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 «Ноутбуки»

Расчетно-пояснительная записка

Разработал студент 28.12.24 Е.А. Ледовской
Подпись, дата Инициалы, фамилия

Руководитель 28.12.24 Н.В.Акамсина
Подпись, дата Инициалы, фамилия

Нормоконтролер _____
Подпись, дата Инициалы, фамилия

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

Защищена 28.12.24 Оценка отлично
дата

Воронеж 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 «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 11, Microsoft VisualStudio 2022, язык программирования С

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

стр, рисунков, таб, приложение

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); оформле- ние пояснительной записки (11.12-14.12.24).

Срок защиты курсового проекта: 28.12.24

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

Задание принял студент 10.09.2024 _____ Е.А.Ледовской
Подпись, дата — Инициалы, фамилия

Замечания руководителя

Содержание

Введение

Файловая база данных представляет собой файл, в котором хранятся упорядоченные записи данных, описывающие заданную предметную область. В данном случае предметная область задана как «Ноутбуки», и каждая запись должна содержать не менее пяти полей различных типов данных: Производитель, Операционная система, Диагональ экрана, Процессор и Объем оперативной памяти.

Целью данной работы является разработка программы для реализации базы данных файлов с заданным набором функций, которые позволят эффективно управлять записями данных о ноутбуках. Для достижения поставленной цели необходимо решить несколько ключевых задач.

Структура данных должна обеспечивать эффективное хранение и быстрый доступ к записям, а формат файла должен быть простым и удобным для чтения и записи данных.

Необходимо реализовать простой и понятный интерфейс для взаимодействия пользователя с программой, который будет работать до тех пор, пока пользователь не захочет выйти из программы. Интерфейс должен быть интуитивно понятным и удобным для пользователя, предоставляя возможность выполнять все необходимые операции без сложностей.

Программа должна позволять пользователю добавлять новые записи о ноутбуках в базу данных, искать записи по определенным критериям, таким как операционная система и объем оперативной памяти, записывать данные в файл и читать их оттуда, обеспечивая сохранность и доступность данных. Пользователь должен иметь возможность просматривать все записи, упорядоченные по определенным критериям, таким как производитель или процессор, редактировать существующие записи, обновляя их данные, а также добавлять несколько новых записей одновременно.

1 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.

Каждая запись в базе данных будет являться структурой «Ноутбуки». Она будет содержать информацию о производителе, операционной системе, оперативной памяти, процессоре и диагонали экрана Структура для хранения данных о ноутбуки содержит:

Производитель – для хранения данных о производителе ноутбука используется массив fabricator[10] типа char. Это позволит хранить строку размером до 10 символов, что достаточно для большинства названий и при этом не расходуется большое количество памяти.

Операционная система – для хранения данных о операционной системе ноутбука используется массив CPU[10] типа char. Это позволит хранить строку размером до 10 символов, что достаточно для большинства названий и при этом не расходуется большое количество памяти

Диагональ экрана – переменная screenDiaogonal типа float необходима для хранения размера экрана в дюймах.

Процессор – для хранения данных о производителе ноутбука используется массив CPU[20] типа char. Это позволит хранить строку размером до 20 символов, что достаточно для большинства названий и при этом не расходуется большое количество памяти.

Оперативная память – переменная RAM типа int необходима для хранения размера оперативной памяти.

Поскольку это переменные различных типов, то для их совместного хранения и использования удобна запись типа структура данных.

Исходная информация представлена в тексовом файле Coursework.txt, в котором каждая строка содержит информацию о производителе, операционной системе, оперативной памяти, процессоре и диагонали экрана Для управления массивом данных необходимо реализовать диалог с пользователем. Блок-схема диалога представлена на рисунке 1.

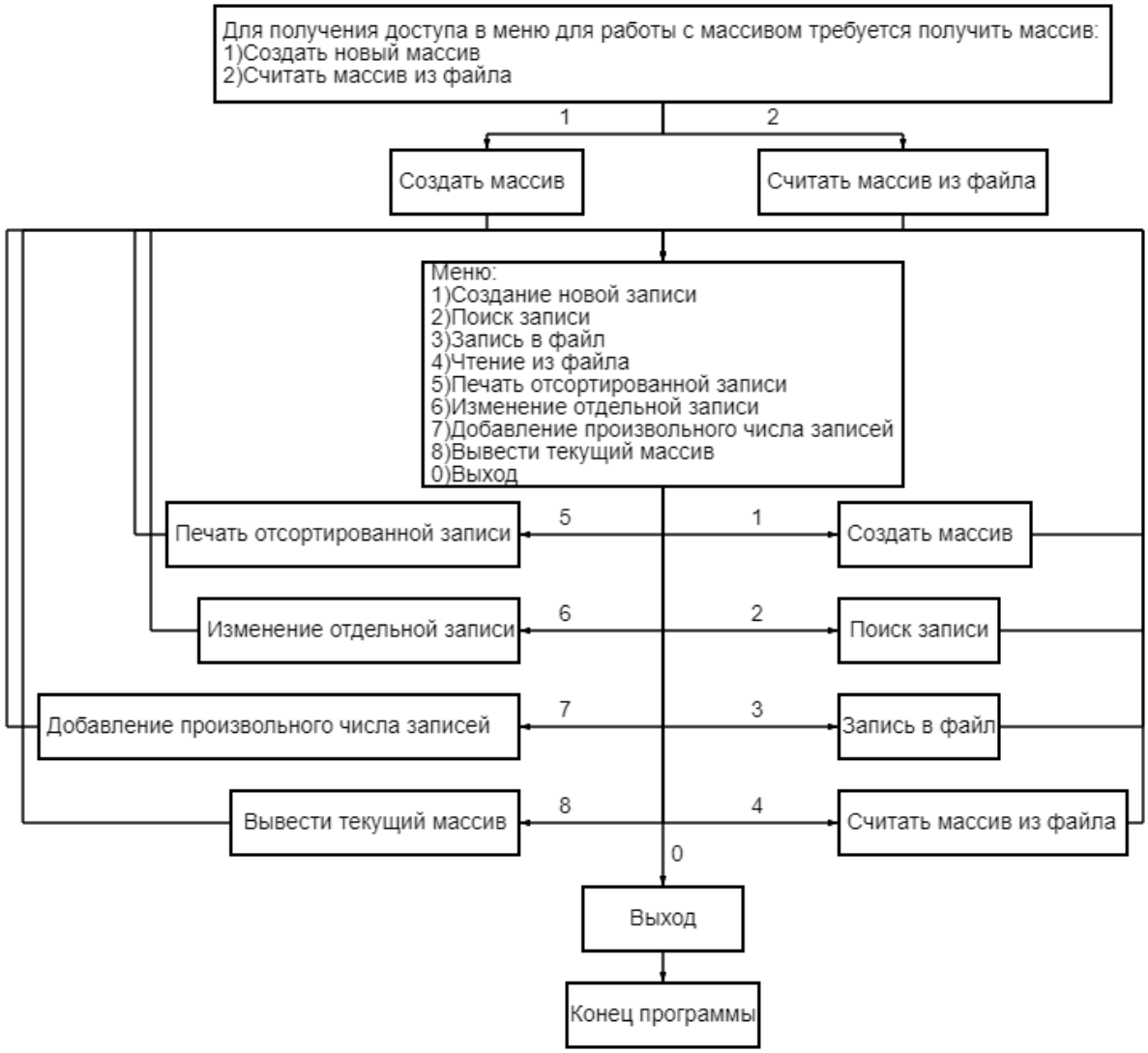

Рисунок 1 – Блок-схема диалога с пользователем

2 Реализация программы

Функция main() – управление всеми действиями и она же является меню общения с пользователем. В ней происходит объявление нужных переменных, диалог с пользователем и вызов других объявленных функций. Блок-схема функции main() представлена на рисунках 2-4.

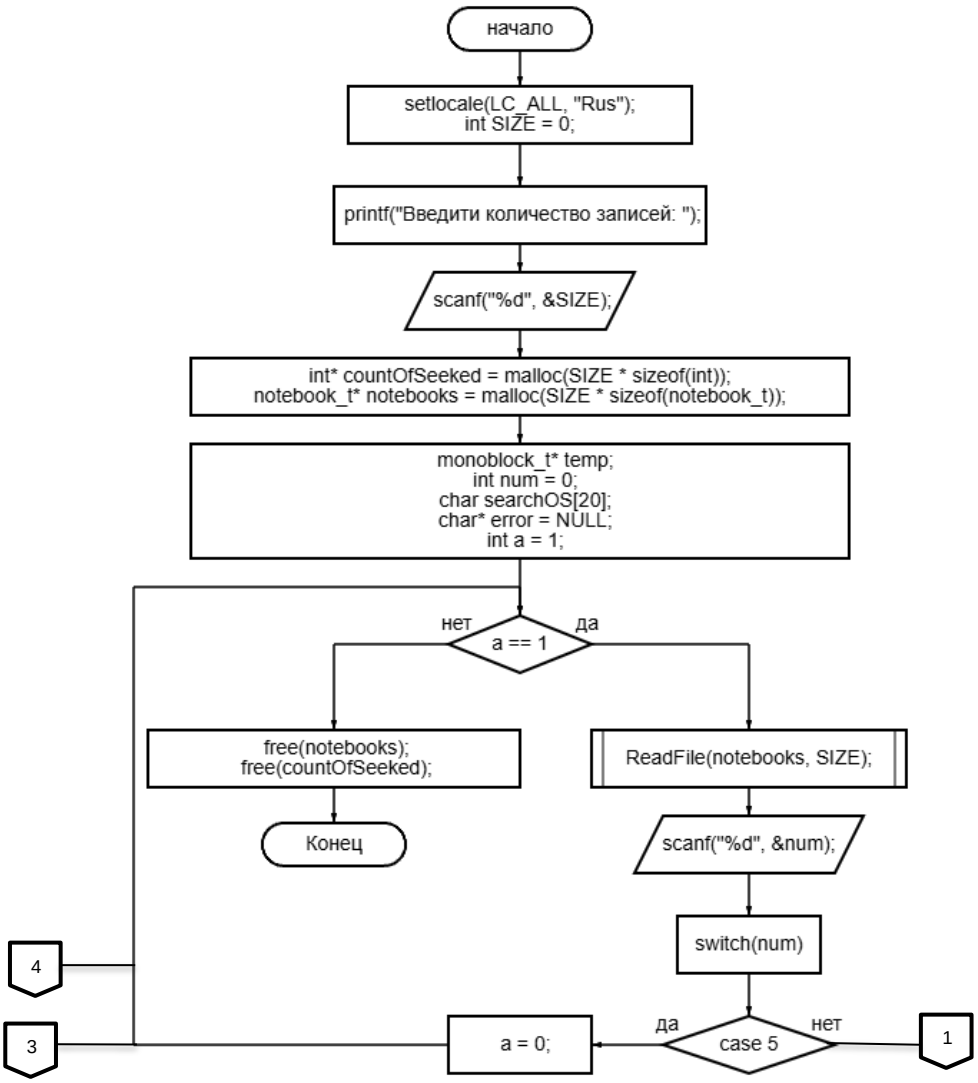

Рисунок 2 - Блок-схема функции main_part1

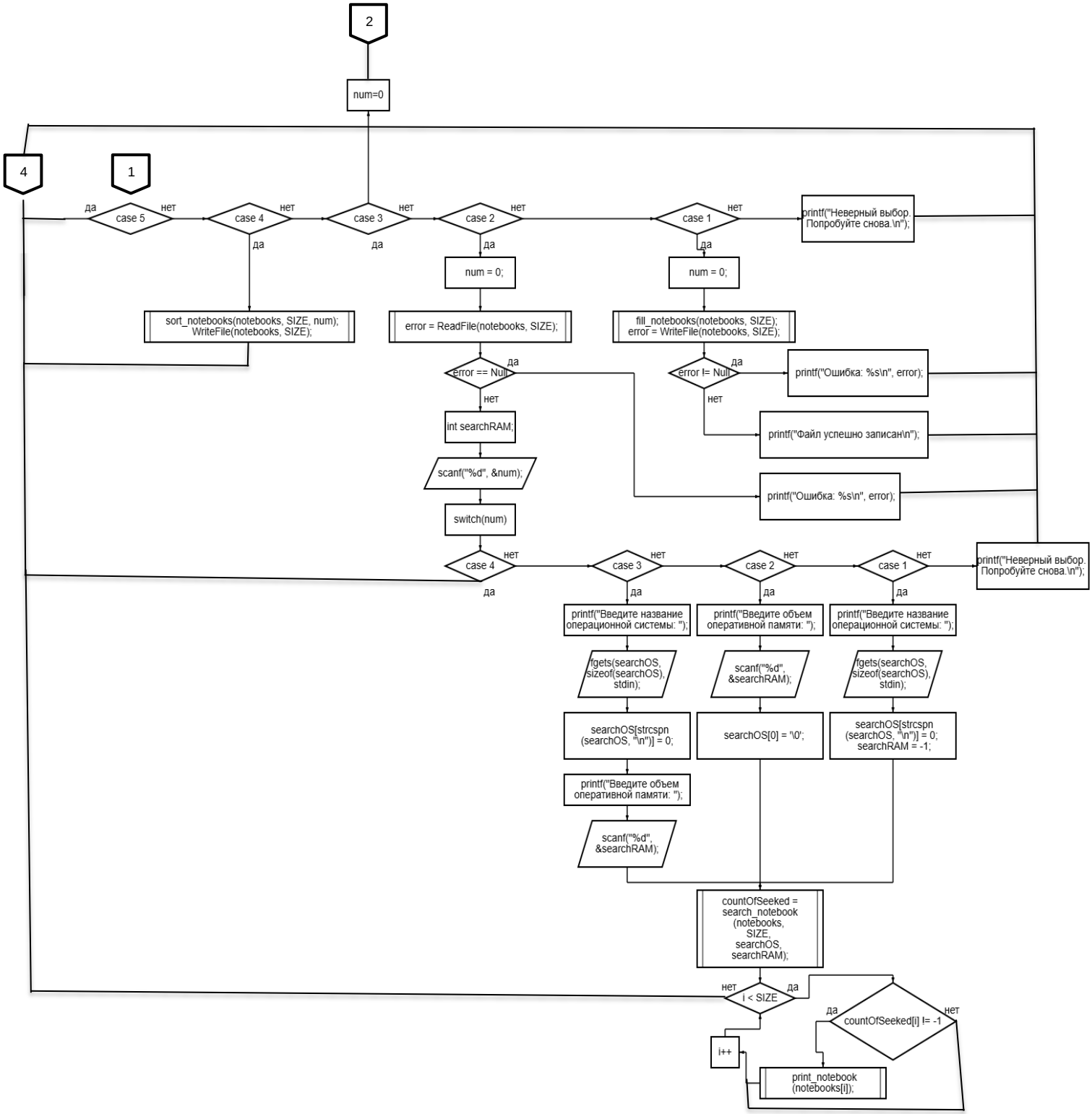

Рисунок 3 - Блок-схема функции main_part2

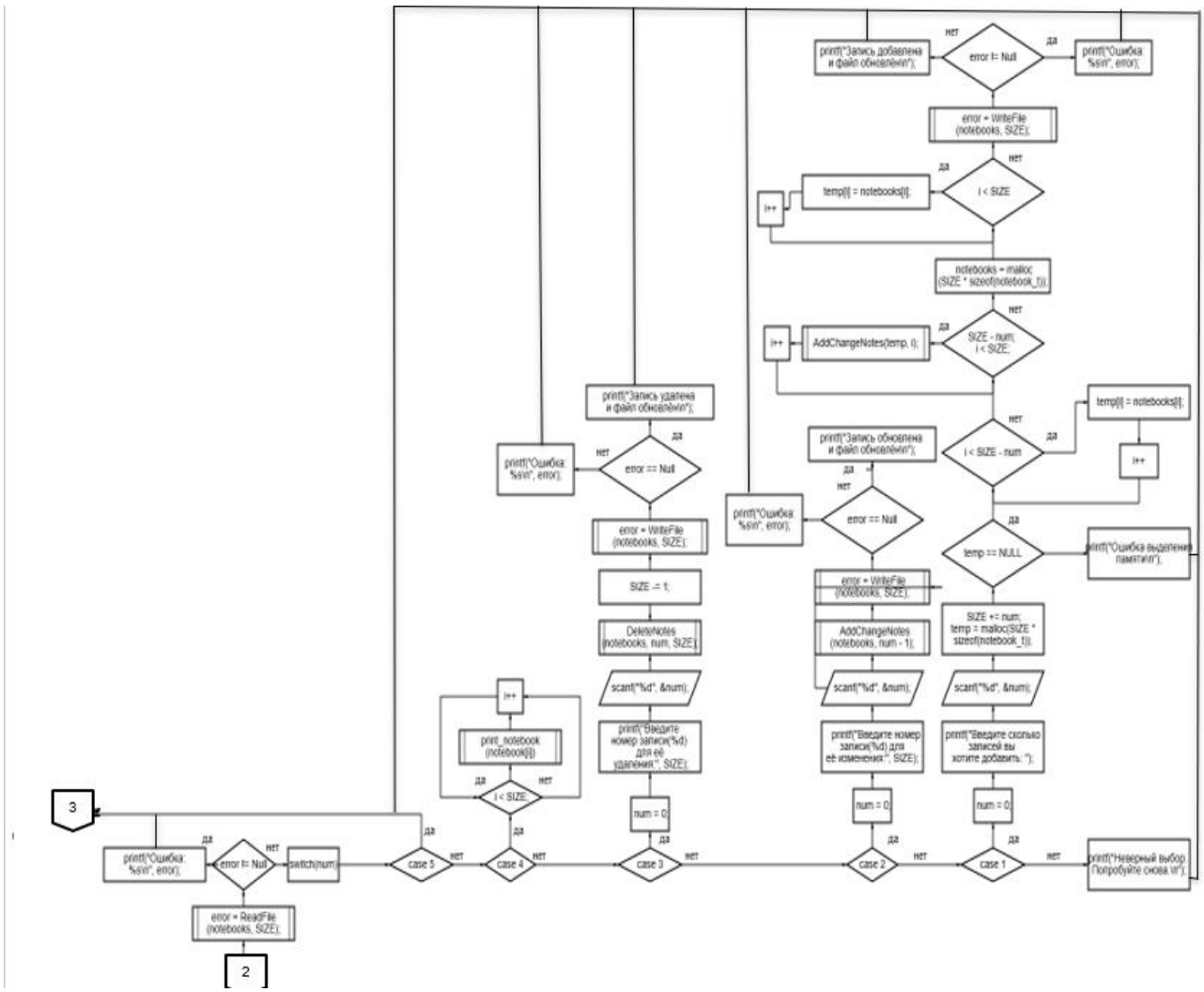

Рисунок 4 - Блок-схема функции main_part3

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. Блок-схема функции представлена на рисунке 5.

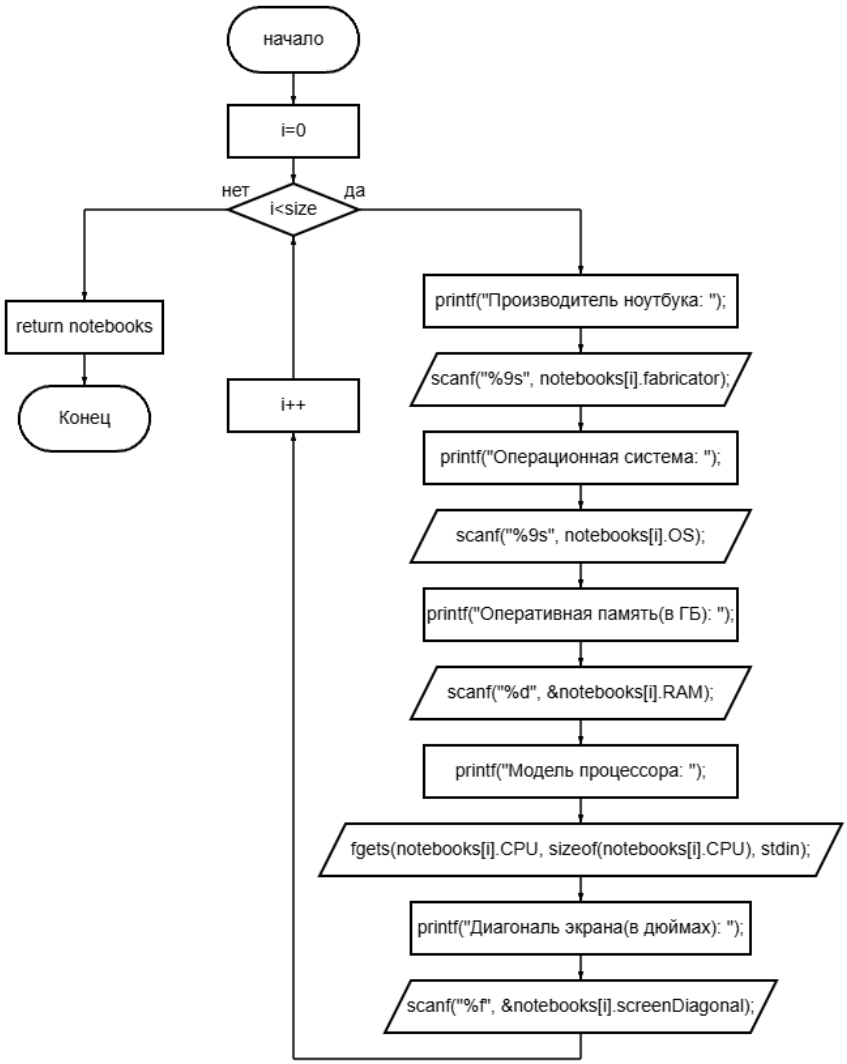

Рисунок 5 - Блок-схема функции fill_notebooks

Функция void print_notebook(notebook_t notebooks) – выводит структуру, на которую указывает параметр notebooks в консоль. Блок-схема функции представлена на рисунке 6.

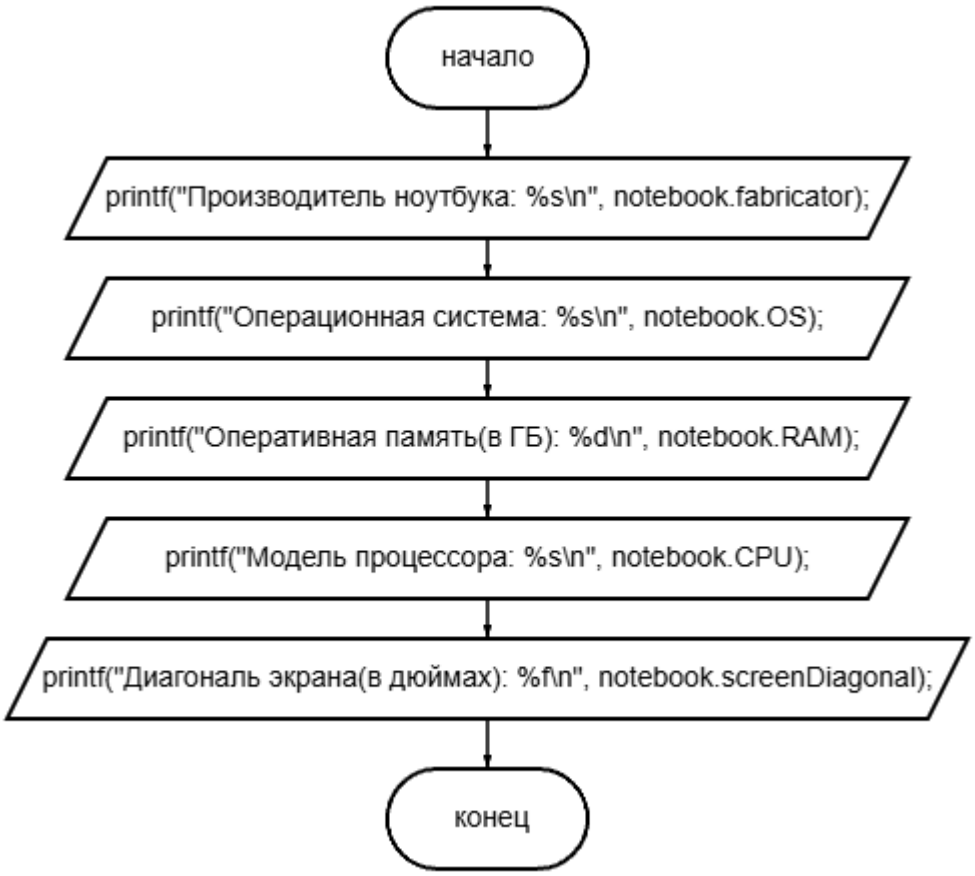

Рисунок 6 – Блок-схема функции print_notebook

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. Блок-схема функции представлена на рисунке 7.

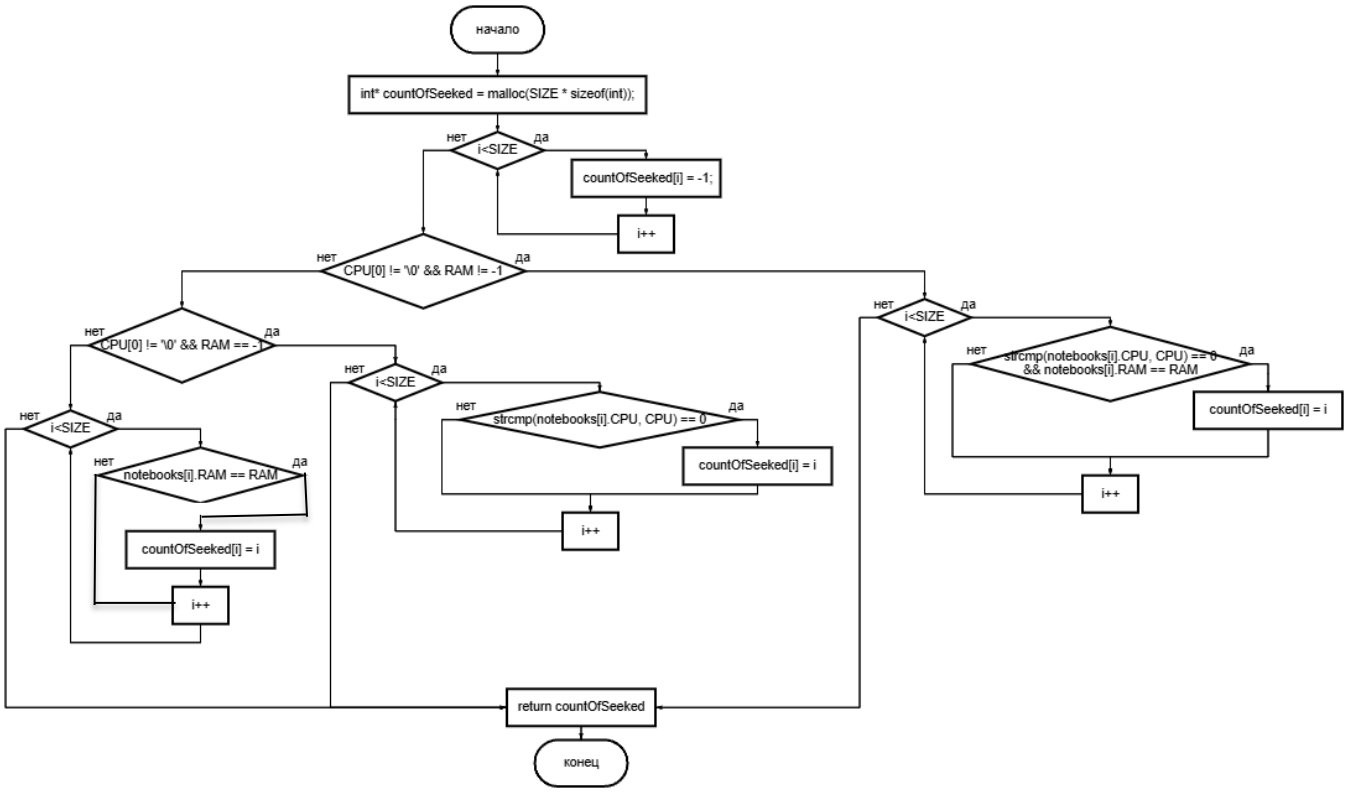

Рисунок 7 – Блок-схема функции search_notebook

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. Блок-схема функции представлена на рисунке 8.

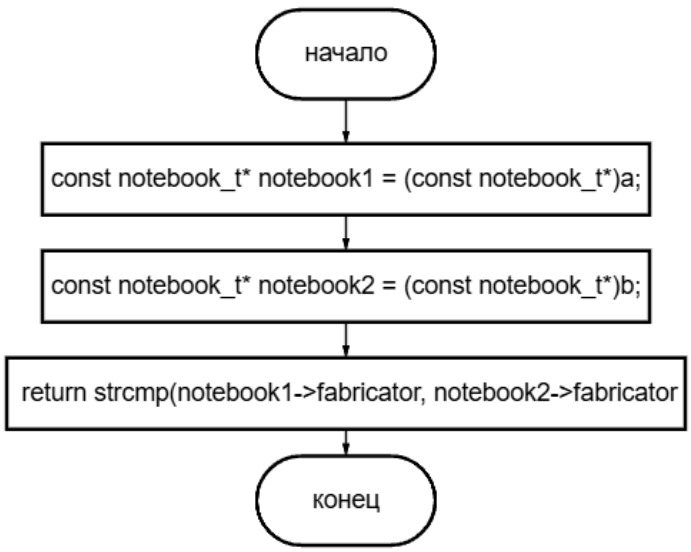

Рисунок 8 – Блок-схема функции compare_fabricator

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. Блок-схема функции представлена на рисунке 9.

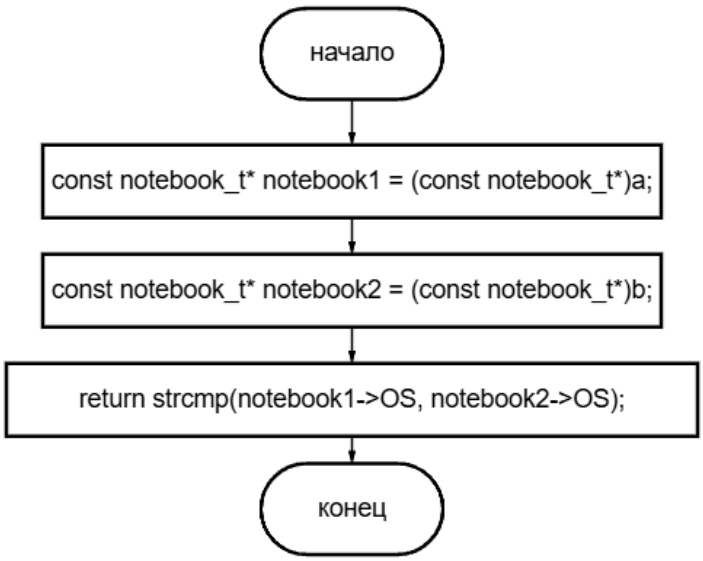

Рисунок 9 – Блок-схема функции compare_OS

Функция int compare_fabricator_and_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. Блок-схема функции представлена на рисунке 10.

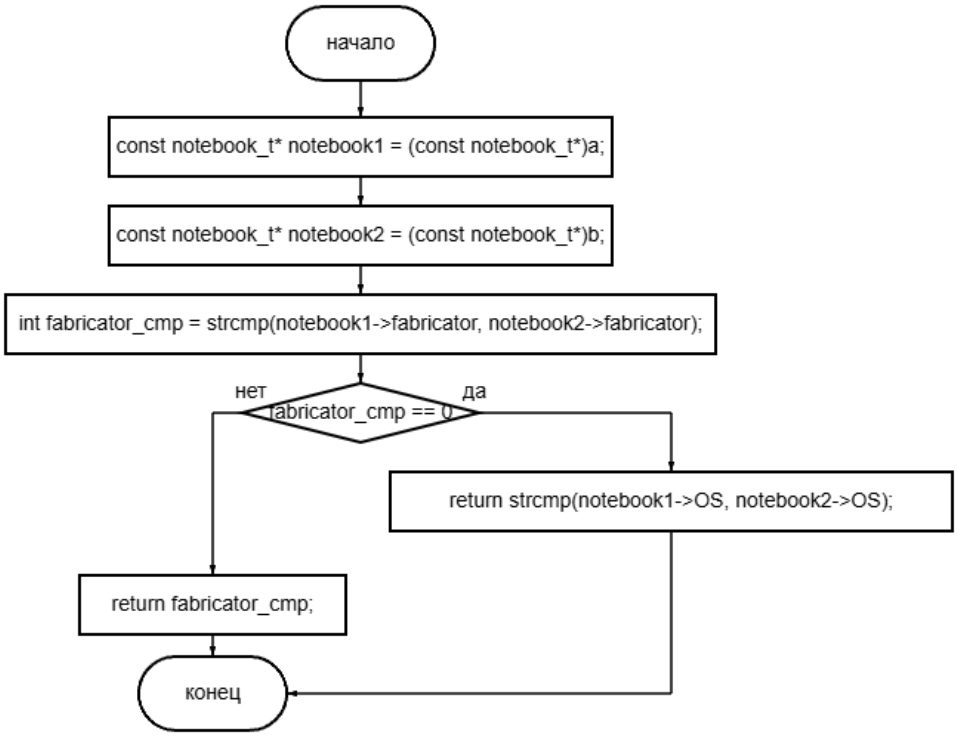

Рисунок 10 - Блок-схема функции compare_fabricator_and_OS

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. Блок-схема функции представлена на рисунке 11.

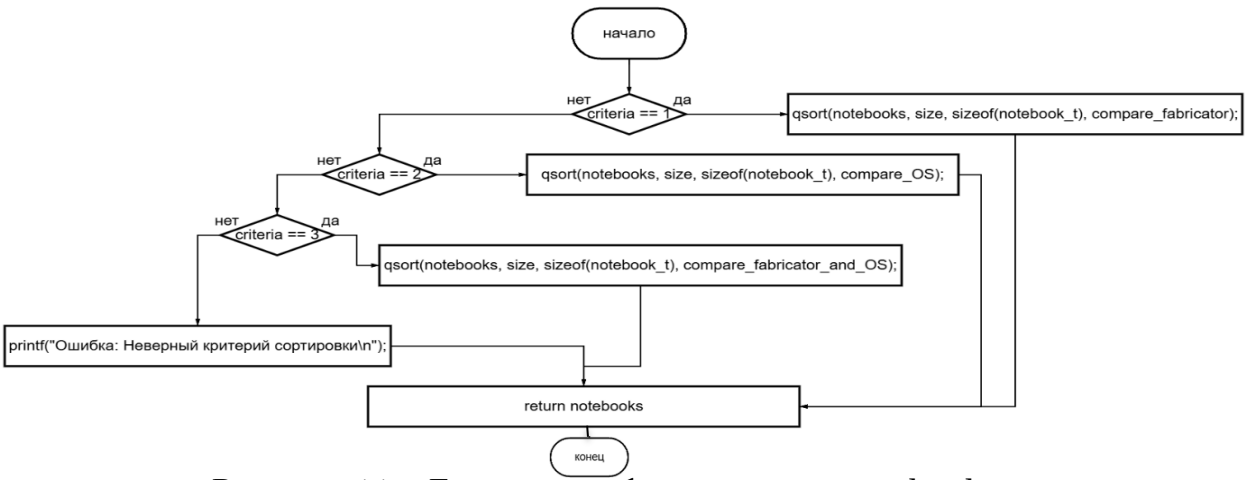

Рисунок 11 – Блок-схема функции sort_notebooks

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. Блок-схема функции представлена на рисунке 12.

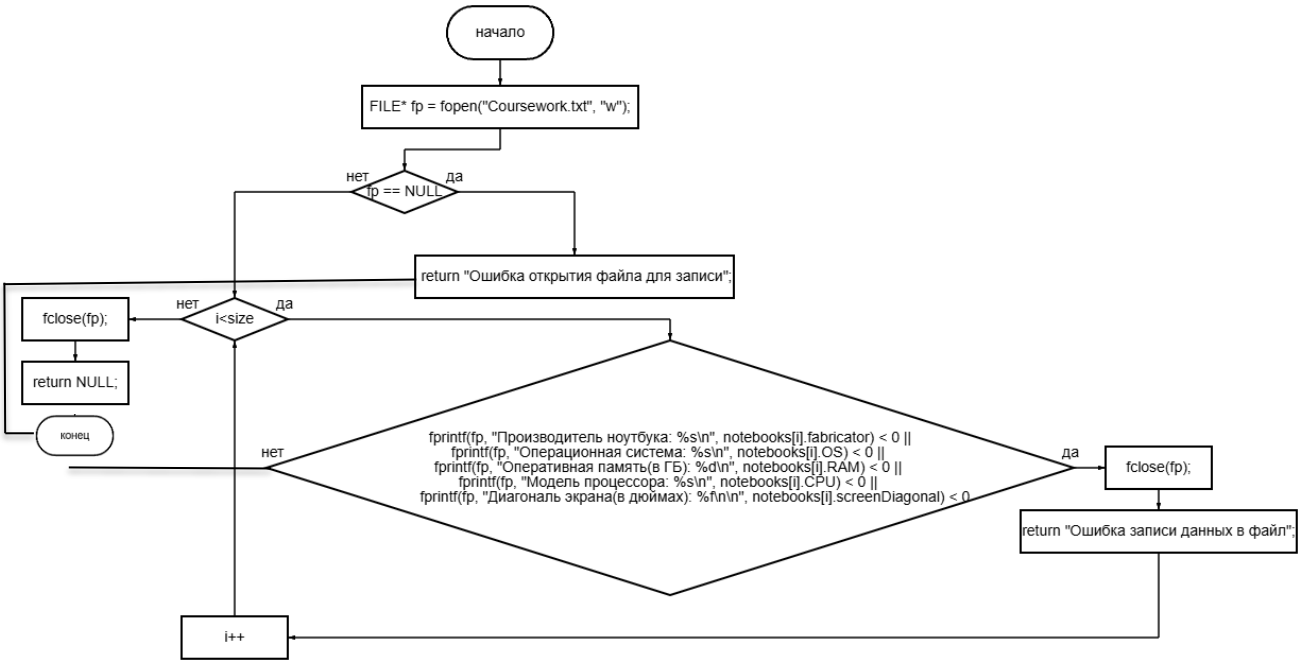

Рисунок 12 – Блок-схема функции WriteFile

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. Блок-схема функции представлена на рисунке 13.

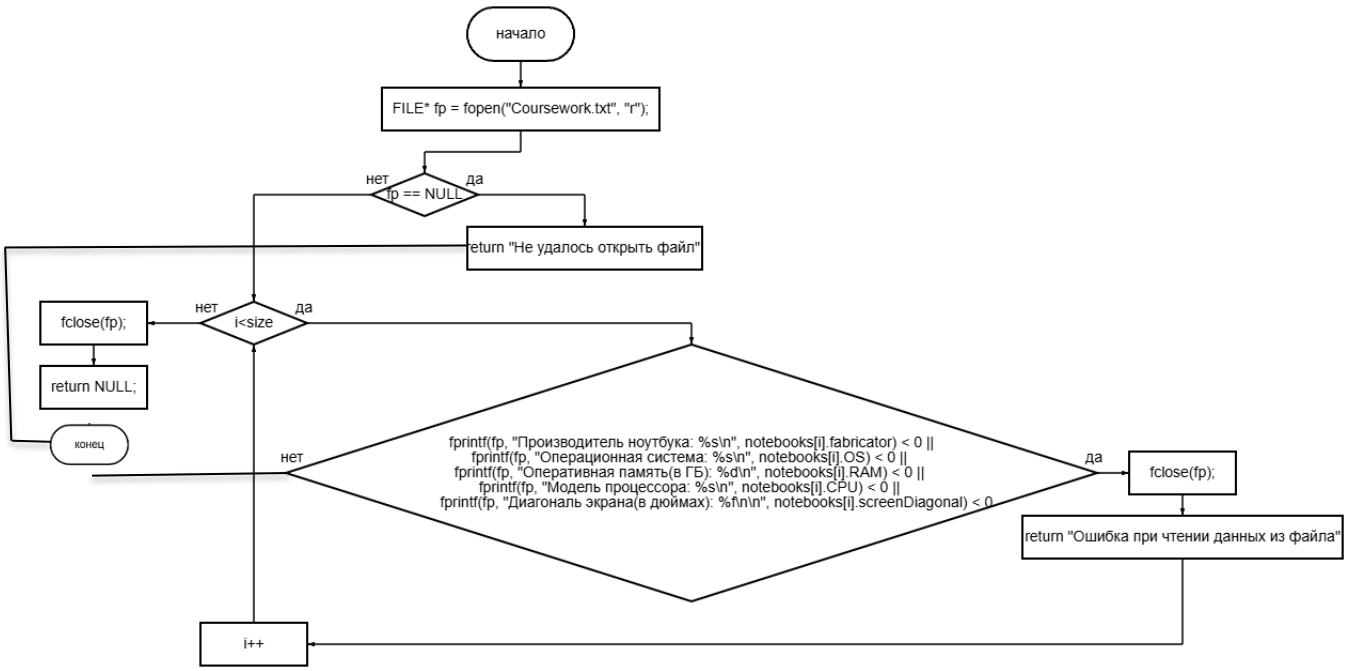

Рисунок 13 – Блок-схема функции ReadFile

Функция notebook_t* AddChangeNotes(notebook_t* notebooks, int num) – обновляет массив структур, на который указывает параметр notebooks , Параметр num указывает на индекс записи которой хотим обновить. Блок-схема функции представлена на рисунке 14.

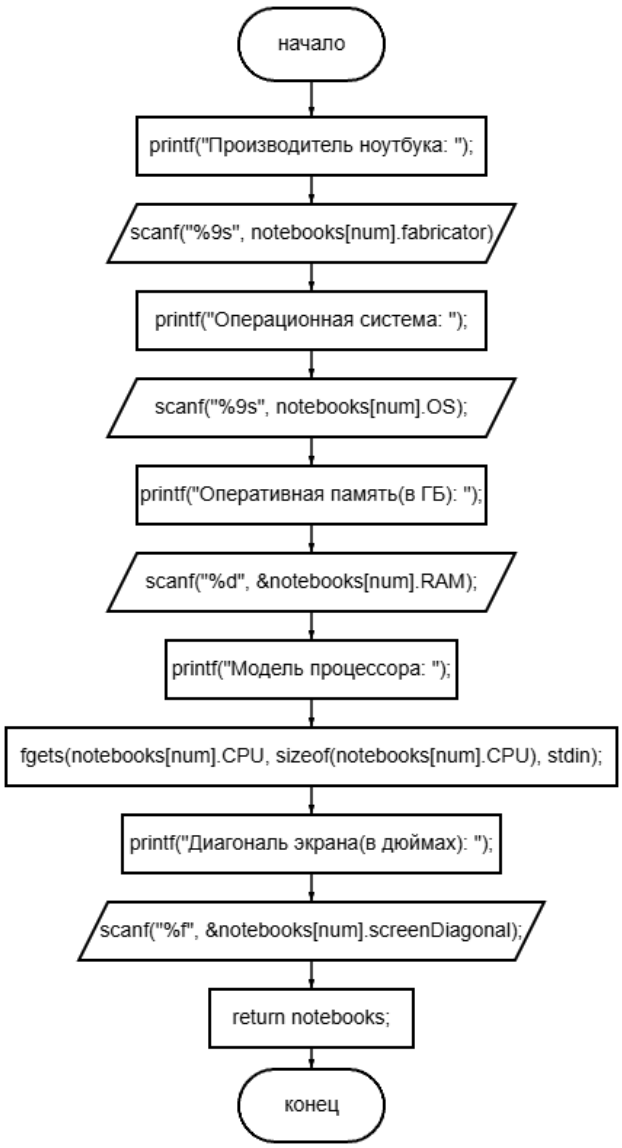

Рисунок 14 – Блок-схема функции AddChangeNotes

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 хотим удалить, параметр SIZE указывает на размерность массива структур, которая изменится при удалении. Блок-схема функции представлена на рисунке 15.

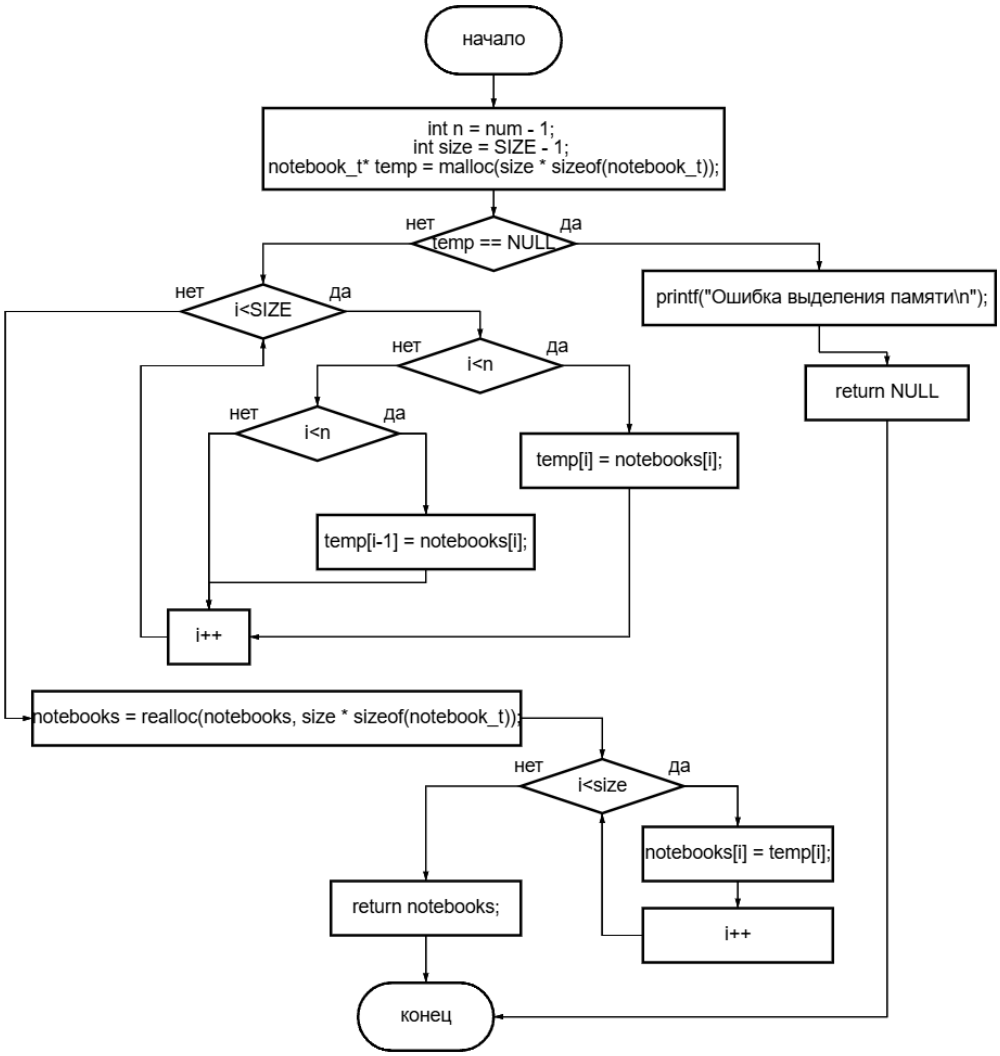

Рисунок 15 – Блок-схема функции DeleteNotes

В таблице 1 представлены собственные функции, которые были введены для работы с массивом, и описано их назначение.

Таблица 1 – Описание собственных функций

Функции	Назначение
notebook_t* fill_notebooks(notebook_t* notebooks, int size)	Заполняет массив данными пользователя
void print_notebook(notebook_t notebooks)	Выводит все данные из массива в консоль

Продолжение таблицы 1

int* search_notebook(notebook_t* notebooks, int SIZE, char* OS, int RAM)	Ищет в массиве данные по указанному параметру и выводит при положительном результате
int compare_fabricator	Сравнивает элементы массива с ключом по производителю
int compare_OS	Сравнивает элементы массива с ключом по операционной системе
int compare_fabricator_and_OS	Сравнивает элементы массива с ключом по производителю и операционной системе
notebook_t* sort_notebooks(notebook_t* notebooks, int size, int criteria)	Сортирует массив по критерию, указанному пользователем
char* WriteFile(notebook_t* notebooks, int size)	Выполняет запись массива структур из параметра notebooks в файл
char* ReadFile(notebook_t* notebooks, int size)	Выполняет чтение из файла текста в массив структур параметра notebooks
notebook_t* AddChangeNotes(notebook_t* notebooks, int num)	Добавляет в массив произвольное количество записей
notebook_t* DeleteNotes(notebook_t* notebooks, int num, int SIZE)	Удаляет в массиве определенную запись

3 Тестирование

При первом запуске приложения выберем количество записей, которые необходимо создать (рисунок 16).

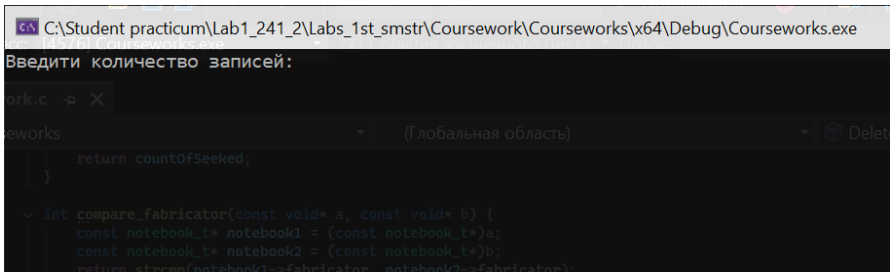

Рисунок 16 – Количество создания записей

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

Результаты представлены на рисунке 17-18.

```
Введити количество записей: 2
*********************************************
*                    Menu                   *
*********************************************
* 1. Создать запись файла.                  *
* 2. Выполнить поиск по файлу.              *
* 3. Запись и чтение файла.                 *
* 4. Сортировать данные файла.              *
* 5. Завершить работу                       *
*********************************************
1
Производитель ноутбука: asus
Операционная система: Windows
Оперативная память(в ГБ): 4
Модель процессора: Intel core i3
Диагональ экрана(в дюймах): 15

Производитель ноутбука: apple
Операционная система: MacOs
Оперативная память(в ГБ): 8
Модель процессора: ryzen 3600
Диагональ экрана(в дюймах): 18

Файл успешно записан
*********************************************
*                    Menu                   *
*********************************************
* 1. Создать запись файла.                  *
* 2. Выполнить поиск по файлу.              *
* 3. Запись и чтение файла.                 *
* 4. Сортировать данные файла.              *
* 5. Завершить работу                       *
*********************************************
```

Рисунок 17 – Процесс заполнение записей через клавиатуру

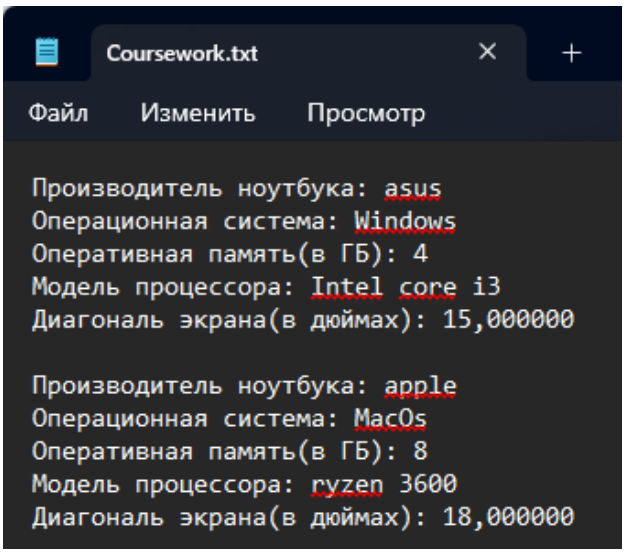

Рисунок 18 – Результат заполнения Coursework.txt

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 хотим добавить. Заполняем с клавиатуры данные о новых записях.

Результаты добавления новых записей представлен на рисунке 19-20.

```
*********************************************
*                  Menu                     *
*********************************************
* 1. Создать запись файла.                  *
* 2. Выполнить поиск по файлу.              *
* 3. Запись и чтение файла.                 *
* 4. Сортировать данные файла.              *
* 5. Завершить работу                       *
*********************************************
3
*********************************************
*            Запись и чтение файла          *
*********************************************
* 1. Добавить запись.                       *
* 2. Изменить записи.                       *
* 3. Удалить записи.                        *
* 4. Чтение файла.                          *
* 5. Вернуться в меню.                      *
*********************************************
1
Введите сколько записей вы хотите добавить: 2
Производитель ноутбука: xiaomi
Операционная система: linux
Оперативная память(в ГБ): 16
Модель процессора: snapdragon we2
Диагональ экрана(в дюймах): 12

Производитель ноутбука: noname
Операционная система: windows
Оперативная память(в ГБ): 20
Модель процессора: Albrus
Диагональ экрана(в дюймах): 12

Запись добавлена и файл обновлён
*********************************************
*                  Menu                     *
*********************************************
* 1. Создать запись файла.                  *
* 2. Выполнить поиск по файлу.              *
* 3. Запись и чтение файла.                 *
* 4. Сортировать данные файла.              *
* 5. Завершить работу                       *
*********************************************
```

Рисунок 19 – Процесс добавления новых записей

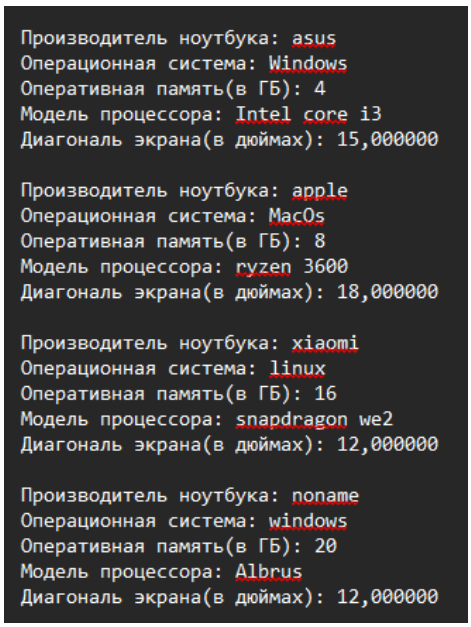

Рисунок 20 – Результат добавления новых записей в файл coursework.txt.

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

Процесс выполнения поиска по процессору представлен на рисунке 21.

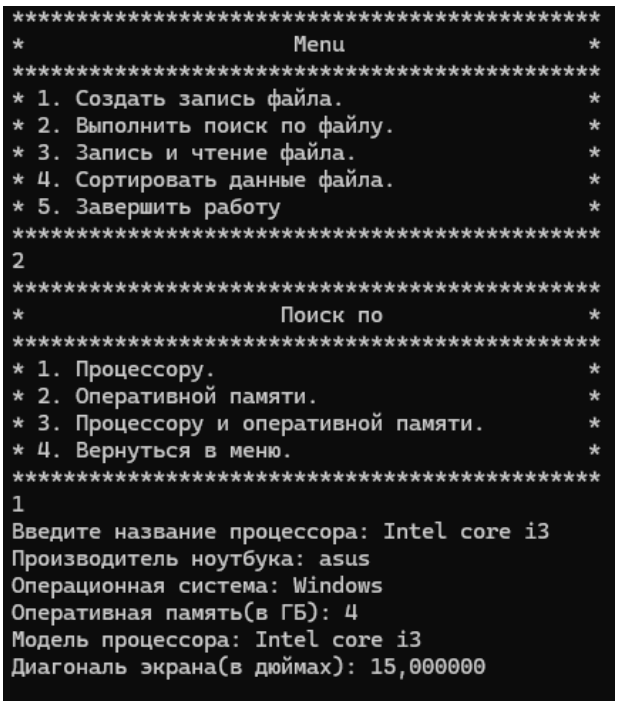

Рисунок 21 – Процесс поиска по процессору

Выполним аналогичным способом поиск по другим критериям. Результаты представлены на рисунках 22-23.

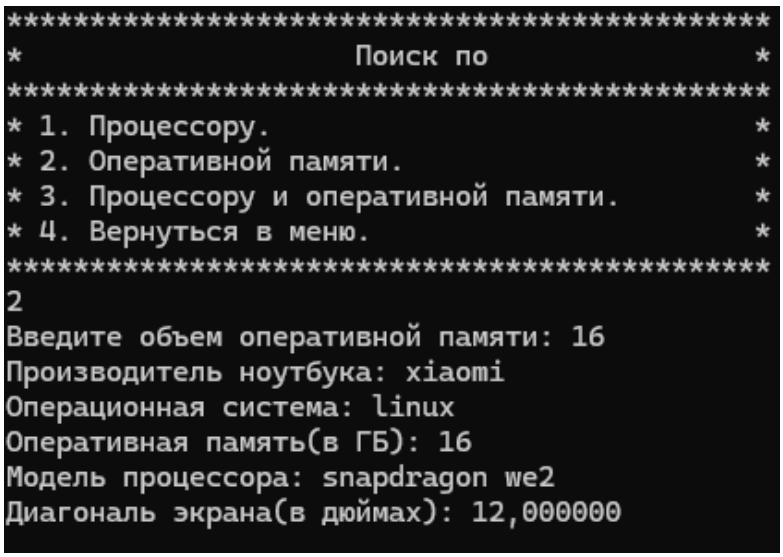

Рисунок 22 – Процесс поиска по оперативной памяти

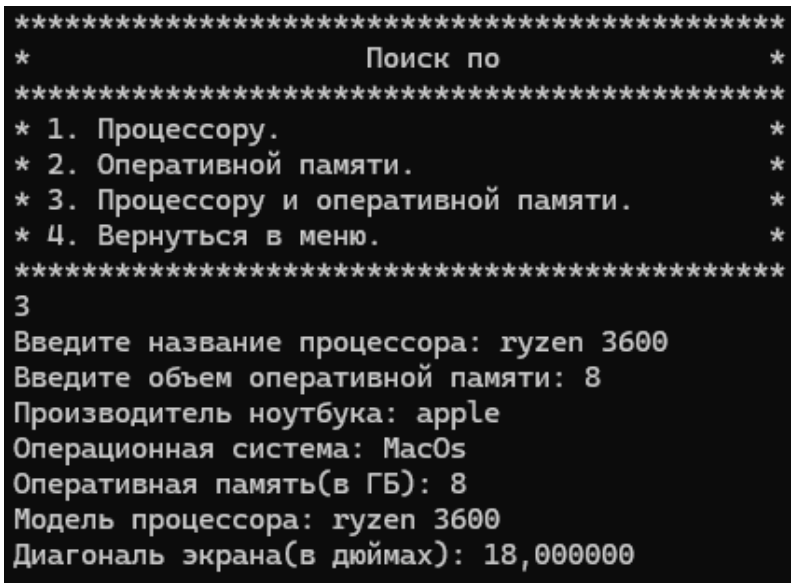

Рисунок 23 – Процесс комбинированного поиска

Для обновления записи номер 3, представленной на рисунке 20, выбираем пункт 3 «Запись и чтение файла», затем пункт 2 «Изменить запись», далее номер записи для изменения и изменяем необходимую запись. Результаты представлены на рисунках 24-25.

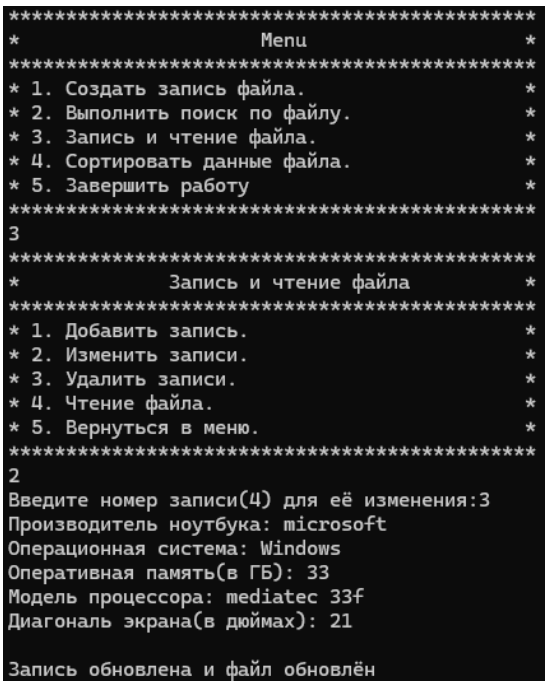

Рисунок 24 – Процесс обновления записи

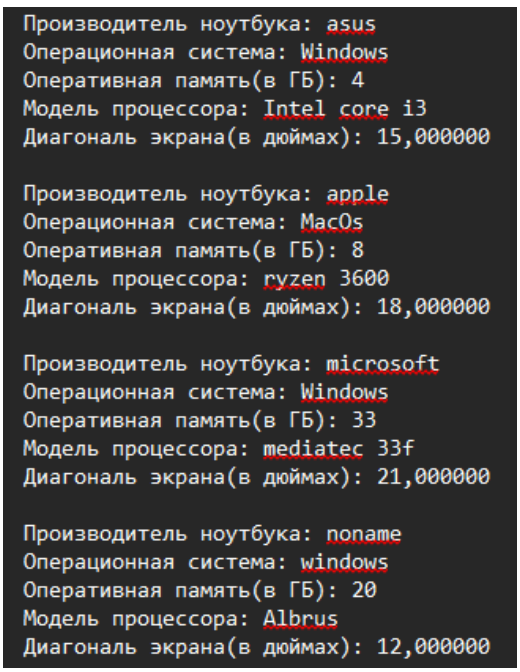

Рисунок 25 – Результат обновления записи файла Coursework.txt

Выполним удаление записи номер 1, представленной на рисунке 24. Выбираем пункт 3 «Запись и чтение файла», далее выбираем пункт 3 «Удалить запись», затем номер записи для удаления. Результаты представлены на рисунках 26-27.

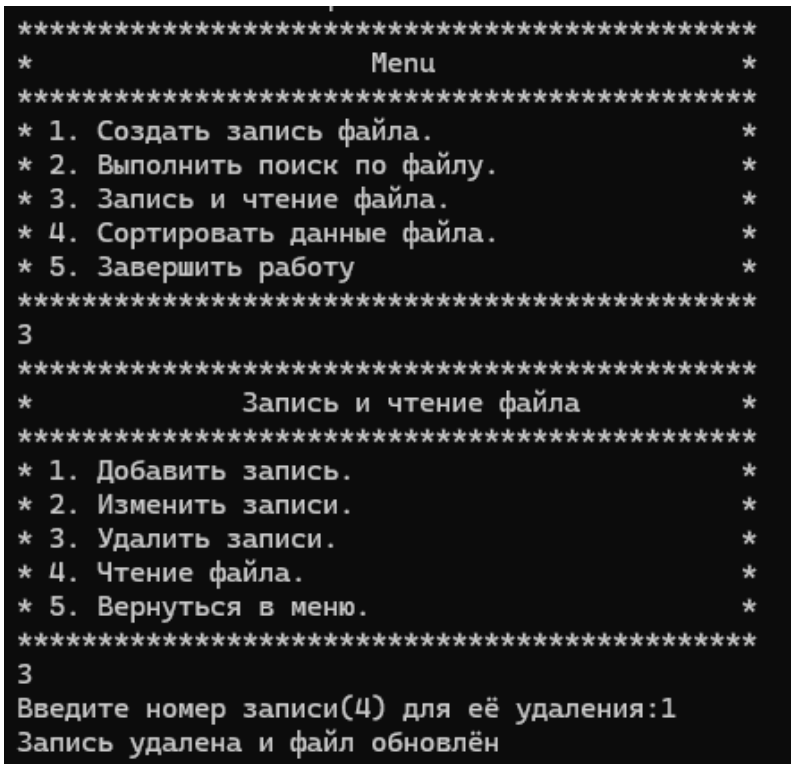

Рисунок 26 – Процесс удаления записи

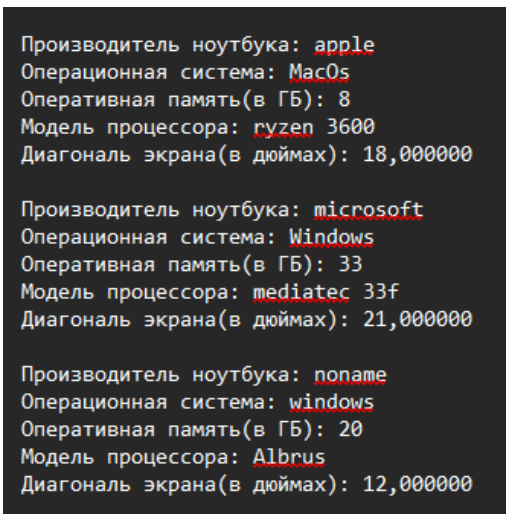

Рисунок 27 – Результат удаления записи в файле Coursework.txt

Выполним чтение записей. Выбираем пункт 3 «Запись и чтение файла», далее выбираем пункт 4 «Чтение файла». Результаты представлены на рисунке 28.

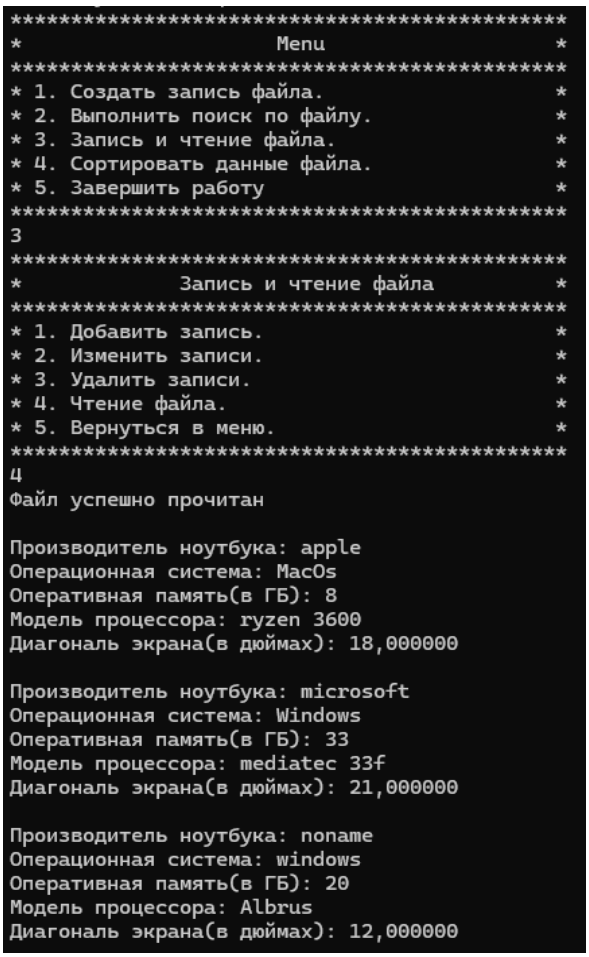

Рисунок 28 – Результат чтения данных

Выполним сортировку записей. Перед этим изменим записи в файле, чтобы было видно результат сортировки (Рис.29). Выбираем пункт 4 «Сортировать данные файла», далее выбираем какой из способов сортировки применить: по производителю или операционной системе или по обеим критериям сразу (приоритетно по производителю). Результаты представлены на рисунках 30-32.

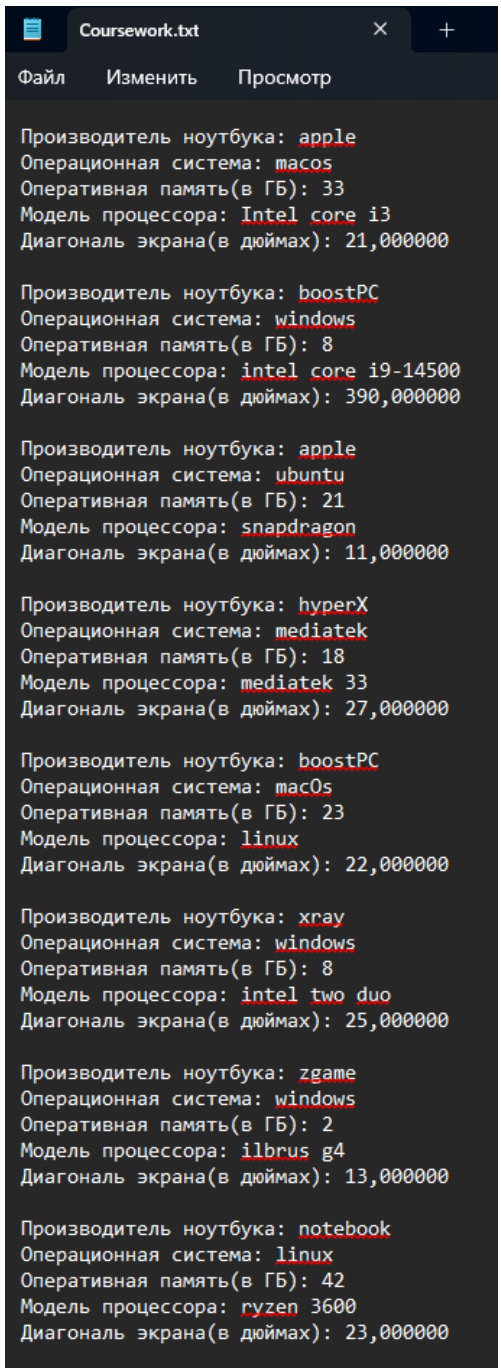

Рисунок 29 – Не отсортированные данные в файле Coursework.txt

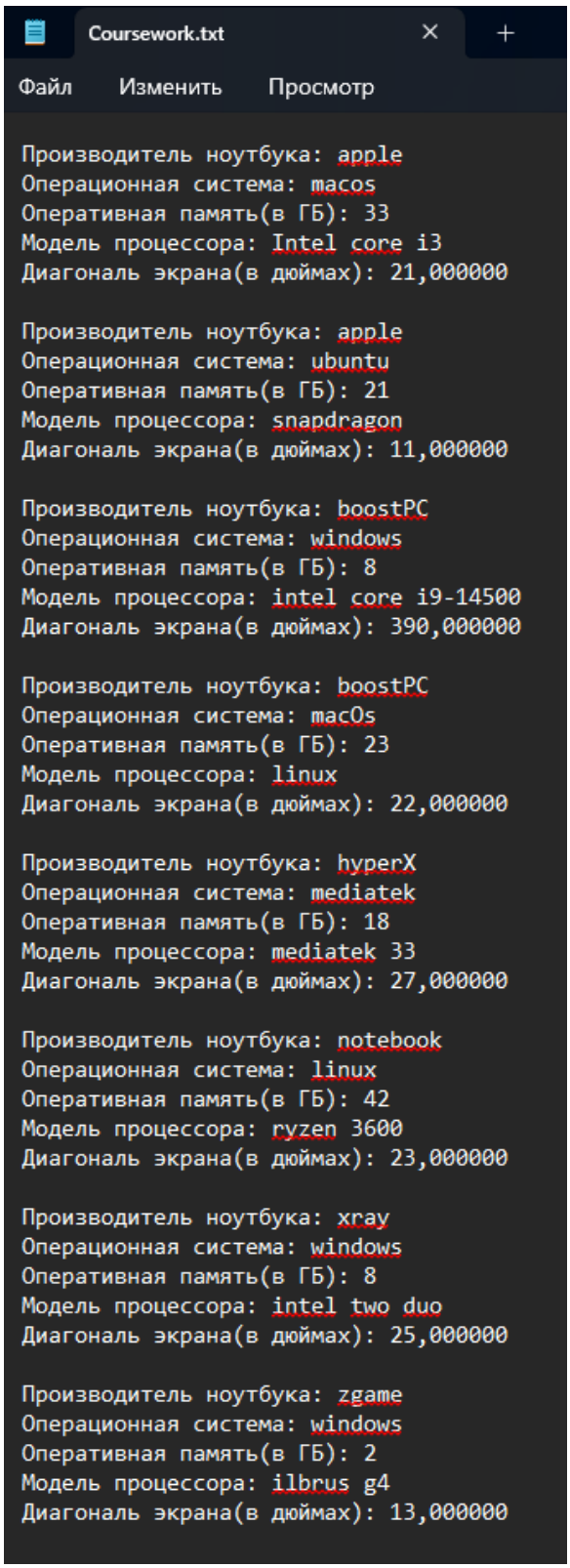

Рисунок 30 – Результат сортировки данных по производителю в файле Coursework.txt

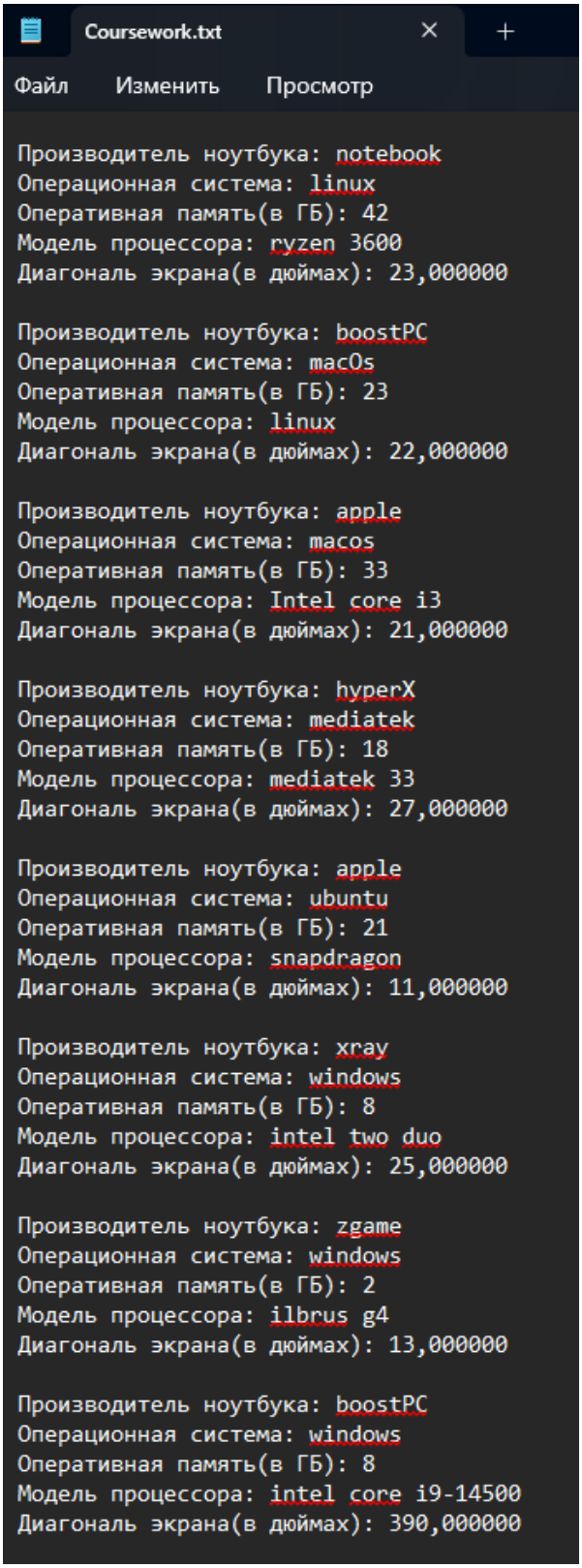

Рисунок 31 – Результат сортировки данных по операционной системе в файле Coursework.txt

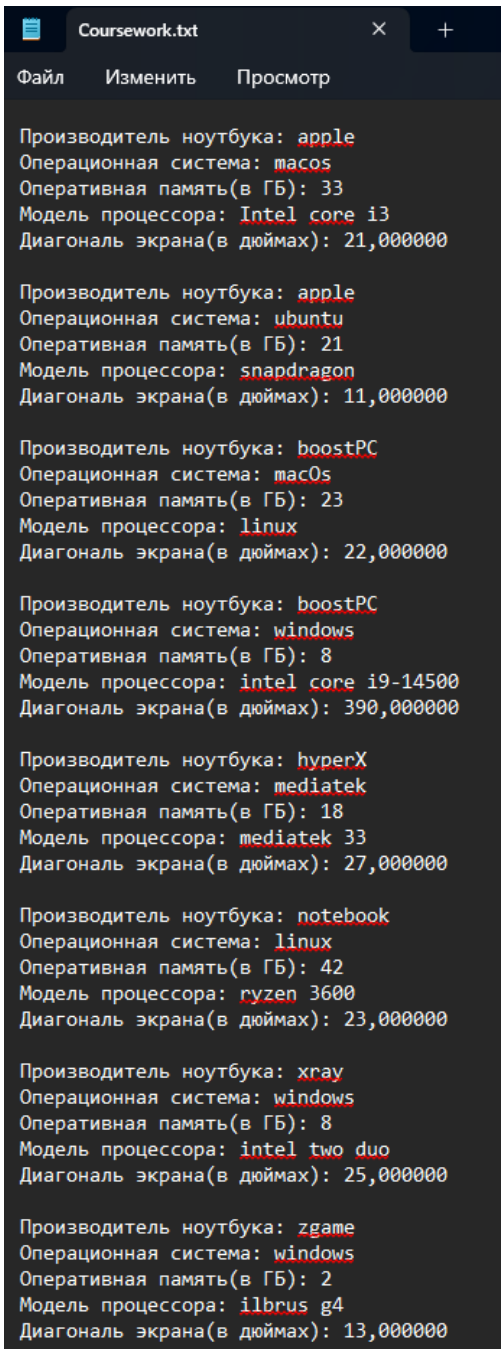

Рисунок 32 – Результат сортировки данных по обеим критериям сразу в файле Coursework.txt

При неправильном указании количества записей может происходить ошибка при удалении записей, связанной с невозможностью выделения памяти (Рисунок 33).

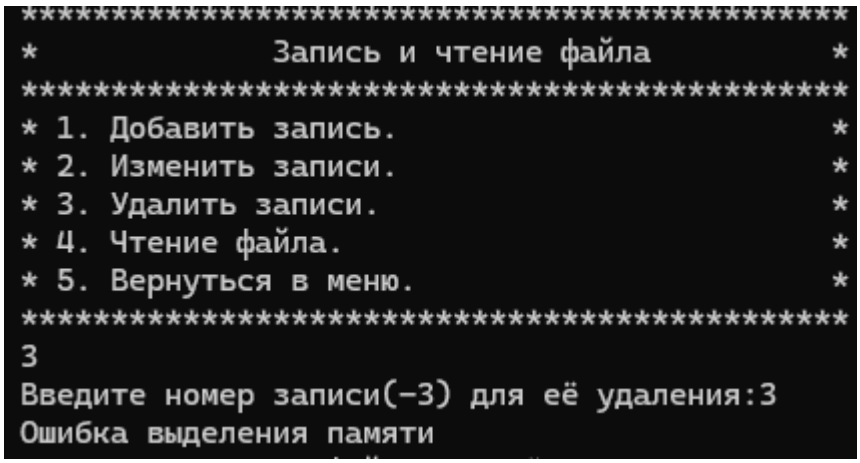

Рисунок 33 – Ошибка выделения памяти

При неудачном чтении из файла или открытии файла программа выводит ошибку на экран представленной на рисунках 34-35.

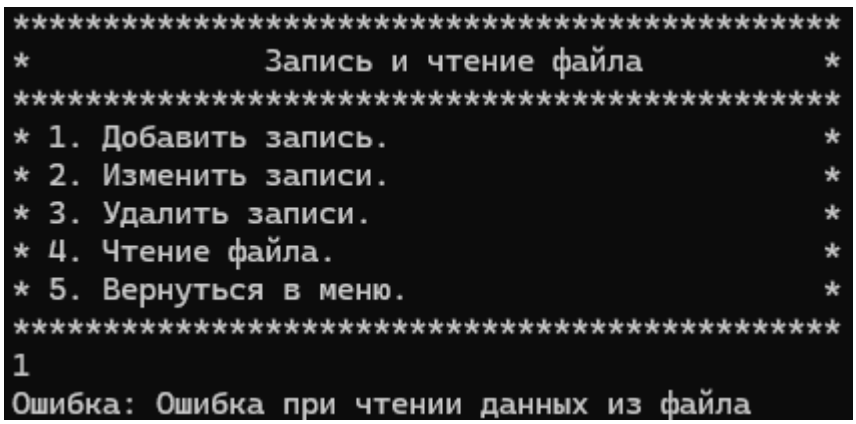

Рисунок 34 – Ошибка чтения из файла

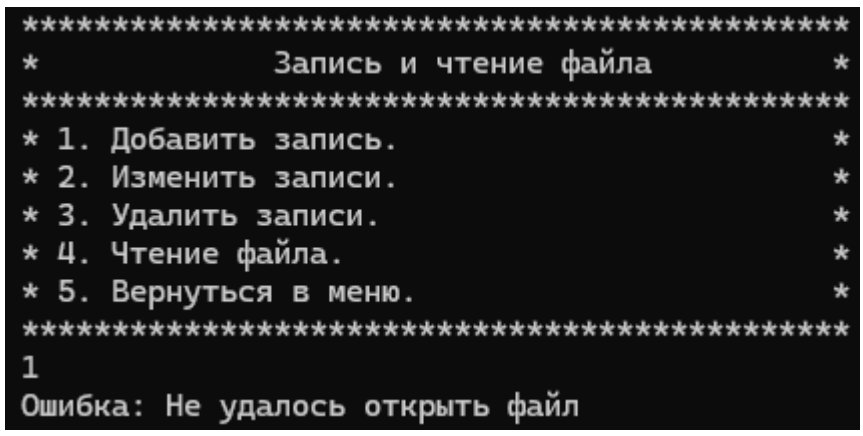

Рисунок 35 – Ошибка открытия файла

При неправильном выборе в меню и подменю на экран выводится ошибка о неправильном выборе действия (рисунок 36).

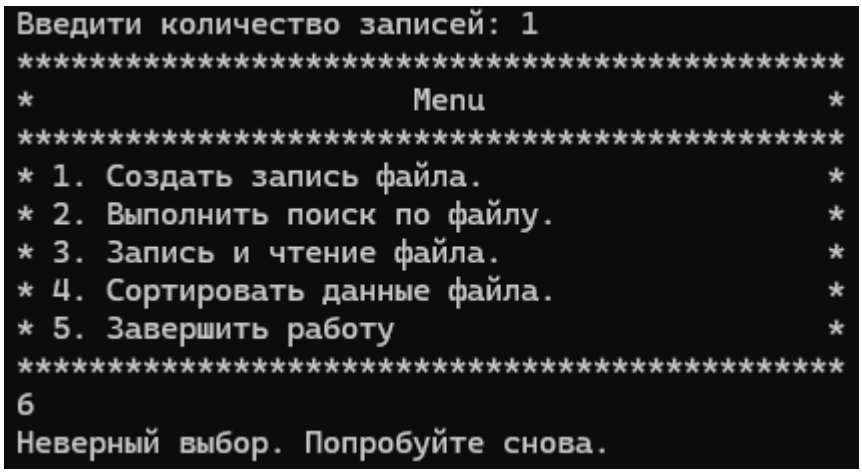

Рисунок 36 – Неправильно выбранное действие

Заключение

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

Ссылка: https://github.com/FalsFord/Coursework_1smstr

Список используемых 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. Подбельский В.В. Программирование на языке Си / В.В. Подбельский, С.С. Фомин. - М.: ФиС, 1999. - 600 с.

3. 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.

4. 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.

5. Дейт К.Дж. Введение в системы баз данных: 7-е изд. / К.Дж. Дейт. - М.: Вильямс, 2001. - 43 с.

6. Microsoft Learn. Справочник по языку C: операторы (C): оператор switch (C) [Электронный ресурс] / Microsoft Learn: официальный сайт. - 2022. - Режим доступа: https://learn.microsoft.com.

7. CppStudio. Язык программирования C++: указатели, массивы и строки: динамический массив в C++ [Электронный ресурс] / CppStudio: электронный журнал. - 2022. - Режим доступа: http://cppstudio.com/post/432/.

8. Указатели и массивы. Глава 5. Указатели и массивы [Электронный ресурс] / cpp: электронный журнал. - 2022. - Режим доступа: https://learnc.info/algorithms/bubblesort.html.

9. Wikipedia. Файловый ввод/вывод в языке Си [Электронный ресурс] / Wikipedia: русскоязычная электронная библиотека. - 2022. – Режим доступа: https://ru.wikipedia.org/wiki/Файловый_ввод-вывод_в _языке_Си.

10. Солдатенко И.С. Основы программирования на языке Си: учеб. пособие. - Тверь: Твер. гос. ун-т, 2017. - 159 с.

Приложение

```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

typedef struct Notebook {
   char fabricator[10];
   char OS[10];
   int RAM;
   char CPU[20];
   float screenDiagonal;
} notebook_t;

notebook_t* fill_notebooks(notebook_t* notebooks, int size) {
   for (int i = 0; i < size; i++) {
      printf("Производитель ноутбука: ");
      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; // Удаляем символ новой строки

      printf("Диагональ экрана(в дюймах): ");
      scanf("%f", &notebooks[i].screenDiagonal);
      getchar();
      printf("\n");
   }
   return notebooks;
}

void print_notebook(notebook_t notebook) {
   printf("Производитель ноутбука: %s\n", notebook.fabricator);
   printf("Операционная система: %s\n", notebook.OS);
   printf("Оперативная память(в ГБ): %d\n", notebook.RAM);
   printf("Модель процессора: %s\n", notebook.CPU);
   printf("Диагональ экрана(в дюймах): %f\n", notebook.screenDiagonal);
   printf("\n");
}

int* search_notebook(notebook_t* notebooks, int SIZE, char* CPU, int RAM) {
   int* countOfSeeked = malloc(SIZE * sizeof(int));
   for (int i = 0; i < SIZE; i++) {
      countOfSeeked[i] = -1;
   }
   if (CPU[0] != '\0' && RAM != -1) {
      for (int i = 0; i < SIZE; i++) {
         if (strcmp(notebooks[i].CPU, CPU) == 0 && notebooks[i].RAM == RAM) countOfSeeked[i] = i;
      }
   }
   else if (CPU[0] != '\0' && RAM == -1) {
      for (int i = 0; i < SIZE; i++) {
         if (strcmp(notebooks[i].CPU, CPU) == 0) countOfSeeked[i] = i;
      }
   }
   else {
```

```
        for (int i = 0; i < SIZE; i++) {
            if (notebooks[i].RAM == RAM) countOfSeeked[i] = i;
        }
    }
    return countOfSeeked;
}

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);
}

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);
}

int compare_fabricator_and_OS(const void* a, const void* b) {
    const notebook_t* notebook1 = (const notebook_t*)a;
    const notebook_t* notebook2 = (const notebook_t*)b;
    int fabricator_cmp = strcmp(notebook1->fabricator, notebook2->fabricator);
    if (fabricator_cmp == 0) {
        return strcmp(notebook1->OS, notebook2->OS);
    }
    return fabricator_cmp;
}

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_and_OS);
    }
    else {
        printf("Ошибка: Неверный критерий сортировки\n");
    }
    return notebooks;
}

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;
}

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;
}

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;
}

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;
}

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);
}
```