ФГБОУ ВО «Пензенский государственный университет»
Политехнический институт
Факультет вычислительной техники
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Программирование»
на тему «База данных»

21.05.2026

Выполнил: ст. гр. 25ВВВ2
Козлов А. В.
Принял: к.т.н., доцент
Генералова А.А.

Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

_____ М.А. Митрохин

«25» февраля 2026г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Козлову Артему Витальевичу Группа 25ВВВ2

Тема проекта: ***Разработка программы сложной структуры методом нисходящего программирования.***

Исходные данные (технические требования) на проектирование

База данных компьютерных комплектующих.
Программа реализует удаление, добавление, редактирование, поиск, сортировка записей; Фильтрация записей по определённому критерию;
Создание новой базы данных; Сохранение текущей базы данных; Загрузка сохраненных баз данных.

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. **Расчетная часть**

Разработка программы.

2. **Графическая часть**

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. **Экспериментальная часть**

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "11" февраля 2026

Дата защиты проекта "12" мая 2026

Руководитель [illegible]

Задание получил "11" февраля 2026г.

Студент Козлов А.В.

Содержание

Введение 5
Постановка задачи 6
Выбор решения 8
Описание разработки программы 11
Отладка и тестирование 12
Описание программы 13
Руководство пользователя 31
Заключение 49
Список изученных источников 50
Приложение А 51
Приложение Б 82

Введение

В современном мире персональный компьютер является неотъемлемой частью учебной, профессиональной и повседневной деятельности человека. Его производительность, надёжность и удобство использования во многом зависят от правильно подобранных комплектующих. Процессор, видеокарта, оперативная память, накопители, материнская плата и другие устройства имеют разные характеристики, стоимость и назначение, поэтому сведения о них необходимо хранить в удобной и систематизированной форме.

Актуальность данной темы заключается в том, что количество моделей компьютерных комплектующих постоянно увеличивается, а пользователю часто требуется быстро найти нужную информацию, сравнить устройства или отобрать их по определённым параметрам. Ручная обработка таких данных занимает много времени и может приводить к ошибкам. Поэтому создание программы, которая позволяет вести базу данных комплектующих, является практически значимой задачей.

Разрабатываемая программа представляет собой информационную систему для хранения и обработки данных о компьютерных комплектующих. В ней предусматривается работа со сложными типами данных, структурами, списками и файлами, что соответствует базовым принципам организации данных в программировании [2], [8], [10]. Это позволяет не только хранить сведения о записях, но и выполнять над ними различные операции: добавление, удаление, редактирование, поиск, сортировку и фильтрацию по заданному критерию.

Постановка задачи

Необходимо разработать программу для работы с базой данных компьютерных комплектующих. Программа должна иметь интуитивно понятный интерфейс для пользователя и обеспечивать удобное хранение, просмотр и обработку информации о комплектующих.

Необходимо определить название пунктов меню, построить алгоритмы работы с массивами структур для фильтрации и сортировки записей по заданным критериям, а также изучить функции для работы с файлами, чтобы сохранять базу данных в файл и загружать её из файла [2], [4], [11].

Программа должна реализовывать следующие основные функции: добавление новых записей, удаление записей, редактирование информации, поиск необходимых комплектующих, сортировку записей, фильтрацию по выбранным критериям, создание новой базы данных, сохранение текущей базы данных и загрузку ранее сохранённых данных.

Многомодульность программы необходима для упрощения разработки, отладки и тестирования. Программа должна быть разделена на логические модули, что позволит легко расширять её функциональность и вносить изменения в отдельные части программы [5].

Использование сложных типов данных — массивов, структур и файлов — необходимо для более простой и удобной обработки информации о компьютерных комплектующих в коде программы.

Режим работы видеосистемы может быть текстовым или графическим. Для начала необходимо определить тип интерфейса и элементы управления, затем изучить способы их реализации [7], [9].

Устройства ввода-вывода — клавиатура и мышь. Необходимо различать и обрабатывать действия пользователя, произведённые с их помощью, что обеспечит удобное управление программой.

Интерфейс должен быть построен на основе меню. Это необходимо для создания понятной структуры программы и быстрого доступа пользователя к основным функциям.

Заставка необходима для того, чтобы пользователь, запустивший программу, смог получить краткую информацию о назначении программы и её возможностях.

Выбор решения

Для реализации программы был выбран язык программирования C++, так как он позволяет эффективно работать со сложными структурами данных, массивами, списками, файлами, а также предоставляет возможность использовать низкоуровневые элементы программирования. Кроме того, C++ поддерживает работу с пользовательскими типами данных, что удобно при создании базы данных компьютерных комплектующих [8], [10], [11].

Разработка программы выполнялась в среде Microsoft Visual Studio 2022. Данная среда разработки была выбрана благодаря удобному редактору кода, встроенным средствам компиляции, отладки и проверки программы. Microsoft Visual Studio 2022 позволяет быстро находить ошибки, запускать программу в режиме отладки и контролировать выполнение отдельных участков кода.

Ассемблерская вставка в программе реализована в функции CompareIntegersAsm и используется для сравнения двух целых чисел. Она применяется при сортировке записей базы данных по идентификатору id. Вставка помещает сравниваемые значения в регистры eax и edx, выполняет команду cmp, после чего с помощью условных переходов определяет результат сравнения. Функция возвращает -1, 0 или 1, что позволяет программе определить порядок расположения записей при сортировке [6], [7].

При запуске программы открывается приветственный экран. Далее появляется главное меню, состоящее из следующих пунктов:

1. Создать новую базу данных. Пользователь выбирает пункт меню «Новая». Программа очищает текущий список записей, сбрасывает активный путь к файлу, а также отключает режим поиска и фильтрации. После этого создаётся новая пустая база данных компьютерных комплектующих.

2. Загрузить базу данных из файла. Пользователь выбирает пункт меню «Загрузить» и вводит путь к сохранённому файлу. Программа открывает файл, проверяет его формат по заголовку DBV1, считывает записи и заносит их в

список. Если файл не найден, имеет неподдерживаемый формат или содержит повреждённые данные, программа выводит сообщение об ошибке.

3. Сохранить базу данных в файл. Пользователь выбирает пункт меню «Сохранить» и вводит путь к файлу с расширением .txt или .db. Если база уже была загружена или сохранена ранее, программа может использовать текущий путь к файлу. Записи сохраняются в текстовом виде в формате UTF-8. Каждая запись содержит поля: ID, название комплектующего, тип комплектующего, производитель и характеристики.

4. Добавить запись. Пользователь выбирает пункт меню «Добавить». Программа предлагает ввести данные нового комплектующего: ID, название комплектующего, тип комплектующего, производителя и характеристики. После ввода программа проверяет уникальность ID. Если запись с таким ID уже существует, выводится сообщение об ошибке. Если ID свободен, новая запись добавляется в базу данных.

5. Изменить запись. Пользователь выбирает пункт меню «Изменить» и вводит ID записи, которую необходимо отредактировать. Программа проверяет наличие записи с указанным ID. Если запись найдена, пользователь вводит новые данные для всех полей. При изменении ID программа проверяет, не занят ли новый идентификатор другой записью. После успешной проверки данные записи обновляются.

6. Удалить запись. Пользователь выбирает пункт меню «Удалить» и вводит ID записи, которую необходимо удалить. Программа выполняет поиск записи по ID. Если запись найдена, она удаляется из списка. Если запись отсутствует, выводится сообщение о том, что указанный ID не найден.

7. Выполнить поиск записей. Пользователь выбирает пункт меню «Поиск» и вводит строку для поиска. Программа ищет совпадения по всем основным полям записи: ID, названию комплектующего, типу, производителю и характеристикам. Поиск выполняется без учёта регистра. Найденные записи отображаются на экране в виде таблицы [2], [10].

8. Отсортировать записи. Пользователь выбирает пункт меню «Сортировка». Далее программа предлагает выбрать поле сортировки:

8.1.Сортировка по ID. Программа упорядочивает записи по идентификатору. Для сравнения ID используется отдельная функция, содержащая ассемблерную вставку для Win32, а в остальных режимах — обычное сравнение целых чисел [2], [6].

8.2.Сортировка по названию комплектующего. Программа сортирует записи по полю «Название комплектующего».

8.3.Сортировка по типу комплектующего. Программа сортирует записи по полю «Тип комплектующего». После выбора поля пользователь указывает направление сортировки: по возрастанию или по убыванию.

9. Отфильтровать записи. Пользователь выбирает пункт меню «Фильтр» и вводит тип комплектующего или его часть. Программа отображает только те записи, у которых поле «Тип комплектующего» содержит введённое значение. Фильтрация выполняется без учёта регистра.

10. Сбросить поиск и фильтрацию. Пользователь выбирает пункт меню «Сброс». Программа очищает активные параметры поиска и фильтрации, после чего снова отображает все записи базы данных.

11. Просмотреть сведения о программе. Пользователь выбирает пункт меню «О программе». На экран выводится краткая информация о назначении программы, её структуре и используемых средствах: многомодульная архитектура, структуры, список std::list, файловое хранение, текстовый интерфейс, управление клавиатурой и мышью.

12. Выход из программы. Пользователь выбирает пункт меню «Выход». Программа завершает работу.

Навигация по меню осуществляется с помощью клавиатуры и мыши. Для быстрого выбора команд используются горячие клавиши: N, L, S, A, E, D, F, T, I, R, H, Q. Также пользователь может нажимать мышью на кнопки панели инструментов в верхней части консольного окна [9].

Описание разработки программы

Разработанная программа состоит из нескольких модулей, которые реализовывались в следующем порядке:

1. main.cpp — основной файл программы. Создаёт объект базы данных и запускает пользовательский интерфейс программы.

2. record.h — файл с описанием структуры записи. Содержит поля: ID, название комплектующего, тип комплектующего, производитель и характеристики.

3. database.h — заголовочный файл класса базы данных. Содержит описание основных методов для добавления, изменения, удаления, поиска, фильтрации, сортировки, сохранения и загрузки записей.

4. database.cpp — реализация работы с базой данных. В данном модуле выполняется обработка списка записей, поиск по полям, фильтрация по типу комплектующего, сортировка и работа с файлами.

5. console_ui.h — заголовочный файл пользовательского интерфейса. Содержит описание класса консольного меню и основных действий пользователя.

6. console_ui.cpp — реализация текстового интерфейса программы. В данном модуле выполняется вывод заставки, отображение меню, таблицы записей, обработка команд клавиатуры и мыши.

7. console_utils.h — заголовочный файл вспомогательных функций для работы с консолью.

8. console_utils.cpp — реализация функций очистки экрана, установки позиции курсора, изменения цвета текста, чтения строк и чисел, вывода сообщений и паузы.

9. asm_utils.h — заголовочный файл функции сравнения целых чисел.

10. asm_utils.cpp — реализация сравнения идентификаторов записей. Для Win32 используется ассемблерная вставка, в остальных случаях применяется обычное сравнение средствами языка C++.

11. sample_components.txt — пример файла базы данных компьютерных комплектующих в текстовом формате.

12. sample_components.db — пример файла базы данных, предназначенный для проверки загрузки и сохранения информации.

По порядку реализации модулей можно понять, что использовался нисходящий подход разработки программы. Сначала были определены основные компоненты программы: структура записи, класс базы данных и главное меню. Затем были реализованы более мелкие детали: обработка пользовательского ввода, вывод информации на экран, поиск, фильтрация, сортировка и работа с файлами [1], [3], [5].

Такой подход позволяет упростить разработку и тестирование программы, так как каждый модуль выполняет отдельную задачу. При необходимости программу можно расширить, добавив новые поля записи, дополнительные виды сортировки или новые функции меню без полной переработки всей структуры проекта [5].

Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Данная среда обладает всеми необходимыми средствами для разработки и отладки программы. Для отладки использовались возможности Visual Studio: точки останова, пошаговое выполнение программы, трассировка и просмотр значений переменных.

Тестирование проводилось во время разработки, а также после завершения основных модулей программы. В ходе тестирования проверялась работа главного меню, добавление новых записей, изменение и удаление данных, поиск комплектующих, фильтрация записей по типу, сортировка по ID, названию и типу комплектующего, а также сохранение базы данных в файл и загрузка данных из файла.

Также была проверена работа пользовательского интерфейса: вывод заставки, отображение таблицы записей, обработка команд с клавиатуры и

нажатий мышью по панели инструментов. Особое внимание уделялось проверке корректности ввода ID, недопущению добавления записей с одинаковыми идентификаторами и обработке ошибок при открытии файлов.

В ходе тестирования серьёзных ошибок, связанных с работой файлов, обработкой списка записей и отображением данных, выявлено не было. Программа выполняет поставленные задачи и корректно работает с базой данных компьютерных комплектующих.

Описание программы

Приложение Data Base является основным модулем программы. При запуске программы происходит вывод заставки-приветствия, после чего пользователю отображается главное меню для работы с базой данных компьютерных комплектующих. Выбор пунктов меню осуществляется с помощью клавиатуры или мыши. Описание состояний программы выполнено в таблицах 1 - 4.

В таблице 1 представлено соответствие между клавишами управления, действиями пользователя и реакцией программы. Главное меню программы построено на выборе команд с клавиатуры. Каждая клавиша отвечает за отдельную операцию с базой данных компьютерных комплектующих: создание новой базы, загрузку и сохранение данных, добавление, изменение и удаление записей, поиск, сортировку и фильтрацию информации. Также в меню предусмотрены команды сброса параметров поиска и фильтрации, вывода сведений о программе и завершения работы.

Таблица 1 – Главное меню программы

Клавиши, вызывающее событие	Действие пользователя	Действие программы
N	Выбран пункт «Новая»	Создаётся новая пустая база данных, текущие записи очищаются
L	Выбран пункт «Загрузить»	Запускается диалог загрузки базы данных из файла
S	Выбран пункт «Сохранить»	Запускается диалог сохранения текущей базы данных в файл
A	Выбран пункт «Добавить»	Запускается диалог добавления новой записи в базу данных
E	Выбран пункт «Изменить»	Запускается диалог изменения выбранной записи
D	Выбран пункт «Удалить»	Запускается диалог удаления записи по ID
F	Выбран пункт «Поиск»	Запускается поиск записей по введённой строке
T	Выбран пункт «Сортировка»	Открывается меню выбора поля и направления сортировки
I	Выбран пункт «Фильтр»	Запускается фильтрация записей по типу комплектующего
R	Выбран пункт «Сброс»	Сбрасываются активные параметры поиска и фильтрации
H	Выбран пункт «О программе»	Выводится справочная информация о программе
Q	Выбран пункт «Выход»	Выполняется закрытие программы

В таблице 2 приведено описание работы с основными функциями программы. В ней указаны клавиши, с помощью которых пользователь

вызывает нужное действие, выбранный пункт меню и последующая реакция программы. Данные функции обеспечивают основную работу с базой данных компьютерных комплектующих: добавление новых записей, изменение и удаление существующих данных, поиск, фильтрацию, сортировку, а также загрузку и сохранение базы данных в файл.

Таблица 2 — Работа с основными функциями программы

Клавиши, вызывающее событие	Действие пользователя	Действие программы
A	Выбран пункт «Добавить»	Пользователь вводит ID, название, тип, производителя и характеристики комплектующего
E	Выбран пункт «Изменить»	Пользователь вводит ID изменяемой записи, затем вводит новые данные
D	Выбран пункт «Удалить»	Пользователь вводит ID записи, которую необходимо удалить
F	Выбран пункт «Поиск»	Пользователь вводит строку для поиска по базе данных
I	Выбран пункт «Фильтр»	Пользователь вводит тип комплектующего или его часть
T	Выбран пункт «Сортировка»	Пользователь выбирает поле сортировки и направление сортировки
L	Выбран пункт «Загрузить»	Пользователь вводит путь к файлу, из которого нужно считать базу данных
S	Выбран пункт «Сохранить»	Пользователь вводит путь к файлу, в который нужно сохранить базу данных

В таблице 3 приведены параметры сортировки записей базы данных компьютерных комплектующих. После выбора пункта «Сортировка» пользователь указывает поле, по которому необходимо упорядочить записи: по идентификатору, названию или типу комплектующего. Затем выбирается направление сортировки — по возрастанию или по убыванию. В результате программа перестраивает список записей в соответствии с выбранными параметрами и выводит данные в заданном порядке [2].

Таблица 3 – Сортировка записей

Клавиши, вызывающее событие	Действие пользователя	Действие программы
1	Выбран пункт «Сортировка по ID»	Записи сортируются по идентификатору
2	Выбран пункт «Сортировка по названию»	Записи сортируются по названию комплектующего
3	Выбран пункт «Сортировка по типу»	Записи сортируются по типу комплектующего
1	Выбрано направление «По возрастанию»	Записи выводятся от меньшего значения к большему
2	Выбрано направление «По убыванию»	Записи выводятся от большего значения к меньшему

В таблице 4 описана работа функций поиска и фильтрации записей базы данных. Данные функции предназначены для быстрого отбора нужной информации среди компьютерных комплектующих. Поиск выполняется по введённой пользователем строке и позволяет находить совпадения в разных полях записи: ID, названии, типе, производителе и характеристиках. Фильтрация используется для вывода записей определённого типа комплектующего или его части [2], [10].

Таблица 4 — Поиск и фильтрация записей.

Клавиши, вызывающее событие	Действие пользователя	Действие программы
F	Выбран пункт «Поиск»	Программа считывает введённую строку
Enter	Подтверждён ввод строки поиска	Программа ищет совпадения по ID, названию, типу, производителю и характеристикам
I	Выбран пункт «Фильтр»	Программа считывает тип комплектующего или его часть
Enter	Подтверждён ввод значения фильтра	На экран выводятся записи, соответствующие указанному типу
R	Выбран пункт «Сброс»	Программа очищает результаты поиска и фильтрации и снова показывает все записи

Далее на рисунке 1 представлена схема взаимодействия программы.

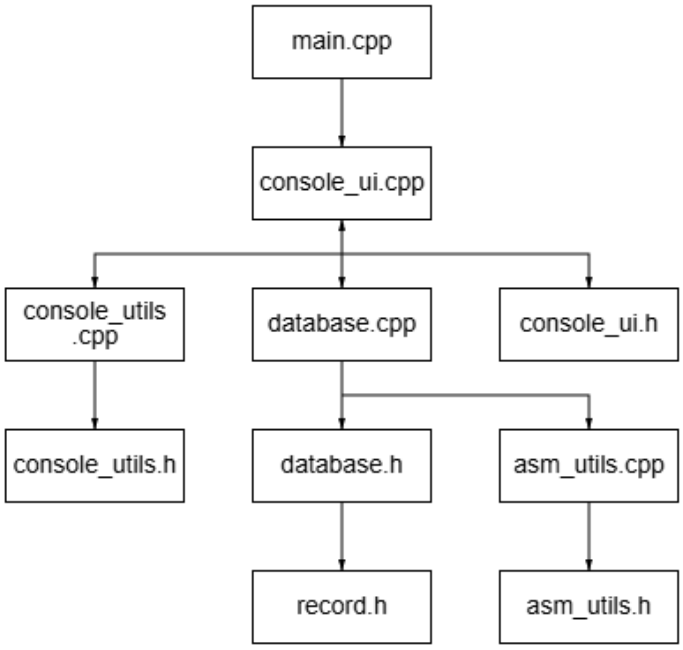

Рисунок 1 – Схема взаимодействия программы

Ниже, на рисунке 2 представлена схема данных.

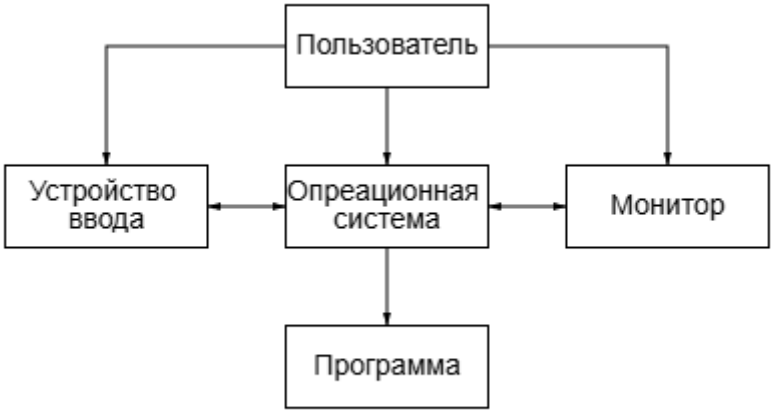

Рисунок 2 – Схема данных

Схема программы. Функция Main представлена на рисунке 3.

Рисунок 3 – Функции main

Схема программы. Функция Database представлена на рисунке 4.

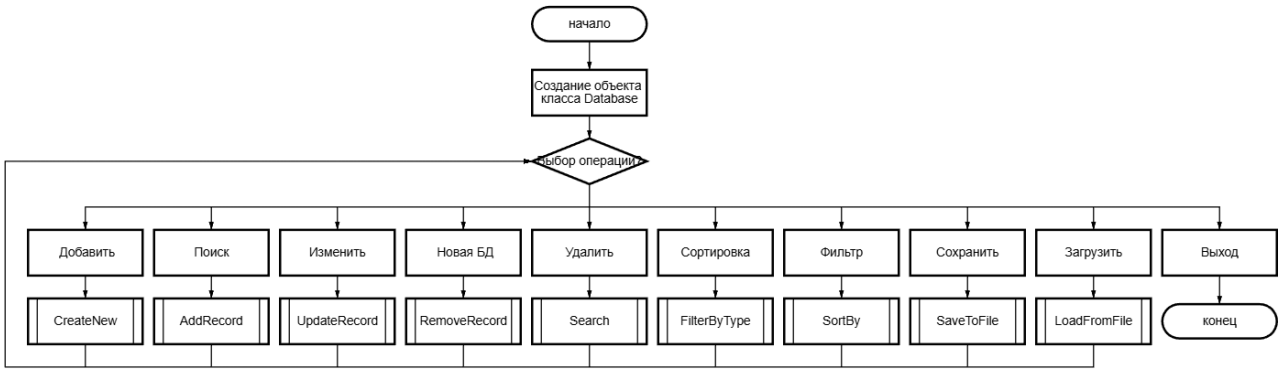

Рисунок 4 – Функции Database

Схема программы. Функция CreateNew представлена на рисунке 5.

Рисунок 5 – Функции CreateNew

Схема программы. Функция AddRecord представлена на рисунке 6.

Рисунок 6 – Функции AddRecord

Схема программы. Функция UpdateRecord представлена на рисунке 7.

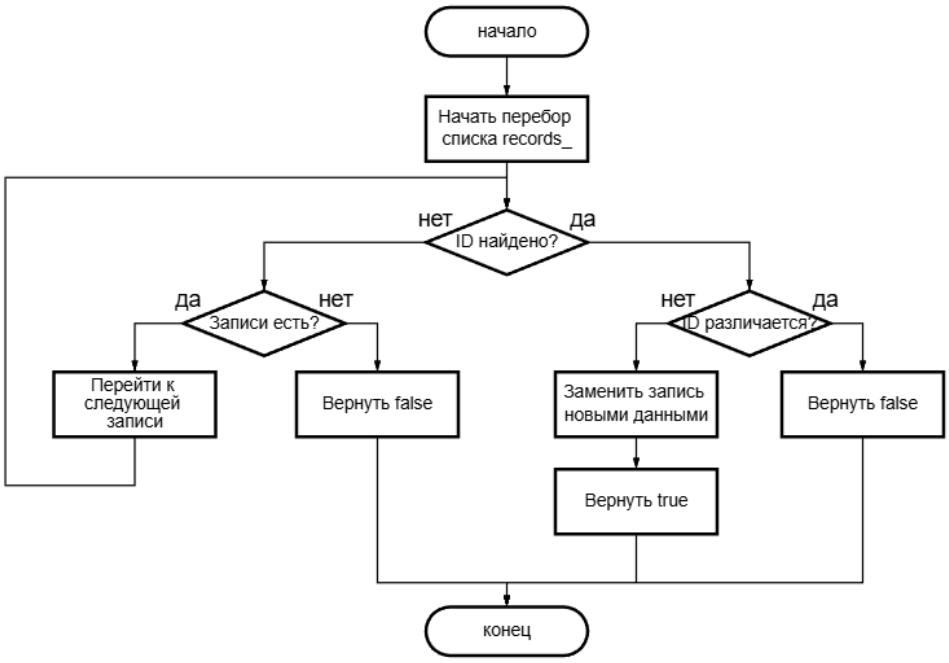

Рисунок 7 – Функции UpdateRecord

Схема программы. Функция RemoveRecord представлена на рисунке 8.

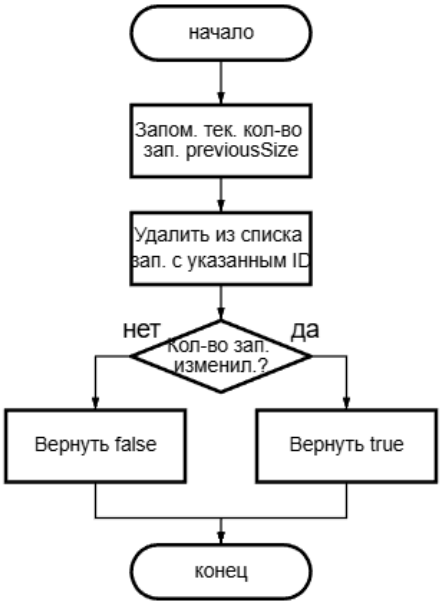

Рисунок 8 – Функции RemoveRecord

Схема программы. Функция ContainsId представлена на рисунке 9.

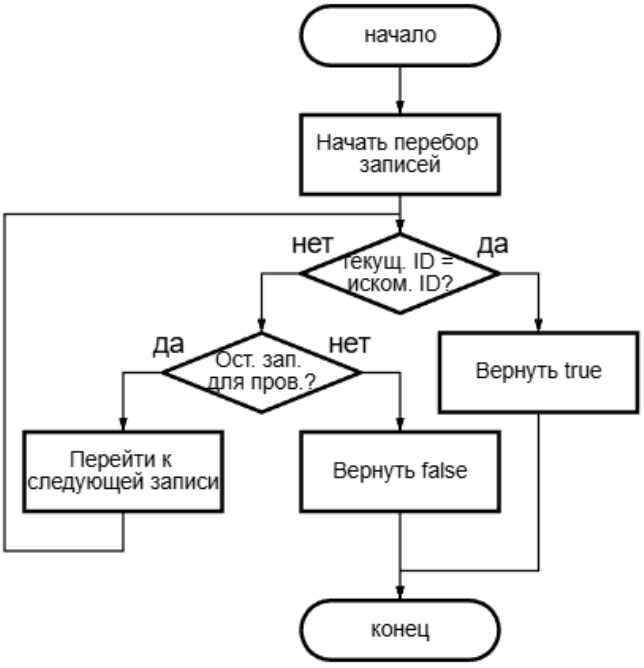

Рисунок 9 – Функции ContainsId

Схема программы. Функция GetAllRecords представлена на рисунке 10.

Рисунок 10 – Функции GetAllRecord

Схема программы. Функция Search представлена на рисунке 11.

Рисунок 11 – Функции Search

Схема программы. Функция FilterByType представлена на рисунке 12.

Рисунок 12 – Функции FilterByType

Схема программы. Функция SortBy представлена на рисунке 13.

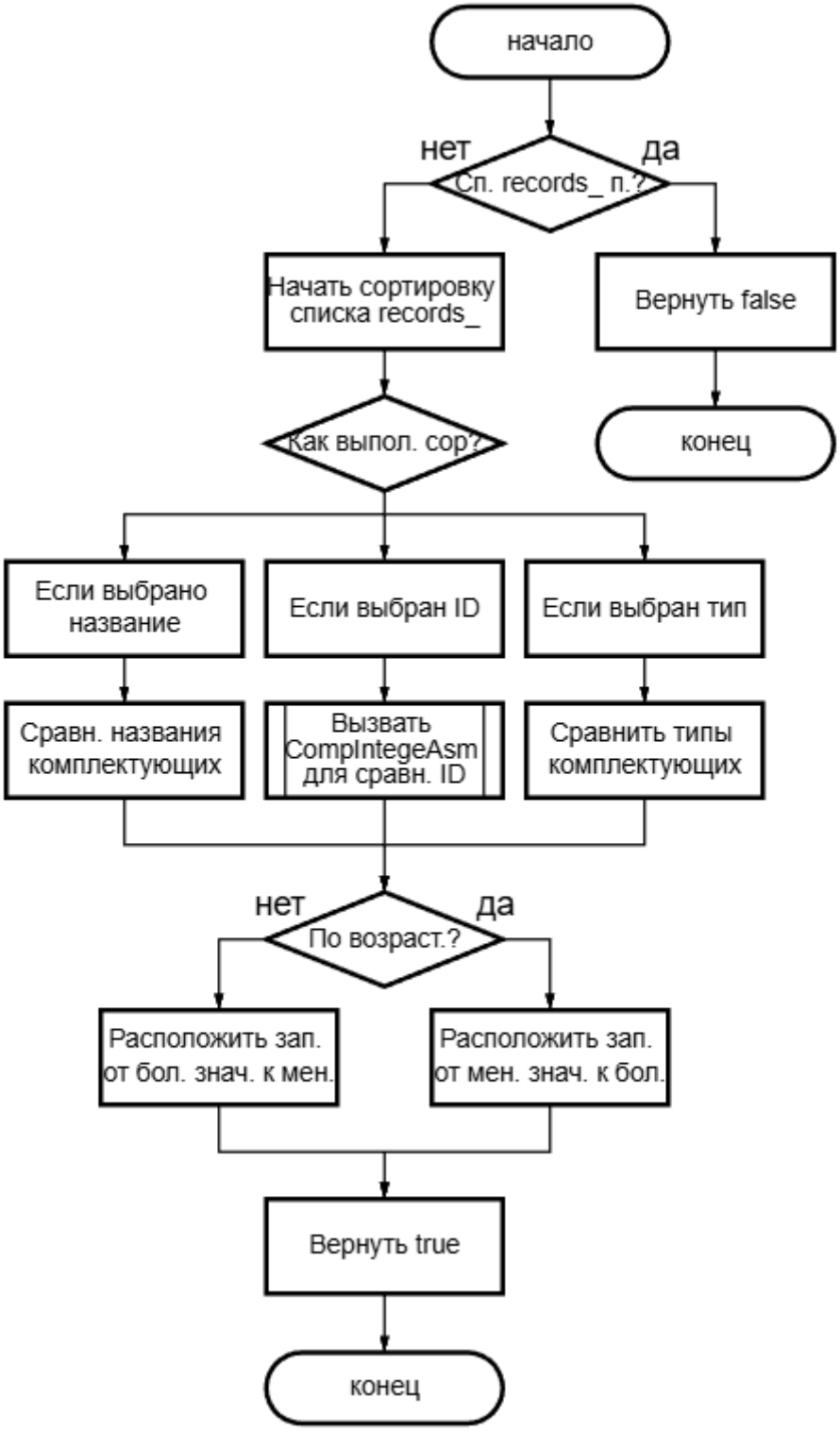

Рисунок 13 – Функции SortBy

Схема программы. Функция SaveToFile представлена на рисунке 14.

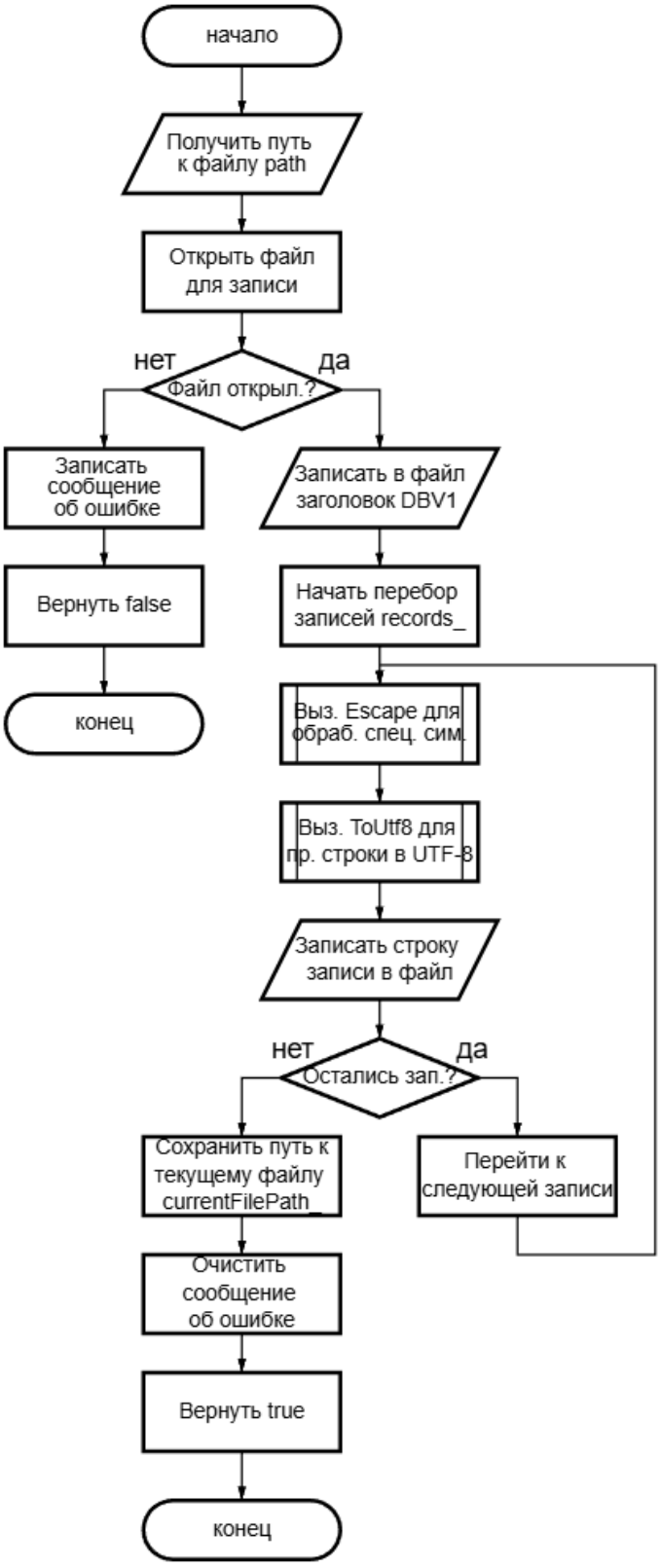

Рисунок 14 – Функции SaveToFile

Схема программы. Функция LoadFromFile представлена на рисунке 15.

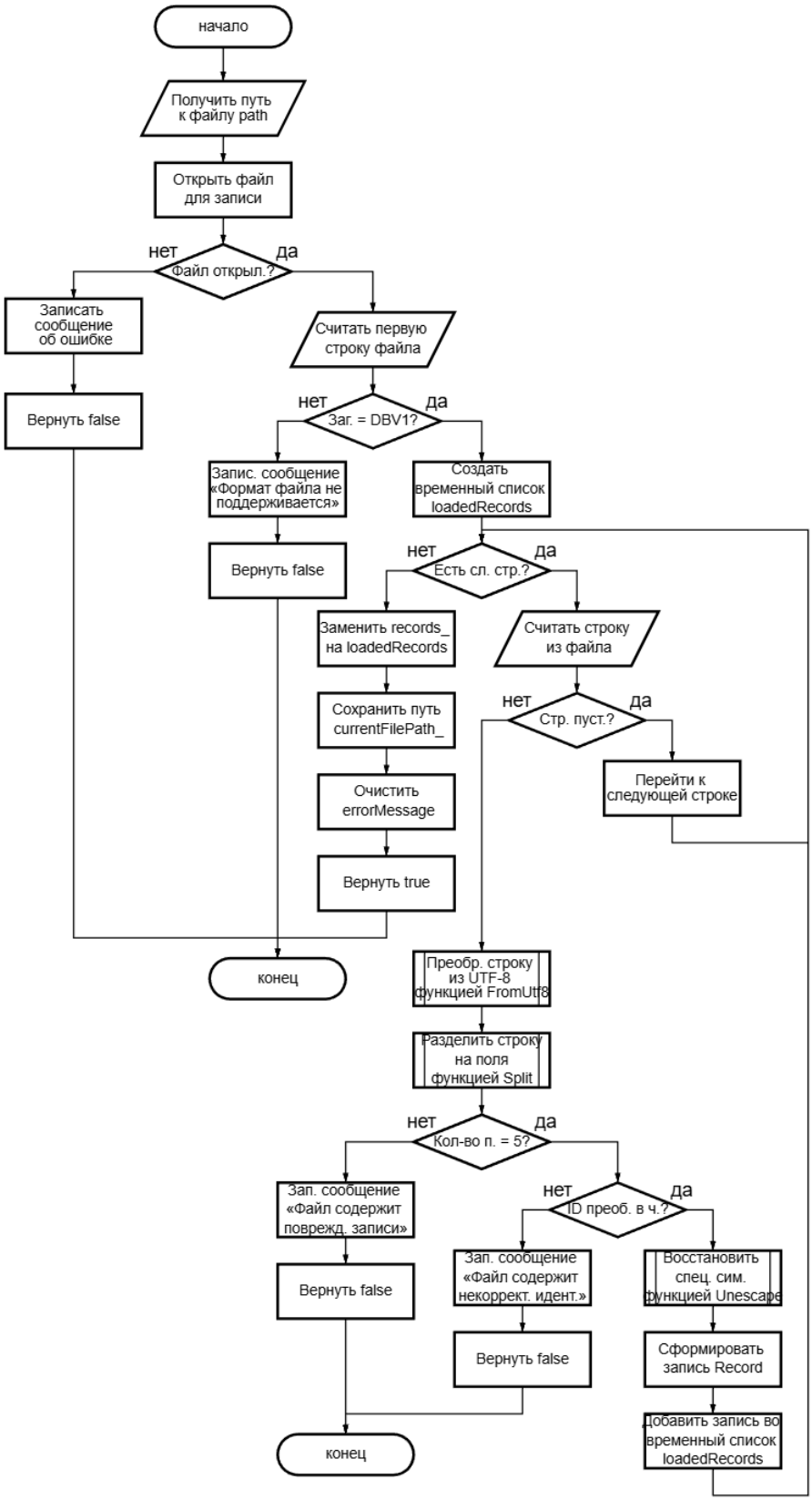

Рисунок 15 – Функции LoadFromFile

Схема программы. Функция CurrentFilePath представлена на рисунке 16.

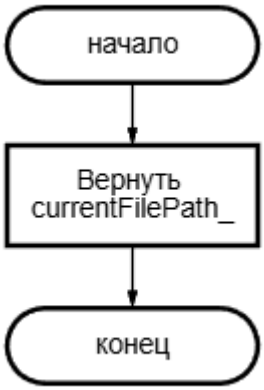

Рисунок 16 – Функции CurrentFilePath

Схема программы. Функция Size представлена на рисунке 17.

Рисунок 17 – Функции Size

Схема программы. Функция Console_utils представлена на рисунке 18.

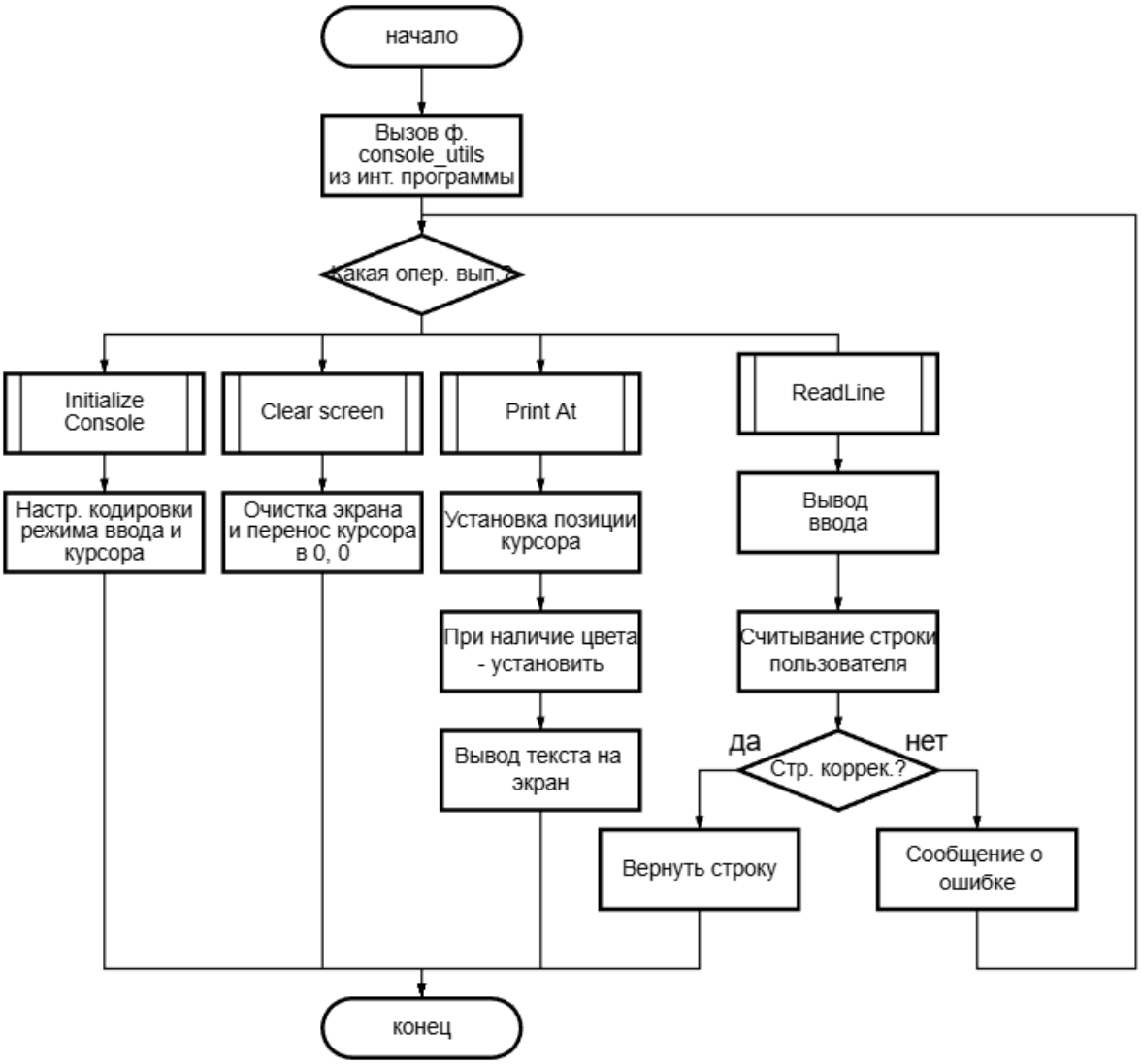

Рисунок 18 – Функции Console_utils

Схема программы. Функция Console_ui представлена на рисунке 19.

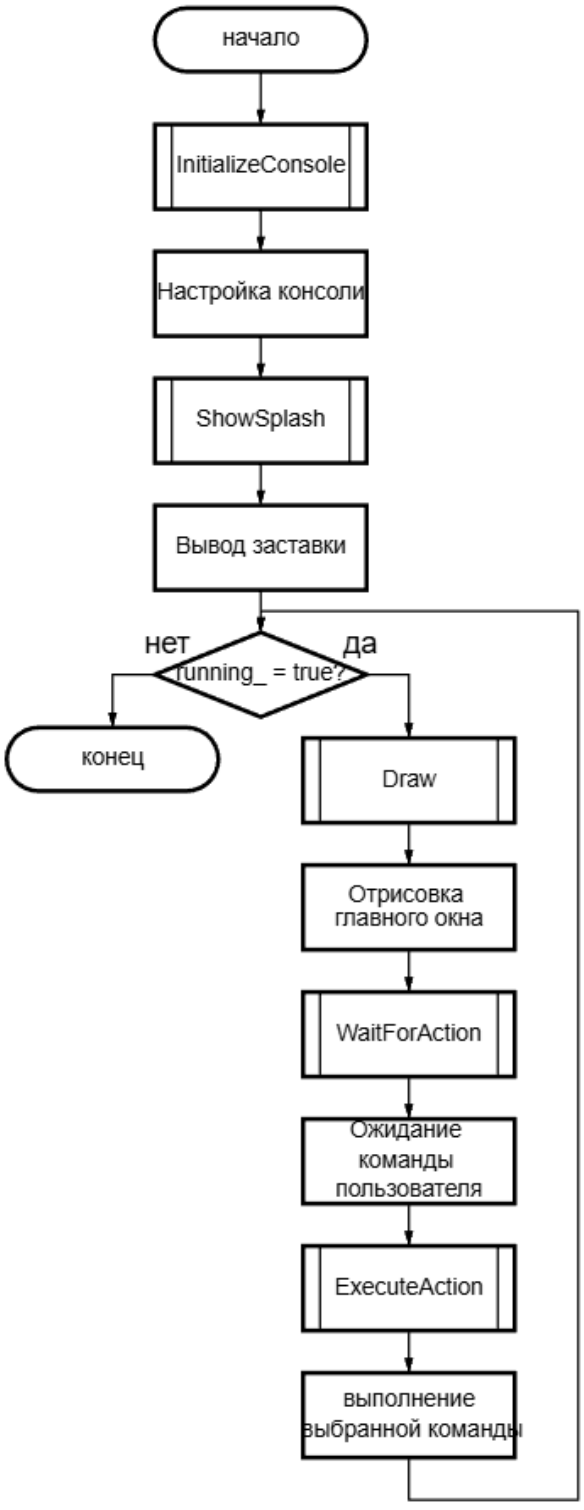

Рисунок 19 – Функции Console_ui

Схема программы. Функция asm_utils представлена на рисунке 20.

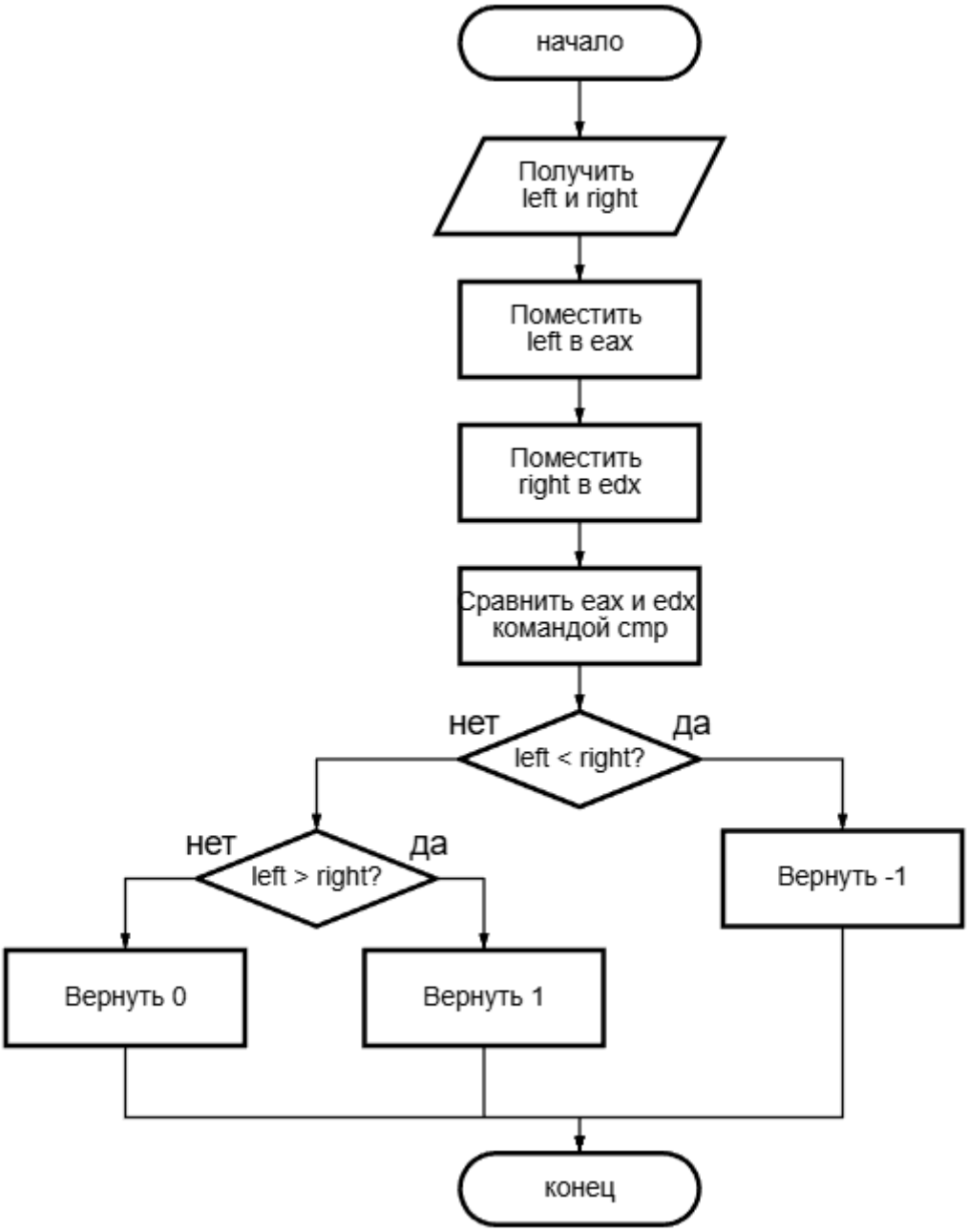

Рисунок 20 – Функции asm_utils

Руководство пользователя

Программа Data Base.exe предназначена для хранения информации о компьютерных комплектующих. Программа имеет интуитивно понятный интерфейс и поддерживает такие операции, как создание новой базы данных, добавление записи в базу данных удаление записи из базы данных, изменение

записи, поиск записей, сортировка записей, фильтрация записей, сохранение базы данных в файл и загрузка базы данных из файла.

При запуске программы на экран выводится заставка-приветствие. После этого открывается главное меню программы с панелью инструментов, представлено на рисунке 21. Управление осуществляется с помощью клавиатуры или мыши. Пользователь может нажимать горячие клавиши N, L, S, A, E, D, F, T, I, R, H, Q, а также выбирать пункты меню с помощью мыши.

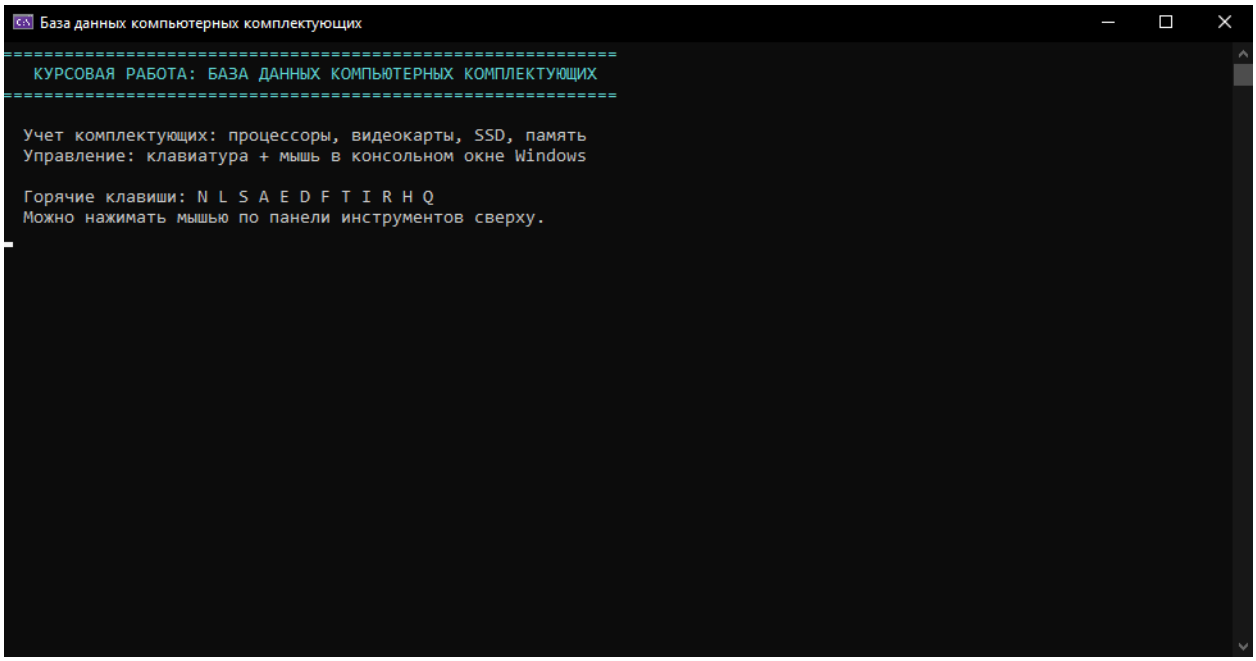

Рисунок 21 – Заставка-приветствие

Для просмотра существующих записей необходимо запустить программу. Все записи текущей базы данных отображаются в главном окне в виде таблицы. Просмотр существующих записей представлен на рисунке 22. В таблице выводятся ID, название комплектующего, тип комплектующего, производитель и характеристики. Если записей нет, программа выводит сообщение о том, что записи не найдены.

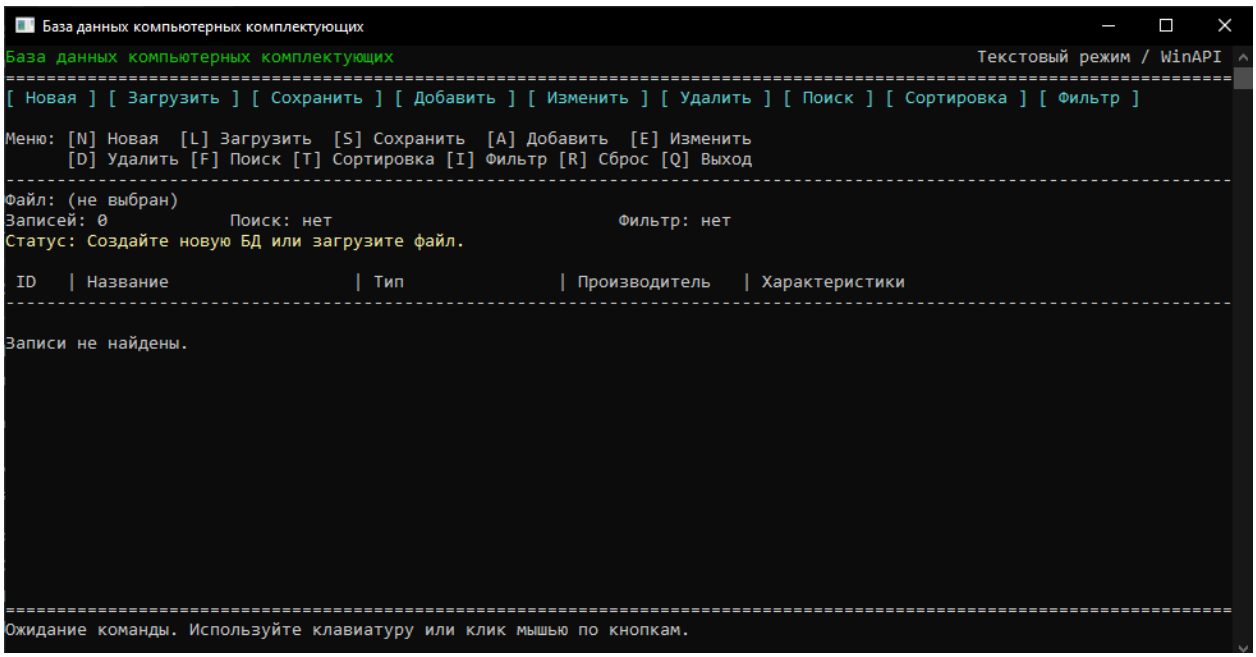

Рисунок 22 – Просмотр существующих записей

Для того чтобы загрузить базу данных из файла, необходимо выбрать пункт меню «Загрузить» или нажать клавишу «L». Далее нужно ввести путь к файлу, из которого производится чтение данных, и нажать клавишу Enter. Ввод пути для загрузки базы данных представлен на рисунке 23. Текущая база данных будет заменена данными из выбранного файла. Если файл не найден, имеет неправильный формат или содержит повреждённые записи, программа выдаст сообщение об ошибке. В случае успешной загрузки на экране появится сообщение «База данных загружена», результат загрузки представлен на рисунке 24.

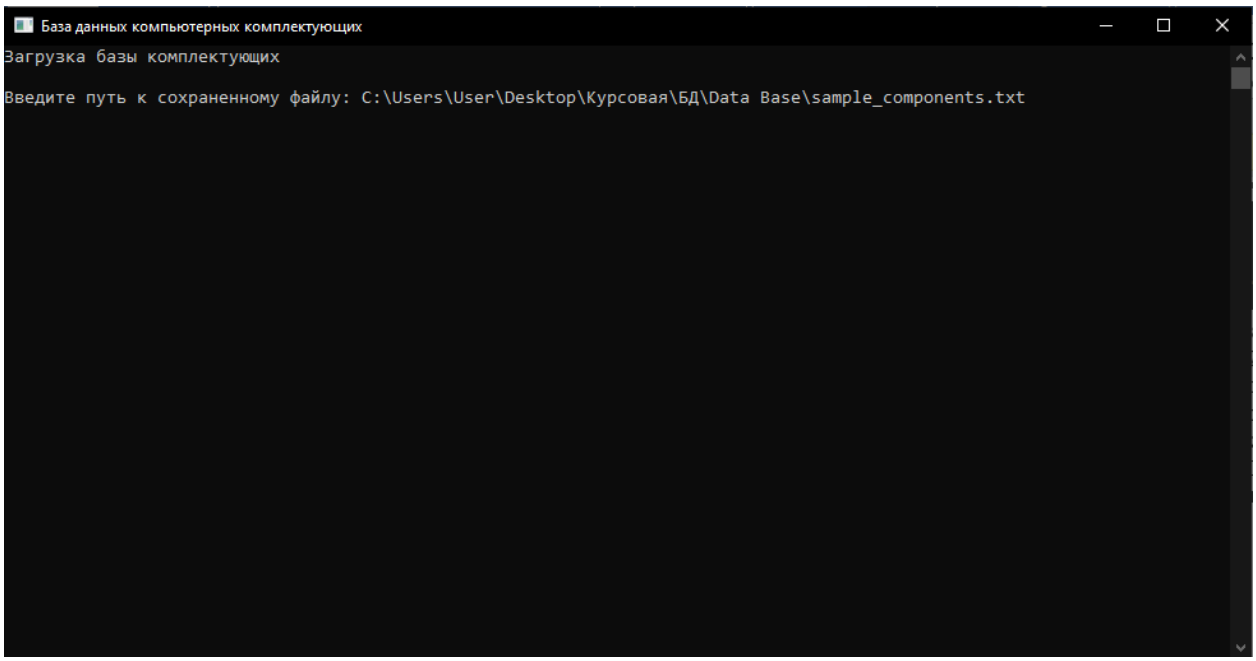

Рисунок 23 – Ввод пути для загрузки базы данных

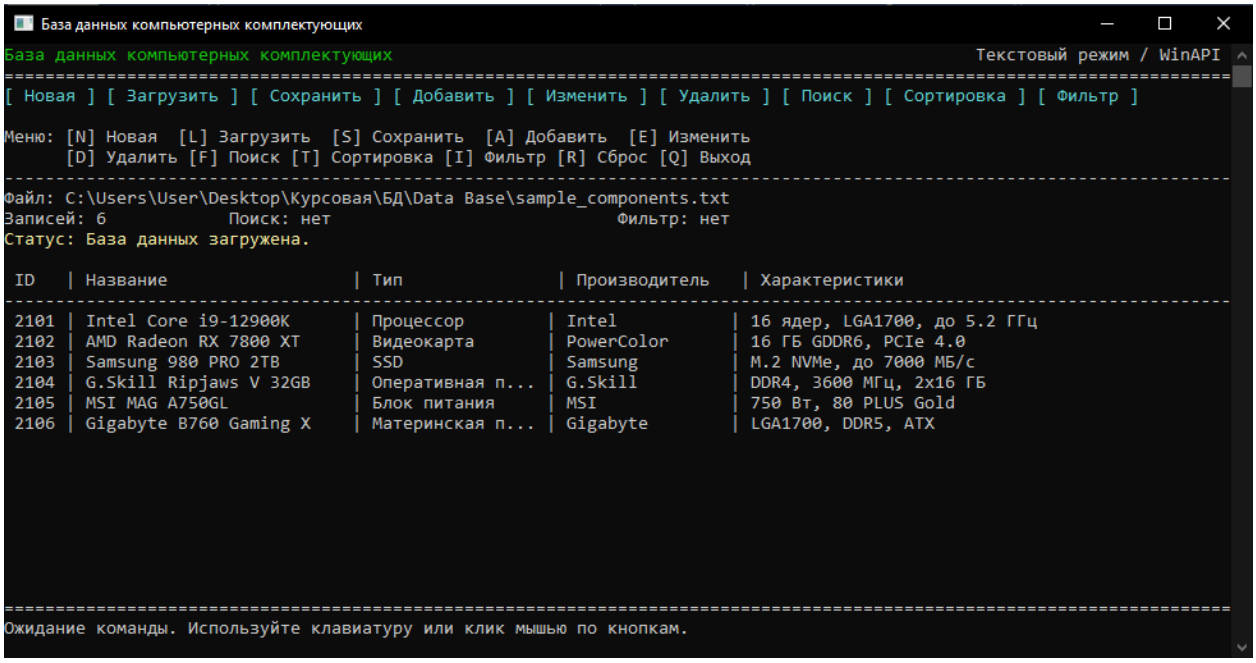

Рисунок 24 – Результат загрузки базы данных

Для добавления записи в базу данных необходимо выбрать пункт меню «Добавить» или нажать горячую клавишу «А». Далее необходимо заполнить форму записи: ввести ID комплектующего, нажать клавишу Enter, ввести название комплектующего, нажать клавишу Enter, ввести тип комплектующего, нажать клавишу Enter, ввести производителя, нажать клавишу Enter, ввести характеристики комплектующего и снова нажать

клавишу Enter. Заполнение формы добавления записи представлено на рисунке 25. Если запись с таким ID уже существует, программа выдаст сообщение об ошибке. Если ошибкине возникли, на экране появится надпись «Запись добавлена», результат добавления представлен на рисунке 26.

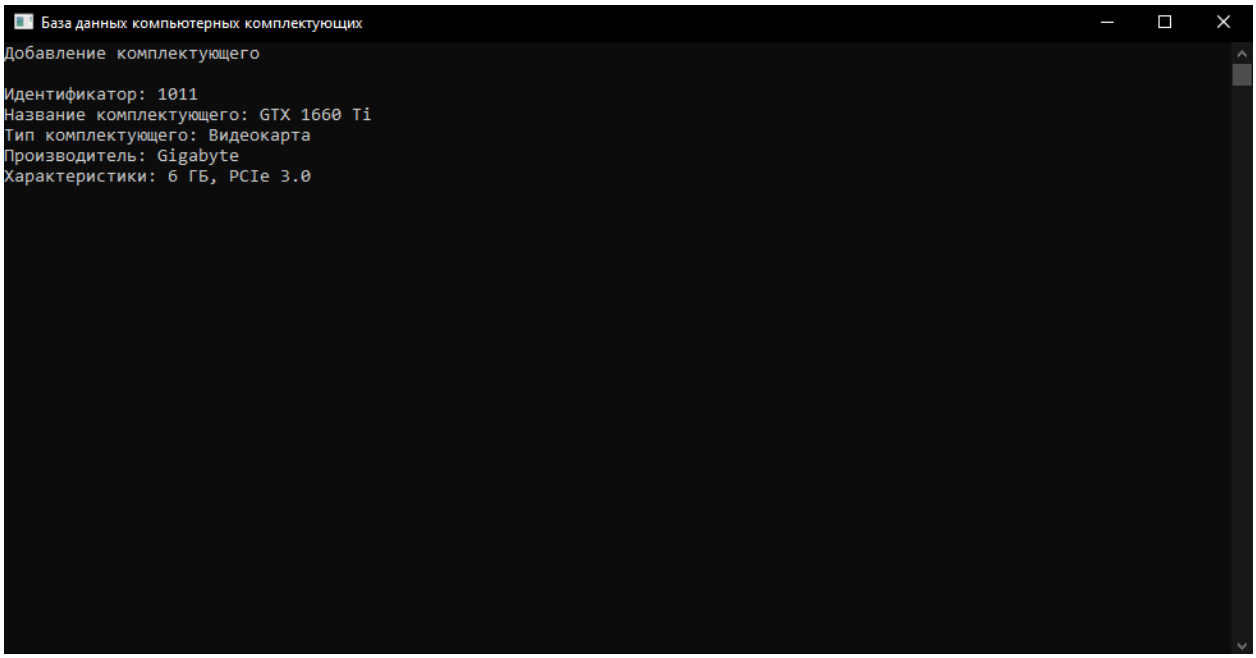

Рисунок 25 – Заполнение формы добавления записи

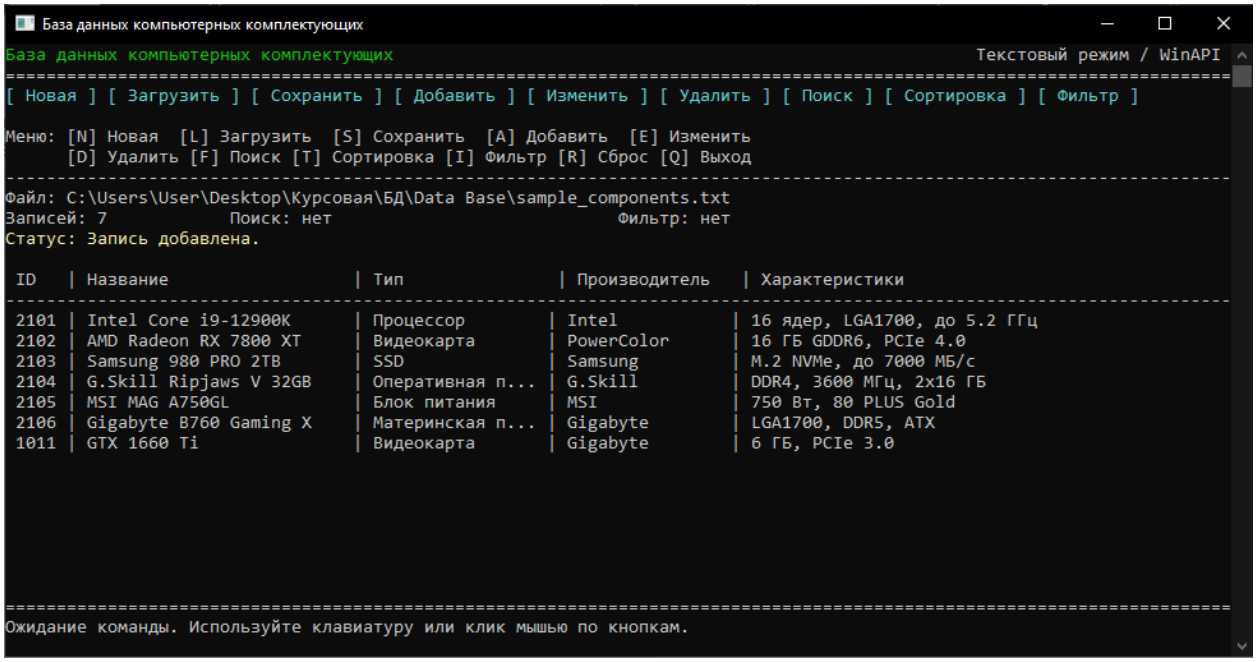

Рисунок 26 – Результат добавления записи

Для удаления записи из базы данных необходимо выбрать пункт меню «Удалить» или нажать горячую клавишу «D». Далее необходимо ввести ID

удаляемой записи и нажать клавишу Enter. Ввод ID удаляемой записи представлен на рисунке 27. Программа выполнит поиск записи с указанным идентификатором. Если запись с таким ID не найдена, программа выдаст сообщение об ошибке. Если ошибки не возникли, на экране появится надпись «Запись удалена», результат удаления представлен на рисунке 28.

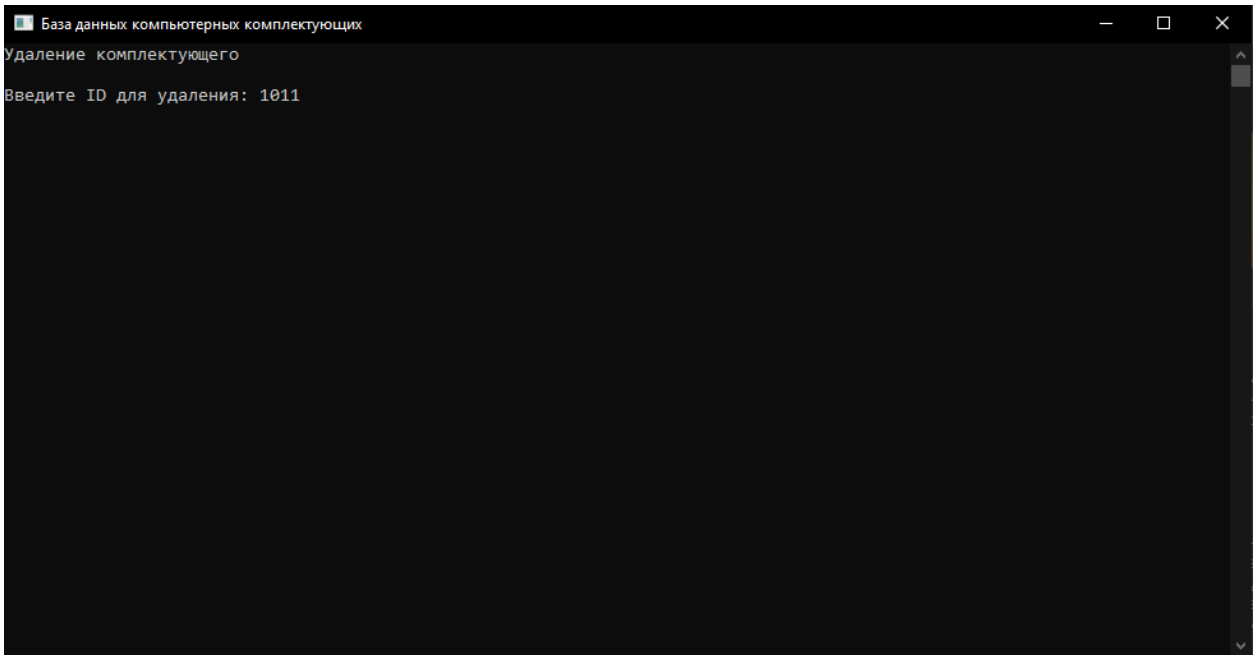

Рисунок 27 – Ввод ID удаляемой записи

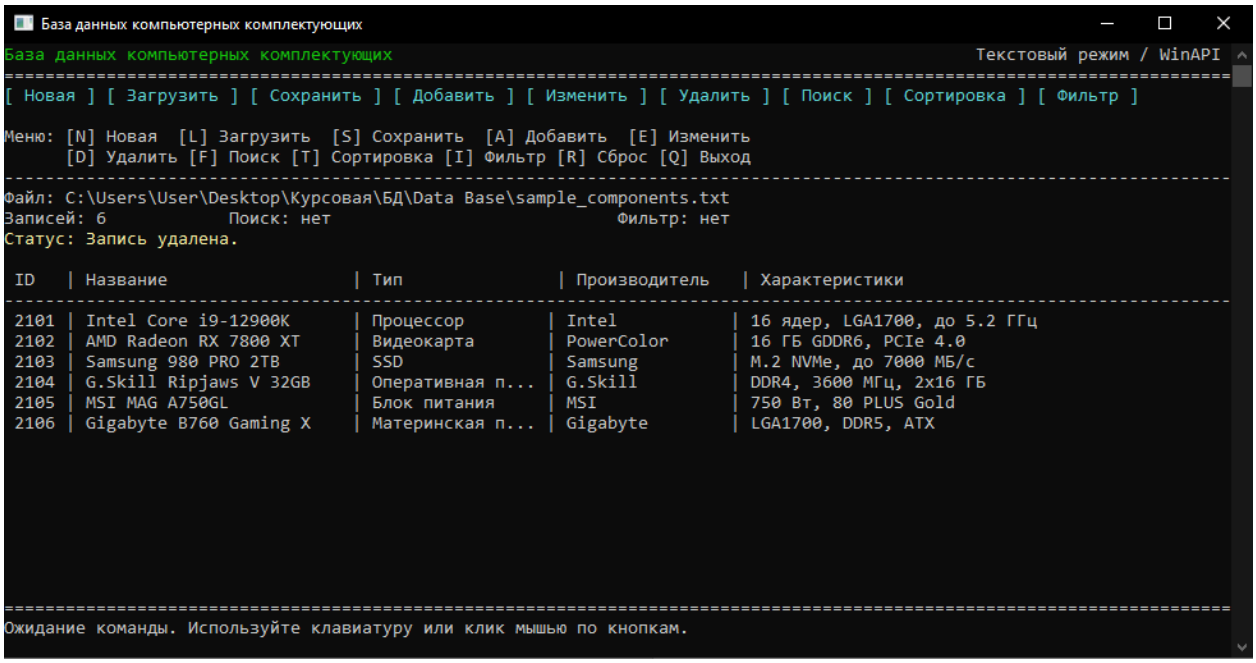

Рисунок 28 – Результат удаления записи

Для изменения записи необходимо выбрать пункт меню «Изменить» или нажать горячую клавишу «E». Далее необходимо ввести ID записи, которую требуется изменить, и нажать клавишу Enter. Ввод ID записи для изменения представлен на рисунке 29. Если запись с указанным ID найдена, программа предложит ввести новые данные: новый ID комплектующего, название комплектующего, тип комплектующего, производителя и характеристики. После ввода каждого значения необходимо нажимать клавишу Enter. Если новый ID уже используется другой записью, программа выдаст сообщение об ошибке. Если ошибки не возникли, на экране появится надпись «Запись изменена», результат изменения представлен на рисунке 30.

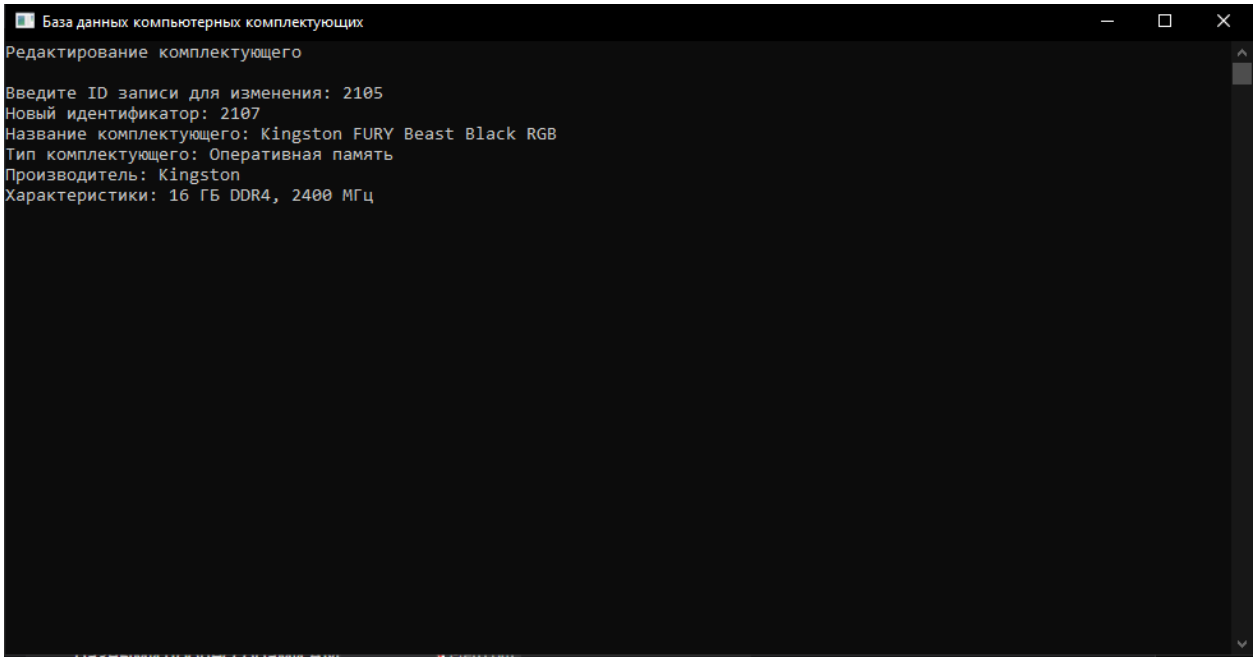

Рисунок 29 – Ввод ID записи для изменения

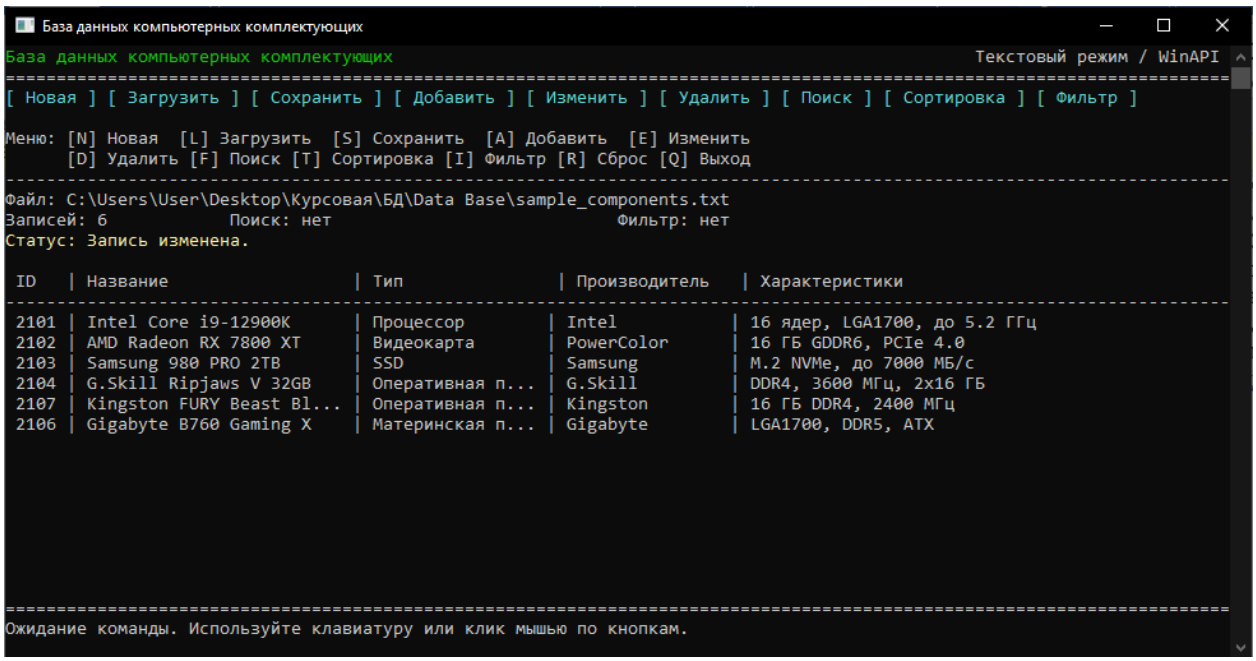

Рисунок 30 – Результат изменения записи

Для того чтобы сохранить текущую базу данных в файл, необходимо выбрать пункт меню «Сохранить» или нажать горячую клавишу «S». Далее нужно ввести путь к файлу, в который будет записана база данных, и нажать клавишу Enter. Ввод пути для сохранения базы данных представлен на рисунке 31. Если файл уже существует, программа перезапишет его и сохранит в него текущие записи базы данных. Если путь к файлу не был введён, программа использует текущий файл, если он был выбран ранее. В случае ошибки программа выдаст сообщение о невозможности открыть файл для записи. Если ошибки не возникли, на экране появится надпись «База данных сохранена», результат сохранения представлен на рисунке 32.

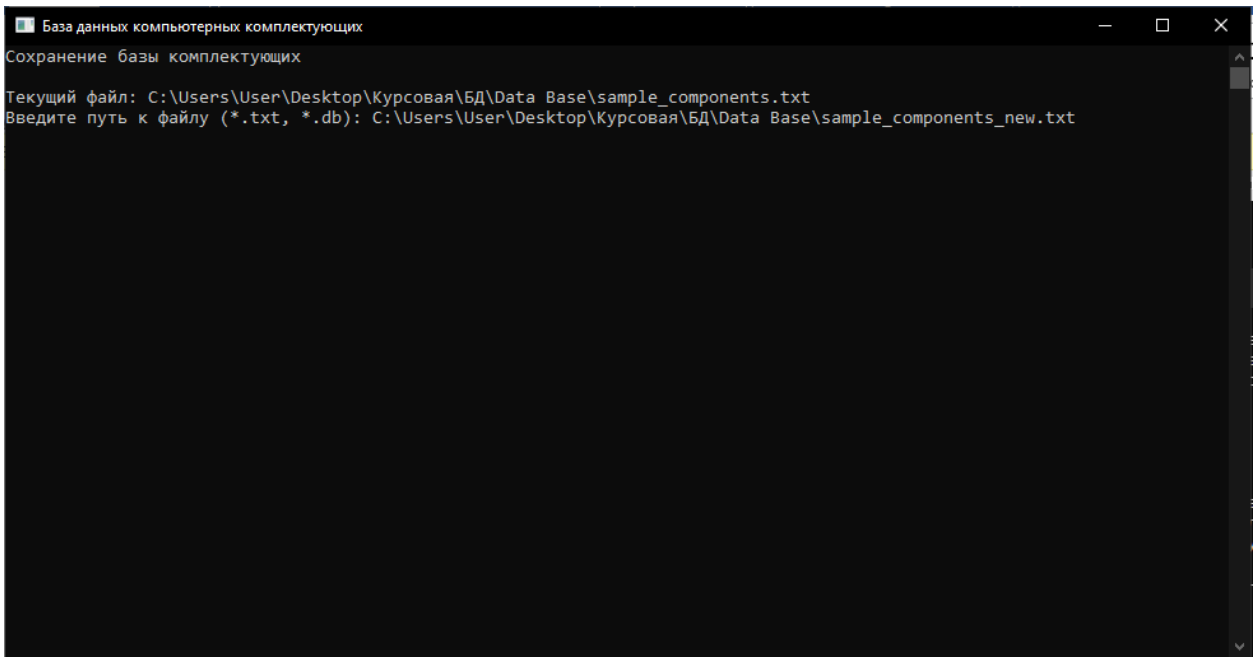

Рисунок 31 – Ввод пути для сохранения базы данных

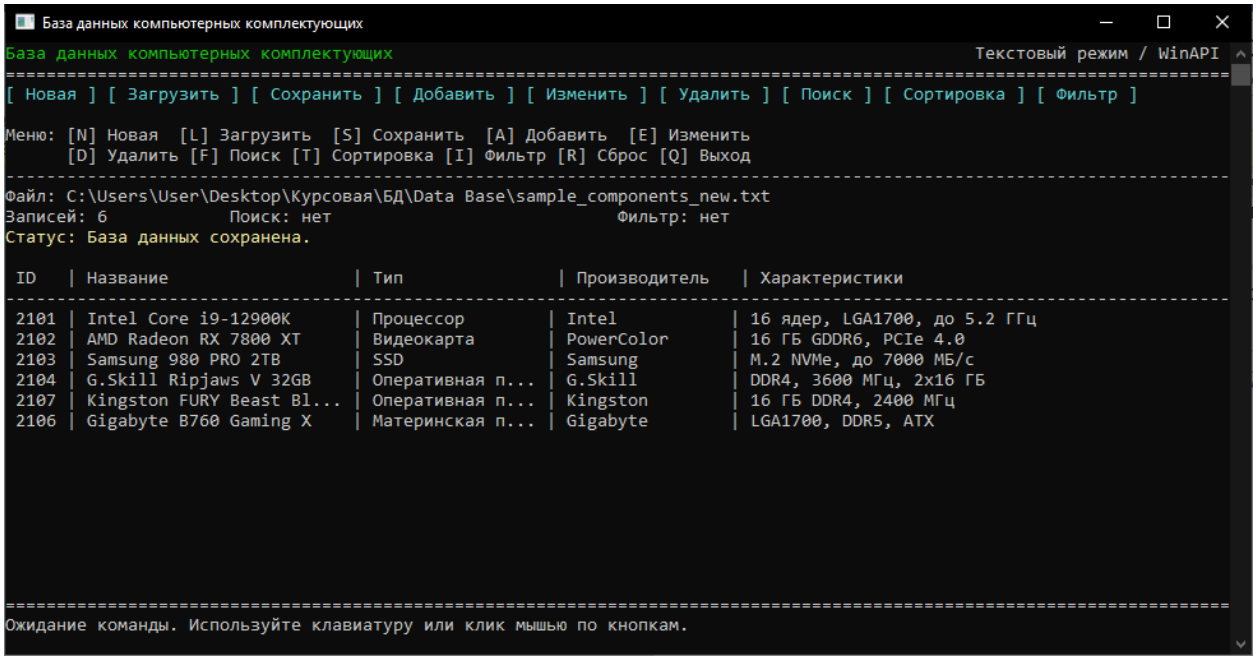

Рисунок 32 – Результат сохранения базы данных

Для того чтобы записи были выведены в отсортированном виде, необходимо выбрать пункт меню «Сортировка» или нажать горячую клавишу «T». Далее программа предложит выбрать один из трёх параметров сортировки: сортировка по ID, сортировка по названию комплектующего или сортировка по типу комплектующего. Выбор параметра сортировки представлен на рисунке 33. После выбора параметра необходимо указать

направление сортировки: по возрастанию или по убыванию, затем нажать клавишу Enter. После выполнения операции записи в таблице будут отображены в выбранном порядке; примеры сортировки представлены на рисунках 34–37.

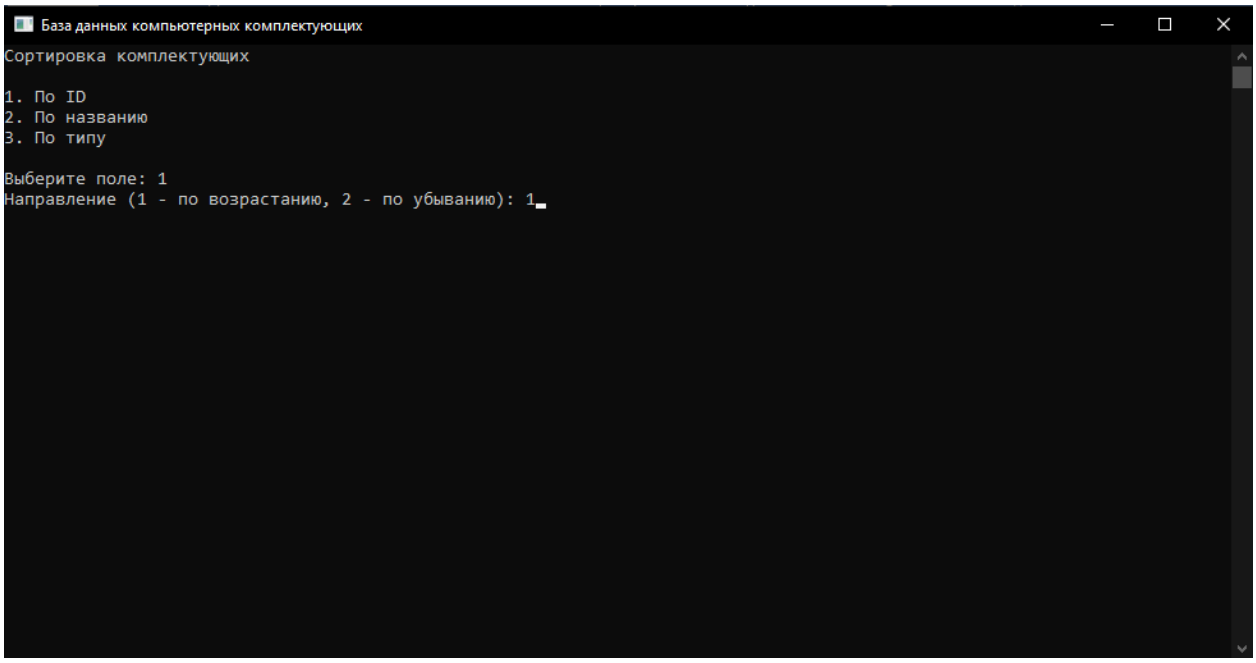

Рисунок 33 – Выбор параметра сортировки

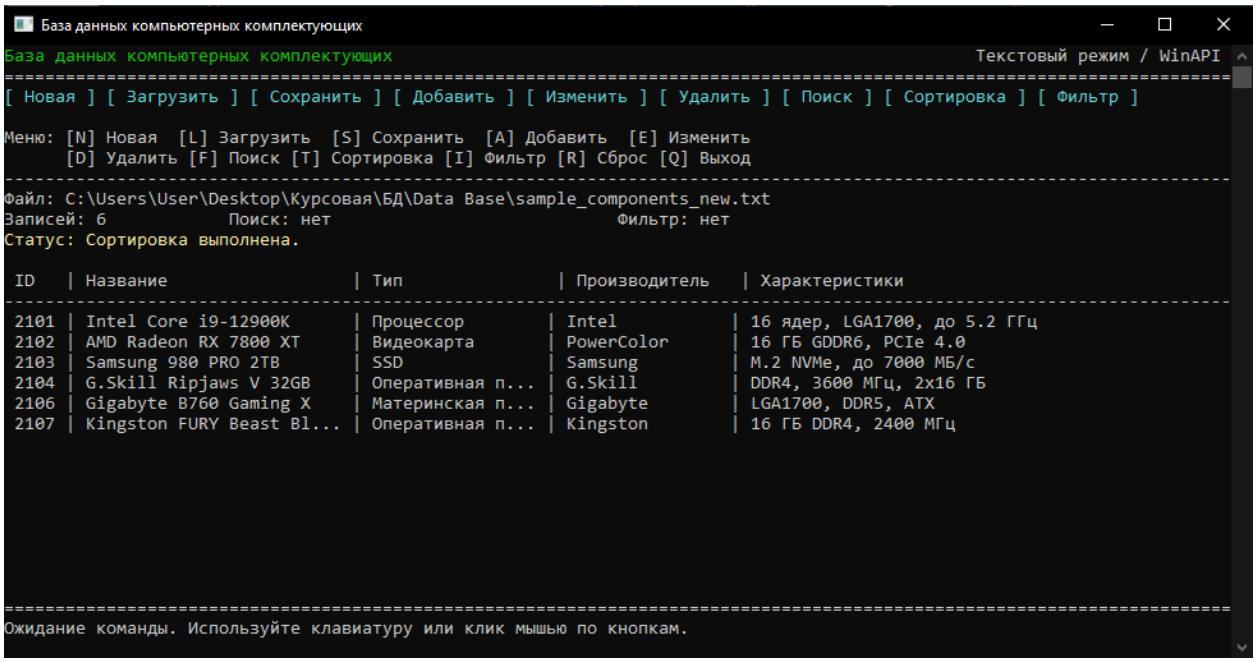

Рисунок 34 – Сортировка ID по возрастанию

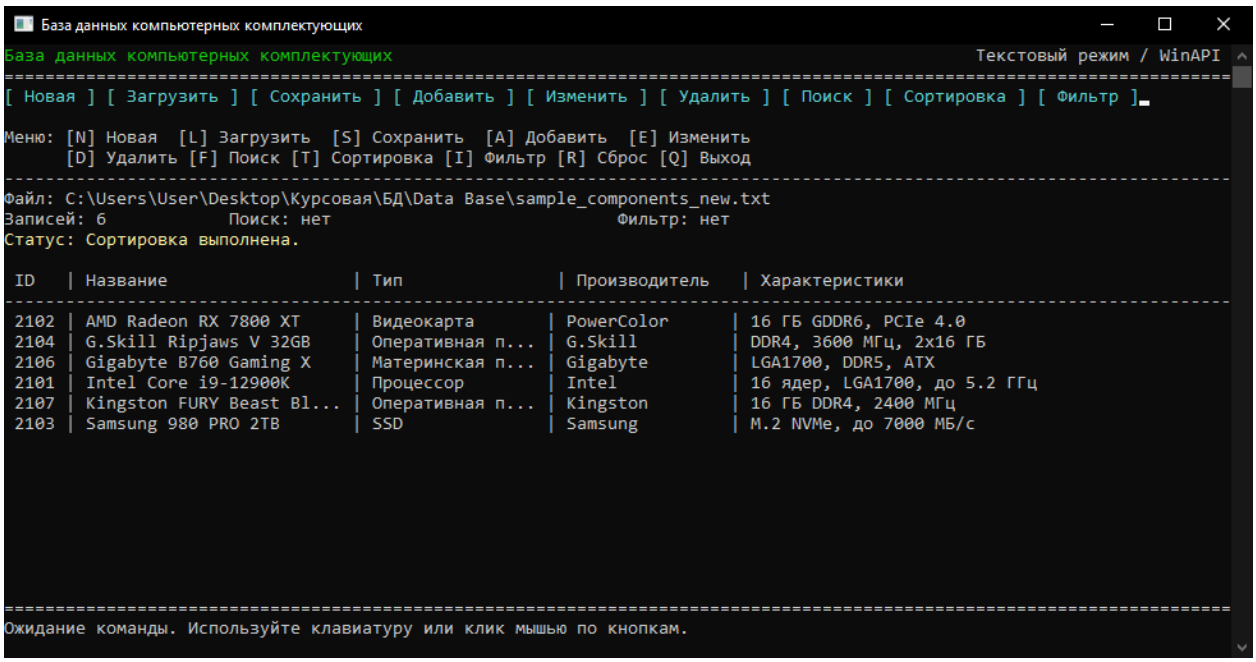

Рисунок 35 – Сортировка названия по возрастанию

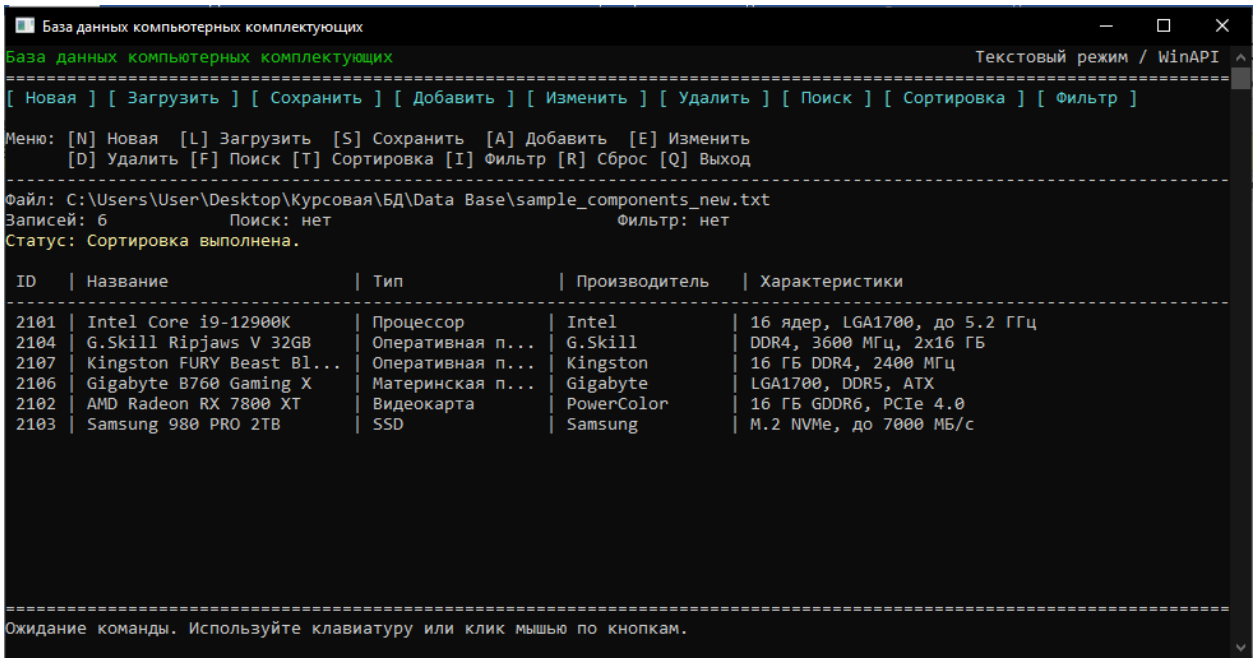

Рисунок 36 – Сортировка типа по убыванию

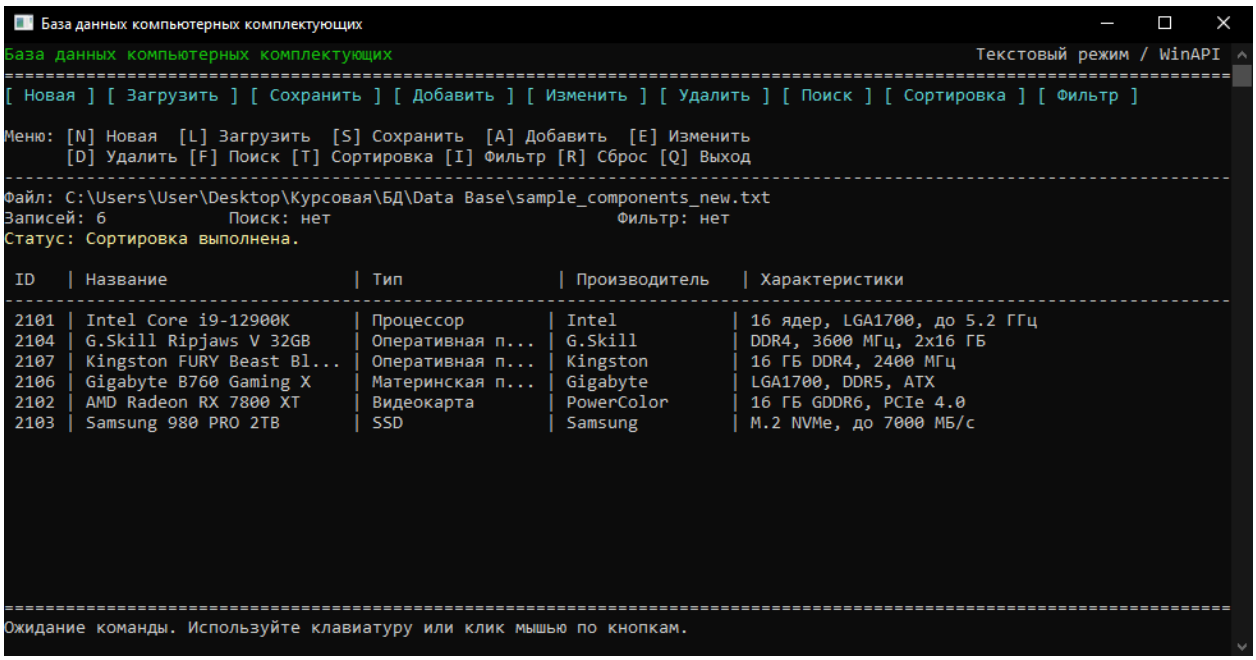

Рисунок 37 – Итоговый вид отсортированных записей

Для того чтобы найти определённую запись или группу записей по заданному критерию, необходимо выбрать пункт меню «Поиск» или нажать горячую клавишу «F». Далее нужно ввести строку, по которой будет выполняться поиск, и нажать клавишу Enter. Ввод поискового запроса по типу представлен на рисунке 38, результат поиска по типу — на рисунке 39. Ввод поискового запроса по характеристике представлен на рисунке 40, результат поиска по характеристике — на рисунке 41. Ввод поискового запроса по названию/производителю представлен на рисунке 42, результат поиска по названию/производителю — на рисунке 43. Программа выполнит поиск совпадений по всем основным полям записи: ID, названию комплектующего, типу комплектующего, производителю и характеристикам.

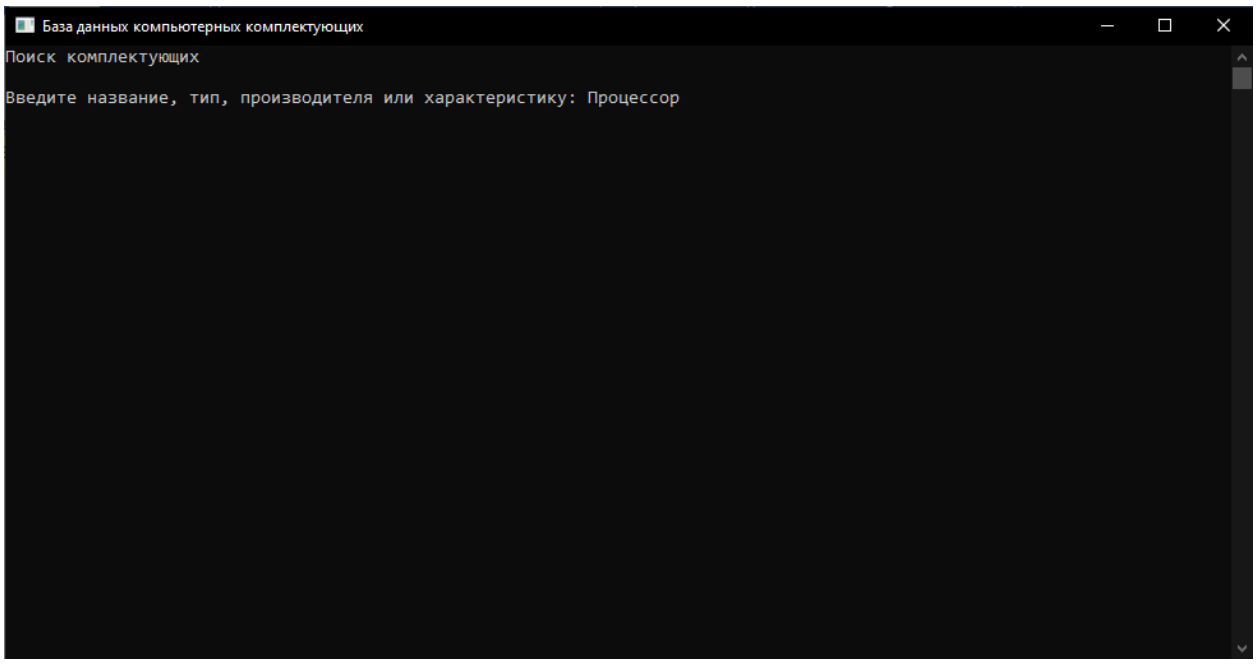

Рисунок 38 – Ввод поискового запроса по типу

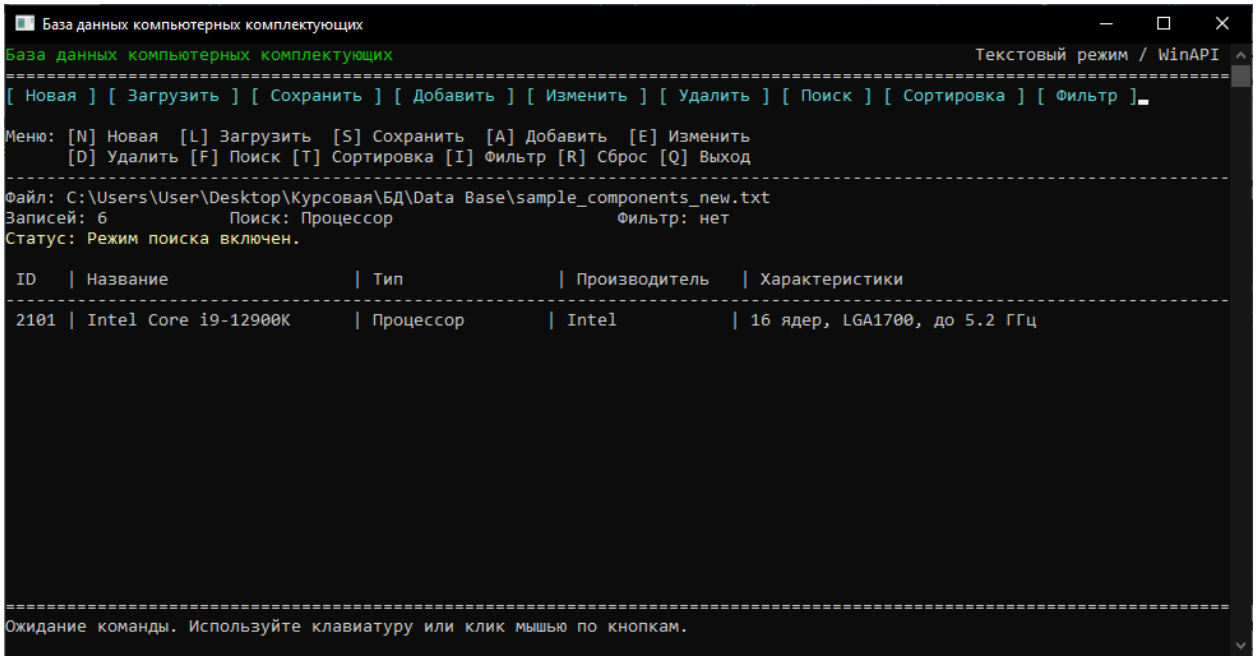

Рисунок 39 – Результат поиска по типу

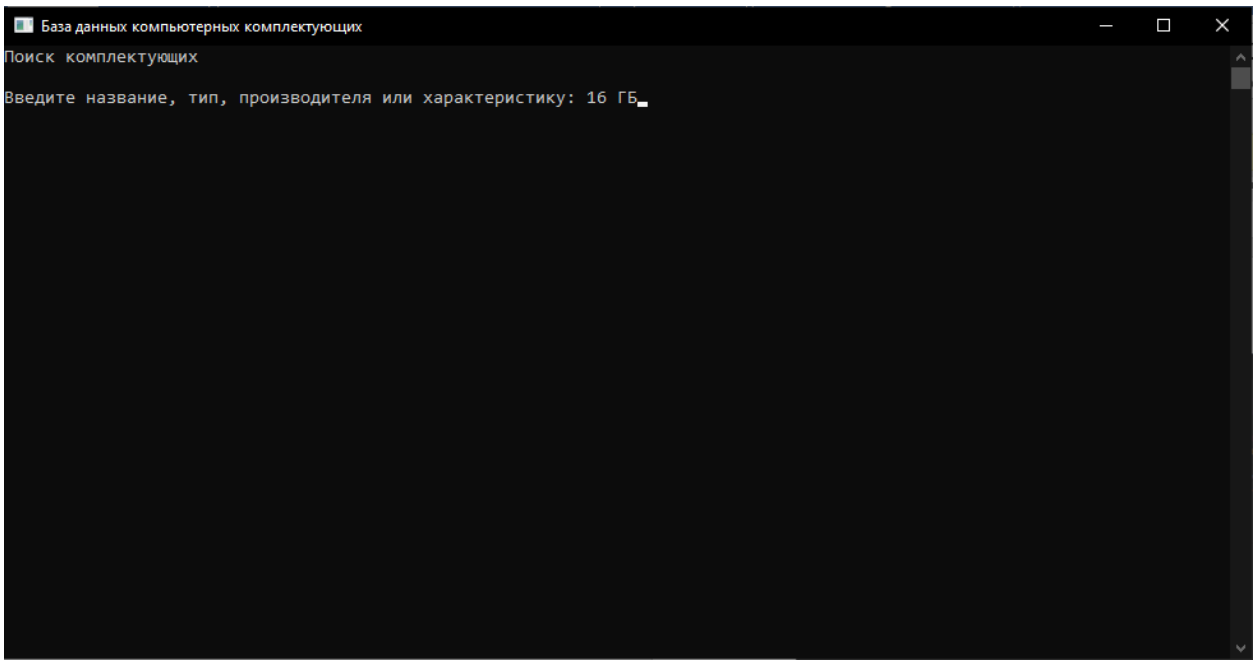

Рисунок 40 – Ввод поискового запроса по характеристике

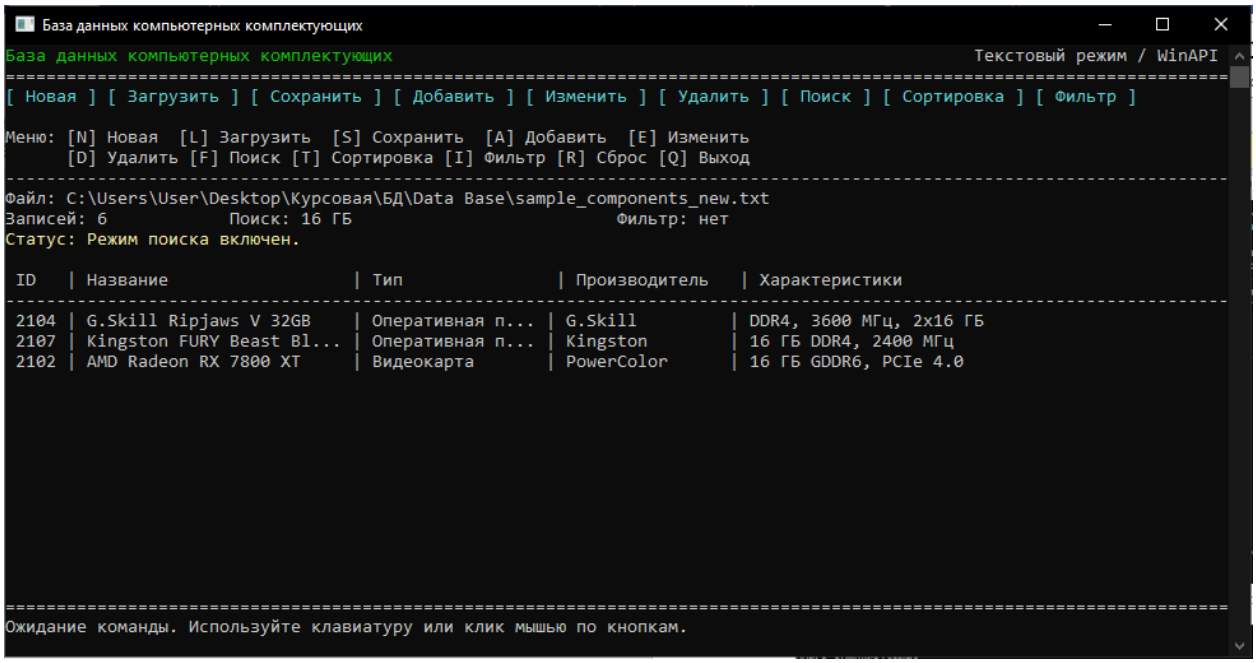

Рисунок 41 – Результат поиска по характеристике

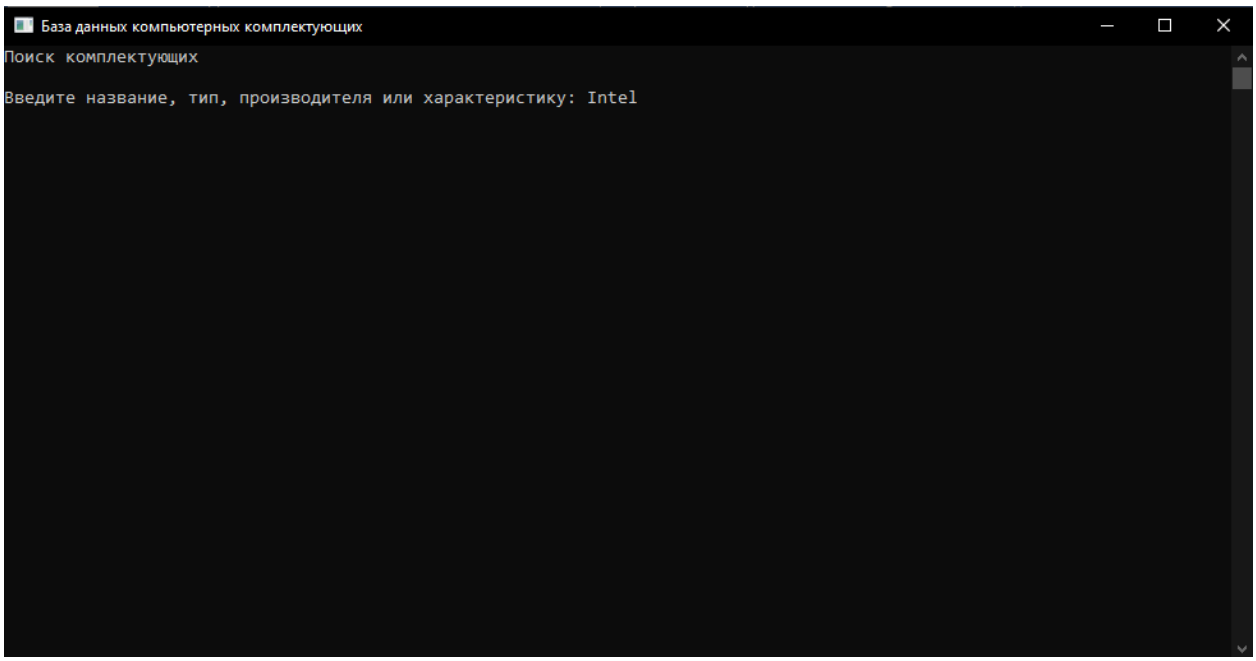

Рисунок 42 – Ввод поискового запроса по названию/производителю

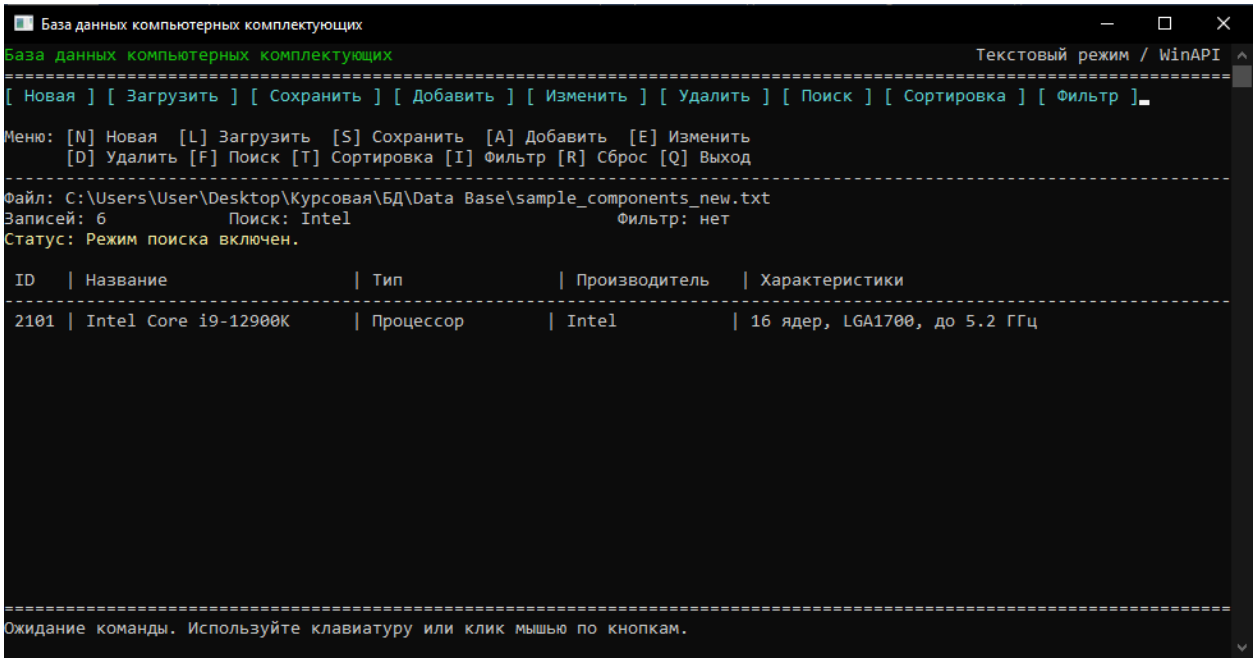

Рисунок 43 – Результат поиска по названию/производителю

Если необходимо вывести записи только определённого типа, следует выбрать пункт меню «Фильтр» или нажать горячую клавишу «I». Далее нужно ввести тип комплектующего или его часть и нажать клавишу Enter. Ввод значения фильтра представлен на рисунке 44. После этого программа отобразит только те записи, у которых поле «Тип комплектующего» соответствует введённому значению; результат фильтрации представлен на

рисунке 45. Например, можно выполнить фильтрацию по значениям «Процессор», «Видеокарта», «SSD», «Оперативная память», «Материнская плата» или «Блок питания».

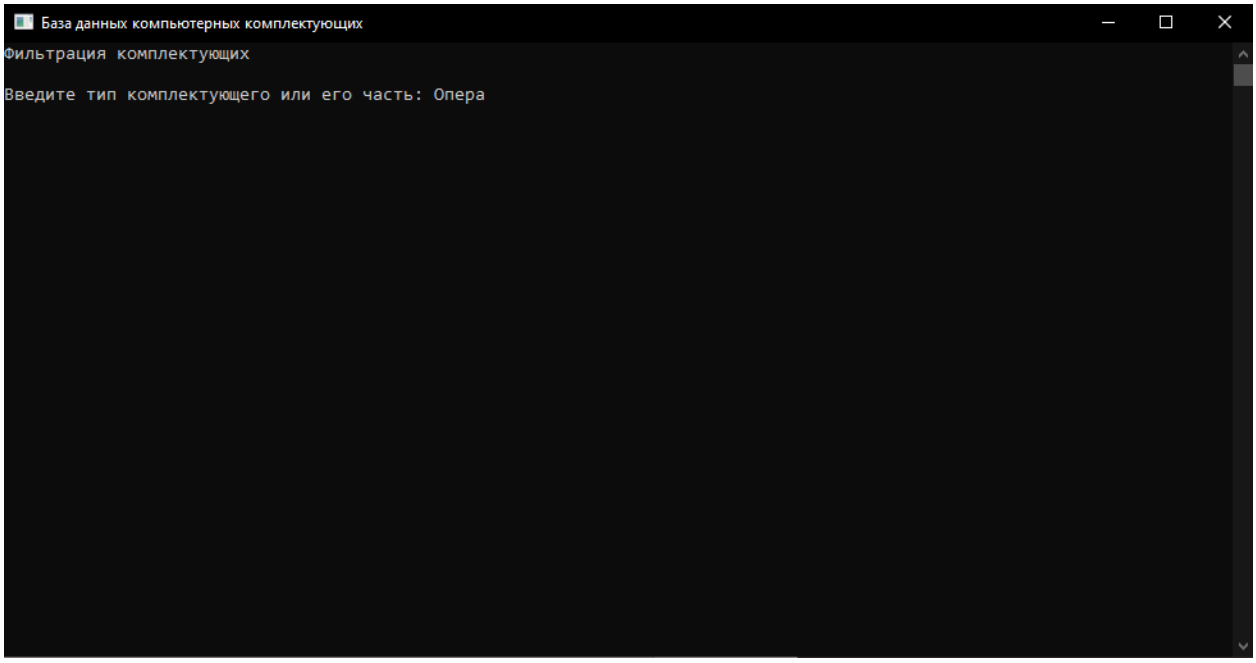

Рисунок 44 – Ввод значения фильтра

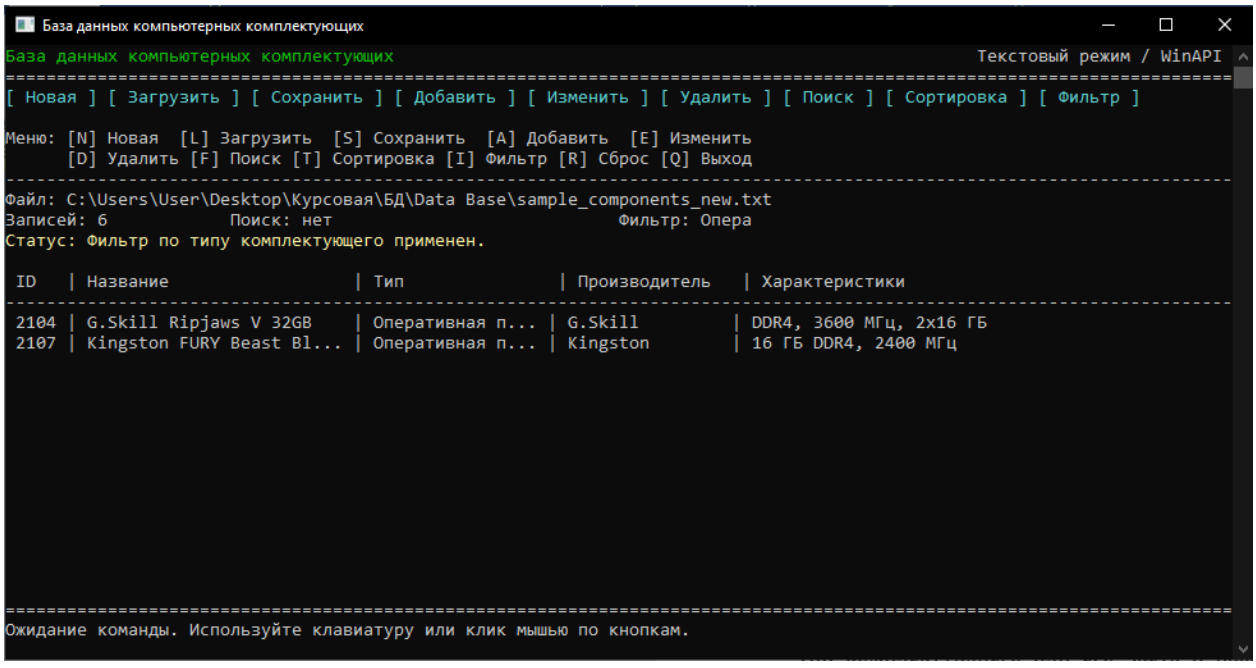

Рисунок 45 – Результат фильтрации по типу

Для сброса результатов поиска и фильтрации необходимо выбрать пункт меню «Сброс» или нажать горячую клавишу «R». Выбор сброса результатов поиска и фильтрации представлен на рисунке 46. После выбора данного

пункта программа очищает активные параметры поиска и фильтрации. Затем на экране снова отображаются все записи текущей базы данных. Если ошибки не возникли, появляется сообщение «Поиск и фильтрация сброшены», результат сброса представлен на рисунке 47.

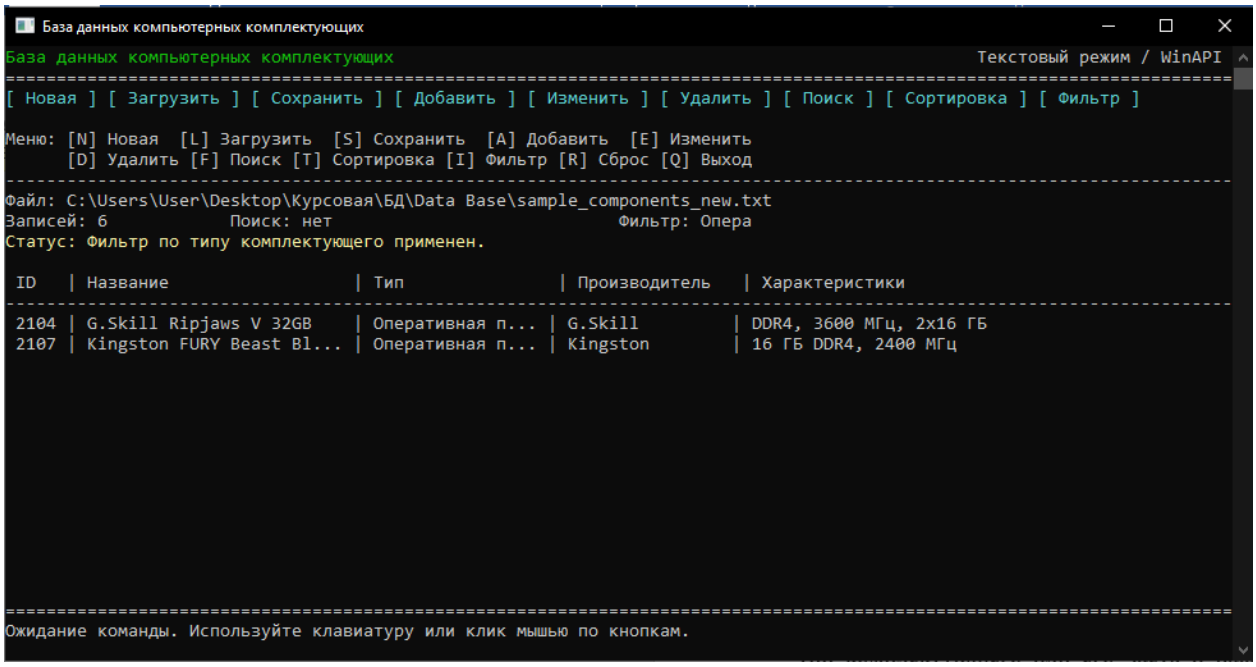

Рисунок 46 – Выбор сброса результатов поиска и фильтрации

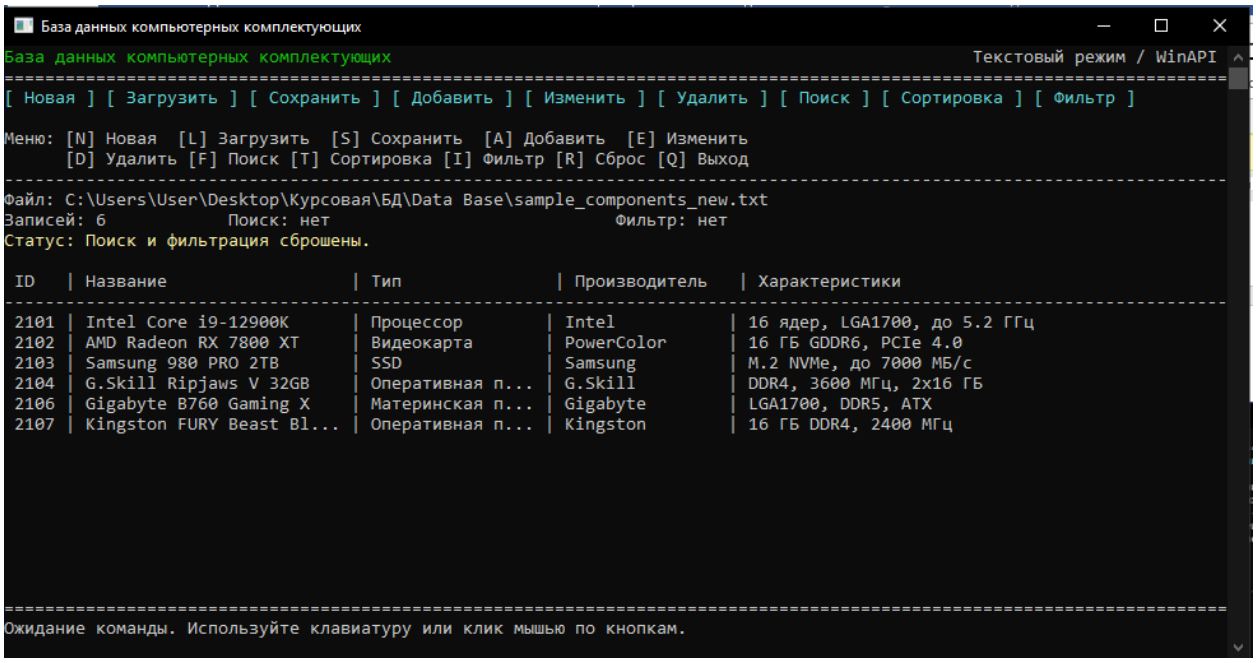

Рисунок 47 – Результат сброса поиска и фильтрации

Для создания новой базы данных необходимо выбрать пункт меню «Новая» или нажать горячую клавишу «N». После выбора данного пункта

программа очищает текущий список записей, сбрасывает выбранный файл, а также удаляет активные параметры поиска и фильтрации. После этого пользователь может заново добавлять записи о компьютерных комплектующих. Если операция выполнена успешно, на экране появится сообщение «Создана новая пустая база комплектующих», результат создания новой базы данных представлен на рисунке 48.

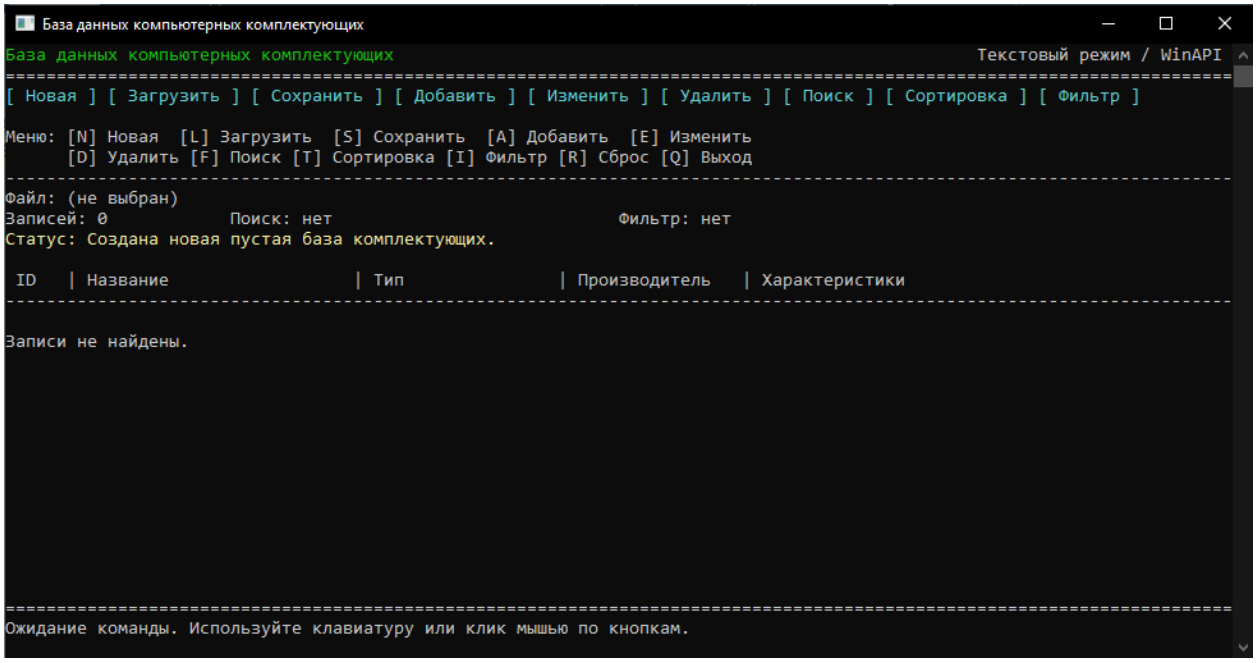

Рисунок 48 – Результат создания новой базы данных

Для выхода из программы необходимо выбрать пункт меню «Выход» или нажать горячую клавишу «Q». После выбора данного пункта программа завершит работу и закроет основное меню.

Заключение

В результате выполнения курсовой работы была разработана текстовая программа для хранения данных о компьютерных комплектующих. Программа реализует создание, добавление, редактирование, удаление, поиск, сортировку, фильтрацию, сохранение и загрузку данных. В работе были использованы язык C++, стандартная библиотека Си/C++, средства WinAPI и ассемблерная вставка [6], [8], [9], [10], [11].

Кроме того, программу можно дополнить графическим интерфейсом, который сделает работу пользователя более удобной и наглядной. Для повышения надёжности можно добавить подтверждение перед удалением записей и перед созданием новой базы данных, чтобы пользователь случайно не потерял введённые данные.

Список изученных источников

Печатные издания

1. Ахо, А. Компиляторы: принципы, технологии и инструменты / А. Ахо, Р. Сети, Дж. Ульман. – 2-е изд. – М. : Вильямс, 2008. – 1184 с.

2. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. – М. : ДМК Пресс, 2010. – 272 с.

3. Касьянов, В. Н. Методы построения трансляторов / В. Н. Касьянов, И. В. Поттосин. – Новосибирск : Наука, 1986. – 344 с.

4. Керниган, Б. Язык программирования С / Б. Керниган, Д. Ритчи. – 2-е изд. – М. : Вильямс, 2009. – 304 с.

5. Макконнелл, С. Совершенный код. Мастер-класс / С. Макконнелл. – М. : Русская редакция, 2010. – 896 с.

6. Пильщиков, В. Н. Программирование на языке ассемблера IBM PC / В. Н. Пильщиков. – М. : Диалог-МИФИ, 2005. – 288 с.

7. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – 5-е изд. – СПб. : Питер, 2007. – 844 с.

Электронные издания

8. Страуструп Б. Язык программирования C++.

9. Microsoft Learn. Документация по WinAPI.

10. Microsoft Learn. Документация по стандартной библиотеке C++.

11. Руководство по Си/C++. Metanit.com

Приложение А
Листинг программы

Файл main.cpp

```
#include "console_ui.h"
#include "database.h"

int wmain() {
    Database database;
    ConsoleUi ui(database);
    ui.Run();
    return 0;
}
```

Файл database.cpp

```
#include "database.h"

#include "asm_utils.h"

#include <Windows.h>

#include <algorithm>
#include <cwctype>
#include <fstream>
#include <sstream>

namespace {

std::string ToUtf8(const std::wstring& value) {
    if (value.empty()) {
        return {};
    }

    const int required = WideCharToMultiByte(CP_UTF8, 0,
value.c_str(), static_cast<int>(value.size()), nullptr, 0,
nullptr, nullptr);
```

```
    std::string result(required, '\0');
    WideCharToMultiByte(CP_UTF8, 0, value.c_str(),
static_cast<int>(value.size()), &result[0], required, nullptr,
nullptr);
    return result;
}

std::wstring FromUtf8(const std::string& value) {
    if (value.empty()) {
        return {};
    }

    const int required = MultiByteToWideChar(CP_UTF8, 0,
value.c_str(), static_cast<int>(value.size()), nullptr, 0);
    std::wstring result(required, L'\0');
    MultiByteToWideChar(CP_UTF8, 0, value.c_str(),
static_cast<int>(value.size()), &result[0], required);
    return result;
}

std::wstring Escape(const std::wstring& value) {
    std::wstring escaped;
    escaped.reserve(value.size());

    for (const wchar_t symbol : value) {
        switch (symbol) {
        case L'\\':
            escaped += L"\\\\";
            break;
        case L'|':
            escaped += L"\\p";
            break;
        case L'\n':
            escaped += L"\\n";
            break;
```

```
        case L'\r':
            break;
        default:
            escaped += symbol;
            break;
        }
    }

    return escaped;
}

std::wstring Unescape(const std::wstring& value) {
    std::wstring result;
    result.reserve(value.size());

    for (size_t index = 0; index < value.size(); ++index) {
        if (value[index] == L'\\' && index + 1 < value.size()) {
            const wchar_t next = value[index + 1];
            if (next == L'\\') {
                result += L'\\';
                ++index;
                continue;
            }
            if (next == L'p') {
                result += L'|';
                ++index;
                continue;
            }
            if (next == L'n') {
                result += L'\n';
                ++index;
                continue;
            }
        }
        result += value[index];
```

```
    }

    return result;
}

std::vector<std::wstring> Split(const std::wstring& line) {
    std::vector<std::wstring> parts;
    std::wstring current;

    for (size_t index = 0; index < line.size(); ++index) {
        if (line[index] == L'|' && (index == 0 || line[index -
1] != L'\\')) {
            parts.push_back(current);
            current.clear();
            continue;
        }
        current += line[index];
    }

    parts.push_back(current);
    return parts;
}

bool ContainsInsensitive(const std::wstring& source, const
std::wstring& query) {
    std::wstring left = source;
    std::wstring right = query;

    std::transform(left.begin(), left.end(), left.begin(),
towlower);
    std::transform(right.begin(), right.end(), right.begin(),
towlower);
    return left.find(right) != std::wstring::npos;
}
```

```
}

void Database::CreateNew() {
    records_.clear();
    currentFilePath_.clear();
}

bool Database::AddRecord(const Record& record) {
    if (ContainsId(record.id)) {
        return false;
    }

    records_.push_back(record);
    return true;
}

bool Database::UpdateRecord(int id, const Record& updatedRecord)
{
    for (Record& record : records_) {
        if (record.id == id) {
            if (updatedRecord.id != id &&
ContainsId(updatedRecord.id)) {
                return false;
            }
            record = updatedRecord;
            return true;
        }
    }
    return false;
}

bool Database::RemoveRecord(int id) {
    const auto previousSize = records_.size();
    records_.remove_if([id](const Record& record) { return
record.id == id; });
```

```
    return previousSize != records_.size();
}

bool Database::ContainsId(int id) const {
    return std::any_of(records_.begin(), records_.end(),
[id](const Record& record) {
        return record.id == id;
        });
}

std::vector<Record> Database::GetAllRecords() const {
    return { records_.begin(), records_.end() };
}

std::vector<Record> Database::Search(const std::wstring& query)
const {
    std::vector<Record> matches;

    for (const Record& record : records_) {
        if (ContainsInsensitive(std::to_wstring(record.id),
query) ||
            ContainsInsensitive(record.componentName, query) ||
            ContainsInsensitive(record.componentType, query) ||
            ContainsInsensitive(record.manufacturer, query) ||
            ContainsInsensitive(record.specifications, query)) {
            matches.push_back(record);
        }
    }

    return matches;
}

std::vector<Record> Database::FilterByType(const std::wstring&
componentType) const {
    std::vector<Record> matches;
```

```
    for (const Record& record : records_) {
        if (ContainsInsensitive(record.componentType,
componentType)) {
            matches.push_back(record);
        }
    }

    return matches;
}

bool Database::SortBy(SortField field, bool ascending) {
    if (records_.empty()) {
        return false;
    }

    records_.sort([field, ascending](const Record& left, const
Record& right) {
        int comparison = 0;

        switch (field) {
        case SortField::Id:
            comparison = CompareIntegersAsm(left.id, right.id);
            break;
        case SortField::ComponentName:
            comparison =
left.componentName.compare(right.componentName);
            break;
        case SortField::ComponentType:
            comparison =
left.componentType.compare(right.componentType);
            break;
        }

        return ascending ? comparison < 0 : comparison > 0;
```

```
        });

    return true;
}

bool Database::SaveToFile(const std::wstring& path,
std::wstring& errorMessage) {
    std::ofstream output(path, std::ios::binary |
std::ios::trunc);
    if (!output) {
        errorMessage = L"Не удалось открыть файл для записи.";
        return false;
    }

    output << "DBV1\n";
    for (const Record& record : records_) {
        const std::wstring line = std::to_wstring(record.id) +
L"|" +
            Escape(record.componentName) + L"|" +
            Escape(record.componentType) + L"|" +
            Escape(record.manufacturer) + L"|" +
            Escape(record.specifications);
        output << ToUtf8(line) << "\n";
    }

    currentFilePath_ = path;
    errorMessage.clear();
    return true;
}

bool Database::LoadFromFile(const std::wstring& path,
std::wstring& errorMessage) {
    std::ifstream input(path, std::ios::binary);
    if (!input) {
        errorMessage = L"Не удалось открыть файл для чтения.";
```

```
        return false;
    }

    std::string header;
    if (!std::getline(input, header) || header != "DBV1") {
        errorMessage = L"Формат файла не поддерживается.";
        return false;
    }

    std::list<Record> loadedRecords;
    std::string lineUtf8;
    while (std::getline(input, lineUtf8)) {
        if (!lineUtf8.empty() && lineUtf8.back() == '\r') {
            lineUtf8.pop_back();
        }

        const std::wstring line = FromUtf8(lineUtf8);
        if (line.empty()) {
            continue;
        }

        const std::vector<std::wstring> parts = Split(line);
        if (parts.size() != 5) {
            errorMessage = L"Файл содержит поврежденные
записи.";
            return false;
        }

        Record record;
        try {
            record.id = std::stoi(parts[0]);
        }
        catch (...) {
            errorMessage = L"Файл содержит некорректный
идентификатор.";
```

```
            return false;
        }

        record.componentName = Unescape(parts[1]);
        record.componentType = Unescape(parts[2]);
        record.manufacturer = Unescape(parts[3]);
        record.specifications = Unescape(parts[4]);
        loadedRecords.push_back(record);
    }

    records_ = std::move(loadedRecords);
    currentFilePath_ = path;
    errorMessage.clear();
    return true;
}

const std::wstring& Database::CurrentFilePath() const {
    return currentFilePath_;
}

size_t Database::Size() const {
    return records_.size();
}
```

Файл console_utils.cpp

```
#include "console_utils.h"

#include <fcntl.h>
#include <io.h>

#include <iostream>
#include <limits>

namespace {
```

```
constexpr WORD kDefaultColor = FOREGROUND_RED | FOREGROUND_GREEN
| FOREGROUND_BLUE;

HANDLE g_inputHandle = nullptr;
HANDLE g_outputHandle = nullptr;

void EnsureHandles() {
    if (g_inputHandle == nullptr) {
        g_inputHandle = GetStdHandle(STD_INPUT_HANDLE);
    }
    if (g_outputHandle == nullptr) {
        g_outputHandle = GetStdHandle(STD_OUTPUT_HANDLE);
    }
}

}

namespace console {

void InitializeConsole() {
    EnsureHandles();

    _setmode(_fileno(stdout), _O_U16TEXT);
    _setmode(_fileno(stdin), _O_U16TEXT);
    _setmode(_fileno(stderr), _O_U16TEXT);

    SetConsoleOutputCP(CP_UTF8);
    SetConsoleCP(CP_UTF8);

    DWORD mode = 0;
    GetConsoleMode(g_inputHandle, &mode);
    mode |= ENABLE_EXTENDED_FLAGS;
    mode &= ~ENABLE_QUICK_EDIT_MODE;
    mode |= ENABLE_MOUSE_INPUT;
    mode |= ENABLE_WINDOW_INPUT;
```

```
    SetConsoleMode(g_inputHandle, mode);

    CONSOLE_CURSOR_INFO cursorInfo{};
    cursorInfo.dwSize = 20;
    cursorInfo.bVisible = TRUE;
    SetConsoleCursorInfo(g_outputHandle, &cursorInfo);

    SetConsoleTitleW(L"База данных компьютерных комплектующих");
}

void ClearScreen() {
    EnsureHandles();

    CONSOLE_SCREEN_BUFFER_INFO info{};
    GetConsoleScreenBufferInfo(g_outputHandle, &info);

    const DWORD size = static_cast<DWORD>(info.dwSize.X) *
static_cast<DWORD>(info.dwSize.Y);
    COORD home{ 0, 0 };
    DWORD written = 0;
    FillConsoleOutputCharacterW(g_outputHandle, L' ', size,
home, &written);
    FillConsoleOutputAttribute(g_outputHandle, info.wAttributes,
size, home, &written);
    SetConsoleCursorPosition(g_outputHandle, home);
}

void SetCursorPosition(short x, short y) {
    EnsureHandles();
    COORD position{ x, y };
    SetConsoleCursorPosition(g_outputHandle, position);
}

void SetTextColor(WORD attributes) {
    EnsureHandles();
```

```
    SetConsoleTextAttribute(g_outputHandle, attributes);
}

void ResetTextColor() {
    SetTextColor(kDefaultColor);
}

void PrintAt(short x, short y, const std::wstring& text, WORD
attributes) {
    SetCursorPosition(x, y);
    if (attributes != 0) {
        SetTextColor(attributes);
    }
    DWORD written = 0;
    WriteConsoleW(g_outputHandle, text.c_str(),
static_cast<DWORD>(text.size()), &written, nullptr);
    if (attributes != 0) {
        ResetTextColor();
    }
}

void DrawHorizontalLine(short y, short width, wchar_t symbol) {
    PrintAt(0, y, std::wstring(width, symbol));
}

short WindowWidth() {
    EnsureHandles();

    CONSOLE_SCREEN_BUFFER_INFO info{};
    GetConsoleScreenBufferInfo(g_outputHandle, &info);
    return static_cast<short>(info.srWindow.Right -
info.srWindow.Left + 1);
}

short WindowHeight() {
```

```
    EnsureHandles();

    CONSOLE_SCREEN_BUFFER_INFO info{};
    GetConsoleScreenBufferInfo(g_outputHandle, &info);
    return static_cast<short>(info.srWindow.Bottom -
info.srWindow.Top + 1);
}

void Pause(const std::wstring& message) {
    std::wcout << L"\n" << message;
    std::wstring line;
    std::getline(std::wcin, line);
}

std::wstring ReadLine(const std::wstring& prompt, bool
allowEmpty) {
    while (true) {
        std::wcout << prompt;
        std::wstring value;
        std::getline(std::wcin, value);
        if (allowEmpty || !value.empty()) {
            return value;
        }
        std::wcout << L"Поле не должно быть пустым.\n";
    }
}

int ReadInt(const std::wstring& prompt) {
    while (true) {
        const std::wstring value = ReadLine(prompt);
        try {
            size_t processed = 0;
            const int result = std::stoi(value, &processed);
            if (processed == value.size()) {
                return result;
```

```
            }
        }
        catch (...) {
        }
        std::wcout << L"Введите корректное целое число.\n";
    }
}

void ShowMessage(const std::wstring& title, const std::wstring&
message, WORD color) {
    std::wcout << L"\n";
    SetTextColor(color);
    std::wcout << L"[" << title << L"] ";
    ResetTextColor();
    std::wcout << message << L"\n";
}

void WaitMilliseconds(DWORD milliseconds) {
    Sleep(milliseconds);
}

HANDLE InputHandle() {
    EnsureHandles();
    return g_inputHandle;
}

HANDLE OutputHandle() {
    EnsureHandles();
    return g_outputHandle;
     }
}
```

Файл console_ui.cpp

```
#include "console_ui.h"

#include "console_utils.h"
```

```
#include <Windows.h>

#include <algorithm>
#include <iostream>
#include <vector>

namespace {

constexpr short kLayoutWidth = 120;
constexpr short kLayoutHeight = 29;
constexpr short kHeaderColor = FOREGROUND_GREEN |
FOREGROUND_INTENSITY;
constexpr WORD kToolbarColor = FOREGROUND_BLUE |
FOREGROUND_GREEN | FOREGROUND_INTENSITY;
constexpr WORD kWarningColor = FOREGROUND_RED | FOREGROUND_GREEN
| FOREGROUND_INTENSITY;

std::wstring Crop(const std::wstring& value, size_t maxLength) {
    if (value.size() <= maxLength) {
        return value;
    }
    if (maxLength <= 3) {
        return value.substr(0, maxLength);
    }
    return value.substr(0, maxLength - 3) + L"...";
}

}

ConsoleUi::ConsoleUi(Database& database) : database_(database) {
}

void ConsoleUi::Run() {
    console::InitializeConsole();
```

```
    ShowSplash();

    while (running_) {
        Draw();
        ExecuteAction(WaitForAction());
    }
}

void ConsoleUi::ShowSplash() const {
    console::ClearScreen();
    console::SetTextColor(FOREGROUND_BLUE | FOREGROUND_GREEN |
FOREGROUND_INTENSITY);
    std::wcout <<
L"============================================================\n
";
    std::wcout << L"   КУРСОВАЯ РАБОТА: БАЗА ДАННЫХ КОМПЬЮТЕРНЫХ
КОМПЛЕКТУЮЩИХ  \n";
    std::wcout <<
L"============================================================\n
";
    console::ResetTextColor();
    std::wcout << L"\n  Учет комплектующих: процессоры,
видеокарты, SSD, память\n";
    std::wcout << L"  Управление: клавиатура + мышь в консольном
окне Windows\n";
    std::wcout << L"\n  Горячие клавиши: N L S A E D F T I R H
Q\n";
    std::wcout << L"  Можно нажимать мышью по панели
инструментов сверху.\n";
    console::WaitMilliseconds(1800);
}

void ConsoleUi::Draw() const {
    UpdateLayoutOrigin();
    console::ClearScreen();
```

```
    PrintLayout(0, 0, L"База данных компьютерных комплектующих",
kHeaderColor);
    const std::wstring modeLabel = L"Текстовый режим / WinAPI";
    const short modeX = static_cast<short>(LayoutWidth() -
static_cast<short>(modeLabel.size()) - 1);
    PrintLayout(modeX, 0, modeLabel, FOREGROUND_RED |
FOREGROUND_GREEN | FOREGROUND_BLUE);
    DrawLayoutLine(1, LayoutWidth(), L'=');

    std::vector<Button> buttons;
    DrawToolbar(buttons);

    PrintLayout(0, 4, L"Меню: [N] Новая  [L] Загрузить  [S]
Сохранить  [A] Добавить  [E] Изменить");
    PrintLayout(0, 5, L"      [D] Удалить [F] Поиск [T]
Сортировка [I] Фильтр [R] Сброс [Q] Выход");
    DrawLayoutLine(6, LayoutWidth(), L'-');

    const std::wstring fileLine = L"Файл: " +
(database_.CurrentFilePath().empty() ? L"(не выбран)" :
database_.CurrentFilePath());
    PrintLayout(0, 7, Crop(fileLine,
static_cast<size_t>(LayoutWidth() - 2)));
    PrintLayout(0, 8, L"Записей: " +
std::to_wstring(database_.Size()));
    PrintLayout(22, 8, L"Поиск: " + (activeSearch_.empty() ?
L"нет" : activeSearch_));
    PrintLayout(60, 8, L"Фильтр: " + (activeFilter_.empty() ?
L"нет" : activeFilter_));
    PrintLayout(0, 9, L"Статус: " + Crop(statusMessage_,
static_cast<size_t>(LayoutWidth() - 10)), kWarningColor);

    DrawRecords();
    DrawLayoutLine(27, LayoutWidth(), L'=');
```

```
    PrintLayout(0, 28, L"Ожидание команды. Используйте
клавиатуру или клик мышью по кнопкам.");
}

void ConsoleUi::DrawToolbar(std::vector<Button>& buttons) const
{
    static const std::vector<std::pair<Action, const wchar_t*>>
toolbarItems = {
        { Action::Create, L" Новая " },
        { Action::Load, L" Загрузить " },
        { Action::Save, L" Сохранить " },
        { Action::Add, L" Добавить " },
        { Action::Edit, L" Изменить " },
        { Action::DeleteRecord, L" Удалить " },
        { Action::Search, L" Поиск " },
        { Action::Sort, L" Сортировка " },
        { Action::Filter, L" Фильтр " },
        { Action::ResetFilter, L" Сброс " },
        { Action::About, L" О программе " },
        { Action::Exit, L" Выход " }
    };

    short x = 0;
    for (const auto& item : toolbarItems) {
        const std::wstring buttonText = L"[" +
std::wstring(item.second) + L"]";
        if (x + static_cast<short>(buttonText.size()) >=
LayoutWidth()) {
            break;
        }
        PrintLayout(x, 2, buttonText, kToolbarColor);
        buttons.push_back(Button{ item.first, x,
static_cast<short>(x + buttonText.size() - 1), 2, 2, item.second
});
        x = static_cast<short>(x + buttonText.size() + 1);
```

```
    }
}

void ConsoleUi::DrawRecords() const {
    const std::wstring header = L" ID   | Название
| Тип               | Производитель   | Характеристики";
    PrintLayout(0, 11, Crop(header,
static_cast<size_t>(LayoutWidth() - 1)));
    DrawLayoutLine(12, LayoutWidth(), L'-');

    const std::vector<Record> records = CurrentView();
    if (records.empty()) {
        PrintLayout(0, 14, L"Записи не найдены.");
        return;
    }

    short row = 13;
    for (const Record& record : records) {
        if (row > 25) {
            PrintLayout(0, 26, L"Показаны первые 13 записей.");
            break;
        }

        std::wstring line = L" " +
Crop(std::to_wstring(record.id), 4);
        if (line.size() < 6) {
            line += std::wstring(6 - line.size(), L' ');
        }
        line += L"| " + Crop(record.componentName, 25);
        if (record.componentName.size() < 25) {
            line += std::wstring(25 -
record.componentName.size(), L' ');
        }
        line += L" | " + Crop(record.componentType, 16);
        if (record.componentType.size() < 16) {
```

```
            line += std::wstring(16 -
record.componentType.size(), L' ');
        }
        line += L" | " + Crop(record.manufacturer, 15);
        if (record.manufacturer.size() < 15) {
            line += std::wstring(15 -
record.manufacturer.size(), L' ');
        }
        line += L" | " + Crop(record.specifications, 35);

        PrintLayout(0, row, Crop(line,
static_cast<size_t>(LayoutWidth() - 1)));
        ++row;
    }
}

ConsoleUi::Action ConsoleUi::WaitForAction() const {
    std::vector<Button> buttons;
    DrawToolbar(buttons);

    while (true) {
        INPUT_RECORD inputRecord{};
        DWORD eventsRead = 0;
        ReadConsoleInputW(console::InputHandle(), &inputRecord,
1, &eventsRead);

        if (inputRecord.EventType == KEY_EVENT &&
inputRecord.Event.KeyEvent.bKeyDown) {
            const wchar_t symbol =
static_cast<wchar_t>(towupper(inputRecord.Event.KeyEvent.uChar.U
nicodeChar));
            switch (symbol) {
            case L'N': return Action::Create;
            case L'L': return Action::Load;
            case L'S': return Action::Save;
```

```
            case L'A': return Action::Add;
            case L'E': return Action::Edit;
            case L'D': return Action::DeleteRecord;
            case L'F': return Action::Search;
            case L'T': return Action::Sort;
            case L'I': return Action::Filter;
            case L'R': return Action::ResetFilter;
            case L'Q': return Action::Exit;
            case L'H': return Action::About;
            default:
                break;
            }
        }

        if (inputRecord.EventType == MOUSE_EVENT) {
            const MOUSE_EVENT_RECORD& mouse =
inputRecord.Event.MouseEvent;
            if ((mouse.dwButtonState &
FROM_LEFT_1ST_BUTTON_PRESSED) != 0) {
                const short localX =
static_cast<short>(mouse.dwMousePosition.X - layoutLeft_);
                const short localY =
static_cast<short>(mouse.dwMousePosition.Y - layoutTop_);
                for (const Button& button : buttons) {
                    if (localY >= button.top && localY <=
button.bottom &&
                        localX >= button.left && localX <=
button.right) {
                        return button.action;
                    }
                }
            }
        }
    }
}
```

```
void ConsoleUi::ExecuteAction(Action action) {
    switch (action) {
    case Action::Create:
        CreateDatabase();
        break;
    case Action::Load:
        LoadDatabase();
        break;
    case Action::Save:
        SaveDatabase();
        break;
    case Action::Add:
        AddRecord();
        break;
    case Action::Edit:
        EditRecord();
        break;
    case Action::DeleteRecord:
        DeleteRecord();
        break;
    case Action::Search:
        SearchRecords();
        break;
    case Action::Sort:
        SortRecords();
        break;
    case Action::Filter:
        FilterRecords();
        break;
    case Action::ResetFilter:
        ResetView();
        break;
    case Action::About:
        ShowAbout();
```

```
        break;
    case Action::Exit:
        running_ = false;
        break;
    case Action::None:
        break;
    }
}

std::vector<Record> ConsoleUi::CurrentView() const {
    if (!activeSearch_.empty()) {
        return database_.Search(activeSearch_);
    }
    if (!activeFilter_.empty()) {
        return database_.FilterByType(activeFilter_);
    }
    return database_.GetAllRecords();
}

void ConsoleUi::AddRecord() {
    console::ClearScreen();
    std::wcout << L"Добавление комплектующего\n\n";

    Record record;
    record.id = console::ReadInt(L"Идентификатор: ");
    record.componentName = console::ReadLine(L"Название
комплектующего: ");
    record.componentType = console::ReadLine(L"Тип
комплектующего: ");
    record.manufacturer = console::ReadLine(L"Производитель: ",
true);
    record.specifications = console::ReadLine(L"Характеристики:
", true);

    if (database_.AddRecord(record)) {
```

```
        statusMessage_ = L"Запись добавлена.";
    }
    else {
        statusMessage_ = L"Запись с таким идентификатором уже
существует.";
    }
}

void ConsoleUi::EditRecord() {
    console::ClearScreen();
    std::wcout << L"Редактирование комплектующего\n\n";

    const int sourceId = console::ReadInt(L"Введите ID записи
для изменения: ");
    if (!database_.ContainsId(sourceId)) {
        statusMessage_ = L"Запись не найдена.";
        return;
    }

    Record record;
    record.id = console::ReadInt(L"Новый идентификатор: ");
    record.componentName = console::ReadLine(L"Название
комплектующего: ");
    record.componentType = console::ReadLine(L"Тип
комплектующего: ");
    record.manufacturer = console::ReadLine(L"Производитель: ",
true);
    record.specifications = console::ReadLine(L"Характеристики:
", true);

    if (database_.UpdateRecord(sourceId, record)) {
        statusMessage_ = L"Запись изменена.";
    }
    else {
```

```
        statusMessage_ = L"Не удалось изменить запись: возможно,
новый ID уже занят.";
    }
}

void ConsoleUi::DeleteRecord() {
    console::ClearScreen();
    std::wcout << L"Удаление комплектующего\n\n";
    const int id = console::ReadInt(L"Введите ID для удаления:
");

    if (database_.RemoveRecord(id)) {
        statusMessage_ = L"Запись удалена.";
    }
    else {
        statusMessage_ = L"Запись с указанным ID не найдена.";
    }
}

void ConsoleUi::SearchRecords() {
    console::ClearScreen();
    std::wcout << L"Поиск комплектующих\n\n";
    activeSearch_ = console::ReadLine(L"Введите название, тип,
производителя или характеристику: ");
    activeFilter_.clear();
    statusMessage_ = L"Режим поиска включен.";
}

void ConsoleUi::SortRecords() {
    console::ClearScreen();
    std::wcout << L"Сортировка комплектующих\n\n";
    std::wcout << L"1. По ID\n";
    std::wcout << L"2. По названию\n";
    std::wcout << L"3. По типу\n\n";
```

```
    const int choice = console::ReadInt(L"Выберите поле: ");
    const int direction = console::ReadInt(L"Направление (1 - по
возрастанию, 2 - по убыванию): ");

    SortField field = SortField::Id;
    if (choice == 2) {
        field = SortField::ComponentName;
    }
    else if (choice == 3) {
        field = SortField::ComponentType;
    }

    if (database_.SortBy(field, direction != 2)) {
        statusMessage_ = L"Сортировка выполнена.";
    }
    else {
        statusMessage_ = L"Сортировка недоступна: база пуста.";
    }
}

void ConsoleUi::FilterRecords() {
    console::ClearScreen();
    std::wcout << L"Фильтрация комплектующих\n\n";
    activeFilter_ = console::ReadLine(L"Введите тип
комплектующего или его часть: ");
    activeSearch_.clear();
    statusMessage_ = L"Фильтр по типу комплектующего применен.";
}

void ConsoleUi::ResetView() {
    activeFilter_.clear();
    activeSearch_.clear();
    statusMessage_ = L"Поиск и фильтрация сброшены.";
}
```

```
void ConsoleUi::SaveDatabase() {
    console::ClearScreen();
    std::wcout << L"Сохранение базы комплектующих\n\n";

    std::wstring path = database_.CurrentFilePath();
    if (!path.empty()) {
        std::wcout << L"Текущий файл: " << path << L"\n";
    }

    path = console::ReadLine(L"Введите путь к файлу (*.txt,
*.db): ", true);
    if (path.empty()) {
        path = database_.CurrentFilePath();
    }

    std::wstring errorMessage;
    if (database_.SaveToFile(path, errorMessage)) {
        statusMessage_ = L"База данных сохранена.";
    }
    else {
        statusMessage_ = errorMessage;
    }
}

void ConsoleUi::LoadDatabase() {
    console::ClearScreen();
    std::wcout << L"Загрузка базы комплектующих\n\n";

    const std::wstring path = console::ReadLine(L"Введите путь к
сохраненному файлу: ");
    std::wstring errorMessage;
    if (database_.LoadFromFile(path, errorMessage)) {
        activeFilter_.clear();
        activeSearch_.clear();
        statusMessage_ = L"База данных загружена.";
```

```
    }
    else {
        statusMessage_ = errorMessage;
    }
}

void ConsoleUi::CreateDatabase() {
    database_.CreateNew();
    activeFilter_.clear();
    activeSearch_.clear();
    statusMessage_ = L"Создана новая пустая база
комплектующих.";
}

void ConsoleUi::ShowAbout() {
    console::ClearScreen();
    std::wcout << L"О программе\n\n";
    std::wcout << L"Проект реализует базу данных компьютерных
комплектующих.\n";
    std::wcout << L"Состав:\n";
    std::wcout << L" - многомодульная архитектура;\n";
    std::wcout << L" - структуры, список std::list и файловое
хранение;\n";
    std::wcout << L" - текстовый интерфейс с меню и панелью
инструментов;\n";
    std::wcout << L" - управление клавиатурой и мышью в
консоли;\n";
    std::wcout << L" - учет процессоров, видеокарт, накопителей
и других комплектующих;\n";
    std::wcout << L" - ассемблерная вставка для сравнения ID при
сортировке (Win32).\n\n";
    console::Pause();
    statusMessage_ = L"Показана информация о программе.";
}
```

```
short ConsoleUi::LayoutWidth() const {
    return kLayoutWidth;
}

short ConsoleUi::LayoutHeight() const {
    return kLayoutHeight;
}

void ConsoleUi::UpdateLayoutOrigin() const {
    layoutLeft_ = static_cast<short>((std::max)(0,
(console::WindowWidth() - LayoutWidth()) / 2));
    layoutTop_ = static_cast<short>((std::max)(0,
(console::WindowHeight() - LayoutHeight()) / 2));
}

void ConsoleUi::PrintLayout(short x, short y, const
std::wstring& text, WORD attributes) const {
    console::PrintAt(static_cast<short>(layoutLeft_ + x),
static_cast<short>(layoutTop_ + y), text, attributes);
}

void ConsoleUi::DrawLayoutLine(short y, short width, wchar_t
symbol) const {
    PrintLayout(0, y, std::wstring(width, symbol));
}
```

Файл asm_utils.cpp

```
#include "asm_utils.h"

int CompareIntegersAsm(int left, int right) {
#if defined(_M_IX86)
    int result = 0;
    __asm {
        mov eax, left
```

```
        mov edx, right
        cmp eax, edx
        jl less_than
        jg greater_than
        mov result, 0
        jmp done
less_than:
        mov result, -1
        jmp done
greater_than:
        mov result, 1
done:
    }
    return result;
#else
    if (left < right) {
        return -1;
    }
    if (left > right) {
        return 1;
    }
    return 0;
#endif
}
```

Приложение Б

Результат работы программы

Файл sample_components_new.txt

```
DBV1

2101|Intel Core i9-12900K|Процессор|Intel|16 ядер, LGA1700, до
5.2 ГГц

2102|AMD Radeon RX 7800 XT|Видеокарта|PowerColor|16 ГБ GDDR6,
PCIe 4.0

2103|Samsung 980 PRO 2TB|SSD|Samsung|M.2 NVMe, до 7000 МБ/с

2104|G.Skill Ripjaws V 32GB|Оперативная память|G.Skill|DDR4,
3600 МГц, 2x16 ГБ

2107|Kingston FURY Beast Black RGB|Оперативная
память|Kingston|16 ГБ DDR4, 2400 МГц

2106|Gigabyte B760 Gaming X|Материнская плата|Gigabyte|LGA1700,
DDR5, ATX
```