», далее выбираем номер записи для удаления и заполняем новую запись. Результаты представлены на рисунках 10–11

**Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание**

Рисунок 10 – Процесс удаления записи

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 11 – Результат удаления записи

Выполним чтение записей. Выбираем пункт 3 «Запись и чтение файла», далее выбираем пункт 4 «Чтение файла». Результаты представлены на рисунке 12

Изображение выглядит как текст, снимок экрана, меню, черно-белый

Автоматически созданное описание

Рисунок 12– Результат чтения данных

Выполним сортировку записей. Перед этим изменим записи в файле, чтобы было видно результат сортировки (Рис.13). Выбираем пункт 4 «Сортировать данные файла», далее выбираем какой из способов сортировки применить. Результаты представлены на рисунках 14–15

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 13 – Не отсортированные данные

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 14 – Результат сортировки по производителю

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 15 – Результат сортировки по операционной системе

**Заключение**

В ходе выполнения курсовой работы была разработана программа, предназначенная для работы с записями данных предметной области «Моноблоки». Программа успешно реализует функции создания, хранения, редактирования и замены записей, что позволяет эффективно управлять базой данных.

Основной функционал программы включает в себя:

Создание новых записей с заполнением всех обязательных полей.

Поиск записей по значениям полей излучателей и объема оперативной памяти, что позволяет быстро находить необходимую информацию.

Запись данных в файл и их последующее чтение обеспечивают долговременное хранение информации.

Вывод всех записей на экран приводит к удостоверению производителю или процессору для удобного анализа данных.

Редактирование существующих записей и добавление новых записей в базы данных.

Интерфейс программы разработан с учетом удобства использования. Он предоставляет возможность выбора функций, ввода данных для новых и редактируемых записей, просмотра результатов работы программы, а также информирует пользователя о возможных ошибках ввода данных. Реализована гибкость в работе с записями: пользователь может выбирать критерии сортировки и выполнять поиск по одному или обоим полям одновременно.

В результате выполнения работы создано простое и функциональное приложение, способное эффективно решать поставленные задачи. Программа соответствует заявленным требованиям и может быть использована для работы с данными в заданной предметной области.

<https://github.com/Dankrenn/kursovay.git>

**Список используемых источников**

1. Язык программирования С - авторы: Брайан Керниган, Денис Ричи, 2007, издательство: Вильямс.

2. С. Программирование для профессионалов - автор: Стивен Прата, 2015, издательство: Питер.

3. Как стать профессиональным программистом на C - автор: Майк Микальчик, 2016, издательство: БХВ-Петербург.

4. Современный язык программирования C - автор: Кеннет Н. Кинг, 2007, издательство: Вильямс.

5. С. Библиотека Стандартных Функций - автор: П. Дж. Плажер, 1992, издательство: Питер.

6. Git. Управление версиями - автор: Скотт Чакон, 2014, издательство: Вильямс.

7. Pro Git - авторы: Скотт Чакон, Бен Страуб, 2015, перевод: Галина Панкратова, издательство: Питер.

8. Гит для профессионалов - автор: Андрей Малашенко, 2017, издательство: БХВ-Петербург.

9. Git. Интуитивное руководство - автор: Аарон Мартин, 2015, издательство: Манн, Иванов и Фербер.

10. Версионный контроль с Git - автор: Джон Лоэлигер, 2018, издательство: Питер.

**Приложение**

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

Структура ноутбуки

typedef struct Notebook {

char fabricator[10];

char OS[10];

int RAM;

char CPU[20];

float screenDiagonal;

} notebook\_t;

Функция добавление структур в массив через клавиатуру

notebook\_t\* fill\_ notebooks(notebook\_t\* notebooks, int size) {

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

printf("Производитель моноблока: ");

scanf("%9s", notebooks[i].fabricator);

getchar();

printf("Операционная система: ");

scanf("%9s", notebooks[i].OS);

getchar();

printf("Оперативная память(в ГБ): ");

scanf("%d", & notebooks[i].RAM);

getchar();

printf("Модель процессора: ");

fgets(notebooks[i].CPU, sizeof(notebooks[i].CPU), stdin);

notebooks[i].CPU[strcspn(notebooks[i].CPU, "\n")] = 0;

printf("Диагональ экрана(в дюймах): ");

scanf("%f", & notebooks[i].screenDiagonal);

getchar();

printf("\n");

}

return notebooks;

}

Функция вывода записи в консоль

void print\_ notebook (notebook\_t notebooks) {

printf("Производитель моноблока: %s\n", notebooks.fabricator);

printf("Операционная система: %s\n", notebooks.OS);

printf("Оперативная память(в ГБ): %d\n", notebooks.RAM);

printf("Модель процессора: %s\n", notebooks.CPU);

printf("Диагональ экрана(в дюймах): %f\n", notebooks.screenDiagonal);

printf("\n");

}

Функция поиска записей по операционной системе и объему оперативной памяти

int\* search\_ notebook (notebook \_t\* notebooks, int SIZE, char\* OS, int RAM) {

int\* countOfSeeked = malloc(SIZE \* sizeof(int));

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

countOfSeeked[i] = -1;

}

if (OS[0] != '\0' && RAM != -1) {

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

if (strcmp(notebooks[i].OS, OS) == 0 && notebooks[i].RAM == RAM) countOfSeeked[i] = i;

}

}

else if (OS[0] != '\0' && RAM == -1) {

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

if (strcmp(notebooks[i].OS, OS) == 0) countOfSeeked[i] = i;

}

}

else {

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

if (notebooks[i].RAM == RAM) countOfSeeked[i] = i;

}

}

return countOfSeeked;

}

Вспомогательная функция для сортировки по производителю

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

const notebook\_t\* notebook1 = (const notebook\_t\*)a;

const notebook\_t\* notebook2 = (const notebook\_t\*)b;

return strcmp(notebook1->fabricator, notebook2->fabricator);

}

Вспомогательная функция для сортировки по процессору

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

const notebook\_t\* notebook1 = (const notebook\_t\*)a;

const notebook\_t\* notebook2 = (const notebook\_t\*)b;

return strcmp(notebook1->OS, notebook2->OS);

}

Функция сортировки

notebook\_t\* sort\_notebooks(notebook\_t\* notebooks, int size, int criteria) {

if (criteria == 1) {

qsort(notebooks, size, sizeof(notebook\_t), compare\_fabricator);

}

else if (criteria == 2) {

qsort(notebooks, size, sizeof(notebook\_t), compare\_OS);

}

else if (criteria == 3) {

qsort(notebooks, size, sizeof(notebook\_t), compare\_fabricator);

qsort(notebooks, size, sizeof(notebook\_t), compare\_OS);

}

else {

printf("Ошибка: Неверный критерий сортировки\n");

}

return notebooks;

}

Функция для записи данных в файл

char\* WriteFile(notebook\_t\* notebooks, int size) {

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

if (fp == NULL) {

return "Ошибка открытия файла для записи";

}

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

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

fprintf(fp, "Операционная система: %s\n", notebooks[i].OS) < 0 ||

fprintf(fp, "Оперативная память(в ГБ): %d\n", notebooks[i].RAM) < 0 ||

fprintf(fp, "Модель процессора: %s\n", notebooks[i].CPU) < 0 ||

fprintf(fp, "Диагональ экрана(в дюймах): %f\n\n", notebooks[i].screenDiagonal) < 0) {

fclose(fp);

return "Ошибка записи данных в файл";

}

}

fclose(fp);

return NULL;

}

Функция для чтения данных из файла

char\* ReadFile(notebook\_t\* notebooks, int size) {

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

if (fp == NULL) {

return "Не удалось открыть файл";

}

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

if (fscanf(fp, "Производитель ноутбука: %9s\n", notebooks[i].fabricator) != 1 ||

fscanf(fp, "Операционная система: %9s\n", notebooks[i].OS) != 1 ||

fscanf(fp, "Оперативная память(в ГБ): %d\n", &notebooks[i].RAM) != 1 ||

fscanf(fp, "Модель процессора: %19[^\n]\n", notebooks[i].CPU) != 1 ||

fscanf(fp, "Диагональ экрана(в дюймах): %f\n\n", &notebooks[i].screenDiagonal) != 1) {

fclose(fp);

return "Ошибка при чтении данных из файла";

}

}

fclose(fp);

return NULL;

}

Функция для добавления и обновления записи

notebook\_t\* AddChangeNotes(notebook\_t\* notebooks, int num) {

printf("Производитель ноутбука: ");

scanf("%9s", notebooks[num].fabricator);

getchar();

printf("Операционная система: ");

scanf("%9s", notebooks[num].OS);

getchar();

printf("Оперативная память(в ГБ): ");

scanf("%d", &notebooks[num].RAM);

getchar();

printf("Модель процессора: ");

fgets(notebooks[num].CPU, sizeof(notebooks[num].CPU), stdin);

notebooks[num].CPU[strcspn(notebooks[num].CPU, "\n")] = 0;

printf("Диагональ экрана(в дюймах): ");

scanf("%f", &notebooks[num].screenDiagonal);

getchar();

printf("\n");

return notebooks;

}

Функция для удаления записи

notebook\_t\* DeleteNotes(notebook\_t\* notebooks, int num, int SIZE)

{

int n = num - 1;

int size = SIZE - 1;

notebook\_t\* temp = malloc(size \* sizeof(notebook\_t));

if (temp == NULL) {

printf("Ошибка выделения памяти\n");

return;

}

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

if (i < n) {

temp[i] = notebooks[i];

}

else if (i > n) {

temp[i - 1] = notebooks[i];

}

}

notebooks = realloc(notebooks, size \* sizeof(notebook\_t));

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

notebooks[i] = temp[i];

}

return notebooks;

}

Основная функция для работы с программой

void main() {

setlocale(LC\_ALL, "Rus");

int SIZE = 0;

printf("Введити количество записей: ");

scanf("%d", &SIZE);

int\* countOfSeeked = malloc(SIZE \* sizeof(int));

notebook\_t\* notebooks = malloc(SIZE \* sizeof(notebook\_t));

notebook\_t\* temp;

int num = 0;

char searchProc[20];

char\* error = NULL;

int a = 1;

while (a) {

ReadFile(notebooks, SIZE);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* Menu \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

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

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

printf("\* 3. Запись и чтение 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\_notebooks(notebooks, SIZE);

error = WriteFile(notebooks, SIZE);

if (error) {

printf("Ошибка: %s\n", error);

}

else {

printf("Файл успешно записан\n");

}

break;

case 2:

num = 0;

int searchRAM;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* Поиск по \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* 1. Процессору. \*\n");

printf("\* 2. Оперативной памяти. \*\n");

printf("\* 3. Процессору и оперативной памяти. \*\n");

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

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d", &num);

switch (num) {

case 1:

printf("Введите название процессора: ");

getchar();

fgets(searchProc, sizeof(searchProc), stdin);

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

searchRAM = -1;

break;

case 2:

printf("Введите объем оперативной памяти: ");

scanf("%d", &searchRAM);

searchProc[0] = '\0';

break;

case 3:

getchar();

printf("Введите название процессора: ");

fgets(searchProc, sizeof(searchProc), stdin);

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

printf("Введите объем оперативной памяти: ");

scanf("%d", &searchRAM);

break;

case 4:

break;

}

countOfSeeked = search\_notebook(notebooks, SIZE, searchProc, searchRAM);

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

if (countOfSeeked[i] != -1) print\_notebook(notebooks[i]);

}

break;

case 3:

num = 0;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* Запись и чтение файла \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* 1. Добавить запись. \*\n");

printf("\* 2. Изменить записи. \*\n");

printf("\* 3. Удалить записи. \*\n");

printf("\* 4. Чтение файла. \*\n");

printf("\* 5. Вернуться в меню. \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d", &num);

switch (num) {

case 1:

error = ReadFile(notebooks, SIZE);

if (error) {

printf("Ошибка: %s\n", error);

break;

}

printf("Введите сколько записей вы хотите добавить: ");

num = 0;

scanf("%d", &num);

SIZE += num;

temp = realloc(notebooks, SIZE \* sizeof(notebook\_t));

if (temp == NULL)

{

printf("Ошибка выделения памяти\n");

continue;

}

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

temp[i] = notebooks[i];

}

for (int i = SIZE - num; i < SIZE; i++)

{

AddChangeNotes(temp, i);

}

notebooks = malloc(SIZE \* sizeof(notebook\_t));

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

notebooks[i] = temp[i];

}

error = WriteFile(notebooks, SIZE);

if (error) {

printf("Ошибка: %s\n", error);

}

else {

printf("Запись добавлена и файл обновлён\n");

}

break;

case 2:

num = 0;

printf("Введите номер записи(%d) для её изменения:", SIZE);

scanf("%d", &num);

AddChangeNotes(notebooks, num - 1);

error = WriteFile(notebooks, SIZE);

if (error) {

printf("Ошибка: %s\n", error);

}

else {

printf("Запись обновлена и файл обновлён\n");

}

break;

case 3:

num = 0;

printf("Введите номер записи(%d) для её удаления:", SIZE);

scanf("%d", &num);

DeleteNotes(notebooks, num, SIZE);

SIZE -= 1;

error = WriteFile(notebooks, SIZE);

if (error) {

printf("Ошибка: %s\n", error);

}

else {

printf("Запись удалена и файл обновлён\n");

}

break;

case 4:

error = ReadFile(notebooks, SIZE);

if (error) {

printf("Ошибка: %s\n", error);

}

else {

printf("Файл успешно прочитан\n");

printf("\n");

}

for (int i = 0; i < SIZE; i++) print\_notebook(notebooks[i]);

break;

case 5:

break;

}

break;

case 4:

num = 0;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* Сортировать по \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* 1. Производителю. \*\n");

printf("\* 2. Операционной системе. \*\n");

printf("\* 3. Производителю и операционной системе. \*\n");

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

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d", &num);

ReadFile(notebooks, SIZE);

switch (num) {

case 1:

sort\_notebooks(notebooks, SIZE, num);

WriteFile(notebooks, SIZE);

break;

case 2:

sort\_notebooks(notebooks, SIZE, num);

WriteFile(notebooks, SIZE);

break;

case 3:

sort\_notebooks(notebooks, SIZE, num);

WriteFile(notebooks, SIZE);

break;

case 4:

break;

}

break;

case 5:

a = 0;

break;

default:

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

}

}

free(notebooks);

free(countOfSeeked);

}