Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Белгородский государственный технологический университет
им. В.Г. Шухова»
Институт энергетики, информационных технологий и управляющих систем
Кафедра информационных технологий

Курсовая работа

по дисциплине визуальное программирование на тему:
«Иерархия лекарственных средств»

Выполнил:
студент группы ИТ-32
Курбатова Софья Андреевна

Проверили
ст.преподаватель
Лазебная Е.А.

Белгород, 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Курсовой проект выполнен с целью решить практическое задание по дисциплине «Визуальное программирование» на языке C# в среде программирования MS Visual Studio.

Задание направлено на реализацию настольного приложения с удобным графическим интерфейсом, которое предназначено для просмотра информации о лекарственных средствах.

Под лекарственным средством понимают вещество или смесь веществ синтетического или природного происхождения в виде лекарственной формы (таблетки, капсулы, растворы, мази и т.д), применяемые для профилактики, диагностики, лечения различных заболеваний. Существует несколько классификаций, основанных на различных признаках лекарственных средств.

– По химическому строению;

– По происхождению – природные, минеральные, синтетические;

– Нозологическая классификация – классификация по заболеваниям, для лечения которых используется лекарственный препарат;

– По фармакологической группе – основана на воздействии препарата на организм человека;

– Анатомо-терапевтическо-химическая классификация (Anatomical Therapeutic Chemical) (АТХ или АТС) – международная классификация, в которой учитывается фармакологическая группа препарата, его химическая природа и нозология заболевания, для лечения которого предназначен препарата.

Для выполнения задания в рамках курсового проекта была выбрана Анатомо-терапевтическо-химическая классификация, в связи с тем, что АТХ подразделяет лекарственные средства на группы, имеющие 5 различных уровней:

– анатомический орган или система;

– основные терапевтические /фармакологические;

– терапевтические/фармакологические;

– терапевтические/фармакологические/основные химические;

– по химической структуре.

Каждая группа в зависимости от уровня имеет буквенный или цифровой код. В большинстве случаев каждому лекарственному средству присваивается только один АТХ-код. Лекарственным средствам, имеющим несколько основных показаний для медицинского применения, может быть присвоено более одного АТХ-кода.

1. ПОСТАНОВКА ЗАДАЧИ И ОПРЕДЕЛЕНИЕ ОСНОВНЫХ ТРЕБОВАНИЙ К РАЗРАБАТЫВАЕМОМУ ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

1.1. Постановка задачи

Составить иерархию объектов «Лекарственные средства» и разработать интерфейс с применением TreeView. Реализовать функции сохранения в файл и загрузки из файла графа TreeView с помощью механизма сериализации. По размеру иерархия классов должна быть не менее чем трехуровневой. Приложение должно использовать минимум 2 разработанных пользовательских элемента управления с собственным набором свойств и методов.

В качестве реализуемых элементов управления выбраны:

– «Убегающий» ЭУ при наступлении события;

– ЭУ, изменяющий свой цвет при наступлении события;

1.2. Основание для разработки

Программа разрабатывается на основе учебного плана кафедры «Информационные технологии» БГТУ им. Шухова для специальности 09.03.02. «Информационные системы и технологии».

1.3. Назначение программного средства «Иерархия лекарственных средств»

Программное средство «Иерархия лекарственных средств» является приложением для визуального представления иерархии лекарственных средств. Программа может быть использована в домашних условиях. .

1.4. Требования к программному средству

1.4.1. Требования к функциональным характеристикам

В приложении должны быть реализованы такие характеристики как:

– Иерархия объектов должна состоять минимум из 3 уровней;

– Для хранения в памяти экземпляров созданных классов использовать обобщенные коллекции LIST<T>;

– Реализовать методы: Конструктор, и еще 2-3 метода работы с объектами;

– Приложение должно реализовывать возможности:

– Создания/ удаления объектов иерархии;

– Изменения характеристик объектов;

– Визуализации объектов;

– Манипуляции объектами на форме при помощи мыши;

– Сохранения/считывания текущего состояния иерархии объектов в формате XML;

– Работы реализованных методов;

Приложение должно отвечать требованиям Windows-приложений:

– Система меню, панель инструментов, горячие клавиши;

– Система подсказок;

– Обработка событий клавиатуры и мыши;

– Переключение фокуса;

– Обработка исключительных ситуаций;

1.4.2.Требования к надежности

– Предусмотреть контроль вводимой информации;

– Блокировка некорректных действий пользователя при работе с программой;

1.4.3. Требования к условиям эксплуатации

Необходимо соблюдать условия эксплуатации персонального компьютера или ноутбука, указанные поставщиком оборудования (не допускать перегрева, попадания жидкости).

1.4.4. Требования к составу и параметрам технических средств

Наличие IBM PC. ПК должен быть совместим с видеоадаптером Intel HD Graphics. Строго необходимо наличие клавиатуры. Необходимое дисковое пространство – не менее 132 Кб. RAM – более 7 МБ. Монитор, поддерживающий расширение от 800x600 до 1920x1080 и больше.

1.4.5. Требования к информационно-программной совместимости

Пользовательский интерфейс должен быть интуитивно понятным и содержать подсказки.

Системные программные средства, используемые программой, должны быть представлены локализованной версией операционной системы Windows (Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10). Базовый язык программирования C#.

2.ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1. Разработка структурной схемы программы

В программе можно выделить 4 основных блока: Помощь, Работа с объектами (включая добавление и удаление), Сохранение состояния объектов, Выход.

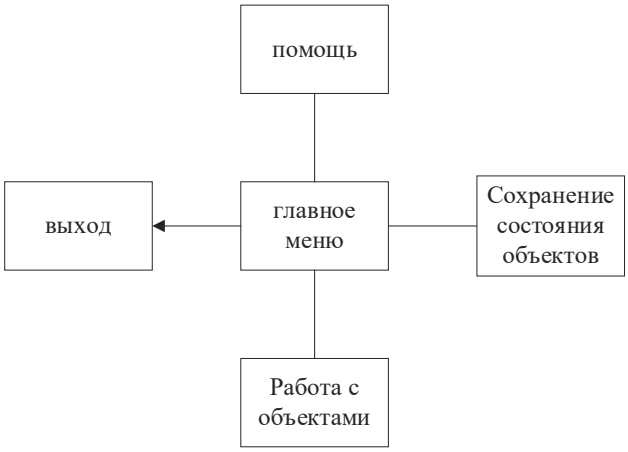

Рис. 2. 1. Структурная схема работы программы

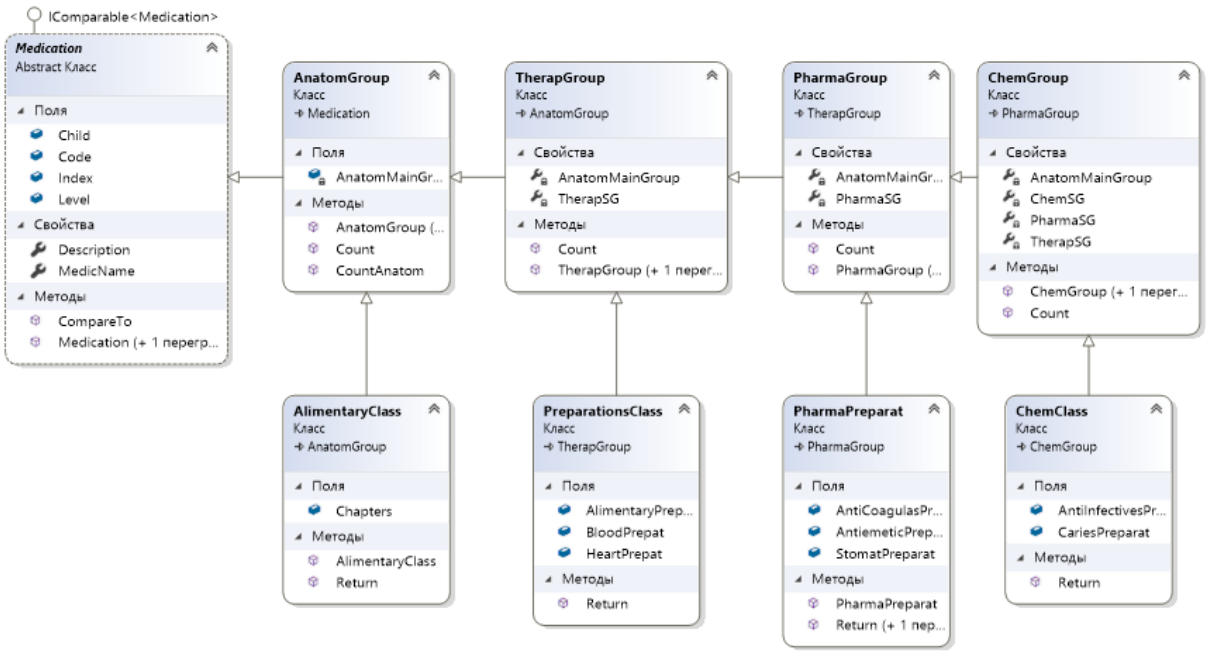

Рис. 2. 2. Диаграмма классов

2.2.Разработка интерфейса программы

В приложении реализовано 4 формы и два пользовательских элемента управления:

2.2.1.Формы

MainWindow – предназначена для отображения иерархии объектов с помощью TreeView. На данной форме (см. рисунок 2.3.)располагаются элементы управления для вызова формы поиска объектов, добавления объектов, удаления объектов. Определено меню (см. рисунок 2.4), для осуществления сохранения и загрузки объектов иерархии.

Форма состоит из следующих элементов:

– MenuStrip – используется для создания меню;

– TreeView – используется для отображения объектов в иерархии;

– Button – для выполнения различных действий;

– StatusStrip- для вывода информации о состоянии программы и текущей даты и времени;

– ToolTip – для вывода всплывающих подсказок при наведении курсора мыши на некоторые объекты;

– DataGridView – для вывода информации, содержащейся в объекте;

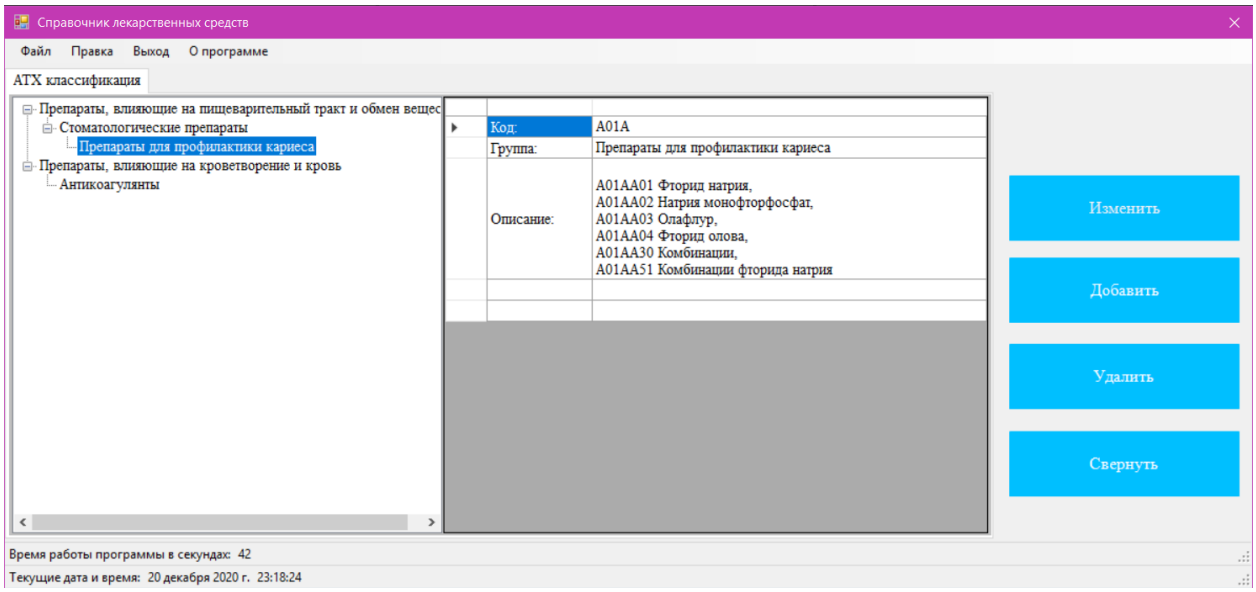

Рис. 2. 3. Главная форма

Главное меню формы состоит из вкладок: Файл, Действия, Выход, О программе.

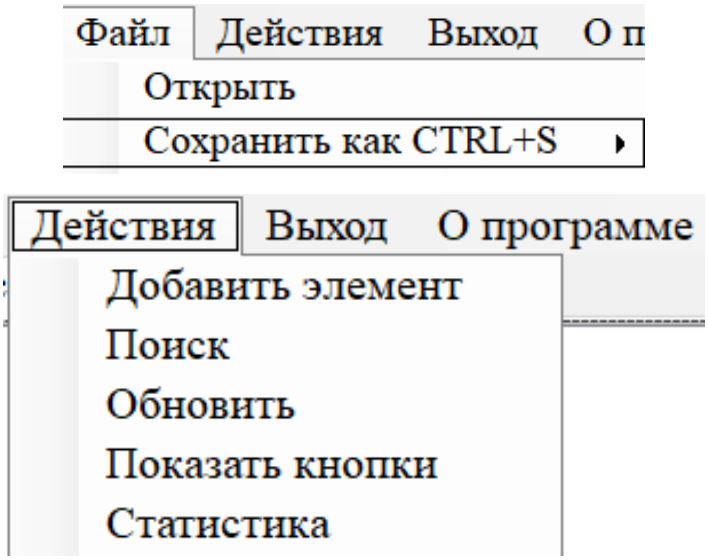

Рис. 2. 4. Пункты Меню

Во вкладке «Файл» перечислены функции, позволяющие выполнить сохранение выводимой иерархии классов в файл с расширением xml, а также выполнить загрузку из сохраненного файла. При нажатии на вкладку «Действия» можно увидеть функции, которые направлены на вызов формы добавления элементов, обновления выводимой информации, демонстрацию кнопок для управления программой. Функция «Поиск» предназначена для поиска в списке добавляемых в иерархию объектов нужного объекта, а «Статистика» выводит на экран количество объектов каждого класса.

AddForm – предназначена для создания и добавления в коллекцию новых объектов (см. рисунок 2.5).

Форма состоит из следующих элементов:

– ComboBox – в них перечислены все имеющиеся наименования объектов иерархии, которые можно использовать для формирования информации о нем;

– TextBox – нужные для ввода информации о таких полях класса как: «Группа» и «Описание»;

– Button – для добавления созданного объекта в список объектов и отмены произведенных действий;

Стоит отметить, что текстовое поле «Код группы» формируется динамически в зависимости от выбранного пользователем наименования группы (см. рисунок 2.6).

TextBox и Button являются здесь разработанными пользовательскими элементами управления: TextBoxNew и ButtonNew. При вводе текста в TextBox у элемента управления появляется заголовок, соответствующий описанию элемента. Кнопки Button меняют свой цвет при щелчке по ним.

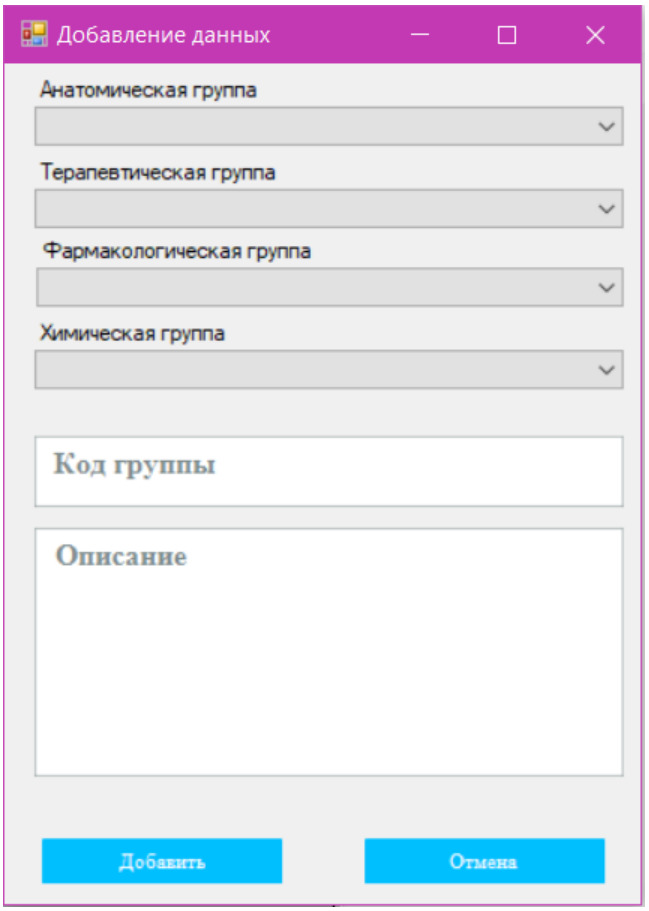

Рис. 2. 5. Форма добавления данных

SearchLine – предназначена для поиска объектов иерархии.

Форма состоит из следующих элементов:

– TextBox – для ввода имени класса, который необходимо найти;

– StaticText – отображает наименование описываемого действия;

– Button – на нее необходимо щелкнуть мышкой, чтобы выполнить поиск нужного элемента;

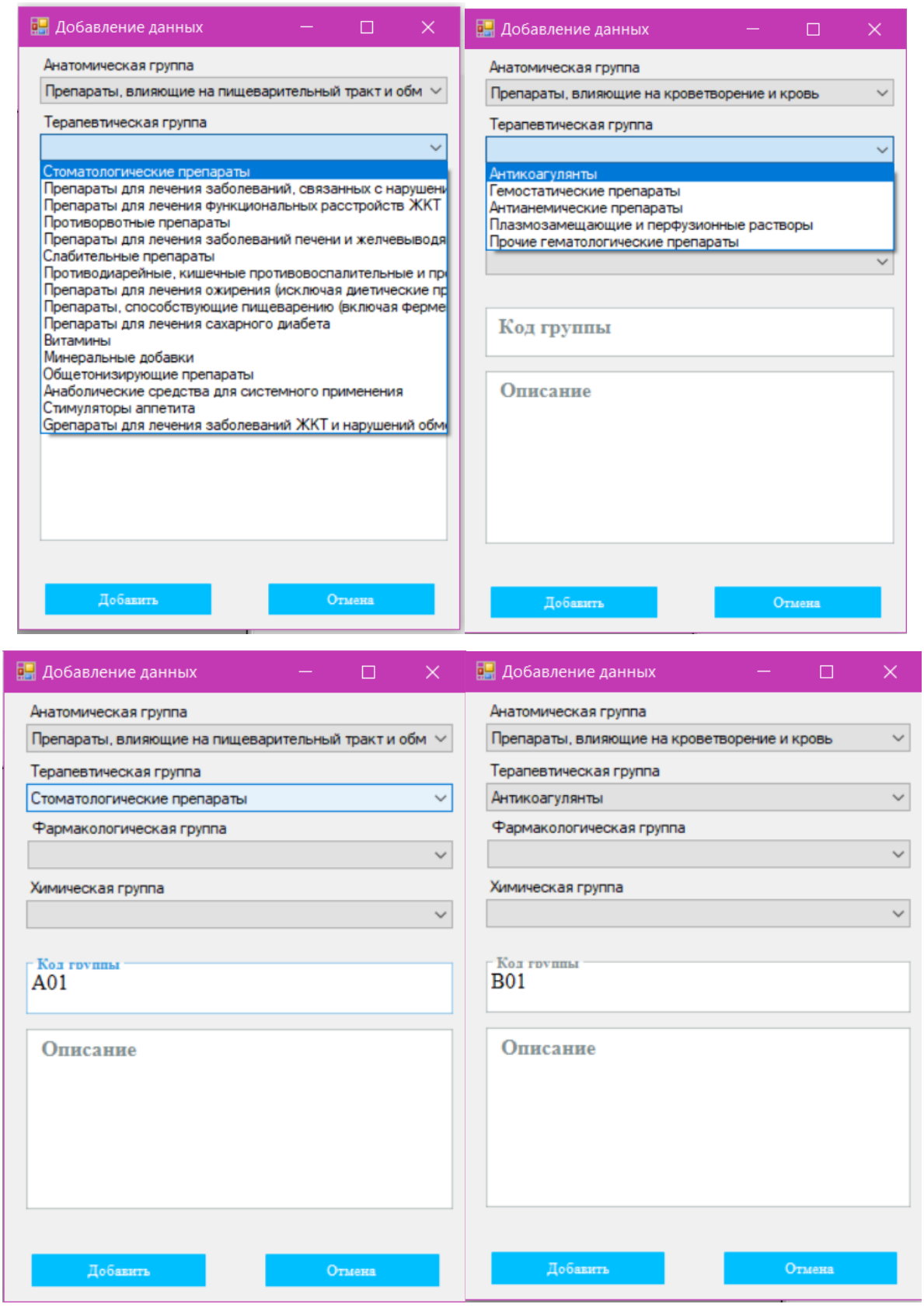

Рис. 2. 6. Демонстрация изменений

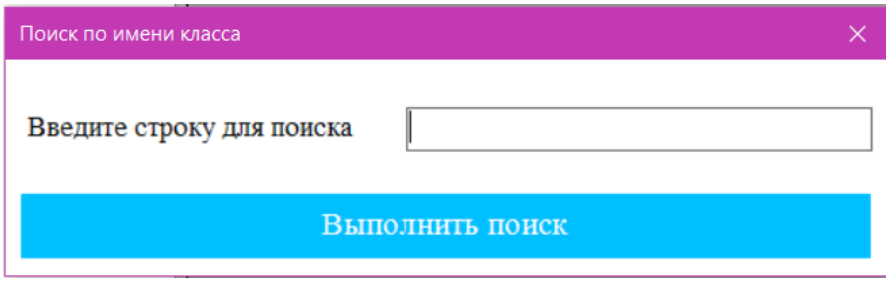

Рис. 2. 7. Форма поиска элементов

2.2.2. Разработка элементов управления

Элемент управления TextBoxNew предназначен для ввода информации от пользователя. Текст, определяющий заголовок элемента управления первоначально определен внутри поля ввода. При наступлении события заголовок «убегает» вверх. Элемент управления был построен на основе базового TextBox.

Элемент управления ButtonNew предназначен для запуска функций, осуществляющих взаимодействие с данными или для изменения текущего состояния формы. При наведении курсора мыши на элемент управления будет изменять своей цвет. Двойной щелчок по элементу управления приведет к появлению на нем эффекта «волны». Элемент управления был построен на основе базового Button.

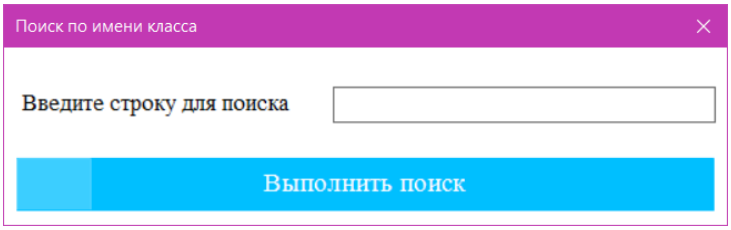

Рис. 2. 8. Появление белой полосы изменения цвета

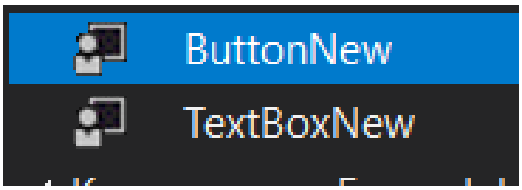

Рис. 2. 9. Демонстрация ЭУ на панели элементов управления

2.3.Разработка алгоритмов программы

2.3.1. Описание обработчиков событий и методов классов главной формы

В таблице перечислены основные обработчики событий и методы классов главной формы.

Таблица 2.1 Обработчики и методы

Наименование метода	Назначение	Входные данные	Выходные данные
1	**2**	**3**	**4**
Работа с содержимым списка объектов			
Public void ParentNodesMed()	вывод содержимого списка объектов в виде иерархической структуры	нет	заполненный TreeView
private void TreeView1_Node MouseDoubleClick():	отображение содержимого класса в таблице на главной форме	object sender, TreeNodeMouse ClickEventArgs e	содержимое объекта класса, перечисленное в таблице
private void StartSearh()	поиск класса по его имени на главной форме	string SearchText, TreeNode StartNode	выделенное имя класса
Работа с данными класса			
private void DataGridInit()	инициализация таблицы для вывода содержимого класса	нет	пустая таблица для вывода
private void DataDescription Grid_KeyDown	разрешение редактирования ячейки по щелчку мыши по ней	object sender, KeyEventArgs e	нет
private void EditButton_Click()	сохранение измененных данных	object sender, EventArgs e	
Общие элементы управления			
private void AddButton _Click_1	открытие формы для добавления данных	object sender, EventArgs e	открытая форма
private void DelButton_Click()	удаление дочерних узлов	object sender, EventArgs e	удаленный узел
private void Minim Button_Click()	закрытие таблицы содержимого класса	object sender, EventArgs e	скрытая таблица содержимого класса
private void СтатистикаTool StripMenuItem_Click()	вывод пользователю информацию о количестве объектов каждого класса	object sender, EventArgs e	количество объектов каждого класса
Работа с файлами			
private void SaveFile()	сохранение файла с объектами класса в файле с расширением xml	список с объектами класса	файл с расширением .xml с содержимым списка
private void XMLToolStrip MenuItem_Click()	загрузка файла с объектами класса	файл с объектами класса с расширением .xml	Заполненный TreeView

2.3.2. Описание обработчиков и методов формы добавления данных:

В таблице перечислены основные обработчики событий формы добавления данных.

Наименование метода	Назначение	Входные данные	Выходные данные
1	**2**	**3**	**4**
Работа с содержимым объектов класса			
private void AnatomCombo Box_SelectedIndexChanged()	формирование списка отображение подгруппы следующего класса, вывод кода класса	object sender, EventArgs e	вывод кода группы
private void TherapCombo Box_SelectedIndexChanged	формирование списка отображение подгруппы следующего класса, вывод кода класса	object sender, EventArgs e	вывод кода группы
private void PharmaCombo Box_SelectedIndexChanged	формирование списка отображение подгруппы следующего класса, вывод кода класса	object sender, EventArgs e	вывод кода группы
private void ChemCombo Box_SelectedIndexChanged	вывод кода класса	object sender, EventArgs e	вывод кода группы

2.3.3. Описание методов созданных классов:

Наименование метода	Назначение	Входные данные	Выходные данные
1	**2**	**3**	**4**
Методы созданных классов			
public static int Count()	подсчет содержимого объектов класса	List<имя класса> <Имя списка>	вывод количества объектов каждого класса
public virtual List<string> Return()	возврат хранимых данных в классе	нет	список данных хранимых в классе

2.3.4. Описание основных алгоритмов программы

Алгоритм представленный на рисунке 2.9. описывает механизм поиска в списке по вводимому имени.

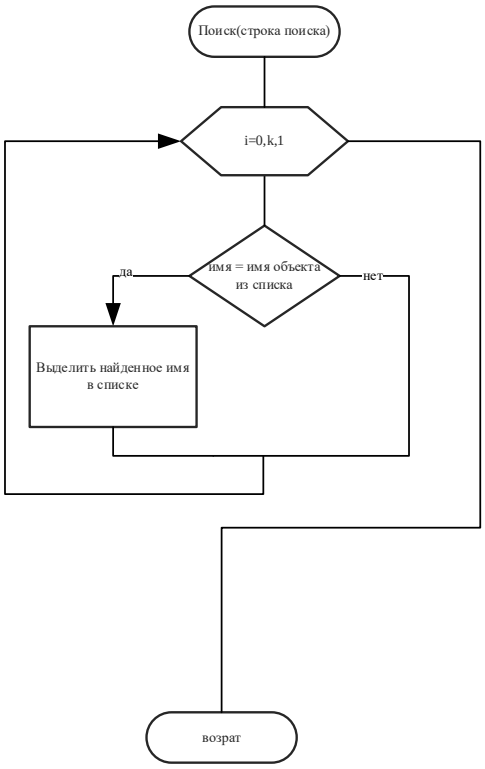

Рис. 2. 10.Блок-схема алгоритма поиск в базе данных по параметру

2.4.Описание структур, типов данных и глобальных переменных

В программе использовались типы данных – int, string, List. В программе используются глобальные переменные: ErrorMess, List<Medication> MedList, List<AnatomGroup> AnatomGroups

, которая определяет имя выходного файла.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе было реализовано настольное приложения, основной целью которого является вывод на экране иерархически организованного списка объектов из предметной области «Лекарственные средства». Реализованное приложение обладает возможностями по добавлению информации, ее удалению. Данная программа может быть использована в процессе исследования предметной области.

В процессе выполнения курсовой работы приобретены и закреплены навыки работы с различными визуальными компонентами приложений C# Windows Forms, закреплены навыки в разработке алгоритмов и в составлении

программ для решения поставленных задач, построения блок-схем. Получены навыки объектно-ориентированного программирования, работы с файлами и другими структурами на языке C#.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ATC/DDD Index 2021 [Электронный ресурс] URL: https://www.whocc.no/atc_ddd_index/

2. Анатомо-терапевтическо-химическая классификация ATX [Электронный ресурс] URL: http://med.rnx.ru/db/db-atx/

ПРИЛОЖЕНИЯ

Приложение А

Листинг 1.1. Разработанные классы

```
    public abstract class Medication : IComparable<Medication>
    {
        [XmlElement("MedicName")]
        public string MedicName { get; set; } //название группы верхнего уровня

        public string Description { get; set; }//текущее название препарата

        public string Code; //код, который состоит из номеров групп верхеного уровня

        List<string> Chapters = new List<string>(); //для объектов или содержимого класса

        /*Поля для формирования дерева*/
        [XmlElement("Level")]
        public int Level;
        [XmlElement("Index")]
        public int Index;
        public int Child;


        /*Если уровень 0, то или A или B и т.д.*/
        public Medication()
        {
            this.MedicName = "V";
            this.Description = " ";
            this.Level = 0;
            this.Index = 0;
        }
        public Medication(string MedicName, string Code, string Description, int Level,
int Index)
        {
            this.MedicName = MedicName;
            this.Code = Code;
            this.Description = Description;
            this.Level = Level;
            this.Index = Index;
        }
        public int CompareTo(Medication ComparedGroup)
        {
            string name1 = this.MedicName;
            string name2 = ComparedGroup.MedicName;
            int n = 0;
            if (name1.Length > name2.Length)
                n = name1.Length;
            else
                n = name2.Length;
            for (int i = 0; i < n; i++)
            {
                if (name1[i] > name2[i])
                    return 1;
                else if (name1[i] < name2[i])
                    return -1;
            }
            if (name1.Length < name2.Length)
                return 1;
            else if (name1.Length > name2.Length)
                return -1;
            else
```

```
                return 0;
        }

        public virtual List<string> Return()
        {
            return Chapters;
        }
        public virtual List<string> Return(string str)
        {
            return Chapters;
        }

    } public class AnatomGroup: Medication
    {
        private string AnatomMainGroup; //GOURPS: A,B,C,D,G,H,J,L,M,N,P,R,S,V


        public AnatomGroup()
        {
            AnatomMainGroup = " ";
            MedicName = "";
            Level = 0;
            Index = 0;
        }

        public AnatomGroup(string MedicName, string Code, string Description, int Level,
int Index)
        {
            this.MedicName = MedicName;
            this.Code = Code;
            this.Description = Description;
            this.Level = Level;
            this.Index = Index;
        }
        public static  int Count(List<Medication> MedicGroups)
        {
            int n = 0;
            foreach (var obj in MedicGroups)
            {
                if (obj is TherapGroup || obj is PharmaGroup || obj is ChemGroup)
                    continue;
                if (obj is AnatomGroup)
                    n++;
            }
            return n;
        }
        public static int CountAnatom(List<AnatomGroup> SubGroups)
        {
            int n = 0;
            foreach (var obj in SubGroups)
            {
                if (obj is AnatomGroup)
                    n++;
            }
            return n;
        }
}
public class TherapGroup: AnatomGroup
   {
        private string AnatomMainGroup { get; set; }
        private string TherapSG { get; set; }

        public TherapGroup()
        {
```

```
            TherapSG = "";
            MedicName = "";
            Code = " ";
            Level = 0;
            Index = 0;
        }
        public TherapGroup(string MedicName, string Code, string Description, int Level,
int Index)
        {
            this.MedicName = MedicName;
            this.Code = Code;
            this.Description = Description;
            this.Level = Level;
            this.Index = Index;
        }

        public static new int Count(List<Medication> MedicGroups)
        {
            int n = 0;
            foreach (var obj in MedicGroups)
            {
                if (obj is PharmaGroup || obj is ChemGroup)
                    continue;
                    if (obj is TherapGroup)
                    n++;
            }
            return n;
        }
    }
public class PharmaGroup: TherapGroup
    {
        private string AnatomMainGroup { get; set; }
        private string PharmaSG { get; set; }
        public PharmaGroup()
        {
            PharmaSG = "";
            MedicName = "";
            Code = "";
            Level = 0;
            Index = 0;
        }
        public PharmaGroup(string MedicName, string Code, string Description, int Level,
int Index)
        {
            this.MedicName = MedicName;
            this.Code = Code;
            this.Description = Description;
            this.Level = Level;
            this.Index = Index;
        }

        public static new int Count(List<Medication> MedicGroups)
        {
            int n = 0;
            foreach (var obj in MedicGroups)
            {
                    if (obj is PharmaGroup)
                    n++;
            }
            return n;
        }
    }
```

```
   public class ChemGroup: PharmaGroup
    {
        private string AnatomMainGroup { get; set; }
        private string TherapSG { get; set; }
        private string PharmaSG { get; set; }
        private string ChemSG { get; set; }

        public ChemGroup()
        {
            PharmaSG = "";
            MedicName = "";
            Code = "";
            Level = 0;
            Index = 0;
        }
        public ChemGroup(string MedicName, string Code, string Description, int Level,
int Index)
        {
            this.MedicName = MedicName;
            this.Code = Code;
            this.Description = Description;
            this.Level = Level;
            this.Index = Index;
        }

        public static new int Count(List<Medication> MedicGroups)
        {
            int n = 0;
            foreach (var obj in MedicGroups)
            {
                /*без этого считает все подряд. почему?*/
                if (obj is TherapGroup || obj is AnatomGroup || obj is PharmaGroup)
                    continue;
                    if (obj is ChemGroup)
                    n++;
            }
            return n;
        }
    }
class AlimentaryClass : AnatomGroup
    {
        /*Здесь перечислены категории препаратов входящих в анатомическую группу*/
        public static List<string> Chapters = new List<string>()
            {
                "Препараты, влияющие на пищеварительный тракт и обмен веществ",
                "Препараты, влияющие на кроветворение и кровь",
                "Препараты для лечения заболеваний сердечно-сосудистой системы",
                "Препараты для лечения заболеваний кожи",
                "Препараты для лечения заболеваний урогенитальных органов и половые
гормоны",
                "Гормональные препараты для системного использования (исключая половые
гормоны)",
                            "Противомикробные препараты для системного использования",
                            "Противоопухолевые препараты и иммуномодуляторы",
                            "Препараты для лечения заболеваний костно-мышечной системы",
                            "Препараты для лечения заболеваний нервной системы",
                            "Противопаразитарные препараты, инсектициды и репелленты",
                            "Препараты для лечения заболеваний респираторной системы",
                            "Препараты для лечения заболеваний органов чувств",
                            "Прочие лекарственные препараты"
            };
        public AlimentaryClass ()
        {
         /*Просто инициализатор класса*/
```

```
        }
        /*Вернем хранящийся список в классе */
        public override List<string> Return()
        {
           return Chapters;
        }
    }


public class PharmaPreparat : PharmaGroup
    {
        /*Стоматологические препараты*/
        public static List<string> StomatPreparat = new List<string>()
        {
            "Препараты для профилактики кариеса",
            "Противомикробные препараты для местного лечения заболеваний",
            "Глюкокортикостероиды для местного лечения заболеваний полости рта",
            "Прочие препараты для лечения заболеваний полости рта",
        };
        /*Противорвотные препараты*/
        public static List<string> AntiemeticPreparat = new List<string>()
        {
            "Блокаторы серотониновых 5-НТ3-рецепторов",
            "Другие противорвотные препараты"

        };
        /*Антикоагулянты*/
        public static List<string> AntiCoagulasPreparat = new List<string>()
        {
            "Антагонисты витамина К",
            "Гепарин и его производные",
            "Ингибиторы агрегации тромбоцитов (исключая гепарин)",
            "Ферментные препараты",
            "Прямые ингибиторы тромбина",
            "Прямые ингибиторы фактора Ха",
            "Прочие антикоагулянты"
        };
        public PharmaPreparat()
        {
            /*Просто инициализатор класса*/
        }
        /*Вернем хранящийся список в классе */
        public override List<string> Return(string Preparat)
        {
            switch (Preparat)
            {
                case "Стоматологические препараты":
                    return StomatPreparat;
                case "Противорвотные препараты ":
                    return AntiemeticPreparat;
                case "Антикоагулянты":
                    return AntiCoagulasPreparat;
                default: return null;
            }
        }
    }
public class PreparationsClass:TherapGroup
    {
        /*Здесь перечислены группы препаратов входящих в терапевтическую группу*/
        /* "Препараты, влияющие на пищеварительный тракт и обмен веществ"*/
        public static List<string> AlimentaryPreparat = new List<string>()
        {
            "Стоматологические препараты",
            "Препараты для лечения заболеваний, связанных с нарушением кислотности",
```

```
            "Препараты для лечения функциональных расстройств ЖКТ",
            "Противорвотные препараты",
            "Препараты для лечения заболеваний печени и желчевыводящих путей",
            "Слабительные препараты",
            "Противодиарейные, кишечные противовоспалительные и противомикробные
препараты",
            "Препараты для лечения ожирения (исключая диетические продукты)",
            "Препараты, способствующие пищеварению (включая ферменты)",
            "Препараты для лечения сахарного диабета",
            "Витамины",
            "Минеральные добавки",
            "Общетонизирующие препараты",
            "Анаболические средства для системного применения",
            "Стимуляторы аппетита",
            "Gпрепараты для лечения заболеваний ЖКТ и нарушений обмена веществ"
        };
        /*Препараты, влияющие на кроветворение и кровь*/
        public static List<string> BloodPrepat = new List<string>()
        {
            "Антикоагулянты",
            "Гемостатические препараты",
            "Антианемические препараты",
            "Плазмозамещающие и перфузионные растворы",
            "Прочие гематологические препараты"
        };
        /*Препараты для лечения заболеваний сердечно-сосудистой системы*/
        public static List<string> HeartPrepat = new List<string>()
        {
            "Препараты для лечения заболеваний сердца",
            "Антигипертензивные препараты",
            "Диуретики",
            "Периферические вазодилататоры",
            "Ангиопротекторы",
            "Бета-адреноблокаторы",
            "Блокаторы кальциевых каналов",
            "Препараты, влияющие на ренин-ангиотензиновую систему",
            "Гиполипидемические препараты"
        };
        public PreparationsClass()
        {
            /*Просто инициализатор класса*/
        }
        /*Вернем хранящийся список в классе */
        public override List<string> Return(string Preparat)
        {
            switch (Preparat)
            {   case "Стоматологические препараты":
                    return StomatPreparat;
                case "Противорвотные препараты ":
                    return AntiemeticPreparat;
                case "Антикоагулянты":
                    return AntiCoagulasPreparat;
                default: return null;
            }
        }
}
    public class ChemClass:ChemGroup
    {
        /*Препараты для профилактики кариеса*/
        public static List<string> CariesPreparat = new List<string>()
        {
            "Фторид натрия",
            "Натрия монофторфосфат",
            "Олафур",
            "Фторид олова",
```

```
            "Комбинации",
            "Комбинации фторида натрия"
        };
        /*Противомикробные препараты для местного лечения заболеваний*/
        public static List<string> AntiInfectivesPreparat = new List<string>()
        {
            "Пероксид водорода",
            "Хлоргексидин",
            "Амфотерицин",
            "Полиноксилин",
            "Домифена бромид",
            "Оксихинолин",
            "Неомицин",
            "Миконазол"
        };
        public override List<string> Return(string Preparat)
        {
            switch (Preparat)
            {
                case "Препараты для профилактики кариеса":
                    return CariesPreparat;
                case "Противомикробные препараты для местного лечения заболеваний":
                    return AntiInfectivesPreparat;
                default: return null;
            }
        }
    }
```

Листинг 1.2. Главная форма приложения

```
public partial class MainWindows : Form
{
public MainWindows()
        {
            InitializeComponent();
            StatusStrip1Initizlization();
            StatusStrip2Initizlization();
            Animator.Start();
        }
  //Error Message
        public string ErrorMess = "Нельзя изменять главный уровень!";
        public string ErrorMess2 = "Нельзя удалить уровень у которого есть подуровни!";
        public string ErrorMess3 = "Нельзя изменить не открытый узел";
        //Списки
        public static List<Medication> MedList = new List<Medication>();
        public static List<AnatomGroup> AnatomGroups = new List<AnatomGroup>();
        public static List<TherapGroup> TherapGroups = new List<TherapGroup>();
        public static List<PharmaGroup> PharmaGroups = new List<PharmaGroup>();
        public static List<ChemGroup> ChemGroups  = new List<ChemGroup>();

        //Переменные
        ToolStripLabel timeStartLabel, dateLabel, timeLabel;
        private static ulong timeSec = 0;
        public static string searchline;//this string may be init after entering text in
searchForm
        ToolStripMenuItem UpdateMenuItem = new ToolStripMenuItem("Обновить");
private void MainWindows_Load_1(object sender, EventArgs e)
        {
            /*Тестовые данные для списка*/
            AnatomGroup anatom1 = new AnatomGroup("Препараты, влияющие на пищеварительный
тракт и обмен веществ","A",
                                                  "Раздел системы буквенно-цифровых кодов
Анатомо-терапевтическо-химической классификации," +
```

```
                                                "разработанных Всемирной организацией
здравоохранения для классификации лекарств и других " +
                                                "медицинских продуктов", 0, 0);
            TherapGroup therap1 = new TherapGroup("Стоматологические препараты", "A01",
"Подгруппа A01 является частью группы препаратов A " +
                                                "«Препараты, влияющие на пищеварительный
тракт и обмен веществ» ", 1, 1);
            PharmaGroup pharma1 = new PharmaGroup("Препараты для профилактики кариеса",
"A01A",
                                                 "\n\tA01AA01 Фторид натрия, " +
                                                 "\n\tA01AA02 Натрия монофторфосфат," +
",2, 2);

            AnatomGroup anatom2 = new AnatomGroup("Препараты, влияющие на кроветворение и
кровь","B"," ",0, 3);
            TherapGroup therap2 = new TherapGroup("Антикоагулянты", "B01"," ", 1, 4);
            MedList.Clear();
            MedList.Add(anatom1);
            MedList.Add(therap1);
            MedList.Add(pharma1);
//внесение данных в списки
            AnatomGroups.Add(anatom1);
            AnatomGroups.Add(anatom2);
            TherapGroups.Add(therap1);
            TherapGroups.Add(therap2);
            PharmaGroups.Add(pharma1);
            contextMenuStrip1.Items.AddRange(new[] { UpdateMenuItem});
            treeView1.ContextMenuStrip = contextMenuStrip1;
            UpdateMenuItem.Click += UpdateMenuItem_Click;
            ParentNodesMed();
        }
       #region Статусы

        // создаем элементы меню

        // добавляем элементы в меню
        //
        // ассоциируем контекстное меню с текстовым полем
        void Timer_Tick(object sender, EventArgs e)
        {
            dateLabel.Text = DateTime.Now.ToLongDateString();
            timeLabel.Text = DateTime.Now.ToLongTimeString();
        }
        void Timer_Tick2(object sender, EventArgs e)
        {
            timeSec += 1;
            timeStartLabel.Text = timeSec.ToString();
        }
        private void StatusStrip1Initizlization()
        {
            ToolStripLabel infoLabel = new ToolStripLabel()
            {
                Text = "Текущие дата и время:"
            };
            dateLabel = new ToolStripLabel();
            timeLabel = new ToolStripLabel();
            statusStrip1.Items.Add(infoLabel);
            statusStrip1.Items.Add(dateLabel);
            statusStrip1.Items.Add(timeLabel);
            Timer timer = new Timer() { Interval = 1000 };
            timer.Tick += Timer_Tick;
            Timer timer2 = new Timer() { Interval = 1000 };
            timer2.Tick += Timer_Tick2;
            timer.Start();
```

```
            timer2.Start();
        }
        private void StatusStrip2Initizlization()
        {
            ToolStripLabel toolStripLabel = new ToolStripLabel();
            ToolStripLabel infoLabel = toolStripLabel;
            infoLabel.Text = "Время работы программы в секундах:";
            timeStartLabel = new ToolStripLabel();
            statusStrip2.Items.Add(infoLabel);
            statusStrip2.Items.Add(timeStartLabel);
            Timer timer = new Timer() { Interval = 1000 };
            timer.Tick += Timer_Tick;
            timer.Start();
        }
        #endregion
      #region TREENODE
        /*--------------Динамическое добавление данных в TreeView-------------------*/
       public void ParentNodesMed()
        {
            int i;

            treeView1.Nodes.Clear();
            treeView1.BeginUpdate();
            for (i = 0; i < MedList.Count(); i++)
            {
                if (MedList[i].Level == 0)
                {
                    treeView1.Nodes.Add(MedList[i].MedicName, MedList[i].MedicName);
                    treeView1.Nodes[treeView1.Nodes.Count - 1].Tag = MedList[i];
                }
            }
            for (i = 0; i < treeView1.Nodes.Count; i++)
            {
                ChildNodesMed(treeView1.Nodes[i]);
                MedList[i].Child++;
            }
            treeView1.EndUpdate();
            treeView1.Refresh();
        }
        private void ChildNodesMed(TreeNode treeNode)
        {
            Medication parentRed = (Medication)treeNode.Tag;
            for (int i = parentRed.Index + 1; i < MedList.Count; i++)
            {
                if (MedList[i].Level == (parentRed.Level + 1))
                {
                    treeNode.Nodes.Add(MedList[i].MedicName, MedList[i].MedicName);
                    treeNode.Nodes[treeNode.Nodes.Count - 1].Tag = MedList[i];
                    ChildNodesMed(treeNode.Nodes[treeNode.Nodes.Count - 1]);
                }
                if (MedList[i].Level <= treeNode.Level) break;
            }
        }
  /*--------------------Отображение содержимого класса-------------------------------*/
        private void TreeView1_NodeMouseDoubleClick(object sender,
TreeNodeMouseClickEventArgs e)
        {
            try
            {

                TreeNode node = treeView1.SelectedNode;// Получение выбранного двойным
щелчком узла дерева.
```

```
                MessageBox.Show(string.Format("You selected: {0}", node.Text)); // Вывод
окна с текстом данного узла.

                DataDescriptionGrid.Rows.Clear();
                DataDescriptionGrid.Columns.Clear();
                DataGridInit();
                DataDescriptionGrid.ReadOnly = true;
                DataDescriptionGrid.Visible = true;

                /*----------Кнопки---------------*/
                AddItem.Visible = true;
                EditItem.Visible = true;//для изменения данных в содержимом узла
                DelItem.Visible = true;
                MinimizeItem.Visible = true;//для закрытия информации об узле
                /*---------Кнопки--------------*/

                var index = MedList.FindIndex(x => x.MedicName.Contains(node.Text));
                DataDescriptionGrid.Rows[0].Cells[1].Value = MedList[index].Code;//код
группы
                DataDescriptionGrid.Rows[1].Cells[1].Value =
MedList[index].MedicName;//название группы/препарата если на его уровне
                DataDescriptionGrid.Rows[2].Cells[1].Value = MedList[index].Description;

                DataDescriptionGrid.Columns[1].DefaultCellStyle.WrapMode =
DataGridViewTriState.True;
                DataDescriptionGrid.Columns[1].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill; //автоматическое выравнивание текста в колонке
                DataDescriptionGrid.AutoSizeRowsMode =
DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка при щелчке по узлу!\nДополнительные
сведения:\n{ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    /*-------------------------------------------------------------------------------*/

    /*---------функция поиска в дереве. Если ничего не найдено - возвращает null------*/
       public static TreeNode SearchNode(string SearchText, TreeNode StartNode)
       {
           TreeNode node=null;
           while (StartNode!= null && SearchText != null)
           {
               if (StartNode.Text.ToLower().Contains(SearchText.ToLower()))
               {
                   node = StartNode; //node was found
                   break;
               };
               if (StartNode.Nodes.Count != 0) //if node has child
               {
                   node = SearchNode(SearchText, StartNode.Nodes[0]);//ищем рекурсивно в
дочерних
                   if (node != null)
                   {
                       break;
                   };
               };
               StartNode = StartNode.NextNode;
           };
           return node;//вернули результат поиска
       }

       private void StartSearh()
       {
```

```
            Forms.SearchLine searchForm = new Forms.SearchLine();
            searchForm.ShowDialog();

            TreeNode SelectedNode = SearchNode(searchline, treeView1.Nodes[0]);//пытаемся
найти в поле Text
            if (SelectedNode != null )
            {
                //нашли, выделяем...
                this.treeView1.SelectedNode = SelectedNode;
                this.treeView1.SelectedNode.Expand();
                this.treeView1.Select();
            };
        }

        private void ПоискToolStripMenuItem_Click(object sender, EventArgs e)//пункт меню
        {
            StartSearh();
        }
        /*-------------------------------------------------------------------------------*/

        #endregion

       #region DataGrids
        private void DataGridInit()
        {

            DataDescriptionGrid.AutoSizeRowsMode =
DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
            DataGridViewCellStyle columnstyle = new DataGridViewCellStyle()
            {
                BackColor = Color.Chocolate,
                Font = new Font("Arial", 11, FontStyle.Regular),
            };

            DataDescriptionGrid.Columns.Add("GroupNameColumn", "");
            DataDescriptionGrid.Columns.Add("DescripColumn", "");
            DataDescriptionGrid.Rows.Add(5);
            DataDescriptionGrid.Rows[0].Cells[0].Value = "Код: ";
            DataDescriptionGrid.Rows[1].Cells[0].Value = "Группа: ";
            DataDescriptionGrid.Rows[2].Cells[0].Value = "Описание: ";

            DataDescriptionGrid.AllowUserToAddRows = false;
            DataDescriptionGrid.Columns[0].ReadOnly = true;
        }
        private void DataDescriptionGrid_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.LButton && DataDescriptionGrid.CurrentCell.ColumnIndex
== 1)
            {
                e.Handled = true;
                DataGridViewCell cell = DataDescriptionGrid.Rows[0].Cells[1];
                DataDescriptionGrid.CurrentCell = cell;
                DataDescriptionGrid.BeginEdit(true);
                DataDescriptionGrid.EditMode = DataGridViewEditMode.EditOnEnter;
            }

        }

        private void DataDescriptionGrid_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            var index = MedList.FindIndex(x =>
x.MedicName.Contains(DataDescriptionGrid.Rows[0].Cells[1].Value.ToString()));
            if (e.ColumnIndex == 1)
            {
```

```
                DataDescriptionGrid.BeginEdit(true);
                DataDescriptionGrid.ReadOnly = false;
            }
            else
            {
                DataDescriptionGrid.ReadOnly = true;
            }
        }

        #endregion

        #region Кнопки

        private void AddButton_Click_1(object sender, EventArgs e)
        {
            Forms.AddForm NewForm = new Forms.AddForm()
            {
                MedList = MedList,
            };
            NewForm.Show();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
           try
           {
                if (DataDescriptionGrid.Rows[1].Cells[1].Value.ToString() == " ")
                {

                    MessageBox.Show(ErrorMess);
                }
                else
                {
                    var index = MedList.FindIndex(x =>
x.Code.Equals(DataDescriptionGrid.Rows[0].Cells[1].Value));
                    MedList[index].Description =
DataDescriptionGrid.Rows[2].Cells[1].Value.ToString();
                    ParentNodesMed();
                }
            }
             catch (Exception ex)
            {
                MessageBox.Show($"{ErrorMess3}!\nДополнительные сведения:\n{ex.Message}",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
/*--------------------Закрыть открытый для редактирования узел--------------------*/
        private void MinimButton_Click(object sender, EventArgs e)
        {
            MinimizeItem.Visible = false;
            EditItem.Visible = false;
            AddItem.Visible = false;
            DelItem.Visible = false;
            DataDescriptionGrid.Rows.Clear();
        }
/*-----------------------------Удаление узла-------------------------------------*/
        private void DelButton_Click(object sender, EventArgs e)
        {
            try
                {
                var index = MedList.FindIndex(x =>
x.Index.Equals(DataDescriptionGrid.Rows[1].Cells[1].Value));
                if (MedList[index].Child > 0)
                    {
                        MessageBox.Show("Нельзя удалять корни!");
```

```
                }
                else
                {
                    MedList.RemoveAt(index);
                }
                ParentNodesMed();
            }
            catch
            {
                MessageBox.Show(ErrorMess2);
            }
        }

        #endregion
        /*Обработка событий клавиатуры*/
        /*KeyDown events*/
        private void MainWindows_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.S)
                SaveFile();
            if (e.Control && e.KeyCode == Keys.F)
                StartSearh();
        }
        private void tabControl1_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Control && e.KeyCode == Keys.S)
                SaveFile();
            if (e.Control && e.KeyCode == Keys.F)
                StartSearh();
            if (e.KeyCode == Keys.F1)
                ОПрограммеToolStripMenuItem_Click(sender, e);
        }

        /*Сохранение в виде XML файла*/
        private void SaveFile()
        {
            try
            {
                List<Type> types = new List<Type>();
                foreach(Medication med in MedList)
                {
                    Type type = med.GetType();
                    if(!types.Contains(type))
                    {
                        types.Add(type);
                    }
                }
                XmlSerializer serializer = new
XmlSerializer(typeof(List<Medication>),types.ToArray());
                SaveFileDialog saveFileDialog1 = new SaveFileDialog();
                saveFileDialog1.DefaultExt = "*.xml";
                saveFileDialog1.Filter = "XML files|*.xml";
                if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK
&& saveFileDialog1.FileName.Length > 0)
                {
                    using (FileStream file = new FileStream(saveFileDialog1.FileName,
FileMode.Create))
                    {
                      file.SetLength(0);
                      file.Flush(); //очищает все буферы данного потока и вызывает запись
данных буферов в базовое устройство.
                      serializer.Serialize(file, MedList);
                    }
```

```
                    MessageBox.Show($"Ваш файл был сохранен!", "Успех",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Что-то пошло не так!\nДополнительные
сведения:\n{ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        private void СохранитьКакToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFile();
        }

        private void ОбновитьToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ParentNodesMed();
        }

        private void ОПрограммеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Программа создана в рамках выполнения курсового
проекта\n\nИсходный текст программы в актуальном виде доступен на GitHub");
        }

        private void ПоказатьКнопкиToolStripMenuItem_MouseDown(object sender,
MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                AddItem.Visible = true;
                EditItem.Visible = true;
                DelItem.Visible = true;
                MinimizeItem.Visible = true;
            }
            else if (e.Button == MouseButtons.Right)
            {
                AddItem.Visible = false;
                EditItem.Visible = false;
                DelItem.Visible = false;
                MinimizeItem.Visible = false;
            }
        }

        private void СтатистикаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            int AnatGroup, ThGroup, PhGroup, ChGroup;
            AnatGroup = AnatomGroup.Count(MedList);
            ThGroup = TherapGroup.Count(MedList);
            PhGroup = PharmaGroup.Count(MedList);
            ChGroup = ChemGroup.Count(MedList);

            MessageBox.Show(
                $"Количество Анатомических групп\n:{AnatGroup}\n" +
                 $"Количество Терапевтических групп\n:{ThGroup}\n" +
                  $"Количество Фармакологических групп\n:{PhGroup}\n" +
                   $"Количество Химических групп\n:{ChGroup}\n",

                "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }


        private void добавитьЭлементToolStripMenuItem_Click(object sender, EventArgs e)
```

```
        {
            AddButton_Click_1(sender, e);
        }

        private void treeView1_DragEnter(object sender, DragEventArgs e)
        {
            if (e.Data.GetDataPresent(DataFormats.FileDrop) && ((e.AllowedEffect &
DragDropEffects.Move) == DragDropEffects.Move))
            {
                e.Effect = DragDropEffects.Move;
            }
        }

        private void treeView1_DragDrop(object sender, DragEventArgs e)
        {
            MessageBox.Show("Файл загружен");
            if (e.Data.GetDataPresent(DataFormats.FileDrop) && e.Effect ==
DragDropEffects.Move)
              {
                string[] obj = (string[])e.Data.GetData(DataFormats.FileDrop); //выясним
сколько файлов
                if (obj.Length >1)
                {
                    MessageBox.Show($"Слишком мгого файлов\n", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                else
                {
                    if(Path.GetExtension(obj[0]) != ".xml")
                    {
                        MessageBox.Show($"Не то расширение файла\nНадо .xml", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                    else
                    {
                        MedList.Clear();
                        XmlDocument doc = new XmlDocument();
                        doc.Load(obj[0]);
                        foreach (XmlNode node in doc.DocumentElement)
                        {
                            string AnatomicalMainGroup = node.Attributes[0].Value;
                            string MedicName = node["MedicName"].InnerText;
                            string Code = node["Code"].InnerText;
                            string Description = " ";
                            int level = int.Parse(node["Level"].InnerText);
                            int index = int.Parse(node["Index"].InnerText);
                            MedList.Add(new AnatomGroup(MedicName, Code, Description,
level, index));
                        }
                        ParentNodesMed();
                    }

                }

            }

        }


        //End work with current Form
        //завершение работы с текущей формой
        private void ВыходToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            Close();
        }
```

```
        private void XMLToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var filePath = string.Empty;
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.DefaultExt = "*.xml";
            openFileDialog.Filter = "XML files|*.xml";
            if (openFileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                filePath = openFileDialog.FileName;
            }
                MedList.Clear();
            XmlDocument doc = new XmlDocument();
            doc.Load(filePath);
            foreach (XmlNode node in doc.DocumentElement)
            {
                string AnatomicalMainGroup = node.Attributes[0].Value;
                string MedicName = node["MedicName"].InnerText;
                string Code = node["Code"].InnerText;
                string Description = " ";
                int level = int.Parse(node["Level"].InnerText);
                int index = int.Parse(node["Index"].InnerText);
                MedList.Add(new AnatomGroup(MedicName, Code, Description, level, index));
            }
            ParentNodesMed();
        }
}
```

Листинг 1.4. Форма добавления данных

```
public partial class AddForm : Form
    {
        string CodeSG;
        List<string> Code = new List<string>();
        public List<Medication>  MedList = new List<Medication>();
        public AddForm()
        {
            InitializeComponent();
            //dataGridView1.Rows[0].Cells[0].Value = LKLIST[0].ShowText;
        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            try
            {
                if (AnatomComboBox.ToString()!= null && TherapComboBox.ToString()!=null
&& PharmaComboBox.ToString()!=null && ChemComboBox.ToString()!=null)
                {

                }

            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка на форме 'Добавить'!\nДополнительные
сведения:\n{ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            if(AnatomComboBox.ToString()!=null)
            {
                CodeTextBox.Text = "A";
            }
            int index = MedList.Count();//пока для последнего элеммента в списке так
```

```
            AnatomGroup anatom = new
AnatomGroup(AnatomComboBox.Text,CodeTextBox.Text,DescriptionBox.Text,0,index);
            MedList.Add(anatom);
            MainWindows.MedList = MedList;
            this.Close();
        }

        /*---------Списки групп--------------*/
        List<string> ANMG = new List<string>();
        List<string> THSG = new List<string>();
        List<string> PHSG = new List<string>();
        List<string> CHSG = new List<string>();

        /*------------Коды групп--------------*/
        List<string> Code_ANMG = new List<string>();
        List<string> Code_THSG = new List<string>();
        List<string> Code_PHSG = new List<string>();
        List<string> Code_CHSG = new List<string>();

        private void AddForm_Load(object sender, EventArgs e)
        {
            AlimentaryClass alimentary = new AlimentaryClass();
            ANMG = alimentary.Return();
            foreach (var str in ANMG)
            {
                AnatomComboBox.Items.Add(str);
            }
            Code_ANMG.Clear();
            Code_ANMG.AddRange(new string[] { "A", "B", "C", "D", "G" });
        }
       private void AnatomComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            PreparationsClass preparations = new PreparationsClass();

            switch ((string)AnatomComboBox.SelectedItem)
            {
                case "Препараты, влияющие на пищеварительный тракт и обмен веществ":
                    /*Был выбран пункт. Сформируем список для следующего поля*/
                    THSG.Clear();
                    TherapComboBox.Items.Clear();
                    THSG = preparations.Return("Препараты, влияющие на пищеварительный
тракт и обмен веществ");
                    Code_THSG.Clear();
                    Code_THSG.AddRange(new string[] { "A01", "A02", "A03", "A04", "A05"
});
                        foreach (var str in THSG)
                        {
                            TherapComboBox.Items.Add(str);
                        }
                    break;
                case "Препараты, влияющие на кроветворение и кровь":
                    THSG.Clear();
                    TherapComboBox.Items.Clear();
                    THSG = preparations.Return("Препараты, влияющие на кроветворение и
кровь");
                    foreach (var str in THSG)
                    {
                        TherapComboBox.Items.Add(str);
                    }

                    /*Сформируем коды для следуюшего списка*/
                    Code_THSG.Clear();
                    Code_THSG.AddRange(new string[] { "B01", "B02", "B03", "B05", "B06"
});
```

```
                    break;
                case "Препараты для лечения заболеваний сердечно-сосудистой системы":
                    /*Был выбран пункт. Сформируем список для следующего поля*/
                    THSG.Clear();
                    TherapComboBox.Items.Clear();
                    THSG = preparations.Return("Препараты для лечения заболеваний
сердечно-сосудистой системы");
                    Code_THSG.Clear();
                    Code_THSG.AddRange(new string[] { "C01", "C02", "C03", "C04", "C05"
});
                    foreach (var str in THSG)
                    {
                        TherapComboBox.Items.Add(str);
                    }
                    break;
                default:
                    TherapComboBox.Items.Clear();
                    break;
            }

            /*Динамически меняем отображаемый код анатомической группы*/
            for (int i = 0; i < ANMG.Count; i++)
            {
                if ((string)AnatomComboBox.SelectedItem == ANMG[i])
                {
                    CodeTextBox.Text = Code_ANMG[i];
                }
            }
        }
        private void TherapComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            PharmaPreparat pharma = new PharmaPreparat();
            switch ((string)TherapComboBox.SelectedItem)
            {
               case "Стоматологические препараты":
                    PharmaComboBox.Items.Clear();
                    PHSG = pharma.Return("Стоматологические препараты");
                    foreach (var str in PHSG)
                    {
                        PharmaComboBox.Items.Add(str);
                    }
                    Code_PHSG.Clear();
                    Code_PHSG.AddRange(new string[] { "A01AA", "A02AB", "A03AC",
"A04AD"});
                   break;
                case "Антикоагулянты":
                    PharmaComboBox.Items.Clear();
                    PHSG = pharma.Return("Антикоагулянты");
                    foreach (var str in PHSG)
                    {
                        PharmaComboBox.Items.Add(str);
                    }
                    Code_PHSG.Clear();
                    Code_PHSG.AddRange(new string[] { "B01AA", "B01AB", "B01AC", "B01AD",
"B01AE","B01AF","B01AX" });
                    break;
                default:
                    PharmaComboBox.Items.Clear();
                    break;
            }
            /*Динамически меняем отображаемый код анатомической группы*/
            for (int i = 0; i < THSG.Count; i++)
            {
                if ((string)TherapComboBox.SelectedItem == THSG[i])
                {
```

```
                    CodeTextBox.Text = Code_THSG[i];
                }
            }
        }


        private void PharmaComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            ChemClass chem = new ChemClass();
            switch ((string)PharmaComboBox.SelectedItem)
            {
                case "Препараты для профилактики кариеса":
                    ChemComboBox.Items.Clear();
                    CHSG = chem.Return("Препараты для профилактики кариеса");
                    foreach (var str in CHSG)
                    {
                        ChemComboBox.Items.Add(str);
                    }
                    Code_CHSG.Clear();
                    Code_CHSG.AddRange(new string[] { "A01AA01", "A01AA02", "A01AA03",
"A01AA04", "A01AA30", "A01AA51" });
                    break;
                case "Противомикробные препараты для местного лечения заболеваний":
                    ChemComboBox.Items.Clear();
                    CHSG = chem.Return("Противомикробные препараты для местного лечения
заболеваний");
                    foreach (var str in CHSG)
                    {
                        PharmaComboBox.Items.Add(str);
                    }
                    Code_CHSG.Clear();
                    Code_CHSG.AddRange(new string[] { "A01AB02", "A01AB03", "A01AB04",
"A01AB05", "A01AB06", "A01AB07", "A01AB08", "A01AB09" });
                    break;
                default:
                    PharmaComboBox.Items.Clear();
                    break;
            }
            /*Динамически меняем отображаемый код анатомической группы*/
            for (int i = 0; i < PHSG.Count; i++)
            {
                if ((string)PharmaComboBox.SelectedItem == PHSG[i])
                {
                    CodeTextBox.Text = Code_PHSG[i];
                }
            }
        }
        private void ChemComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            /*Динамически меняем отображаемый код анатомической группы*/
            for (int i = 0; i < CHSG.Count; i++)
            {
                if ((string)ChemComboBox.SelectedItem == CHSG[i])
                {
                    CodeTextBox.Text = Code_CHSG[i];
                }
            }
        }

    private void CancelButton_Click(object sender, EventArgs e)
        {
            DialogResult dialog = DialogResult.Cancel;
        }
     }
```

Листинг 1.5. Форма поиска

```
public partial class SearchLine : Form
    {
        public SearchLine()
        {
            InitializeComponent();
        }
       // private bool Button_click = false;

        public void OkButton_SearchLine_Click(object sender, EventArgs e)
        {
            //Button_click = true;
            string SearchText = this.TextBox_SearchLine.Text;

            if (SearchText == "")

            {
                SearLineToolTip.Show("Поле не может быть пустым!",TextBox_SearchLine,
10000);
                return;
            };
            //передаем введенную строку на глобальную форму
            //return search string in main form
            LekarList.MainWindows.searchline = SearchText;
            this.Close();

        }



    }
```

Приложение Б

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1.1.Требования к установке

Системные программные средства, используемые программой, должны быть представлены локализованной версией операционной системы Windows (Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10).

1.2. Запуск

Запуск приложения производится двойным щелчком мыши на файле .exe (см. рис. 1. 1). После запуска на экране главная входа (см. рис. 1. 2).

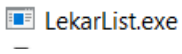

Рис. 1. 1. Исполняемый файл

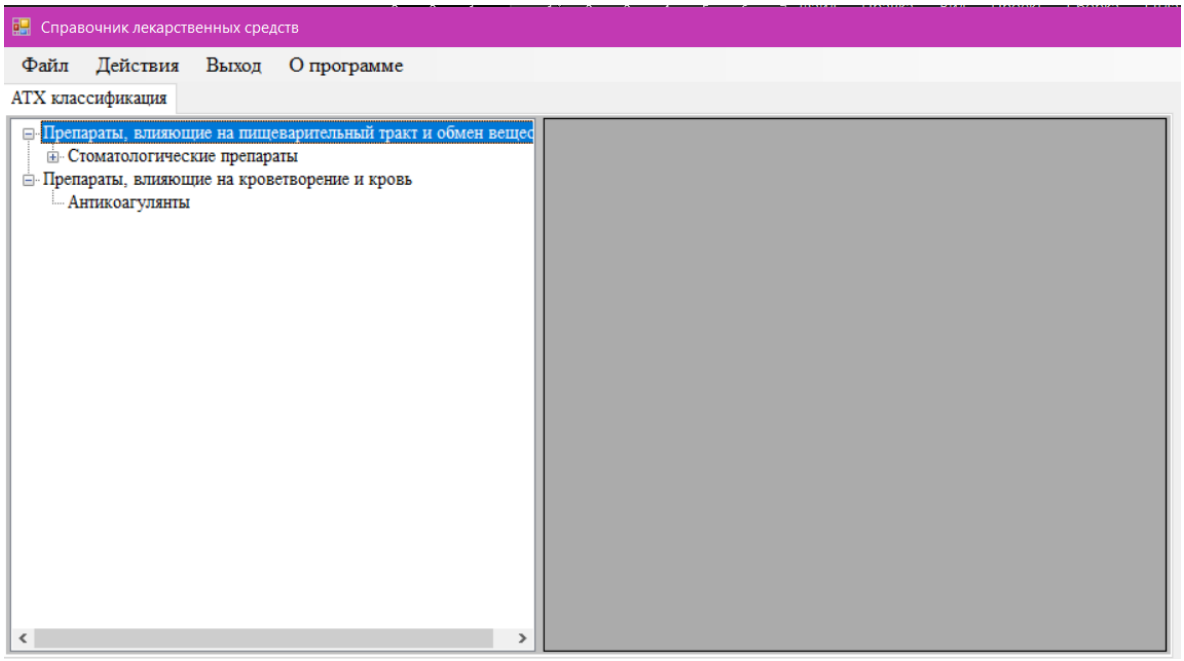

Рис. 1. 2. Окно входа

1.3.Основные команды

Для пользователя возможны следующие такие действия как (см. рис. 1.3.):

– Добавить элемент – после нажатия на эту кнопку открывается форма добавления данных;

– Поиск – появляется форма для поиска данных;

– Показать кнопки – после нажатия на эту кнопку на экране появляются кнопки для управления программой;

– Статистика – после нажатия на эту кнопку на экране появится форма, которая выводит информацию о количестве объектов каждого класса.

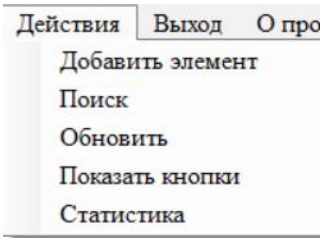

Рис. 1. 3. Действия пользователей в программе

1.4.Работа с формой «Добавление данных»

Для того, чтобы добавить из данной формы данные в таблицу необходимо нажать на кнопку «Добавить элемент». В полях подстановки можно выбрать нужное имя класса. Далее нужно нажать на кнопку «Добавить». После этого данные будут записаны в список.

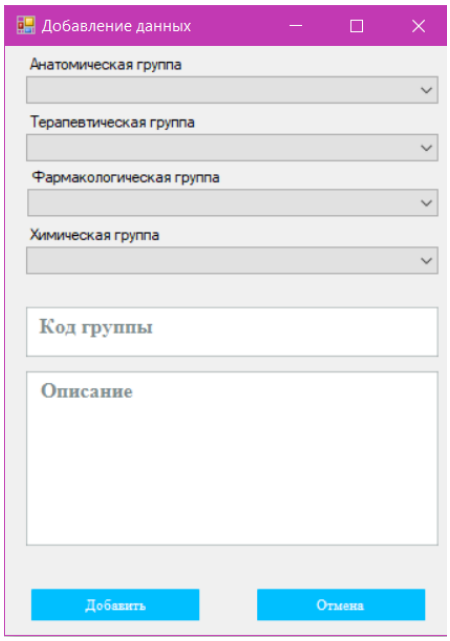

Рис. 1. 4. Общий вид формы

1.5.Работа с формой «Поиск»

Для того, чтобы осуществить поиск нужно ввести фразу, которую надо найти в поле для ввода и нажать на кнопку «Выполнить поиск».

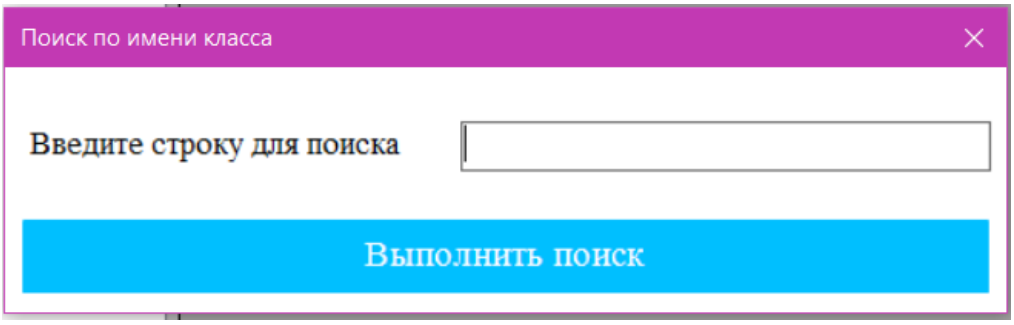

Рис. 1. 5. Общий вид формы

1.6. Сохранение и открытие файлов

Для того, чтобы выполнить загрузку данных из файла необходимо выбрать пункт меню «Открыть» на вкладке «Файл». Для выполнения сохранения файла необходимо выбрать пункт меню «Сохранить как» или выполнить нажатие указанных клавиш.

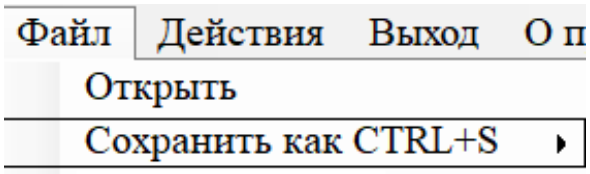

Рис. 1. 6. Сохранение загрузка файлов

Кроме того, форма поддерживает технологию Drag&Drop, поэтому выбранный для загрузки файл можно перетащить на форму в область с выведенным на экран списком.

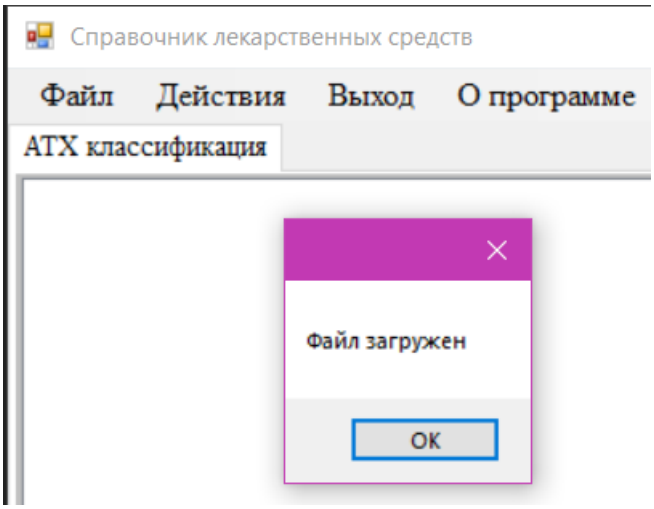

Рис. 1. 7. Сообщение информирующее об успешной загрузке

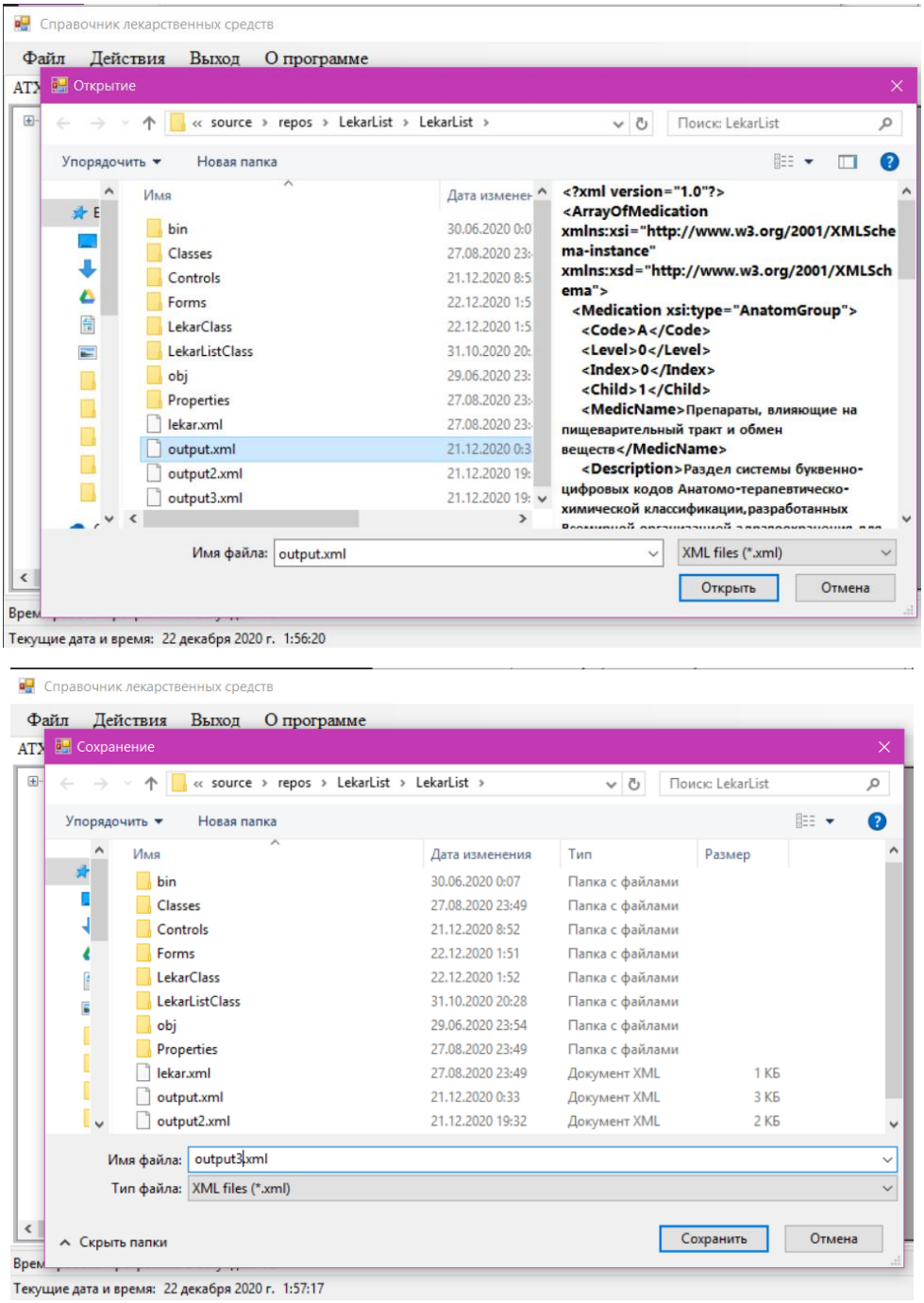

Рис. 1. 8. Диалоговые окна