**Министерство образования Российской Федерации**

**Пензенский государственный университет**

Кафедра «Математическое обеспечение и применение ЭВМ»

Пояснительная записка к курсовому проекту   
по дисциплине

«Объектно-ориентированное программирование».

Разработка программы с использованием объектно-ориентированного подхода.

ИС “Аптека”

Автор работы:

студенты группы 20ВП1

Нешкин Д.А.

Принял:

к.т.н. Афонин А.Ю.

Работа сдана \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пенза 2022 г.

**Оглавление**

[**Введение** 3](#_Toc106212860)

[**1.** **Постановка задачи и анализ предметной области** 4](#_Toc106212861)

[**1.1.** **Анализ требований** 4](#_Toc106212862)

[**1.1.1** **Требования к интерфейсу пользователя** 4](#_Toc106212863)

[**1.2** **Технология разработки программного обеспечения** 6](#_Toc106212864)

[**2. Проектирование программы** 11](#_Toc106212865)

[**2.1 Модель интерфейса** 11](#_Toc106212866)

[**2.2** **Структура программного обеспечения** 13](#_Toc106212867)

[**3.** **Реализация программы** 16](#_Toc106212868)

[**4.** **Тестирование программы** 18](#_Toc106212869)

[**5.** **Руководство пользователя** 25](#_Toc106212870)

[**Заключение** 26](#_Toc106212871)

[**Список используемых источников** 27](#_Toc106212872)

[**Приложение А. Код программы** 28](#_Toc106212873)

# **Введение**

Информационная система (ИС) — это формальная, социотехническая, организационная система, предназначенная для сбора, обработки, хранения и распространения информации. С социотехнической точки зрения информационные системы состоят из четырех компонентов: задача, люди, структура (или роли) и технология. Информационные системы можно определить, как интеграцию компонентов для сбора, хранения и обработки данных, данные которых используются для предоставления информации, вклада в знания, а также цифровых продуктов, облегчающих принятие решений.

"Информационные системы» также является академическим полевым исследованием о системах с конкретной ссылкой на информацию и дополнительные сети компьютерного оборудования и программного обеспечения, которые люди и организации используют для сбора, фильтрации, обработки, создания и распространения данных. Акцент делается на информационной системе, имеющей определенную границу, пользователей, процессоры, хранилища, входы, выходы и вышеупомянутые сети связи.

Во многих организациях отдел или подразделение, отвечающее за информационные системы и обработку данных, известно как «информационные услуги».

Таким образом, информационные системы взаимосвязаны с системами данных, с одной стороны, и системами деятельности, с другой. Информационная система – это форма системы коммуникации, в которой данные представляют и обрабатываются как форма социальной памяти. Информационную систему также можно считать полуформальным языком, который поддерживает принятие решений и действия человека.

Информационные системы являются основным направлением изучения организационной информатики. [1]

1. **Постановка задачи и анализ предметной области**
   1. **Анализ требований**
      1. **Требования к интерфейсу пользователя**

Приложение должно предоставлять пользователю удобный интерфейс, позволяющий:

* Добавлять лекарственное средство в базу данных
* Удалять базу данных целиком
* Удалять определенные элементы из базы данных
* Редактировать записи в базе данных
* Искать определенные записи в БД
* Различными способами сортировать записи в БД
* Фильтровать записи по определенному критерию
  + 1. **Требования к программным средствам**

Анализ задания на разработку позволяет выделить следующие варианты использования (рисунок 1).

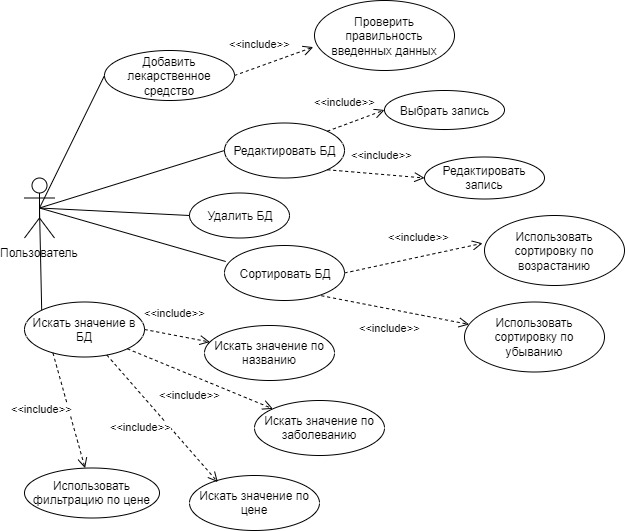


Рисунок 1 – Диаграмма вариантов использования

Была составлена спецификация для некоторых прецедентов из диаграммы вариантов использования (Таблица 1-3).

Таблица 1 – Спецификация прецедента «Добавить лекарственное средство»

|  |
| --- |
| Наименование: Добавить лекарственное средство |
| ID: 2 |
| Краткое описание: Пользователь добавляет медикамент в БД |
| Действующие лица: Пользователь |
| Предусловие: Пользователь выбрал страничку «Создание БД» |
| Основной поток:   1. Пользователь вводит необходимые данные 2. Начинается проверка корректного ввода данных 3. Если данные корректны    1. Введенные данные заносятся в БД 4. Иначе приложение выводит сообщение об ошибке |
| Постусловие: Пользователь добавил медикамент |

Таблица 2 – Спецификация прецедента «Удалить БД»

|  |
| --- |
| Название: Удалить БД |
| ID: 6 |
| Краткое описание: Пользователь очищает все данные БД |
| Действующие лица: Пользователь |
| Предусловие: Пользователь создал БД с медикаментами |
| Основной поток:   1. Пользователь выбирает функцию удаления БД 2. Файл с БД очищается |
| Постусловие: Пользователь получает пустой файл |

Таблица 3 – Спецификация прецедента «Редактировать БД»

|  |
| --- |
| Название: Редактировать БД |
| ID: 4 |
| Краткое описание: Пользователь редактирует записи в БД |
| Действующие лица: Пользователь |
| Предусловие: Пользователь создал БД с медикаментами |
| Основной поток:   1. Пользователь выбирает строку, которую необходимо отредактировать 2. Открывается окно редактирования 3. Пользователь вводит нужные ему данные 4. Если данные корректны    1. Данные заносятся в БД 5. Иначе приложение выводит сообщение об ошибке |
| Постусловие: Пользователь получает отредактированную БД |

* 1. **Технология разработки программного обеспечения**

Технология разработки программного обеспечения представляет собой инженерный подход к разработке программных средств ЭВМ, охватывающий методологию программирования, проблемы обеспечения надежности программ, оценки рабочих характеристик и качества проектов.

Технология разработки программного обеспечения рассматривает вопросы управления проектированием систем программного обеспечения, а также средства и стандарты разработки программ.

1) RUP (Rational Unified Process).

Один из самых известных процессов, использующих итеративную модель разработки – RUP. Он был создан во второй половине 1990-x годов в компании Rational Software. Термином RUP обозначает как методологию, так и продукт компании IBM (ранее Rational) для управления процессом разработки.

Методология RUP описывает абстрактный общий процесс, на основе которого организация или проектная команда должна создать специализированный процесс, ориентированный на ее потребности.

Основные характеристики:

* Разработка требований, для описания требований в RUP используются прецеденты использования (use cases). Полный набор прецедентов использования системы вместе с логическими отношениями между ними называется моделью прецедентов использования. Каждый прецедент использования – это описание сценариев взаимодействия пользователя с системой, полностью выполняющего конкретную пользовательскую задачу. Согласно RUP все функциональные требования должны быть представлены в виде прецедентов использования.
* Итеративная разработка, проект RUP состоит из последовательности итераций с рекомендованной продолжительностью от 2 до 6 недель. Перед началом очередной итерации определяется набор прецедентов использования, которые будут реализованы к её завершению.

Жизненный цикл проекта RUP состоит из четырех фаз. Последовательность этих фаз фиксирована, но число итераций, необходимых для завершения каждой фазы, определяется индивидуально для каждого конкретного проекта. Фазы RUP нельзя отождествлять с фазами водопадной модели – их назначение и содержание принципиально различны. [2]

2) Scrum

Scrum предоставляет эмпирический подход к разработке ПО. Этот процесс быстр, адаптивен, умеет подстраиваться и отличен от каскадной модели. Scrum основан на повторяющихся циклах, это делает его более гибким и предсказуемым.

Для начала определим роли, которые участвуют в процессе:

Scrum мастер (Scrum Master), Владелец продукта (Product Owner), Команда (Team).

Scrum Мастер - самая важная роль в методологии. Scrum Мастер отвечает за успех Scrum в проекте. Как правило, эту роль в проекте играет менеджер проекта или лидер команды (Team Leader). Важно подчеркнуть, что Scrum Мастер не раздает задачи членам команды. В Scrum команда является самоорганизующейся и самоуправляемой.

Основные обязанности Scrum Мастера таковы:

* создает атмосферу доверия,
* участвует в митингах в качестве фасилитатора - человека, обеспечивающий успешную групповую коммуникацию
* устраняет препятствия − делает проблемы и открытые вопросы видимыми
* отвечает за соблюдение практик и процесса в команде

Scrum Мастер отслеживает прогресс команды при помощи Sprint Backlog, отмечая статус всех задач в спринте.

3)Crystal Clear.

Легковесная гибкая методология, созданная Алистером Коуберном, которая предназначена для небольших команд в 6-8 человек для разработки некритичных бизнес-приложений. Как и все гибкие методологии, Crystal Clear больше опирается на людей, чем на процессы 38 и артефакты.

Crystal Clear использует семь методов/практик, три из которых являются обязательными:

* Частая поставка продукта;
* Улучшения через рефлексию;
* Личные коммуникации;
* Чувство безопасности;
* Фокусировка;
* Простой доступ к экспертам;
* Качественное техническое окружение.

4)RAD – для разработки интерфейса

RAD (англ. rapidapplicationdevelopment — быстрая разработка приложений) — концепция создания средств разработки программных продуктов, уделяющая особое внимание быстроте и удобству программирования, созданию технологического процесса, позволяющего программисту максимально быстро создавать компьютерные программы.

RAD предполагает, что разработка ПО осуществляется небольшой командой разработчиков за срок порядка трёх-четырёх месяцев путём использования инкрементного прототипирования с применением инструментальных средств визуального моделирования и разработки. Технология RAD предусматривает активное привлечение заказчика уже на ранних стадиях — обследование организации, выработка требований к системе. Последнее из указанных свойств подразумевает полное выполнение требований заказчика как функциональных, так и нефункциональных, с учётом их возможных изменений в период разработки системы, а также получение качественной документации, обеспечивающей удобство эксплуатации и сопровождения системы. Это означает, что дополнительные затраты на сопровождение сразу после поставки будут значительно меньше. Таким образом, полное время от начала разработки до получения приемлемого продукта при использовании этого метода значительно сокращается. [3]

В результате изучения технологий разработки программного продукта, была выбрана технология RAD по следующим причинам:

* Интерфейс, устраивающий пользователя;
* Легкая адаптируемость проекта к изменяющимся требованиям;
* Быстрота продвижения программного продукта.

# **2. Проектирование программы**

## **2.1 Модель интерфейса**

В результате проектирования программы были реализованы формы, представленные на рисунках 2-7.

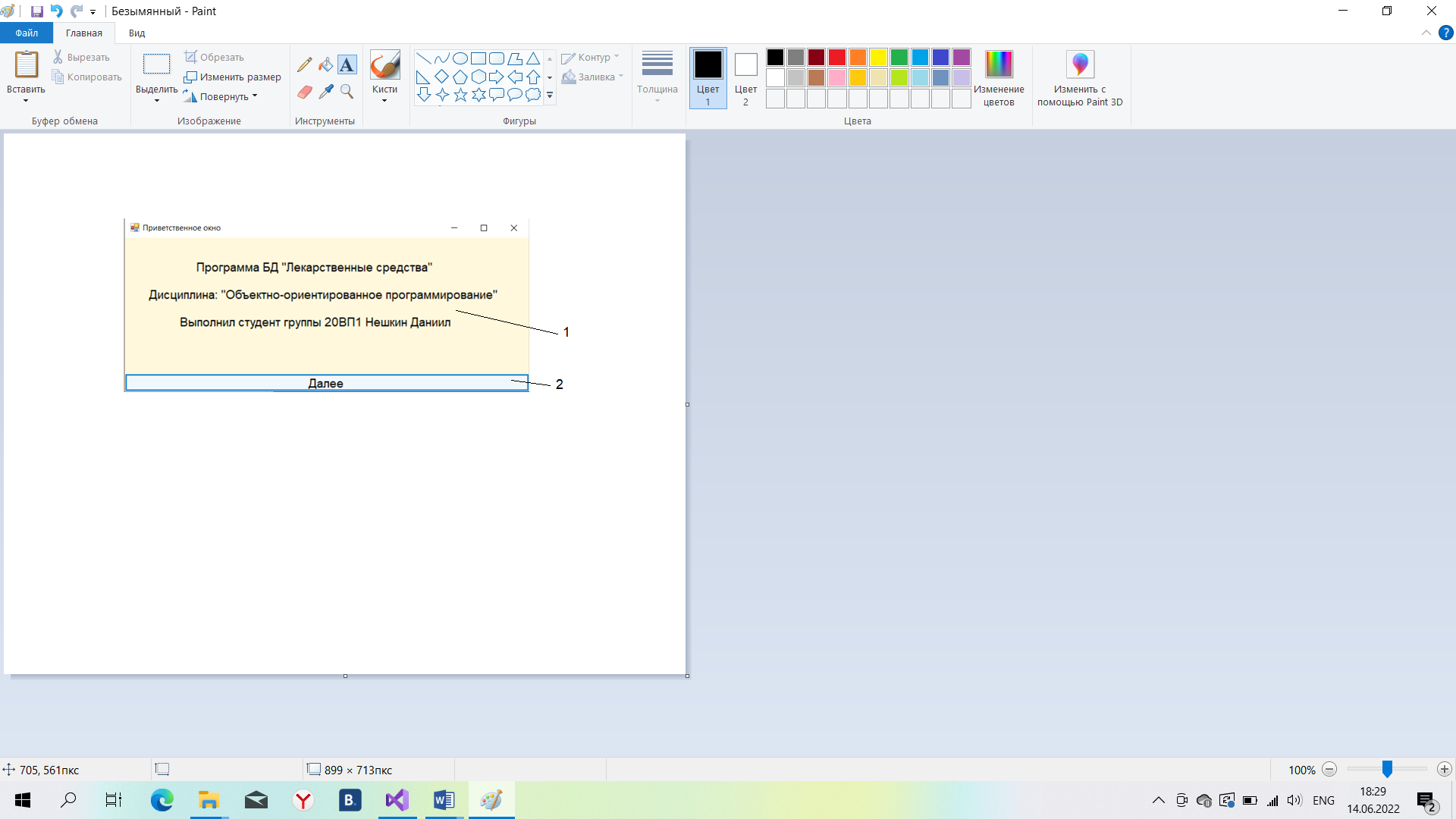


Рисунок 2 – Приветственное окно

1. Информация о приложении
2. Кнопка для перехода к основной части приложения



Рисунок 3 – Приложение «Лекарственные средства (1-й раздел)»

1. Разделы приложения
2. Поля для ввода данных
3. Кнопка записи введенных данных в БД

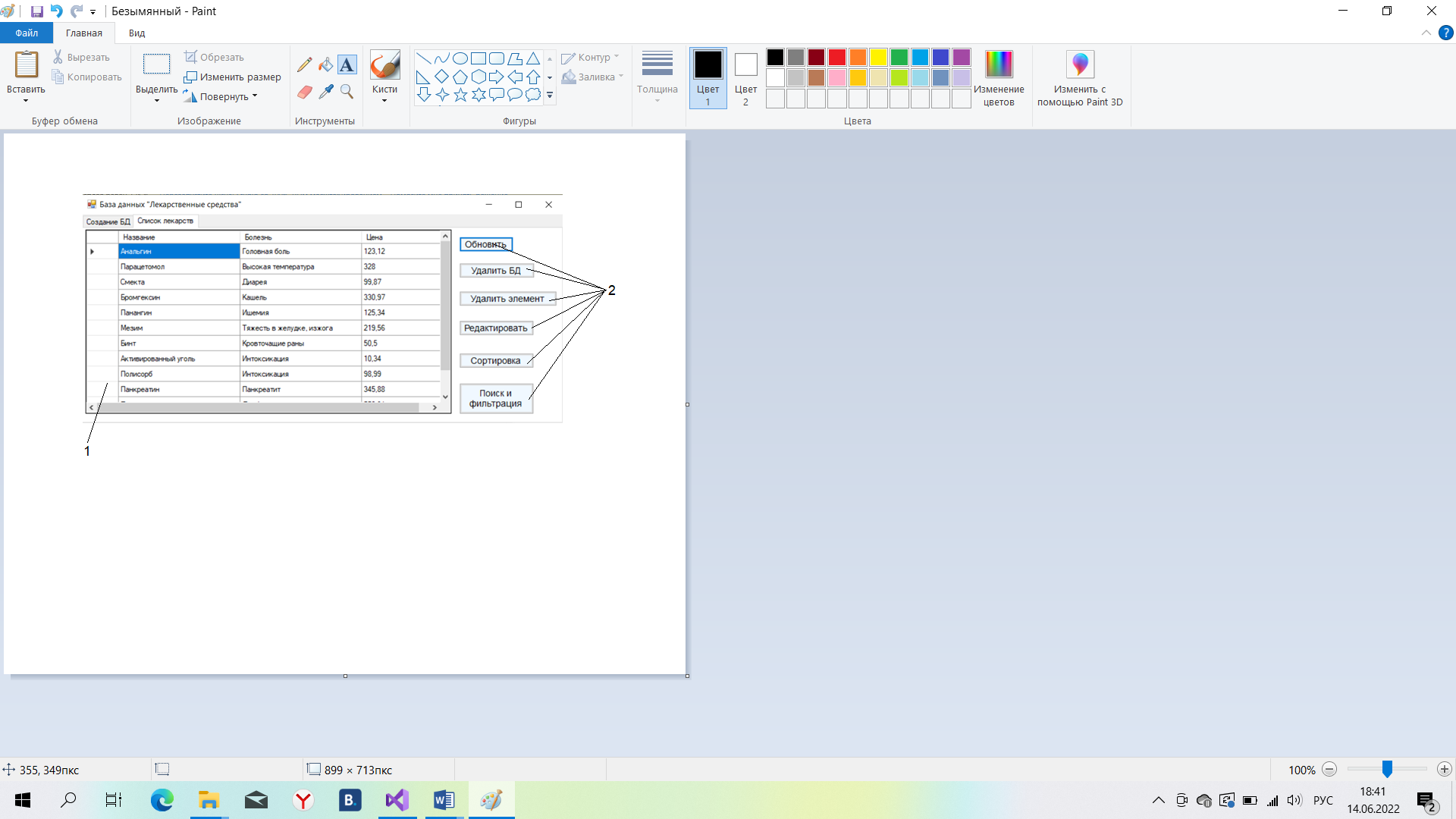


Рисунок 4 – Приложение «Лекарственные средства (2-ой раздел)»

1. DataGridView таблица для отображения БД из файла
2. Кнопки взаимодействия с БД

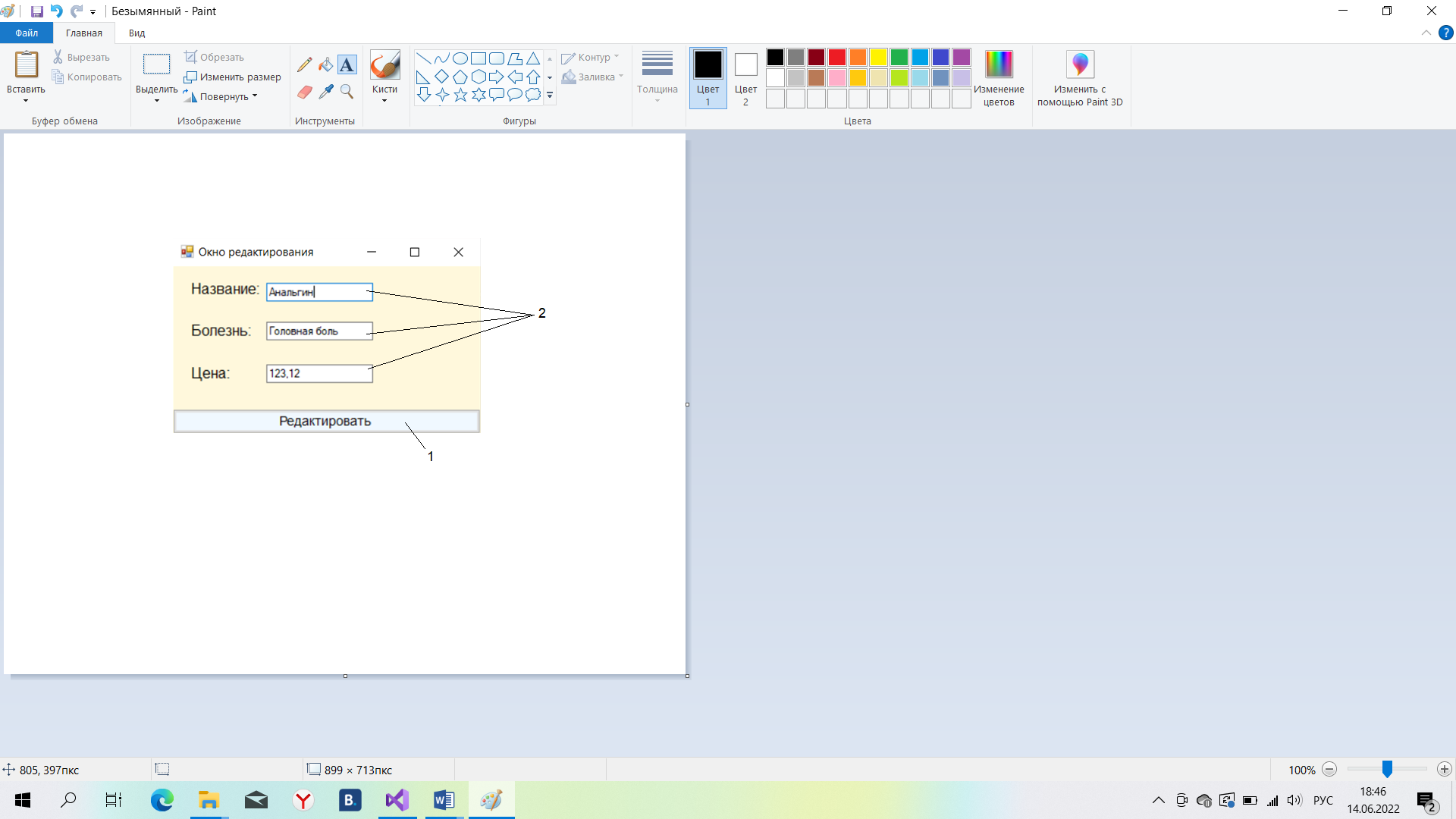


Рисунок 5 – Окно редактирования

1. Кнопка, запускающая процесс записи отредактированных данных в БД
2. Поля для ввода новых(отредактированных) значений

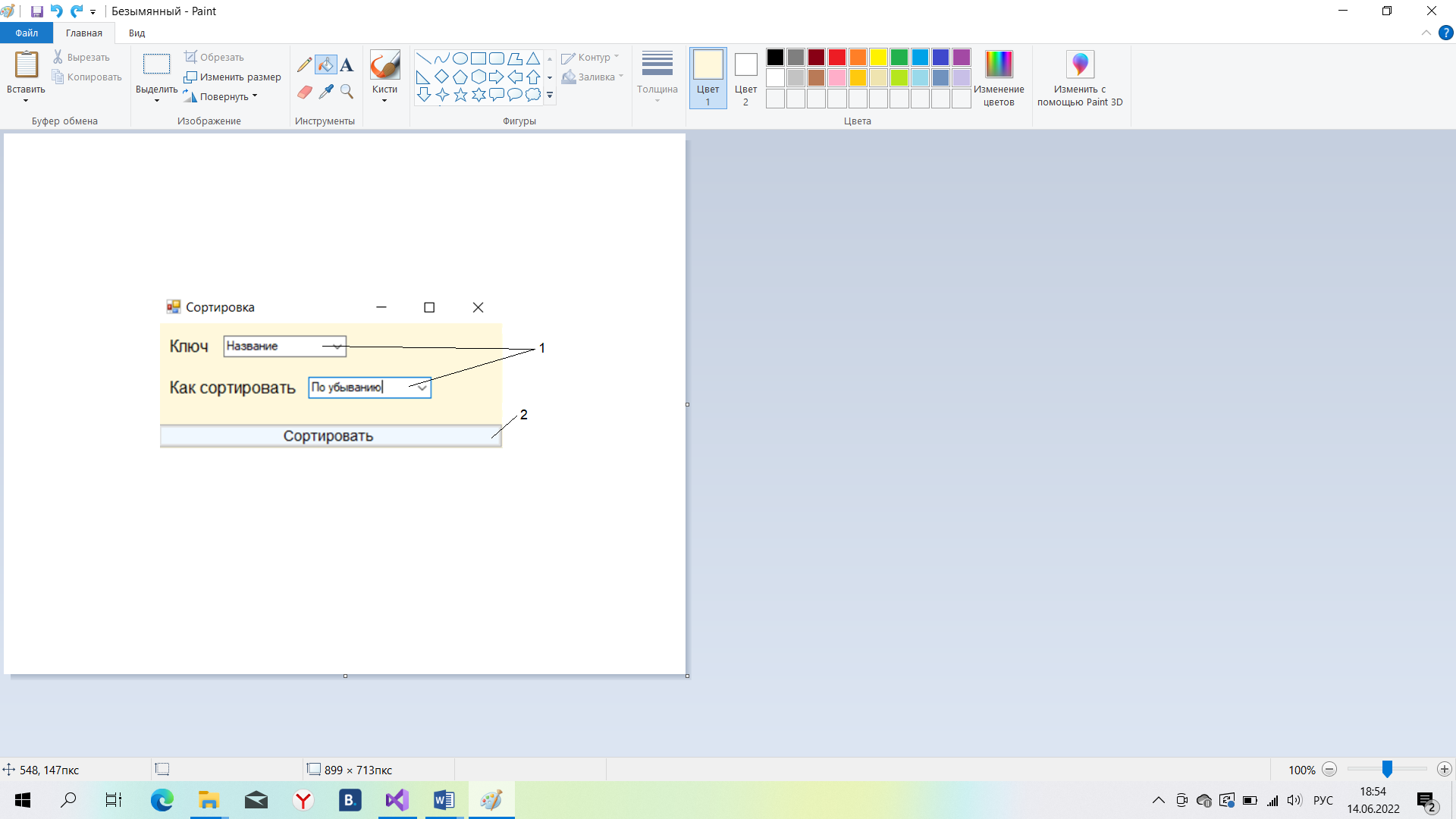


Рисунок 6 – Окно сортировки

1. Поля для ввода параметров сортировки
2. Кнопка, запускающая процесс сортировки базы данных



Рисунок 7 – Окно поиска и фильтрации

1. Поля ввода характеристик для поиска записей
2. Кнопка, инициирующая процесс поиска/фильтрации
   1. **Структура программного обеспечения**

Структура программного обеспечения описана диаграммой классов, представленной на рисунке 8.

На диаграмме классов представлены все созданные классы на основе которых была реализована ИС “Аптека”.

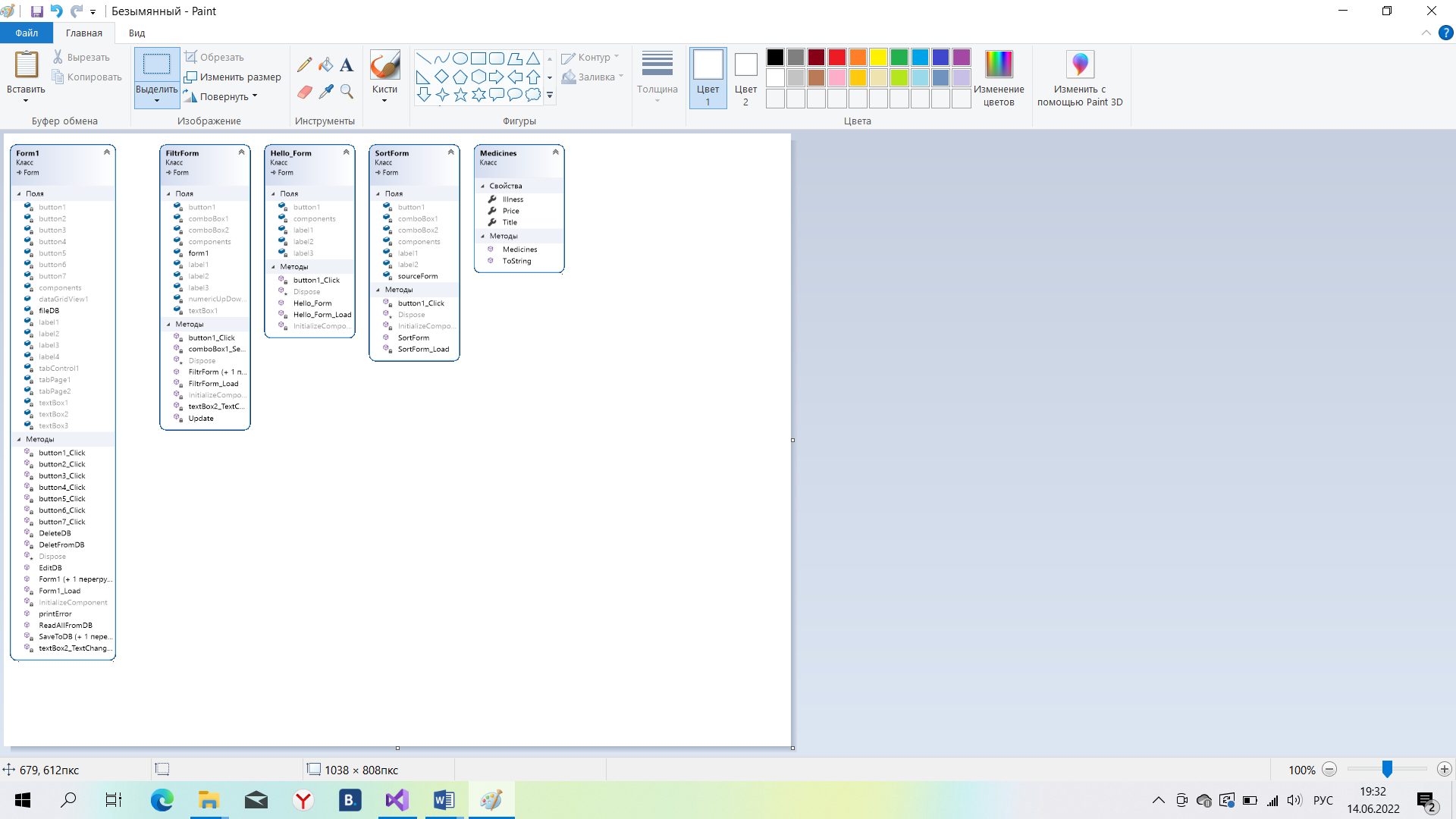


Рисунок 8 – Диаграмма классов

Описание компонентов диаграммы классов:

1. Form1 – Главная форма. С помощью этой формы пользователь взаимодействует с базой данных.

Методы:

* button1\_Click – button7\_Click – Методы кнопок для взаимодействия с БД
* DeleteDB – метод, очищающий файл с БД
* DeleteFromDB – метод, удаляющий конкретную запись из БД
* EditDB – метод редактирования записей в БД
* printError – метод, выводящий окно ошибки
* RaedAllFromDB – метод, позволяющий считать все записи из БД
* SaveToDB – позволяет сохранить запись или коллекцию записей в БД

1. FiltrForm – форма, позволяющая производить поиск и фильтрацию в базе данных

Методы:

* button1\_Click – метод, инициирующий процесс поиска и фильтрации
* comboBox1\_Selected – блокирует поле «Значение», если выбраны ключ «Цена»

1. Hello\_Form – приветственное окно с информацией о проекте

Методы:

* button1\_Click – закрывает приветственное окно и выводит основную форму

1. SortForm – форма, отвечающая за сортировку базы данных

Методы:

* button1\_Click – инициирует процесс сортировки базы данных по определенному критерию

1. Medicines – класс описывающий строку в базе данных, то есть само лекарственное средство
2. **Реализация программы**

Для решения поставленной задачи было написано приложение в среде Visual Studio 2022 с использованием языка C#, а весь интерфейс реализован в WinForms. Исходный код приведен в приложении А. Скриншоты программы представлены в руководстве пользователя.

Диаграмма компонентов реализованного приложения представлена на рисунке 9.

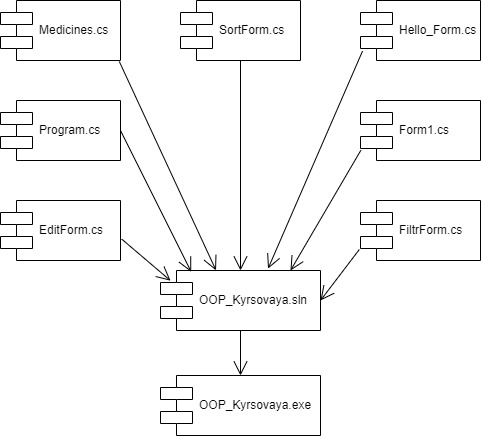


Рисунок 9 – Диаграмма компонентов

Описание компонентов представлено в таблице 4

Таблица 4 – Описание компонентов

|  |  |
| --- | --- |
| **Название** | **Описание** |
| OOP\_Kyrsovaya.exe | Исполняемый файл |
| OOP\_Kyrsovaya.sln | Файл решения |

Продолжение таблицы 4

|  |  |
| --- | --- |
| EditForm.cs | Файл с реализацией формы «Редактирование» |
| Program.cs | Исходный файл |
| Medicines.cs | Файл с реализацией класса «Медикаменты» |
| SortForm.cs | Файл с реализацией формы «Сортировка» |
| Hello\_Form.cs | Файл приветственного окна |
| Form1.cs | Файл реализации приложения |
| FiltForm.cs | Файл с реализацией формы «Фильтрация и поиск» |

1. **Тестирование программы**

В курсовом проекте было проведено функциональное тестирование. Результаты тестирования представлены в таблице 5.

Таблица 5 – Тестирование приложения

|  |  |  |
| --- | --- | --- |
| **Состав теста** | **Ожидаемый результат** | **Наблюдаемый результат** |
| При добавлении медикамента отсутствуют некоторые данные | Приложение выведет сообщение об ошибке | Тест прошел успешно.  Было выведено сообщение об ошибке (Рисунок 10) |
| При добавлении медикамента цена введена не в числовом формате | Приложение выведет сообщение об ошибке | Тест прошел успешно.  Было выведено сообщение об ошибке (Рисунок 11) |
| Удаление пользователем базы данных | Приложение выведет сообщение о выполненной операции и очистит таблицу приложения | Тест прошел успешно.  База данных была удалена (Рисунок 12) |
| Удаление пользователем определенной записи из базы данных | Приложение удалит запись из файла и таблицы | Тест пройден.  Запись удалена (Рисунки 13 - 14) |
| Редактирование записи пользователем в базе данных | Приложение изменит запись | Тест прошел успешно.  Запись изменена (Рисунки 15 – 16) |
| При редактировании были введены не все данные | Приложение выведет сообщение об ошибке | Тест прошел успешно.  Выведено сообщение об ошибке (Рисунок 17) |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| **Состав теста** | **Ожидаемый результат** | **Наблюдаемый результат** |
| Сортировка данных по названию в обратном алфавитном порядке | Приложение выведет отсортированную таблицу данных | Тест прошел успешно.  Приложение отсортировало данные (Рисунок 18) |
| Поиск данных, начинающихся на букву «И» | Приложение выведет таблицу с подходящими записями | Тест прошел успешно.  Приложение вывело таблицу с результатами (Рисунок 19) |
| Поиск данных, начинающихся на букву «П», с фильтрацией на цену | Приложение выведет таблицу с подходящими записями | Тест прошел успешно.  Приложение вывело таблицу с результатами (Рисунок 20) |

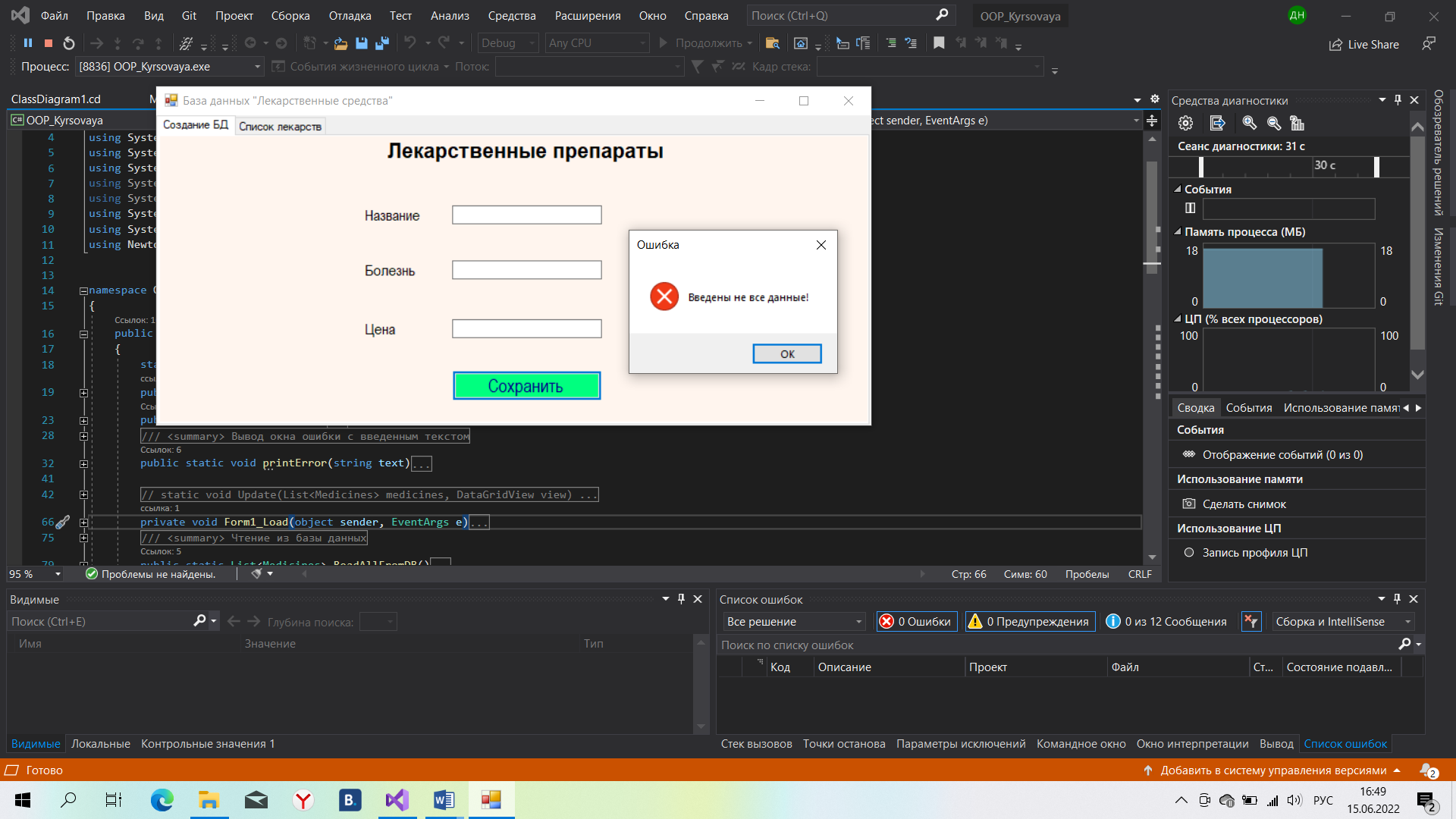


Рисунок 10 – Ошибка. Заполнены не все поля

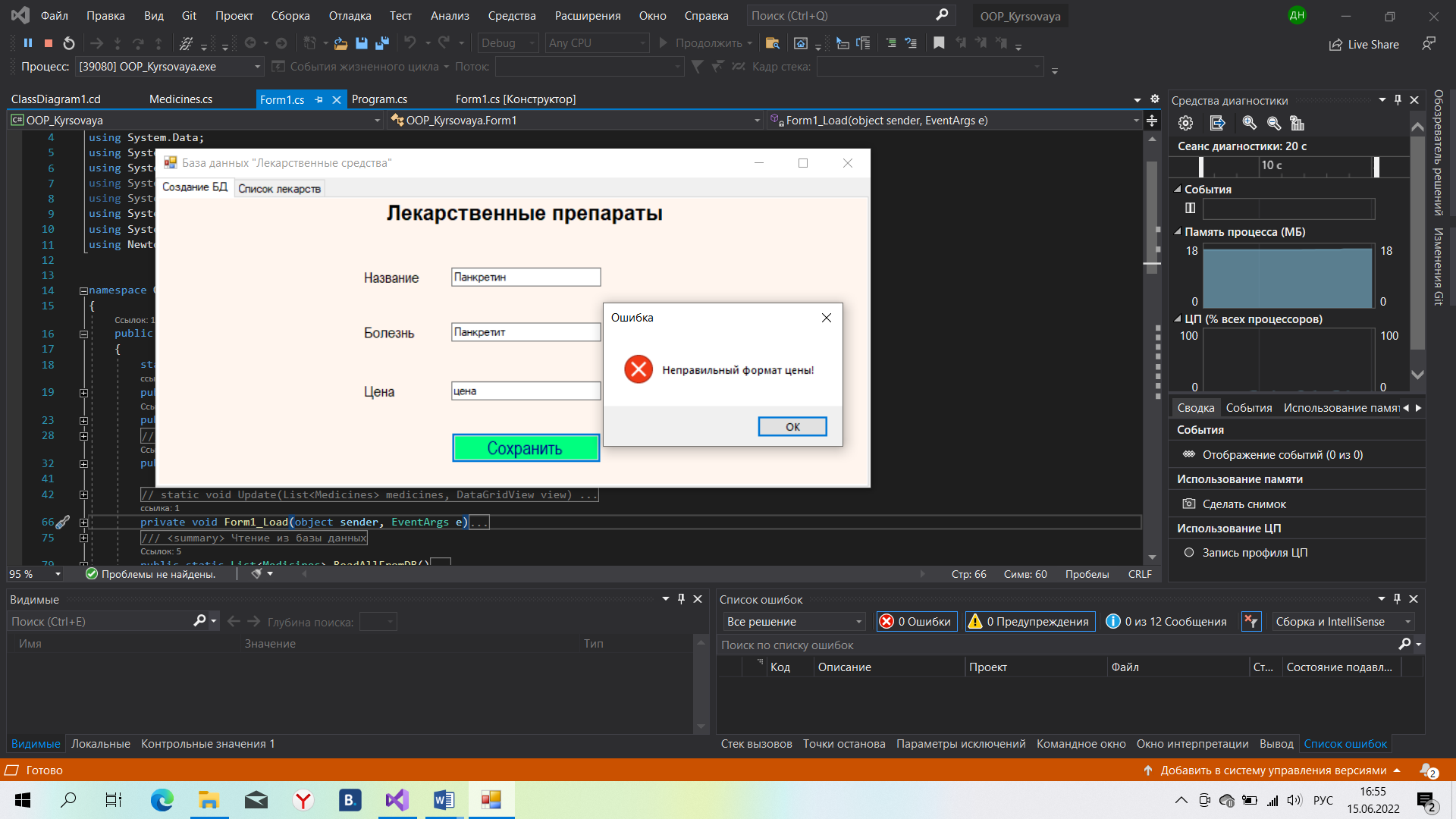


Рисунок 11 – Введена неправильная цена

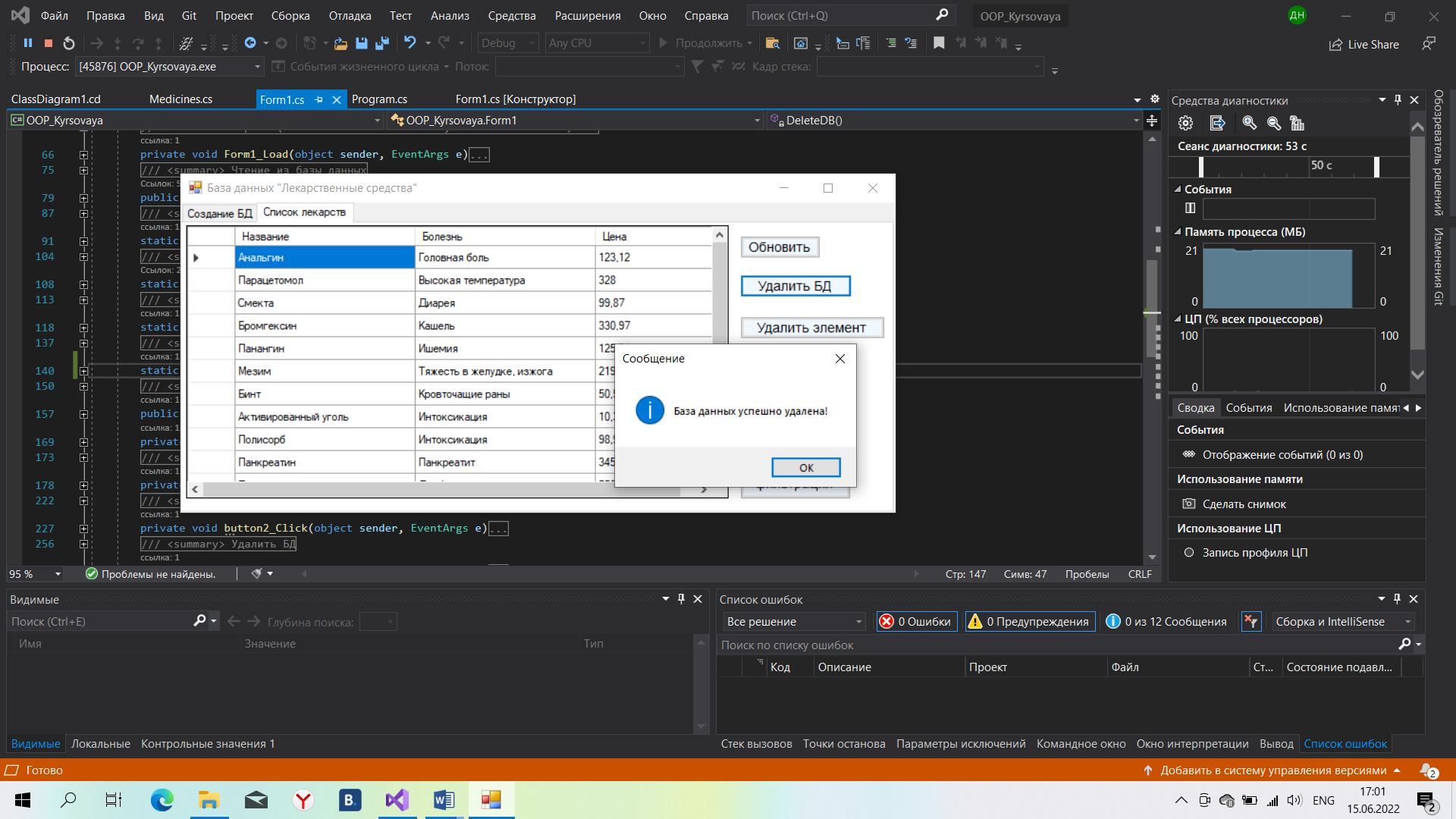


Рисунок 12 – Удаление базы данных

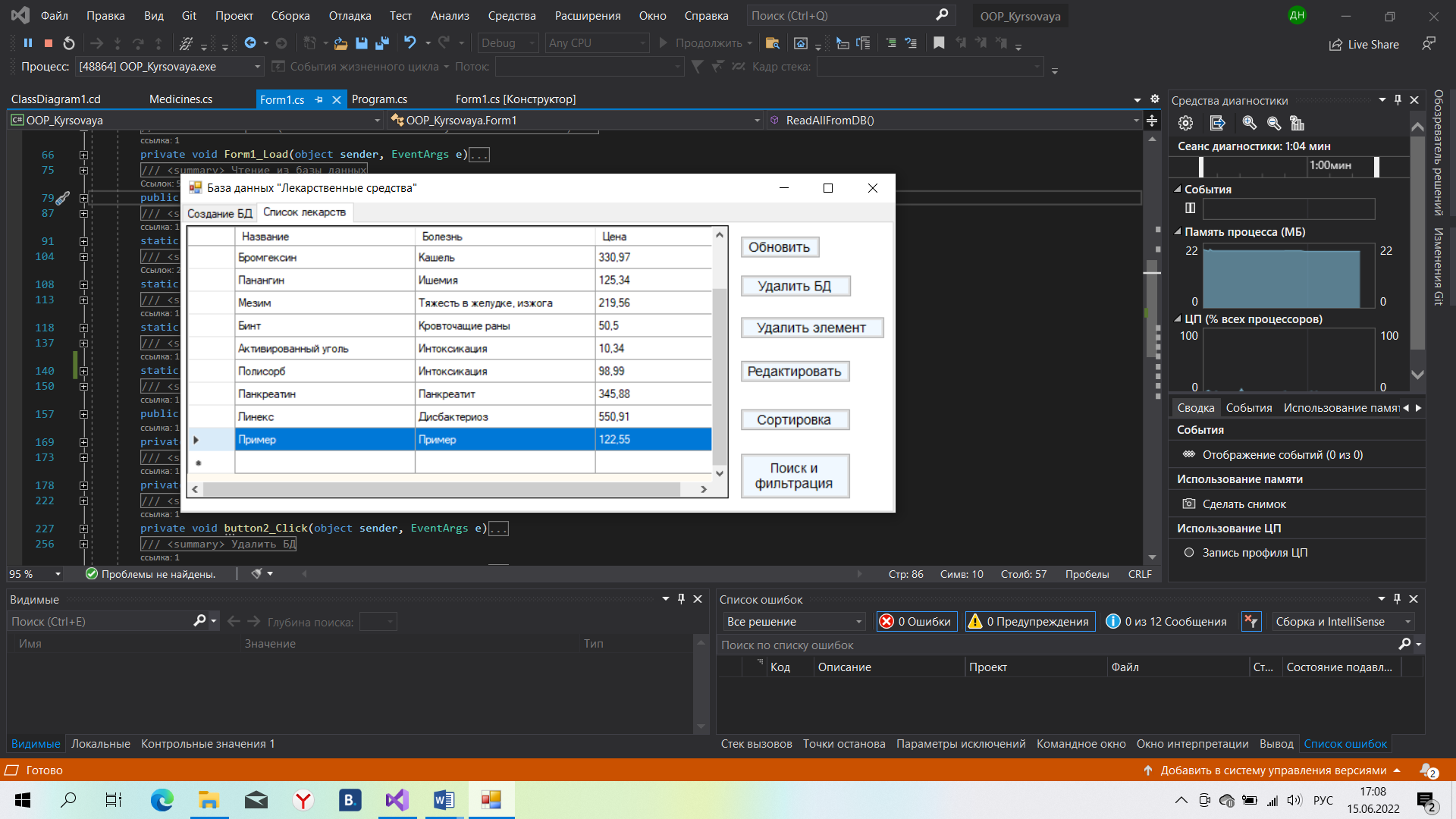


Рисунок 13 – Приложение перед удалением записи

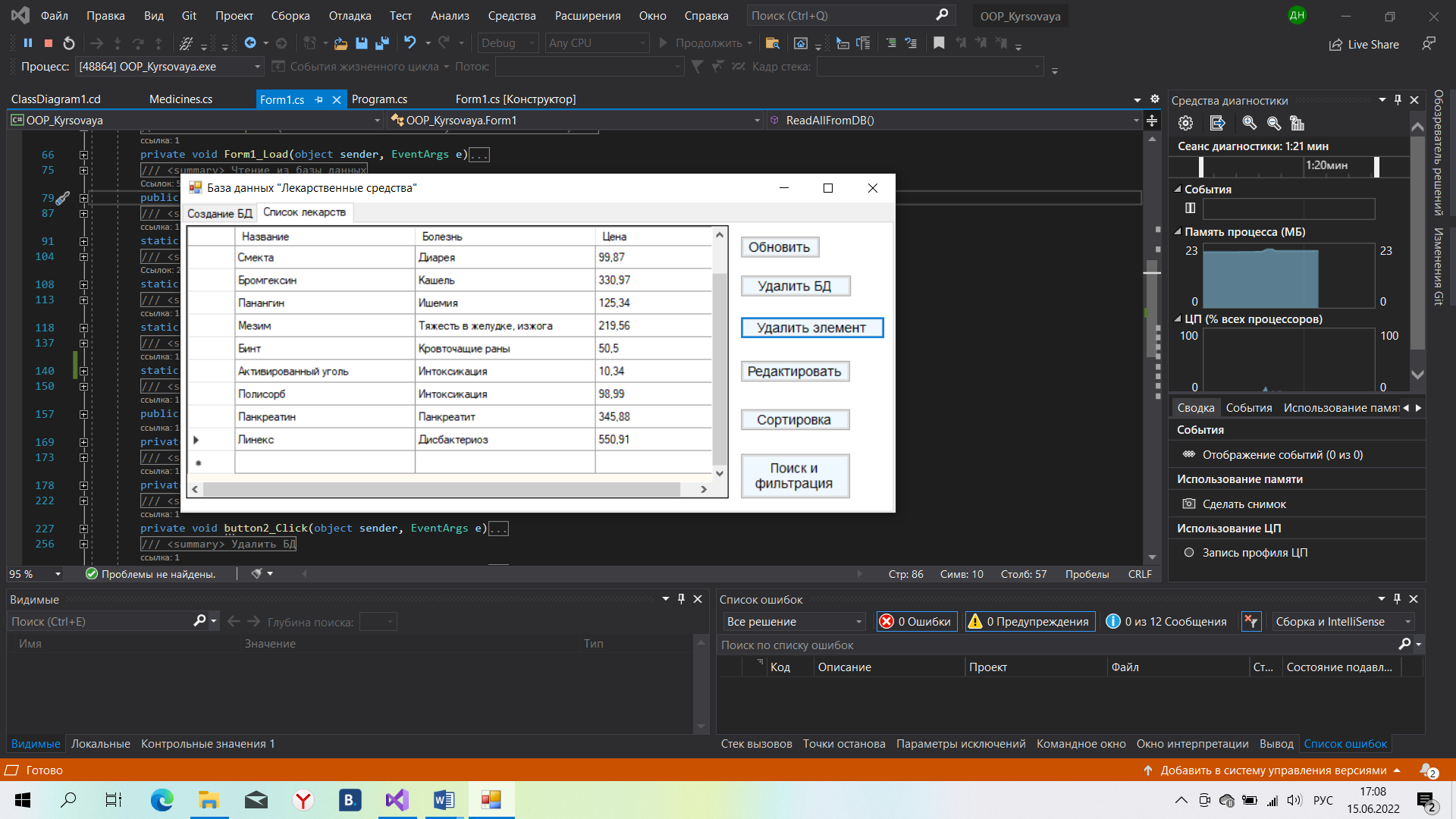


Рисунок 14 – Приложение после удаления

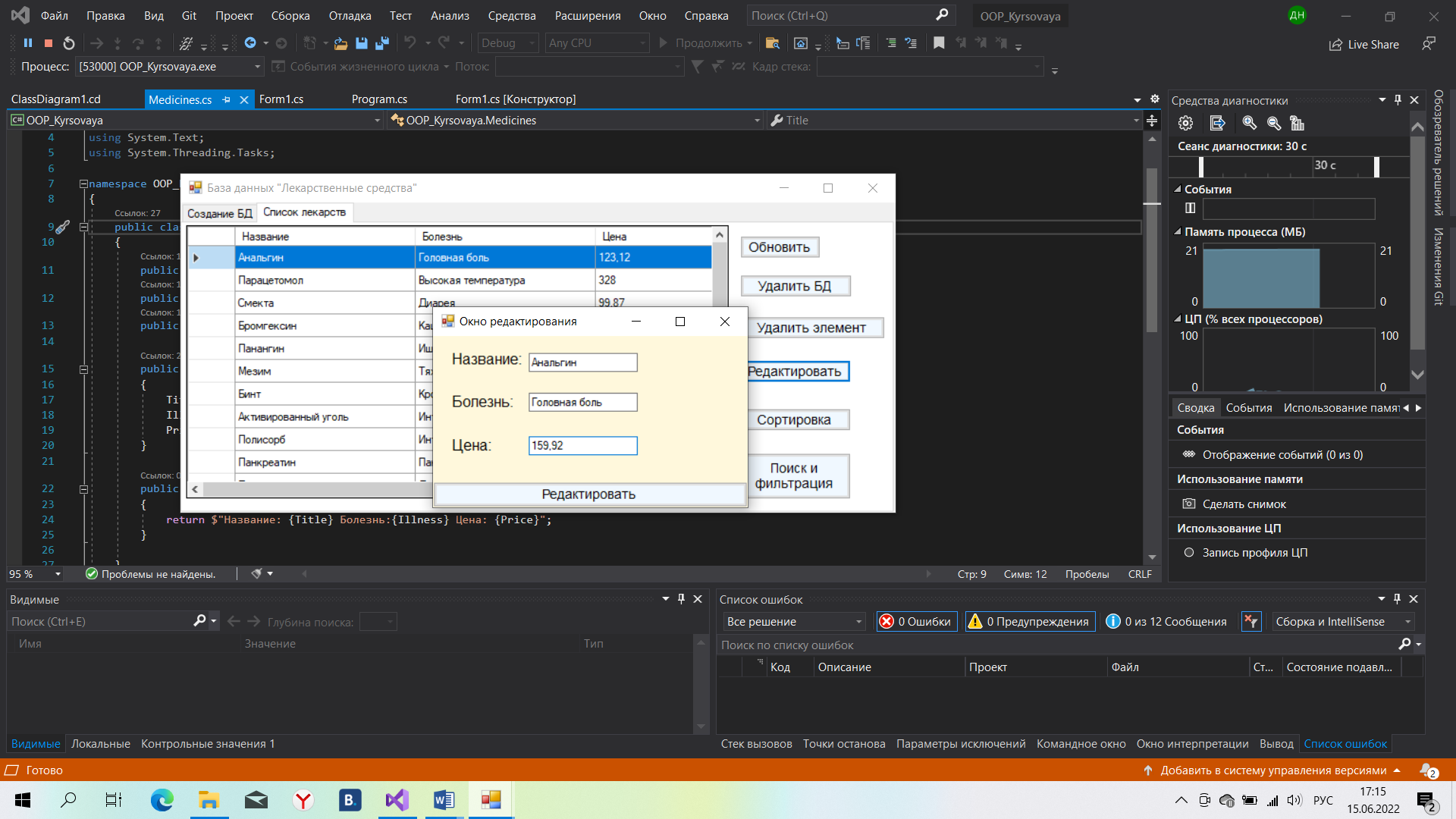


Рисунок 15 – Приложение перед редактированием

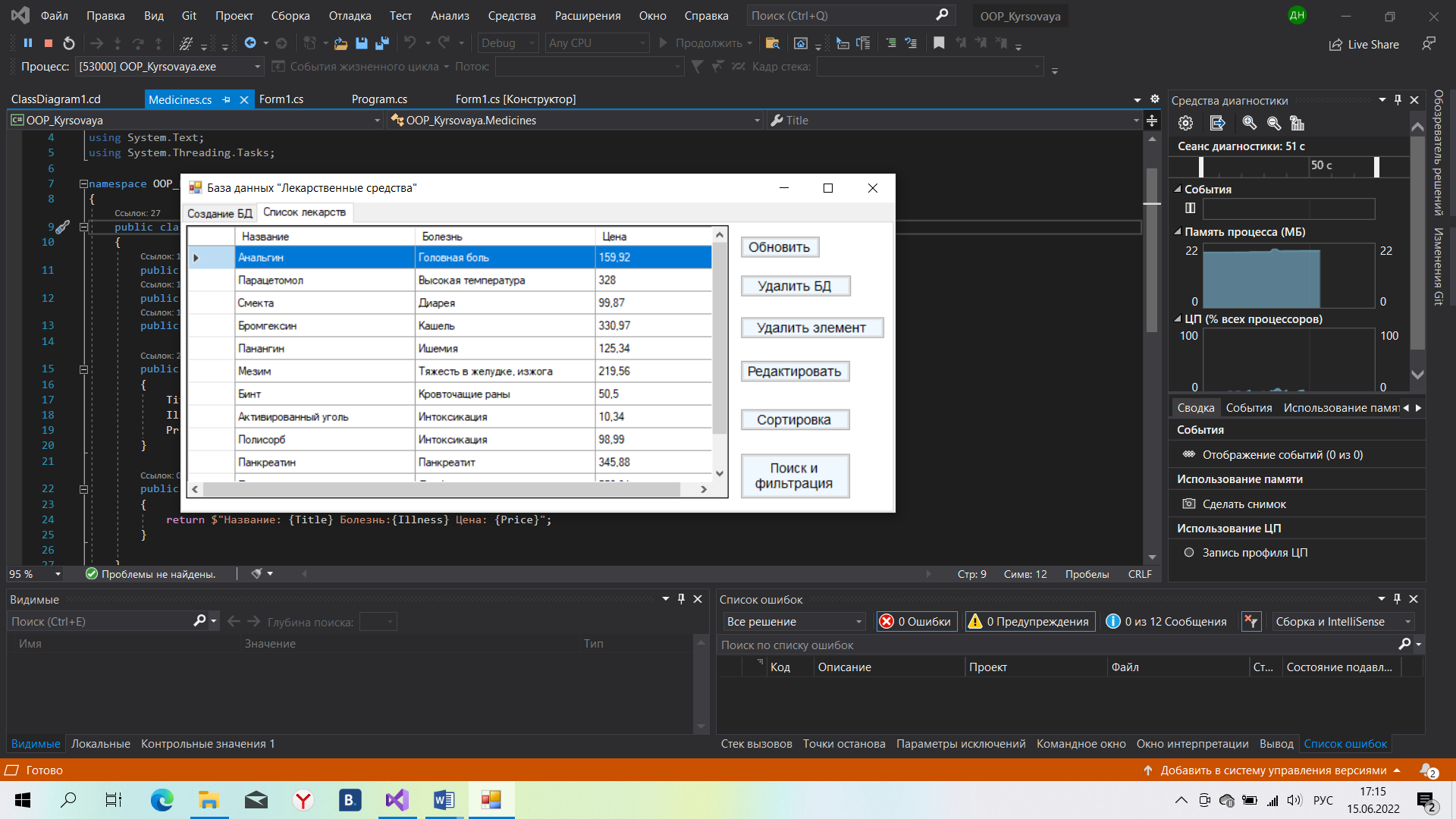


Рисунок 16 – Приложение после редактирования

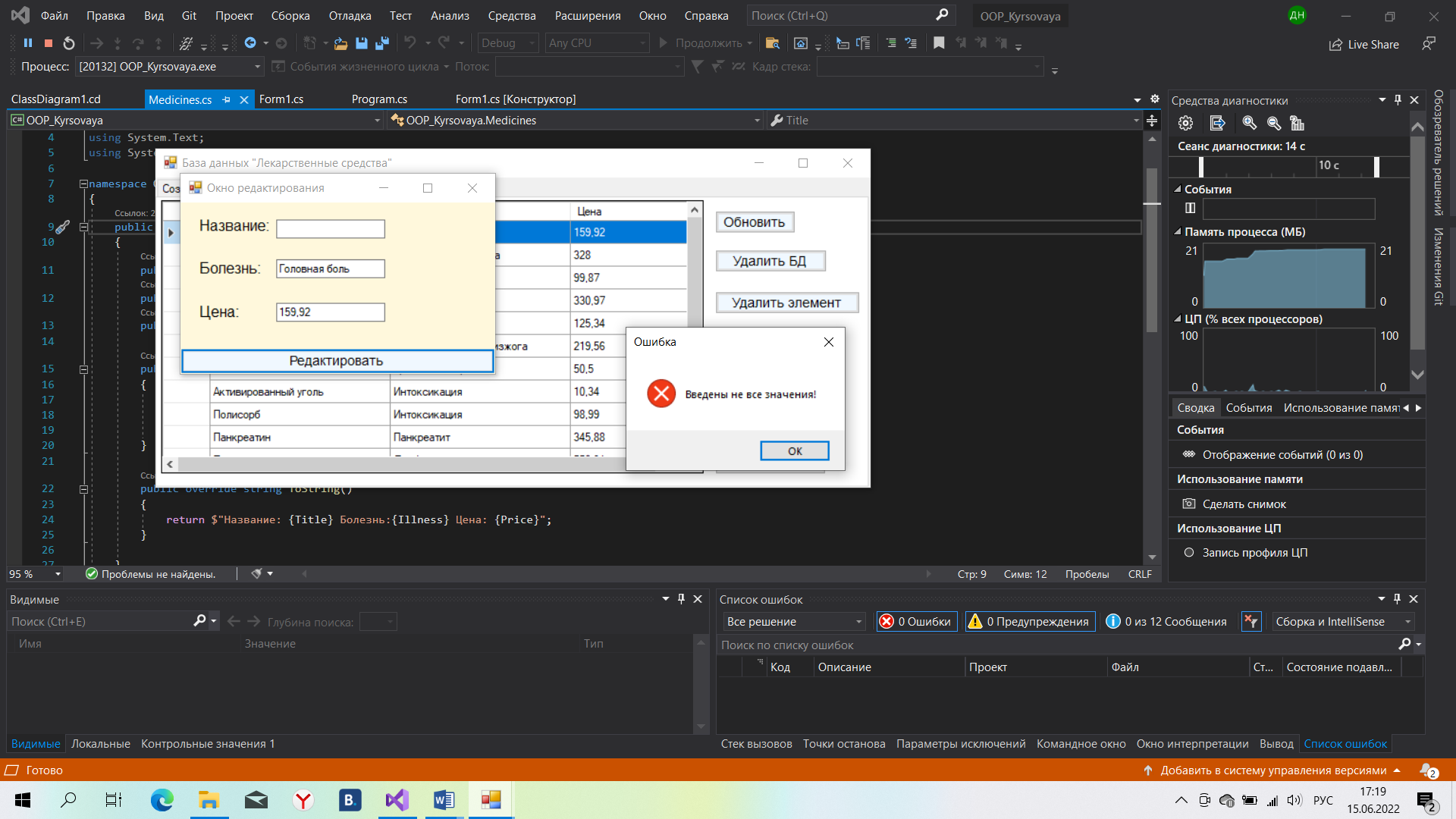


Рисунок 17 – Ошибка при редактировании

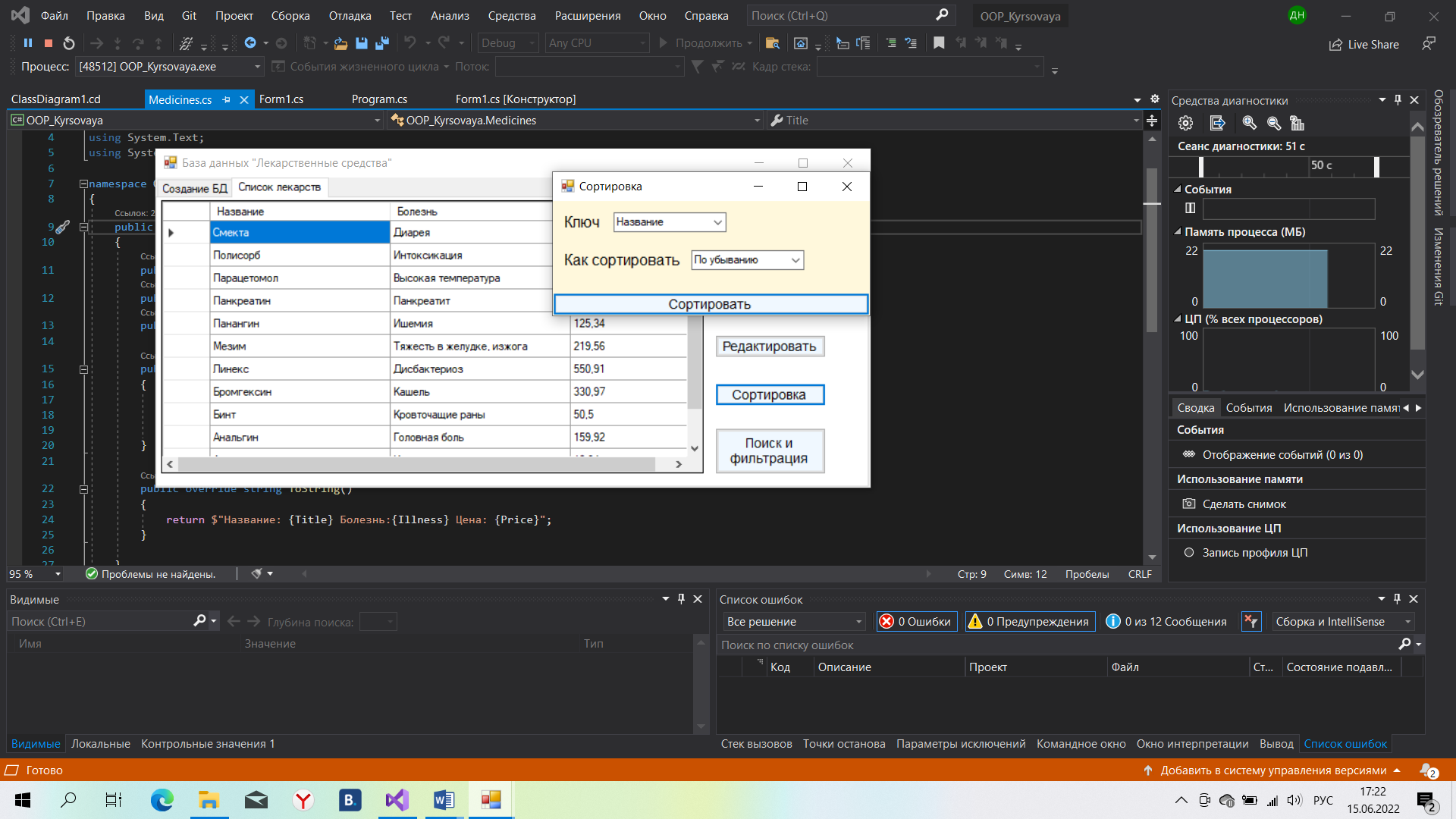


Рисунок 18 – Сортировка данных в обратном алфавитном порядке

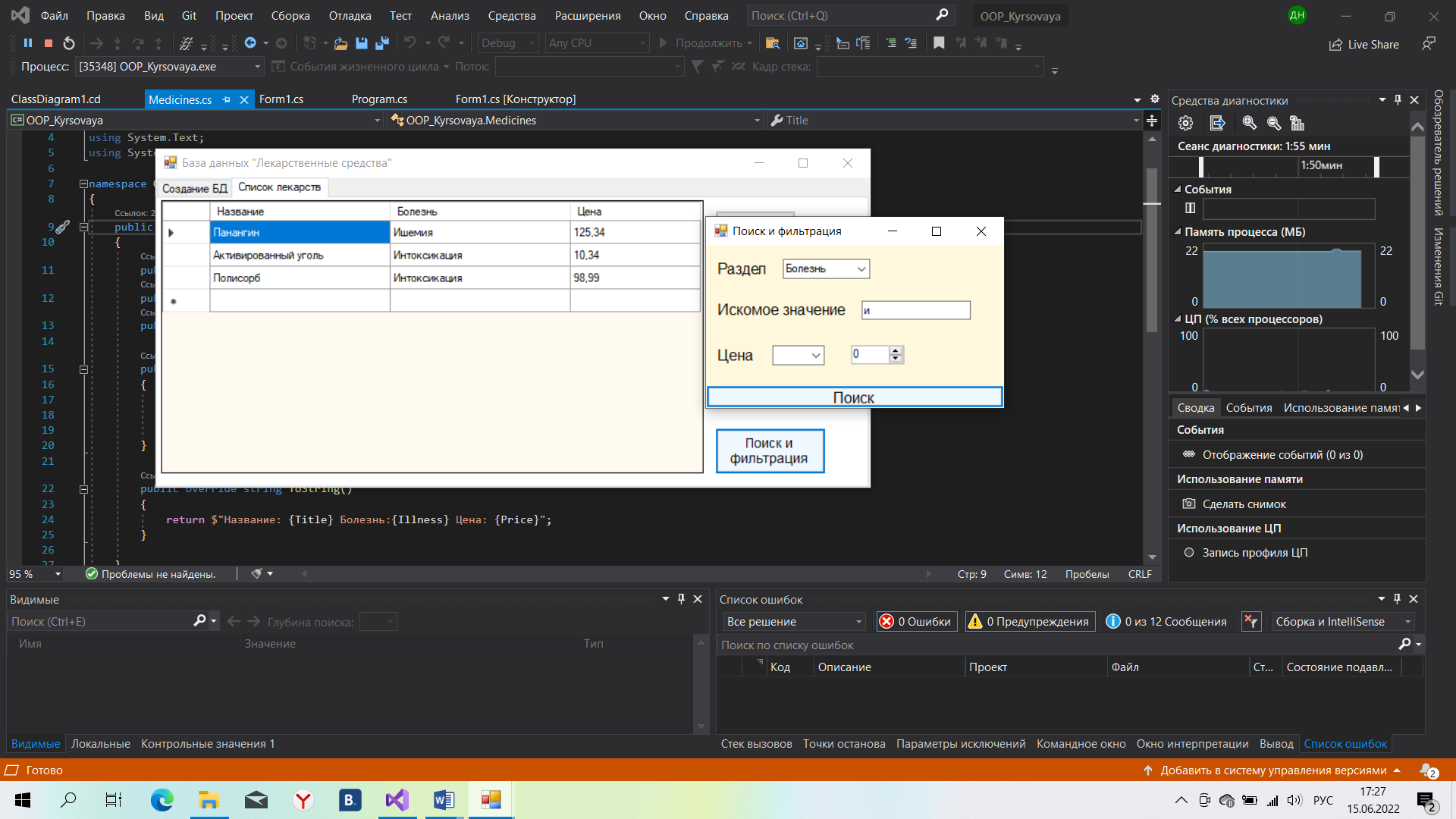


Рисунок 19 – Поиск данных

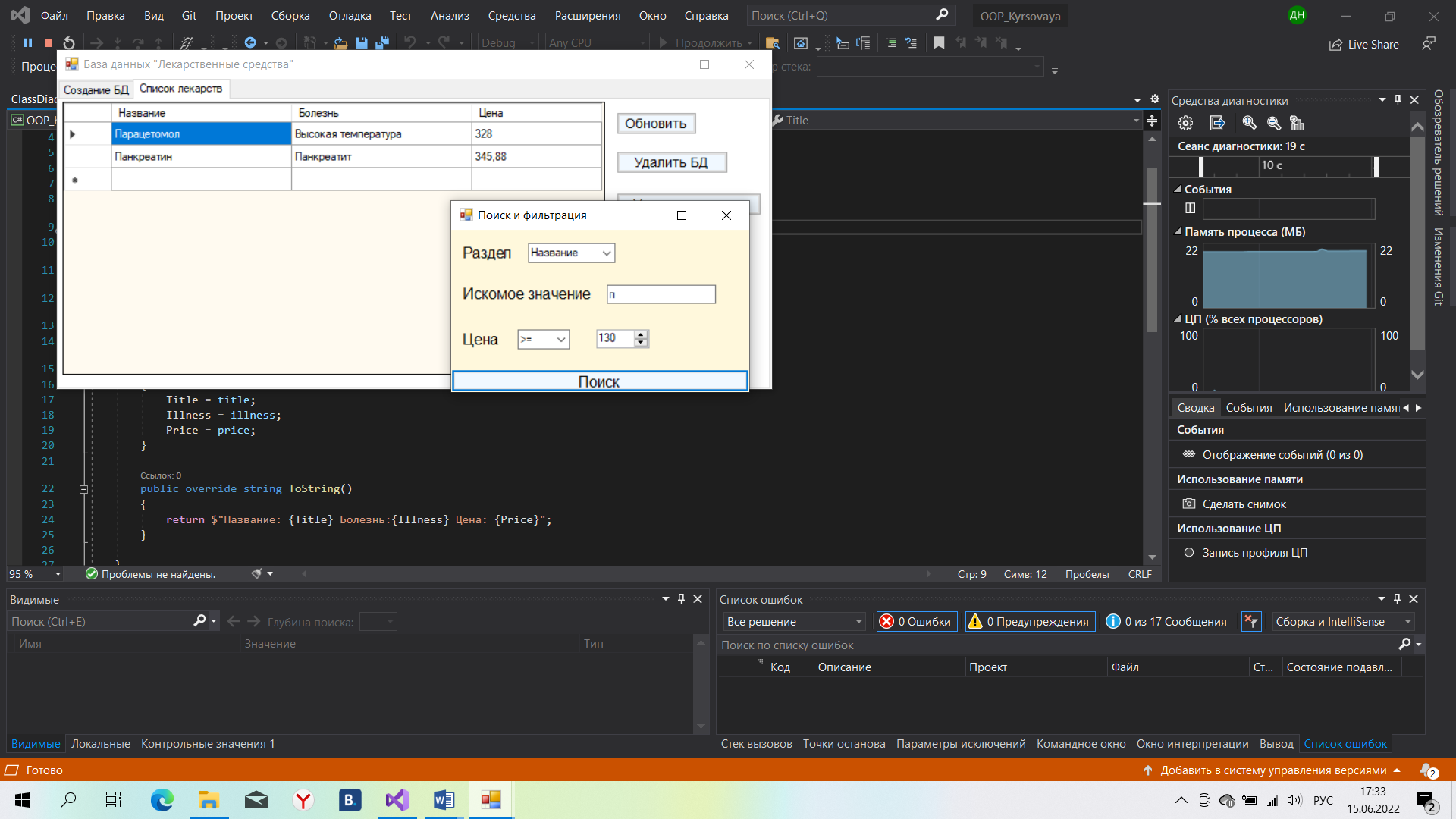


Рисунок 20 – Поиск данных с фильтрацией

В ходе тестирования было выявлено, что ожидаемый результат совпадает с наблюдаемым. Программа работает корректно.

1. **Руководство пользователя**

После запуска приложения появится информация об авторе и потом откроется главное меню (рисунки 21 – 23).

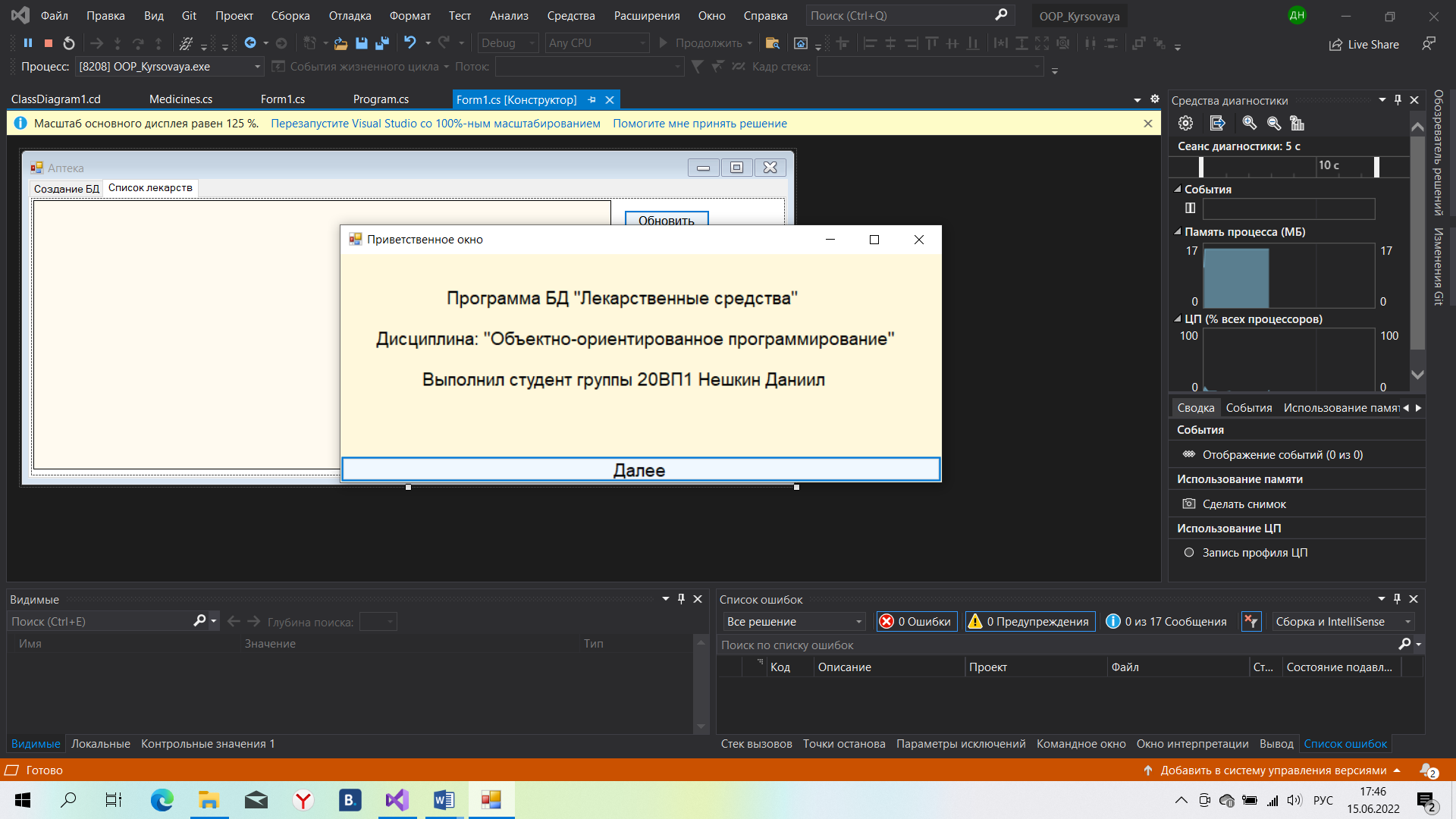


Рисунок 21 – Информация о проекте

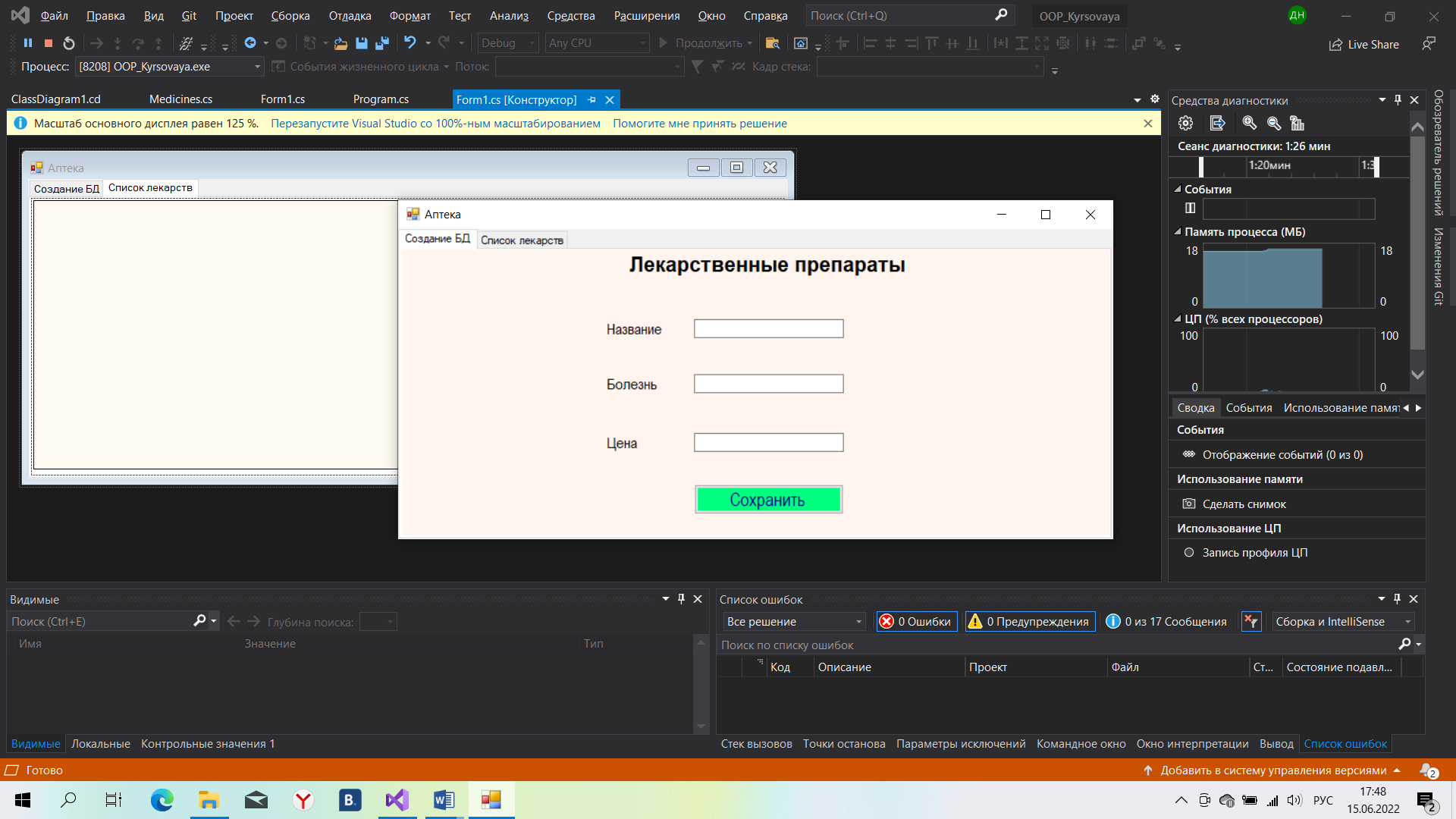


Рисунок 22 – Меню добавления записей в базу данных

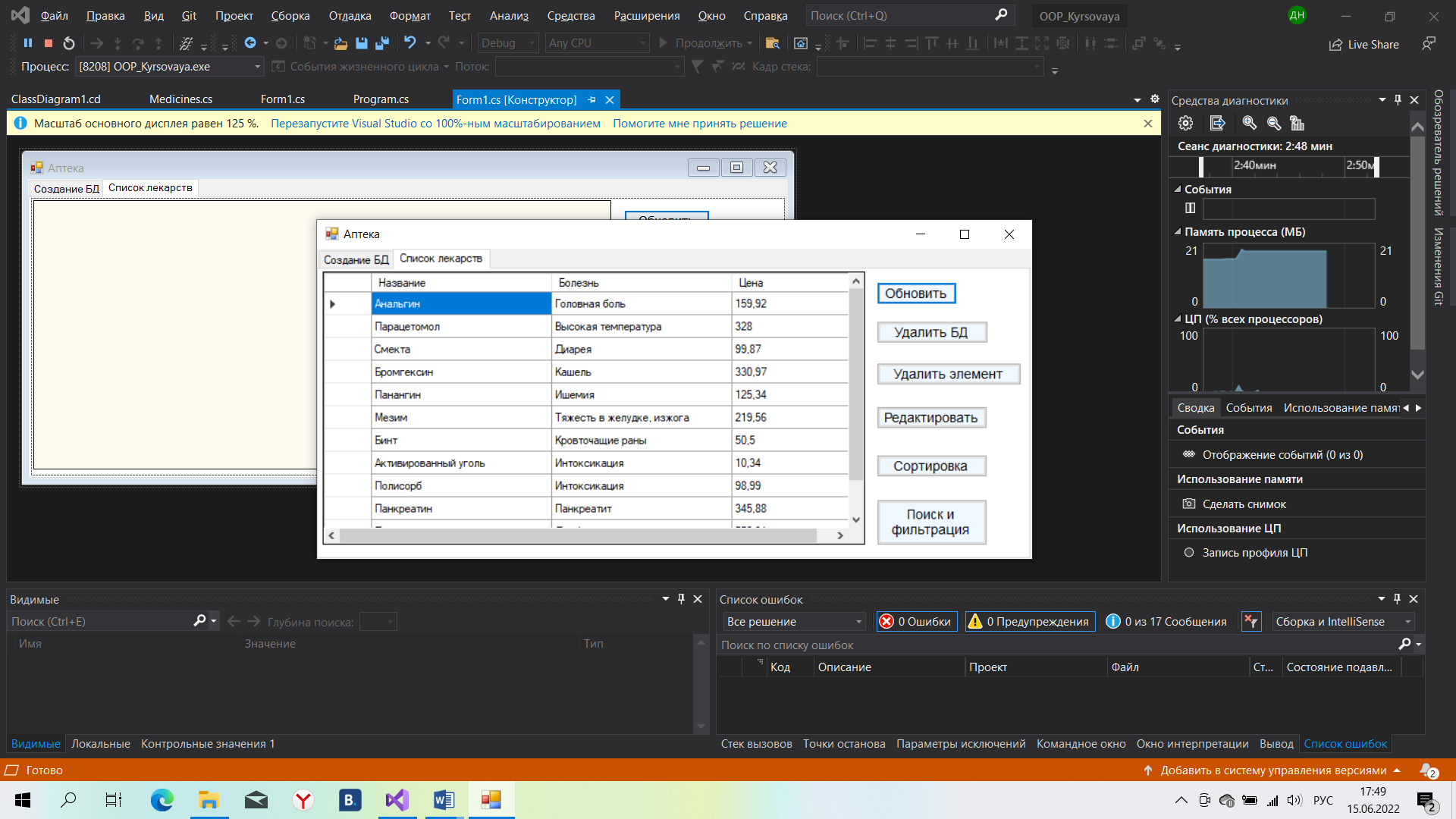


Рисунок 23 – Отображение базы данных. Взаимодействие с базой данных

# **Заключение**

В рамках данной курсового проекта были выполнены все поставленные задачи:

* Составлены требования к программе
* Спроектирована программа
* Реализована программа
* Протестирована программа
* Проведено сравнение эффективности реализованных методов
* Проанализированы результаты.

Результатом курсовой работы является программа, которая позволяет:

* Добавлять медикаменты в базу данных
* Выводить все медикаменты на экран в табличном виде
* Удалять медикаменты
* Удалять БД
* Сортировать медикаменты в различных порядках
* Сортировать медикаменты по различным значениям
* Редактировать данные о медикаментах
* Искать медикаменты в базе данных
* Фильтровать данные о медикаментах по различным значениям

Полученное приложение удовлетворяет всем изложенным требованиям.

# **Список используемых источников**

1. Информационная система. // Википедия [Электронный ресурс]. URL: [Информационная система - Википедия (turbopages.org)](https://translated.turbopages.org/proxy_u/en-ru.ru.2b82aa95-62a9f3ea-23bbe245-74722d776562/https/en.wikipedia.org/wiki/Information_systems_%28discipline%29)
2. Жизненный цикл разработки ПО. // Apmath [Электронный ресурс]. URL: [zhiznennyy\_cikl\_razrabotki\_programmnogo\_obespecheniya\_chast\_1\_.docx (live.com)](https://view.officeapps.live.com/op/view.aspx?src=http%3A%2F%2Fwww.apmath.spbu.ru%2Fru%2Fstaff%2Fdolzhikov_vv%2Ffiles%2Fzhiznennyy_cikl_razrabotki_programmnogo_obespecheniya_chast_1_.docx%23%3A~%3Atext%3D%25D0%2596%25D0%25B8%25D0%25B7%25D0%25BD%25D0%25B5%25D0%25BD%25D0%25BD%25D1%258B%25D0%25B9%2520%25D1%2586%25D0%25B8%25D0%25BA%25D0%25BB%2520%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B5%25D0%25BA%25D1%2582%25D0%25B0%2520RUP%2520%25D1%2581%25D0%25BE%25D1%2581%25D1%2582%25D0%25BE%25D0%25B8%25D1%2582%2C%25D0%25BD%25D0%25B0%25D0%25B7%25D0%25BD%25D0%25B0%25D1%2587%25D0%25B5%25D0%25BD%25D0%25B8%25D0%25B5%2520%25D0%25B8%2520%25D1%2581%25D0%25BE%25D0%25B4%25D0%25B5%25D1%2580%25D0%25B6%25D0%25B0%25D0%25BD%25D0%25B8%25D0%25B5%2520%25D0%25BF%25D1%2580%25D0%25B8%25D0%25BD%25D1%2586%25D0%25B8%25D0%25BF%25D0%25B8%25D0%25B0%25D0%25BB%25D1%258C%25D0%25BD%25D0%25BE%2520%25D1%2580%25D0%25B0%25D0%25B7%25D0%25BB%25D0%25B8%25D1%2587%25D0%25BD%25D1%258B&wdOrigin=BROWSELINK)
3. Гибкие методологии разработки URL: <https://test-otg.dev.dosvit.org.ua/storage/survey/attachments/3d3fac769c788420b9f25286fdb1eb48.pdf>
4. Как сериализировать и десериализировать (маршалирование и демаршалирование) JSON в .NET // Microsoft [Электронный ресурс]. URL: [Как сериализировать и десериализировать JSON с помощью C# для .NET | Microsoft Docs](https://docs.microsoft.com/ru-ru/dotnet/standard/serialization/system-text-json-how-to?pivots=dotnet-6-0)

# **Приложение А. Код программы**

**Program.cs**

using System;

using System.Windows.Forms;

namespace OOP\_Kyrsovaya

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Hello\_Form());

Application.Run(new Form1());

}

}

}

**Medicines.cs**

namespace OOP\_Kyrsovaya

{

public class Medicines

{

public string Title { get; set; }

public string Illness { get; set; }

public double Price { get; set; }

/// <summary>

/// Конструктор

/// </summary>

/// <param name="title"></param>

/// <param name="illness"></param>

/// <param name="price"></param>

public Medicines(string title, string illness, double price)

{

Title = title;

Illness = illness;

Price = price;

}

/// <summary>

/// Вывод класса

/// </summary>

/// <returns></returns>

public override string ToString()

{

return $"Название: {Title} Болезнь:{Illness} Цена: {Price}";

}

}

}

**Hello\_Form.cs**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace OOP\_Kyrsovaya

{

public partial class Hello\_Form : Form

{

public Hello\_Form()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Close();

}

private void Hello\_Form\_Load(object sender, EventArgs e)

{

this.MinimumSize = new Size(650, 280);

this.MaximumSize = new Size(670, 287);

}

}

}

**SortForm.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace OOP\_Kyrsovaya

{

public partial class SortForm : Form

{

static Form1 sourceForm;

public SortForm(Form1 form)

{

InitializeComponent();

sourceForm = form;

}

/// <summary>

/// сортировка

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button1\_Click(object sender, EventArgs e)

{

string key = comboBox1.Text;

string how = comboBox2.Text;

List<Medicines> list = new List<Medicines>();

List<Medicines> medicines = new List<Medicines>();

DataGridView view = sourceForm.dataGridView1;

for (int i = 0; i < view.RowCount - 1; i++)

{

Medicines med = new Medicines(view[0, i].Value.ToString(), view[1, i].Value.ToString(), double.Parse(view[2, i].Value.ToString()));

list.Add(med);

}

switch (key)

{

case "Название":

if (how == "По возрастанию")

{

var data = from m in list

orderby m.Title

select m;

foreach (var m in data)

medicines.Add(m);

}

else if (how == "По убыванию")

{

var data = from m in list

orderby m.Title descending

select m;

foreach (var m in data)

medicines.Add(m);

}

break;

case "Болезнь":

if (how == "По возрастанию")

{

var data = from m in list

orderby m.Illness

select m;

foreach (var m in data)

medicines.Add(m);

}

else if (how == "По убыванию")

{

var data = from m in list

orderby m.Illness descending

select m;

foreach (var m in data)

medicines.Add(m);

}

break;

case "Цена":

if (how == "По возрастанию")

{

var data = from m in list

orderby m.Price

select m;

foreach (var m in data)

medicines.Add(m);

}

else if (how == "По убыванию")

{

var data = from m in list

orderby m.Price descending

select m;

foreach (var m in data)

medicines.Add(m);

}

break;

default:

break;

}

DataSet medData = new DataSet("MedicStore");

DataTable medTable = new DataTable("Lekarstva");

// добавляем таблицу в dataSet

medData.Tables.Add(medTable);

DataColumn nameColumn = new DataColumn("Название", Type.GetType("System.String"));

nameColumn.Unique = true;

DataColumn illColumn = new DataColumn("Болезнь", Type.GetType("System.String"));

DataColumn priceColumn = new DataColumn("Цена", Type.GetType("System.Double"));

medTable.Columns.Add(nameColumn);

medTable.Columns.Add(illColumn);

medTable.Columns.Add(priceColumn);

medTable.PrimaryKey = new DataColumn[] { medTable.Columns["Title"] };

sourceForm.dataGridView1.DataSource = medData.Tables[0];

sourceForm.dataGridView1.Columns[0].Width = 190;

sourceForm.dataGridView1.Columns[1].Width = 190;

sourceForm.dataGridView1.Columns[2].Width = 137;

foreach (var med in medicines)

{

medTable.Rows.Add(new object[] { med.Title, med.Illness, med.Price });

}

}

/// <summary>

/// Задает размеры окна

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void SortForm\_Load(object sender, EventArgs e)

{

this.MaximumSize = new System.Drawing.Size(350, 160);

this.MinimumSize = new System.Drawing.Size(350, 160);

}

}

}

**FiltrForm.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace OOP\_Kyrsovaya

{

public partial class FiltrForm : Form

{

Form1 form1;

public FiltrForm()

{

InitializeComponent();

}

public FiltrForm(Form1 form)

{

InitializeComponent();

form1 = form;

}

/// <summary>

/// Обновить таблицу

/// </summary>

/// <param name="medicines"></param>

private void Update(List<Medicines> medicines)

{

DataSet medData = new DataSet("MedicStore");

DataTable medTable = new DataTable("Lekarstva");

// добавляем таблицу в dataSet

medData.Tables.Add(medTable);

DataColumn nameColumn = new DataColumn("Название", Type.GetType("System.String"));

nameColumn.Unique = true;

DataColumn illColumn = new DataColumn("Болезнь", Type.GetType("System.String"));

DataColumn priceColumn = new DataColumn("Цена", Type.GetType("System.Double"));

medTable.Columns.Add(nameColumn);

medTable.Columns.Add(illColumn);

medTable.Columns.Add(priceColumn);

medTable.PrimaryKey = new DataColumn[] { medTable.Columns["Title"] };

form1.dataGridView1.DataSource = medData.Tables[0];

form1.dataGridView1.Columns[0].Width = 190;

form1.dataGridView1.Columns[1].Width = 190;

form1.dataGridView1.Columns[2].Width = 137;

//List<Medicines> medicines = Form1.ReadAllFromDB();

foreach (var med in medicines)

{

medTable.Rows.Add(new object[] { med.Title, med.Illness, med.Price });

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

/// <summary>

/// Кнопка поиска

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button1\_Click(object sender, EventArgs e)

{

string razdel = comboBox1.Text;

string znach = textBox1.Text;

string sign = comboBox2.Text;

double price = (double)numericUpDown1.Value;

List<Medicines> values = new List<Medicines>();

List<Medicines> buffer = new List<Medicines>();

buffer = Form1.ReadAllFromDB();

if (razdel == "Название" && znach != "")

{

var find = buffer.Where(f => f.Title.ToUpper().StartsWith(znach.ToUpper()));

foreach (var i in find)

values.Add(i);

}else if(razdel == "Болезнь" && znach != "")

{

var find = buffer.Where(f => f.Illness.ToUpper().StartsWith(znach.ToUpper()));

foreach (var i in find)

values.Add(i);

}

if(sign != "" && comboBox1.SelectedIndex != 2)

{

buffer.Clear();

if (sign == ">=")

{

var find = values.Where(f => f.Price >= price);

foreach (var i in find)

buffer.Add(i);

}else if(sign == "<=")

{

var find = values.Where(f => f.Price <= price);

foreach (var i in find)

buffer.Add(i);

}

Update(buffer);

}

else if(sign == "")

{

Update(values);

}else if(comboBox1.SelectedIndex == 2)

{

if (sign == ">=")

{

var find = buffer.Where(f => f.Price >= price);

foreach (var i in find)

values.Add(i);

}

else if (sign == "<=")

{

var find = buffer.Where(f => f.Price <= price);

foreach (var i in find)

values.Add(i);

}

Update(values);

}

}

private void FiltrForm\_Load(object sender, EventArgs e)

{

this.MinimumSize = new Size(330, 210);

this.MaximumSize = new Size(330, 210);

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 2)

textBox1.Enabled = false;

else

textBox1.Enabled = true;

}

}

}

**EditForm.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

namespace OOP\_Kyrsovaya

{

public partial class EditForm : Form

{

static Form1 sourceForm;

/// <summary>

/// Старое название

/// </summary>

static string oldTitle;

public EditForm()

{

InitializeComponent();

}

/// <summary>

/// Считываем данные из полей

/// </summary>

/// <param name="form"></param>

public EditForm(Form1 form)

{

InitializeComponent();

foreach (DataGridViewRow row in form.dataGridView1.SelectedRows)

{

textBox1.Text = form.dataGridView1[0, row.Index].Value.ToString();

oldTitle = form.dataGridView1[0, row.Index].Value.ToString();

textBox2.Text = form.dataGridView1[1, row.Index].Value.ToString();

textBox3.Text = form.dataGridView1[2, row.Index].Value.ToString();

}

sourceForm = form;

}

private void EditForm\_Load(object sender, EventArgs e)

{

this.MaximumSize = new Size(348, 220);

this.MinimumSize = new Size(348, 220);

}

/// <summary>

/// Редактирование

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button1\_Click(object sender, EventArgs e)

{

string newTitle = textBox1.Text;

string newIllness = textBox2.Text;

if(newTitle == "" || newIllness == "")

{

Form1.printError("Введены не все значения!");

return;

}

string newStrPrice = textBox3.Text;

double newPrice = 0;

try

{

newPrice = double.Parse(newStrPrice);

}

catch (FormatException)

{

Form1.printError("Неправильный формат числа!");

return;

}

if(newPrice < 0)

{

Form1.printError("Неправильный формат числа!");

return;

}

Form1.EditDB(oldTitle, newTitle, newIllness, newPrice);

}

private void EditForm\_FormClosed(object sender, FormClosedEventArgs e)

{

DataSet medData = new DataSet("MedicStore");

DataTable medTable = new DataTable("Lekarstva");

// добавляем таблицу в dataSet

medData.Tables.Add(medTable);

DataColumn nameColumn = new DataColumn("Название", Type.GetType("System.String"));

nameColumn.Unique = true;

DataColumn illColumn = new DataColumn("Болезнь", Type.GetType("System.String"));

DataColumn priceColumn = new DataColumn("Цена", Type.GetType("System.Double"));

medTable.Columns.Add(nameColumn);

medTable.Columns.Add(illColumn);

medTable.Columns.Add(priceColumn);

medTable.PrimaryKey = new DataColumn[] { medTable.Columns["Title"] };

sourceForm.dataGridView1.DataSource = medData.Tables[0];

sourceForm.dataGridView1.Columns[0].Width = 190;

sourceForm.dataGridView1.Columns[1].Width = 190;

sourceForm.dataGridView1.Columns[2].Width = 137;

List<Medicines> medicines = Form1.ReadAllFromDB();

foreach (var med in medicines)

{

medTable.Rows.Add(new object[] { med.Title, med.Illness, med.Price });

}

}

}

}

**Form1.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using System.IO;

using Newtonsoft.Json;

namespace OOP\_Kyrsovaya

{

public partial class Form1 : Form

{

/// <summary>

/// Имя файла

/// </summary>

static string fileDB = "Medicaments.txt";

public Form1()

{

InitializeComponent();

}

public Form1(Hello\_Form form)

{

InitializeComponent();

form.Close();

}

/// <summary>

/// Вывод окна ошибки с введенным текстом

/// </summary>

/// <param name="text">Текст ошибки</param>

public static void printError(string text)

{

MessageBox.Show(

text,

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error

);

}

private void Form1\_Load(object sender, EventArgs e)

{

this.MinimumSize = new Size(770, 366);

this.MaximumSize = new Size(770, 366);

if (!File.Exists(fileDB))

{

File.Create(fileDB);

}

}

/// <summary>

/// Чтение из базы данных

/// </summary>

/// <returns>Коллекция медикаментов</returns>

public static List<Medicines> ReadAllFromDB()

{

string json = File.ReadAllText(fileDB);

List<Medicines> allMedicaments = JsonConvert.DeserializeObject<List<Medicines>>(json); // перевод обратно из json

return allMedicaments ?? new List<Medicines>();

}

/// <summary>

/// Запись в базу данных

/// </summary>

/// <param name="medicament">Медикамент</param>

static void SaveToDB(Medicines medicament)

{

List<Medicines> allMedicaments = ReadAllFromDB();

if(allMedicaments.Count(u => u.Title == medicament.Title) > 0)

{

printError("Такой медикамент уже есть в БД!");

return;

}

allMedicaments.Add(medicament);

string serializedMedicaments = JsonConvert.SerializeObject(allMedicaments); // перевод в json

File.WriteAllText(fileDB, serializedMedicaments);

}

/// <summary>

/// Запись в базу данных

/// </summary>

/// <param name="medicaments">Коллекция медикаментов</param>

static void SaveToDB(List<Medicines> medicaments)

{

string serializedMedicaments = JsonConvert.SerializeObject(medicaments);

File.WriteAllText(fileDB, serializedMedicaments);

}

/// <summary>

/// Удаление из БД

/// </summary>

/// <param name="title"></param>

/// <returns></returns>

static bool DeletFromDB(string title)

{

bool result;

List<Medicines> allMedicaments = ReadAllFromDB();

Medicines medicamentForDeletion = allMedicaments.FirstOrDefault(u => u.Title == title);

if (medicamentForDeletion != null)

{

allMedicaments.Remove(medicamentForDeletion);

SaveToDB(allMedicaments);

result = true;

}

else

{

result = false;

}

return result;

}

/// <summary>

/// Удаление БД

/// </summary>

static void DeleteDB()

{

File.WriteAllText(fileDB, "");

MessageBox.Show(

"База данных успешно удалена!",

"Сообщение",

MessageBoxButtons.OK,

MessageBoxIcon.Information

);

}

/// <summary>

/// Редактировать все

/// </summary>

/// <param name="title">Ключ записи для редактирования</param>

/// <param name="newTitle">Новое название</param>

/// <param name="newIllness">Новая болезнь</param>

/// <param name="newPrice">Новая цена</param>

public static void EditDB(string title, string newTitle, string newIllness, double newPrice)

{

string json = File.ReadAllText(fileDB);

List<Medicines> allMedicaments = JsonConvert.DeserializeObject<List<Medicines>>(json);

int index = allMedicaments.FindIndex(m => m.Title == title);

allMedicaments[index].Title = newTitle;

allMedicaments[index].Illness = newIllness;

allMedicaments[index].Price = newPrice;

SaveToDB(allMedicaments);

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

/// <summary>

/// Сохранение введенных данных в БД

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button1\_Click(object sender, EventArgs e)

{

string title = textBox1.Text;

string illness = textBox2.Text;

double price = 0;

if(title == "" || illness == "")

{

MessageBox.Show(

"Введены не все данные!",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error

);

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

return;

}

try

{

price = double.Parse(textBox3.Text);

}

catch (FormatException)

{

printError("Неправильный формат цены!");

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

return;

}

if(price < 0)

{

printError("Неправильный формат цены!");

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

return;

}

Medicines medicament = new Medicines(title, illness, price);

SaveToDB(medicament);

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

/// <summary>

/// Обновить

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button2\_Click(object sender, EventArgs e)

{

DataSet medData = new DataSet("MedicStore");

DataTable medTable = new DataTable("Lekarstva");

// добавляем таблицу в dataSet

medData.Tables.Add(medTable);

DataColumn nameColumn = new DataColumn("Название", Type.GetType("System.String"));

nameColumn.Unique = true;

DataColumn illColumn = new DataColumn("Болезнь", Type.GetType("System.String"));

DataColumn priceColumn = new DataColumn("Цена", Type.GetType("System.Double"));

medTable.Columns.Add(nameColumn);

medTable.Columns.Add(illColumn);

medTable.Columns.Add(priceColumn);

medTable.PrimaryKey = new DataColumn[] { medTable.Columns["Title"] };

dataGridView1.DataSource = medData.Tables[0];

dataGridView1.Columns[0].Width = 190;

dataGridView1.Columns[1].Width = 190;

dataGridView1.Columns[2].Width = 137;

List<Medicines> medicines = ReadAllFromDB();

foreach(var med in medicines)

{

medTable.Rows.Add(new object[] { med.Title, med.Illness, med.Price });

}

}

/// <summary>

/// Удалить БД

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button3\_Click(object sender, EventArgs e)

{

DeleteDB();

dataGridView1.DataSource = null;

}

/// <summary>

/// Удалить элемент(-ы)

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button4\_Click(object sender, EventArgs e)

{

foreach (DataGridViewRow row in dataGridView1.SelectedRows)

{

DeletFromDB(dataGridView1[0, row.Index].Value.ToString());

dataGridView1.Rows.Remove(row);

}

}

/// <summary>

/// Кнопка редактировать

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button5\_Click(object sender, EventArgs e)

{

int count = 0;

foreach (DataGridViewRow row in dataGridView1.SelectedRows)

{

if(row.Index != -1) count++;

}

if (count > 1 || count == 0) return;

else

{

EditForm editForm = new EditForm(this);

editForm.Show();

}

}

/// <summary>

/// Кнопка сортировка

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button6\_Click(object sender, EventArgs e)

{

SortForm sortForm = new SortForm(this);

sortForm.Show();

}

/// <summary>

/// Поиск и фильтрация

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button7\_Click(object sender, EventArgs e)

{

FiltrForm filtr = new FiltrForm(this);

filtr.Show();

}

}

}