Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА
ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни «Об’єктно-орієнтоване програмування»

ДОВІДНИК ГЕОГРАФА

Керівник, ас. каф ПІ, Ляпота В.М.

Студент гр. ПІ-15-2 Гребенник В.В.

Комісія:

проф. _____ Дудар З.В.

проф. _____ Бондарев В.М.

Харків 2016

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра: ***Програмної інженерії***

Дисципліна: Об'єктно-орієнтоване програмування

Спеціальність: ***Програмна інженерія***

Курс 1 . Група *ПІ-15-2* . Семестр 2 .

ЗАВДАННЯ

на курсовий проект студента

Гребенник Влади Володимирівни

(Прізвище, Ім'я, По батькові)

1. Тема проекту:

   Довідник географа

2. Термін здачі студентом закінченого проекту: ***"28" - травня - 2016 р.***

3. Вихідні дані до проекту:

   Специфікація програми, методичні вказівки до виконання курсової роботи

4. Зміст розрахунково-пояснювальної записки:

   Вступ, специфікація програми, проектна специфікація, інструкція , висновки.

5. Перелік графічного матеріалу

Екранні форми, схема об'єктної моделі, діаграми класів

КАЛЕНДАРНИЙ ПЛАН

№	*Назва етапу*	*Термін виконання*
1	Видача теми, узгодження і затвердження теми	1-03-2016 р.
2	Аналіз предметної області	1-03-2016 – 15-03-2016 р.
3	Розробка постановки задачі	15-03-2016 – 25-03-2016 р.
4	Розробка об’єктної моделі	25-03-2016 – 10-04-2016 р.
5	Кодування програмної системи	10-04-2016 – 5-05-2016 р.
6	Тестування і доопрацювання розробленої програмної системи.	5-05-2016 – 20-05-2016 р.
7	Оформлення пояснювальної записки	20-05-2016 – 27-05-2016 р.
8	Публічний захист проекту перед комісією	28-05-2016 р.

Студент _____

Керівник _____ Ляпота В.М.

(Прізвище, Ім'я, По батькові)

« 1 »_березня__ 2016 р.

РЕФЕРАТ / ABSTRACT

Пояснювальна записка: 27 с., 25 рис., 5 джерел, 1 додаток.

Мета роботи: закріплення знань, отриманих під час вивчення дисципліни «Об’єктно-орієнтовне програмування», шляхом розробки програмної системи під назвою «Довідник географа».

Методи розробки: Microsoft VisualStudio Enterprise 2015, Windows Forms, .NET Framework 4.6

В результаті розробки отримана програмна система під назвою «Довідник географа» для роботи з інформаційною базою даних, чию роль виконують *bin файли. Розроблена програма дає швидкий та зручний доступ до географічних одиниць, звідної інформації про кількість населення, площу та ін. параметрів міст, регіонів, материків та країн, які існують в базі. Також є можливість додавання нових одиниць; їх пошук у базі; додавання баз та ін.

КУРСОВА РОБОТА, ДОВІДНИК ГЕОГРАФА, БАЗА, ДОВІДНИК, ПРОГРАММА.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

«Без географии мы нигде», – сказал американский певец Джимми Баффетт. И действительно, возникновению этой науки мы обязаны тем, где мы есть. География – одна из древнейших наук. Люди всегда интересовались тем, что их окружает, пытались изучать близлежащие и отдаленные земли. Первые исследователи привозили из своих экспедиций не только карты и планы местности, но и описания рельефа, почвы, растительности. Они изучали и население других территорий, культуру и быт людей, их численность…

Казалось бы, Земля изучена уже вдоль и поперек. Однако предметы исследований географов очень изменчивы и подвержены множеству факторов. Поэтому данным специалистам вряд ли когда-нибудь придется бездельничать. А посему весьма актуальной является проблема систематизации и структуризации информации, с которой по долгу службы приходится работать географам.

В данной курсовой работе требуется разработать программу «Справочник географа». Целью данной курсовой работы является разработка программы-справочника, с помощью которого можно добавлять географические единицы в иерархической последовательности, наполнять информацией такие параметры, как население, площадь, форма правления, координаты и др.; осуществлять поиск по имеющейся базе данных, перезаписывать и объединять базы. За основу взят объектно-ориентированный подход.

Задачи выполнения работы:

a) исследование предметной области с целью выявления основных принципов данной сферы;

b) проектирование иерархии классов, интерфейсов, взаимодействия компонентов на основе выделенных принципов и данных средств;

c) использование встроенные элементы среды для структуризации классов и оптимизации кода;

d) применение принципов объектно-ориентированной парадигмы к классам;

e) реализовать программное взаимодействие с базой данной формата *bin.

Объектно-ориентированный подход требует глубокого понимания основных принципов, или, иначе, концепций, на которых он базируется. В данном подходе основными концепциями являются понятия объектов и классов [1].

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования. Использование этого подхода предоставляет программисту широкие возможности в функциональности и в сопровождаемости проекта.

В качестве основного инструмента разработки применяется Microsoft Visual Studio 2015 Enterprise. Visual Studio представляет собой интегрированную среду разработки программ, созданную корпорацией Microsoft [5]. Язык программирования C#.

1 СПЕЦИФИКАЦИЯ ПРОГРАММЫ

1.1 Главное окно, добавление элементов

Работа с программой начинается с главного окна (рис. 1.1). Интерфейс программы прост в использовании и интуитивно понятен. Это является необходимым условием при разработке пользовательских интерфейсов [3].

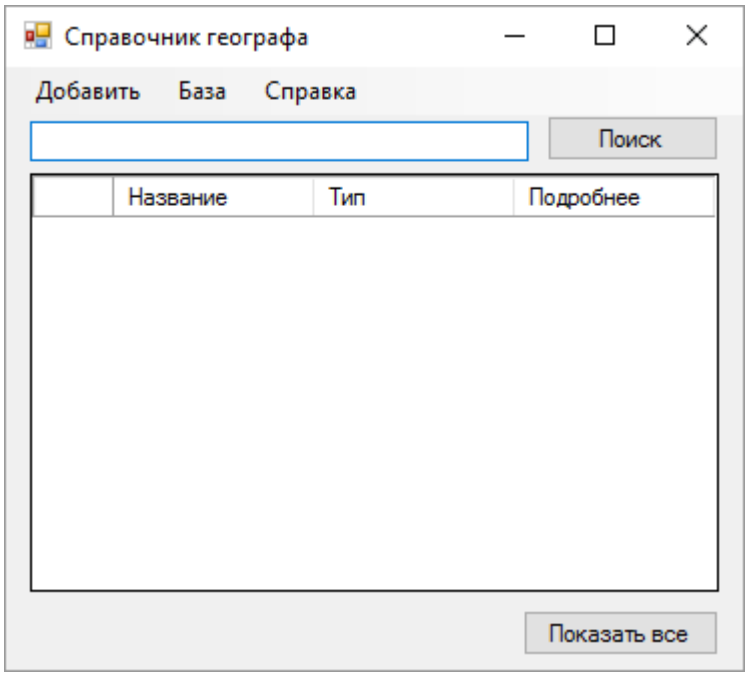

Рисунок 1.1 – Главное окно

Нажав на вкладку «Добавить», пользователь перейдет к окну с полями заполнения географических параметров для разных территориальных типов, начиная с материка (рис. 1.2). Окно также содержит в себе вкладки «Страна» (рис. 1.3), «Регион» (рис. 1.4) с подвкладками «Область», «Штат» и «Провинция», которые состоят из одинаковых полей; вкладку «Город» (рис. 1.5).

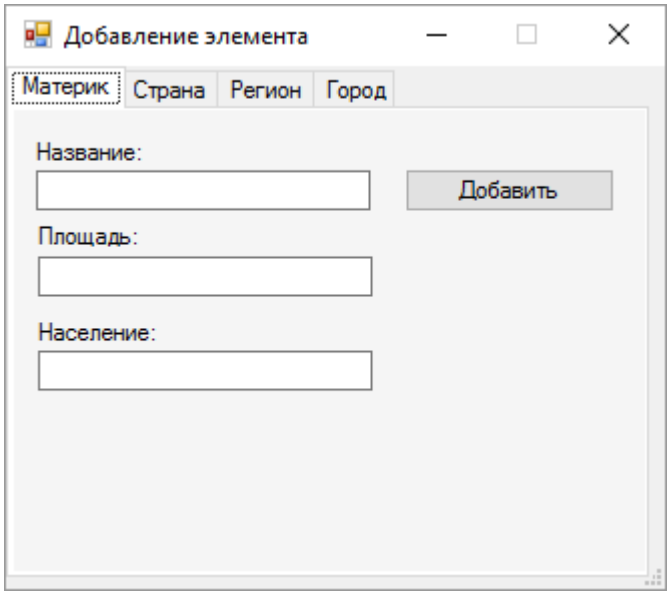

Рисунок 1.2 – Добавление материка

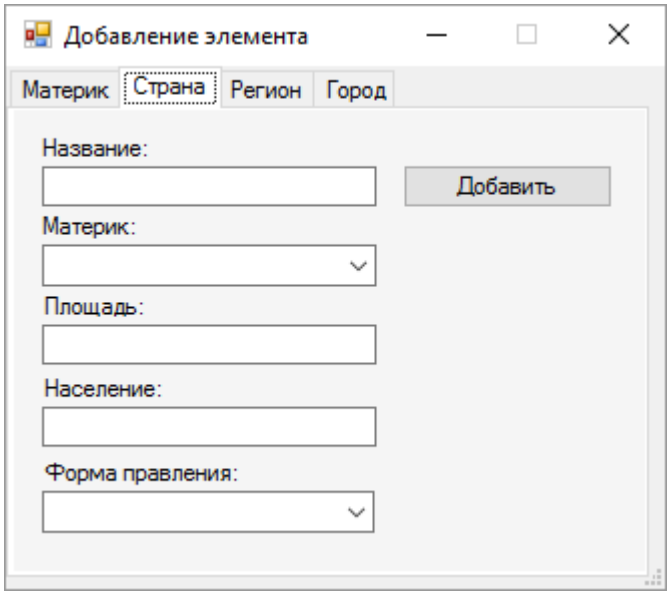

Рисунок 1.3 – Добавление страны

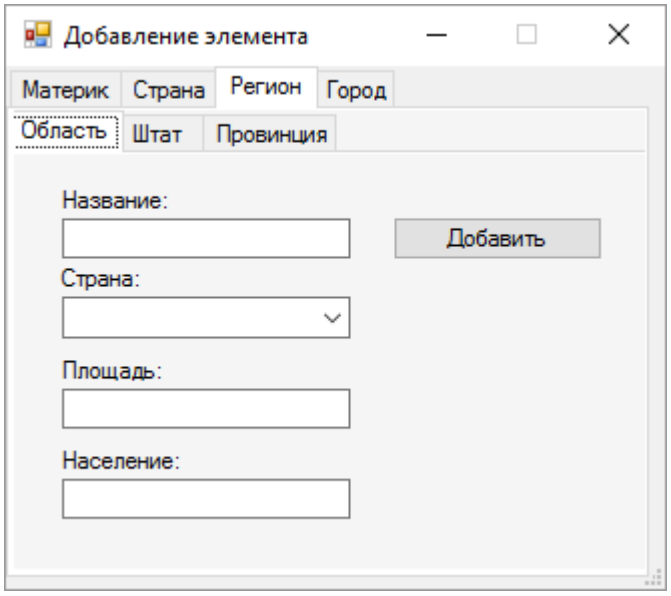

Рисунок 1.4 – Добавление региона

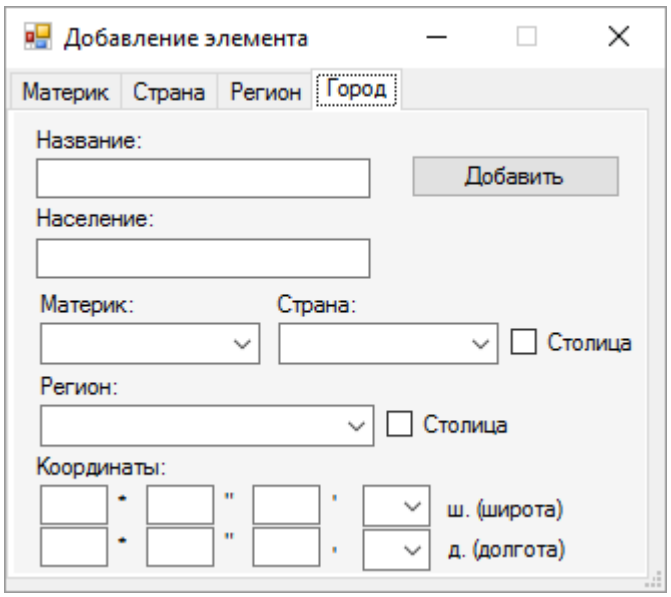

Рисунок 1.5 – Добавление города

1.2 Работа с базой, поиск в базе

Вкладка «База» содержит в себе подвкладки «Открыть», «Сохранить» и «Добавить» (рис. 1.6), которые при нажатии открывают диалоговое окно проводника, где пользователь может выбрать/сохранить/выбрать базу.

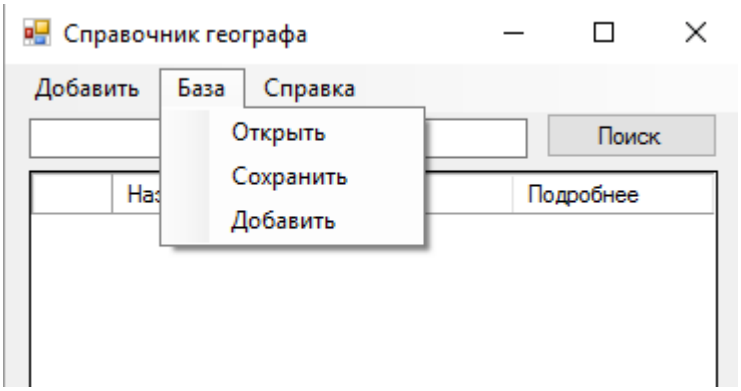

Рисунок 1.6 – Опции вкладки «База»

После открытия какой-либо базы пользователь может осуществить поиск по географическому типу, введя топоним и нажав кнопку «Поиск» (рис. 1.7).

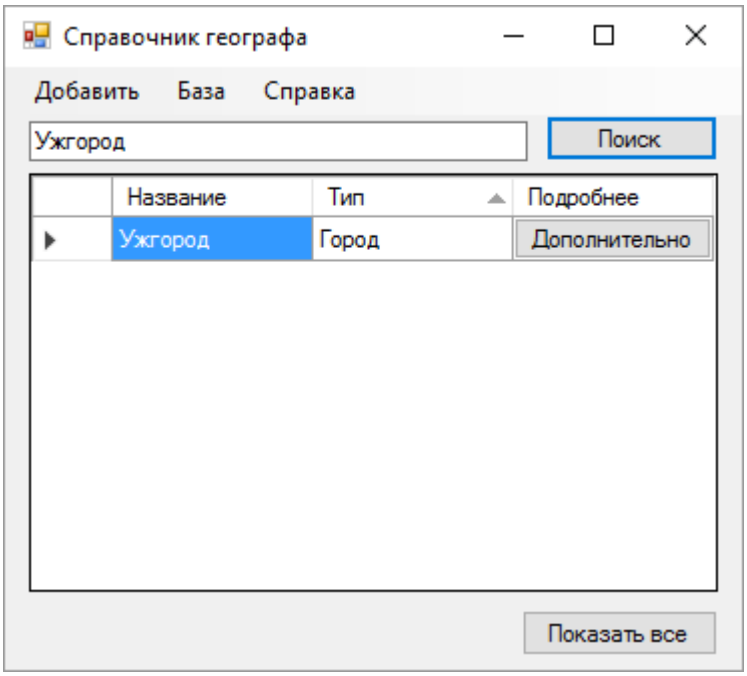

Рисунок 1.7 – Результат поиска

При желании получить вывод всех элементов, содержащихся в базе, необходимо нажать «Показать все». Программа выдает результат в формате списка табличного вида (рис. 1.8).

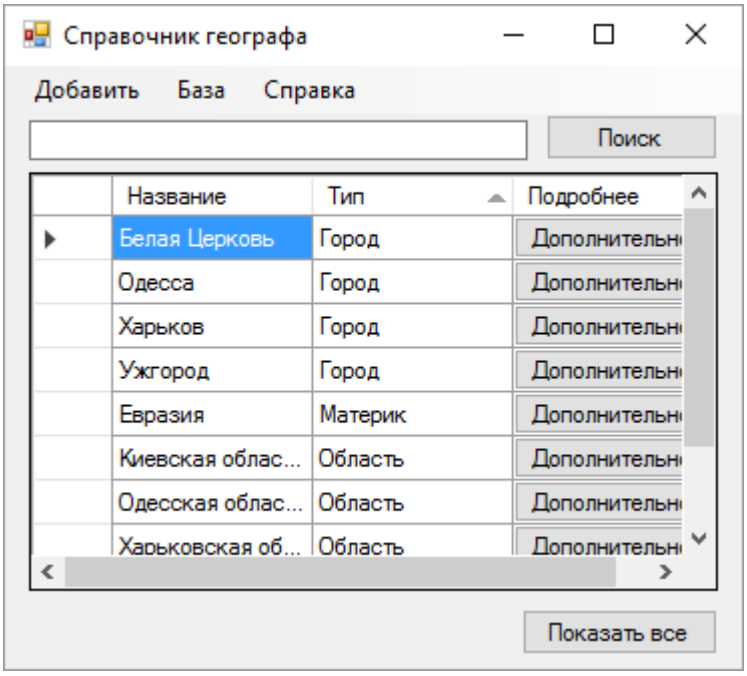

Рисунок 1.8 – Список всех элементов базы

1.3 Просмотр информации об элементе базы, справка

Нажав кнопку «Дополнительно», пользователь может изучить информацию об элементах, относящихся к различным категориям (материк, страна, регион или город), а также перейти от одной категории к другой. На примере дополнительной информации о городе (рис. 1.9) видно, что пользователь может перейти к просмотру информации об области (рис. 1.10), стране (рис. 1.11) и материке (рис. 1.12).

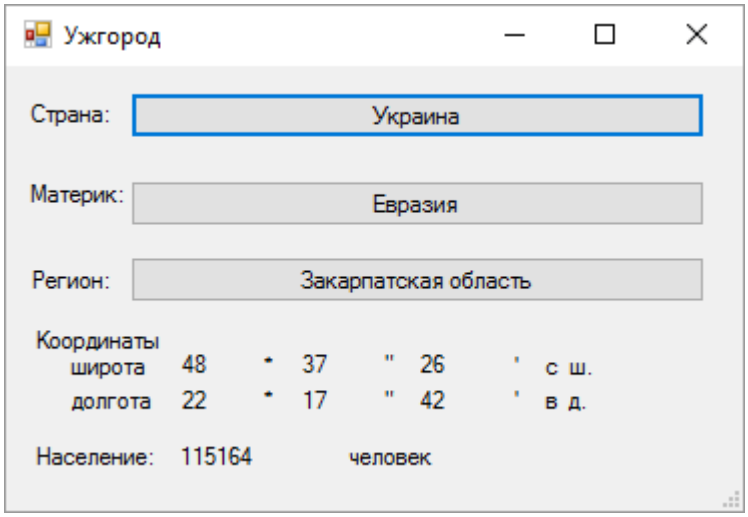

Рисунок 1.9 – Информация о городе

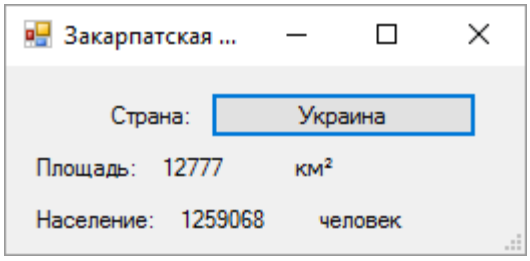

Рисунок 1.10 – Информация о регионе

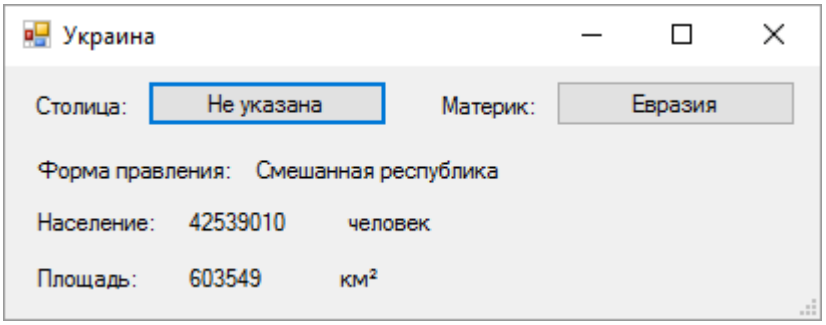

Рисунок 1.11 – Информация о стране

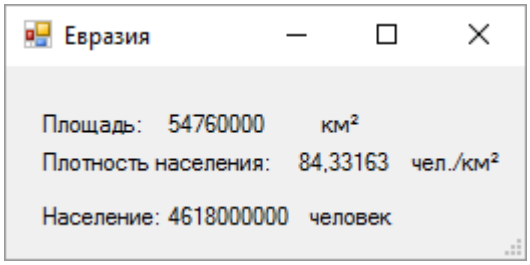

Рисунок 1.12 – Информация о материке

Вкладка «Справка» содержит подвкладки «О программе» и «Инструкция» (рис. 1.13), позволяющие узнать больше информации о программе (рис. 1.14), а также изучить руководство пользователя (рис. 1.15).

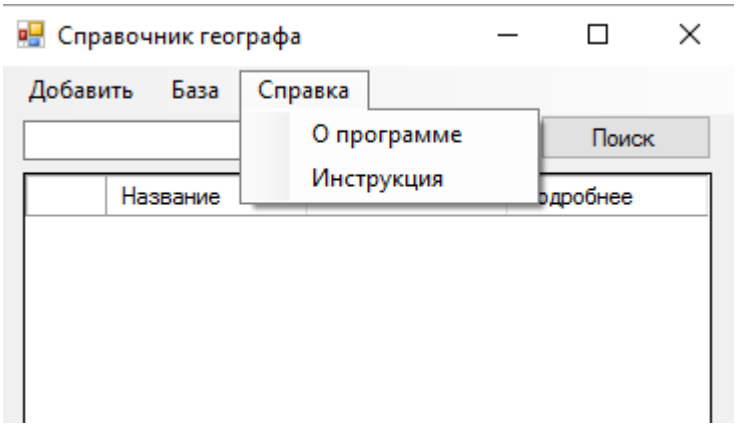

Рисунок 1.13 – Опции вкладки «Справка»

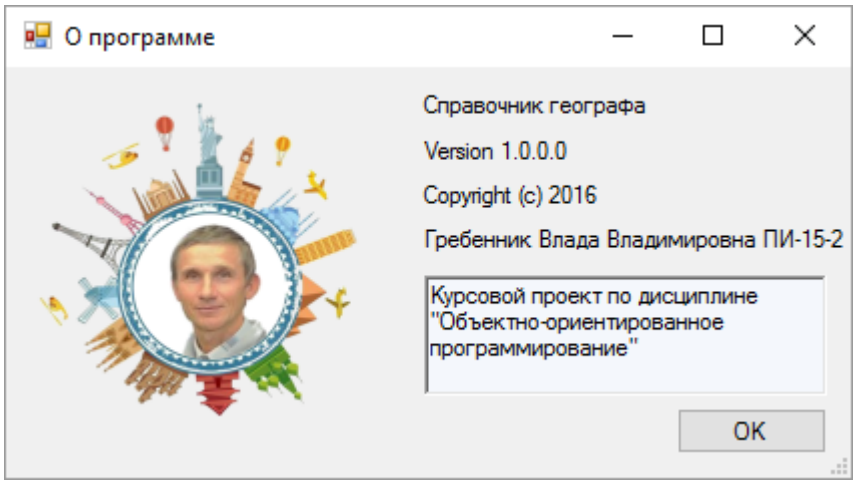

Рисунок 1.14 – Информация о программе

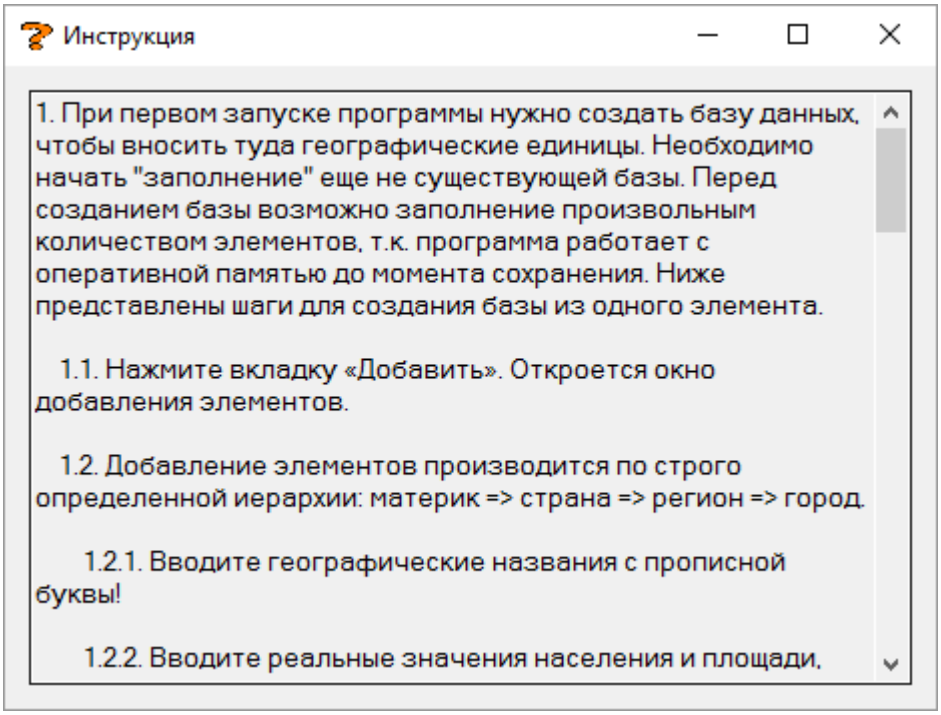

Рисунок 1.15 – Инструкция

2 ПРОЕКТНАЯ СПЕЦИФИКАЦИЯ

2.1 Объектная модель программы

Основным конструктивным элементом в языках ООП является модуль (module), представляющий собой логически связанную совокупность классов и объектов, а не подпрограмм, как в более ранних языках [2]. В графическом интерфейсе данной программы содержатся несколько форм заполнения и редактирования данных, следовательно в проекте присутствуют специальные вспомогательные функции, которые считывают данные, заполняют коллекции, сохраняют данные. Считывание данных происходит при загрузке окон. Методом записи и считывания соответственно является сериализация и десериализация.

При записи данных в документ с расширением *bin сериализируются соответственные классы Hesh, eNode, Mainland, Country, Region, Oblast, State, Provinces, City (рис. 2.1).

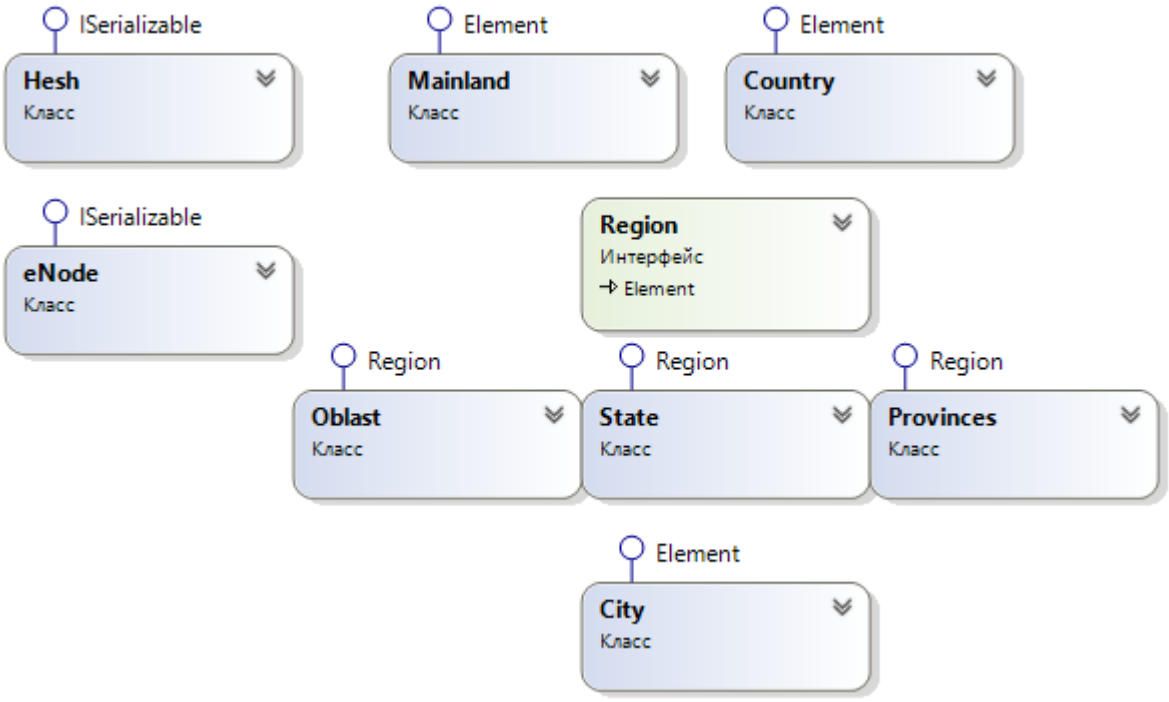

Рисунок 2.1 – Классы для работы с данными

Классы Hesh и eNode реализуют интерфейс ISerializible и содержат в себе следующие поля и методы, необходимые для создания хэш-таблицы – скелета программы (рис. 2.2):

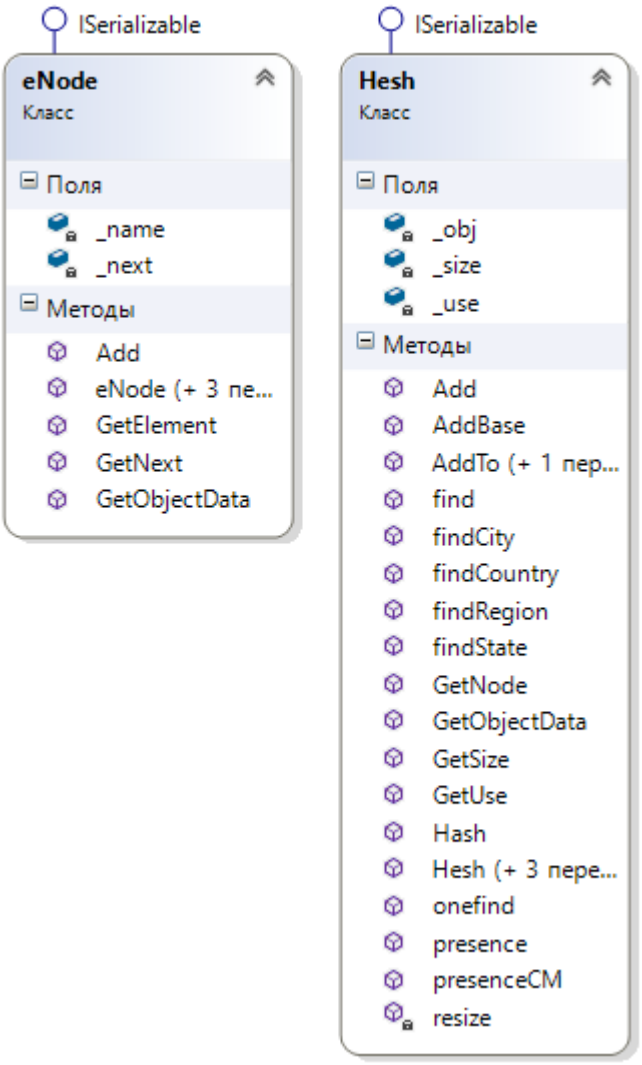

Рисунок 2.2 – Поля и методы классов Hesh и eNode

Классы Mainland, Country и City реализуют интерфейс Element и содержат в себе следующие поля и методы: (рис. 2.3). Класс City содержит также структуру для внесения координат.

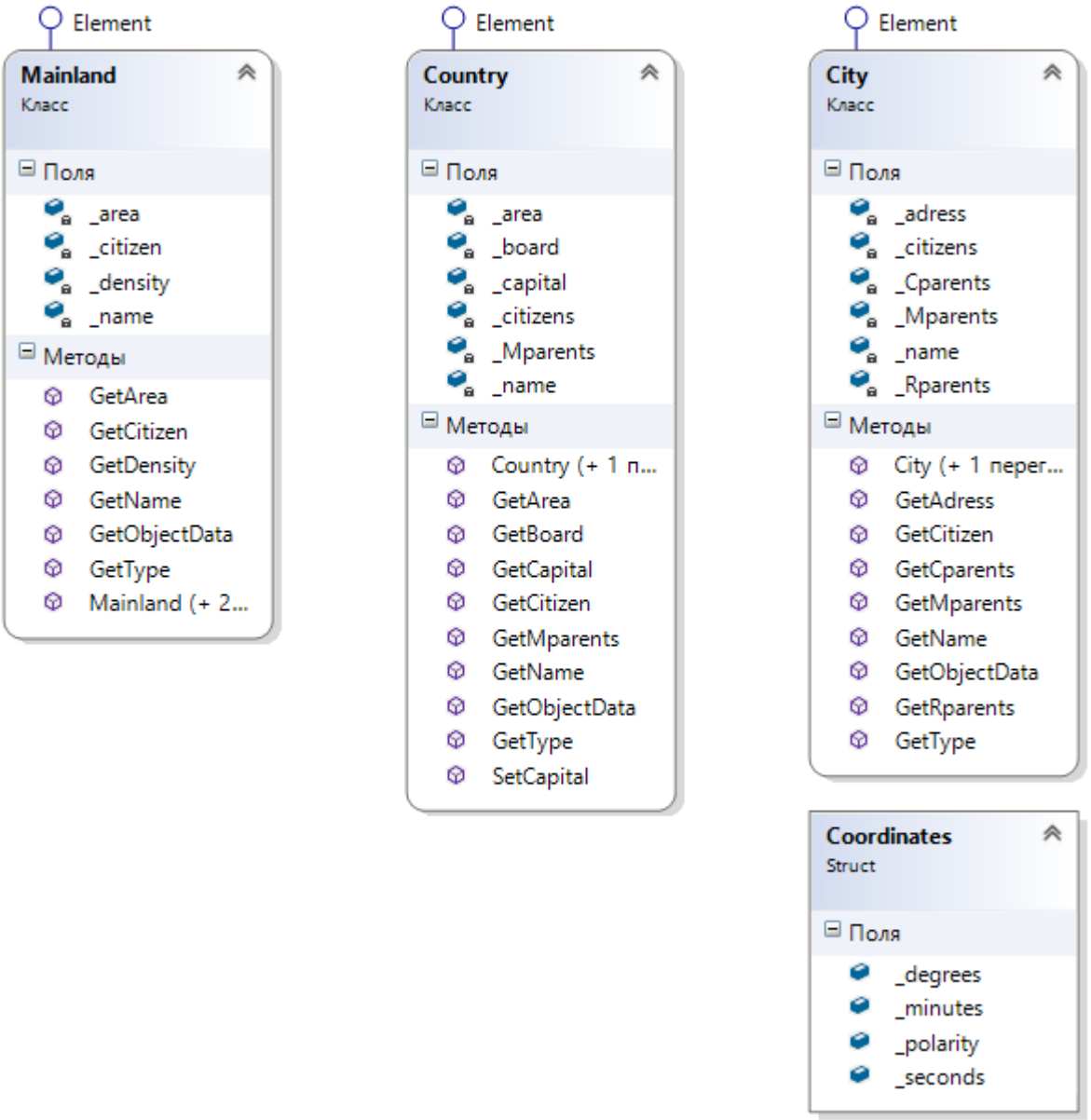

Рисунок 2.3 – Поля и методы классов Mainland, Country, City; структура Coordinates

Уже упомянутый класс-интерфейс Element является наследуемым для класса Region. Он содержит в себе методы для получения типа географического элемента, количества жителей и названия (рис. 2.4). Наследование позволяет определять в родительском классе общую функциональность, которая может применяться и, возможно, изменяться в дочерних классах [4].

Класс Region также является интерфейсом и реализуется подклассами Oblast, State и Provinces, которые имеют поля со ссылками и методы получения площади, названия, количества жителей и т.д. Класс Region содержит в себе методы для определения страны-родителя и получения площади (рис. 2.4).

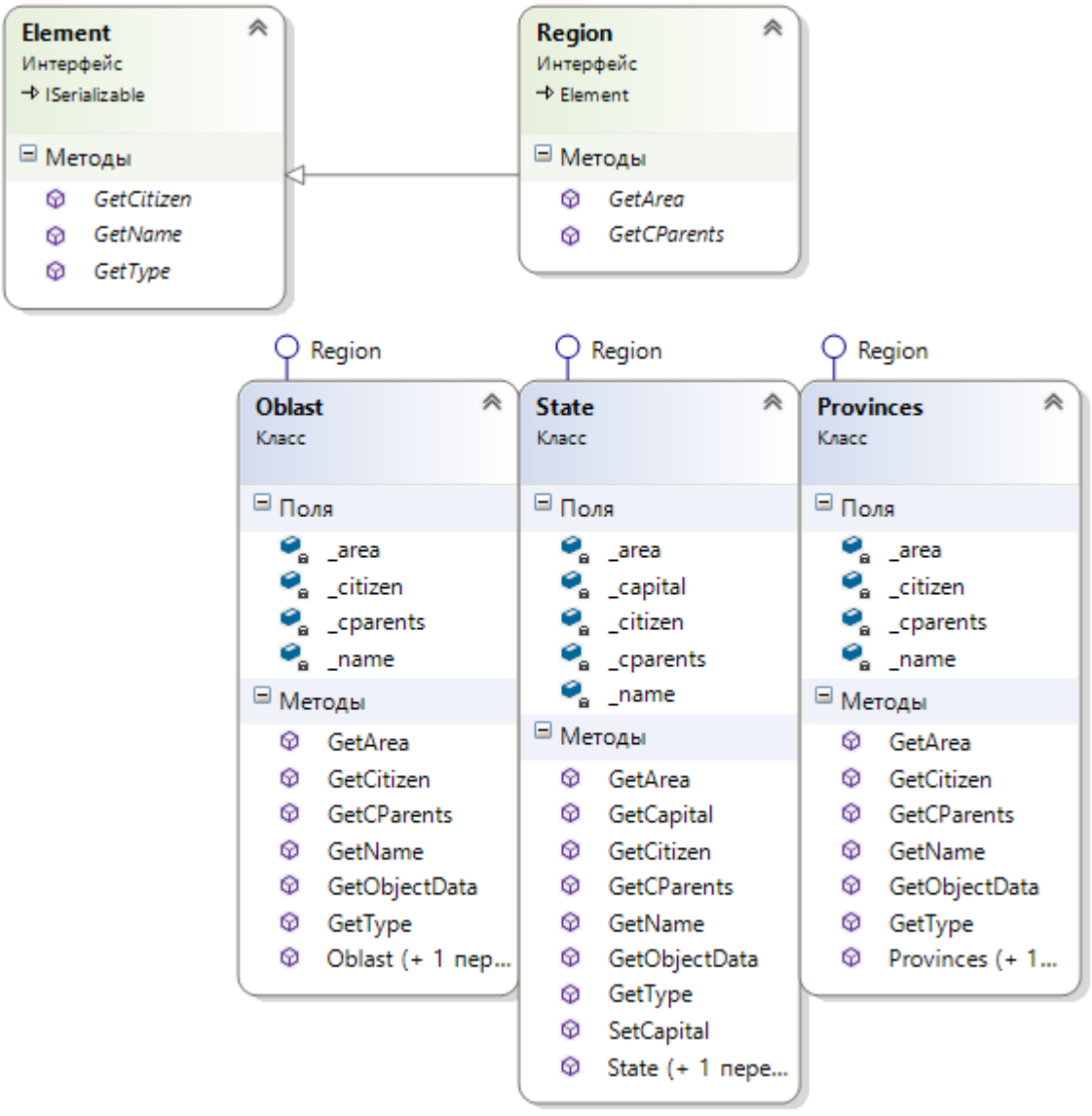

Рисунок 2.4 – Классы Element, Region, Oblast, State и Provinces

2.2 Реализация функций программы

Как упоминалось ранее, данная программа состоит из нескольких компонентов, которые тесно взаимосвязаны друг с другом. В этом разделе будут рассмотрены ключевые функции. Полный набор методов можно увидеть в коде программы (приложение А).

Скелетной основой программы является реализация хэш-таблицы. Она чрезвычайно удобна для хранения и считывания огромного количества информации в виде узлов с ключами и имеет более выигрышную алгоритмическую сложность по сравнению с записью данных в текстовые файлы. Ниже представлен один из стандартных алгоритмов для реализации хэш-функции (рис 2.5), а также методы для проверки наличия элемента в хэш-таблице (рис 2.6), добавления нового элемента в хэш-таблицу (рис 2.7) и метод возвращения всех элементов с заданным ключом (рис 2.8):

```
public int Hash(string v)
        {
            char[]k = v.ToCharArray();
            int hashsum, i;

            for (hashsum = i = 0;i<k.LongLength; i++)
                hashsum = (hashsum * 31) ^ k[i];
            return (hashsum & 0x7fffffff) % _size;
        }
```

Рисунок 2.5 – Хэш-функция

```
public bool presenceCM(string _elem)
       {
           eNode temp = Program._main.GetNode(Program._main.Hash(_elem));
           while (temp != null)
           {
               if (temp.GetElement().GetName().Equals(_elem)) return true;
               temp = temp.GetNext();
           }
```

Рисунок 2.6 – Проверка наличия элемента в хэш-таблице

```
public void Add(Element name)
       {
           try
           {
               if (_use > (_size / 2)) resize(_obj);
               if (presence(name)) throw new MyException("Элемент уже внесен в
базу!");
               if (_obj[Hash(name.GetName())] == null) _obj[Hash(name.GetName())] =
new eNode(name);
               else _obj[Hash(name.GetName())].Add(name);
               _use++;
           }
           catch(MyException exp)
           {
               MessageBoxButtons buttons = MessageBoxButtons.OK;
               MessageBox.Show(exp.GetMessage(), "Ошибка", buttons);
           }

        }
```

Рисунок 2.7 – Добавление нового элемента

```
public Element[] find(string key)
       {
           Element[] back;
           int size = 0;
           eNode temp = Program._main.GetNode(Program._main.Hash(key));
           while(temp!=null)
           {
               if (temp.GetElement().GetName().Equals(key)) size++;
               temp = temp.GetNext();
           }
           back = new Element[size];
           temp = Program._main.GetNode(Program._main.Hash(key));
           size = 0;
           while (temp != null)
           {
               if (temp.GetElement().GetName().Equals(key)) back[size++] =
temp.GetElement();
               temp = temp.GetNext();
           }
           return back;
       }
```

Рисунок 2.8 – Возвращение всех элементов с заданным ключом

Также одним из важнейших методов программы являются методы сериализации и десериализации объектов. Поскольку данные методы содержатся в нескольких классах, рассмотрим их на примере одного класса (рис. 2.9, рис. 2.10).

```
public virtual void GetObjectData(SerializationInfo info, StreamingContext context)
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            info.AddValue("size", _citizens);
            info.AddValue("capital", _capital);
            info.AddValue("area", _area);
            info.AddValue("Mparents", _Mparents);
            info.AddValue("name", _name);
            info.AddValue("board", _board);
         }
```

Рисунок 2.9 – Сериализация

```
   protected Country(SerializationInfo info, StreamingContext context)
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            _citizens   = (uint)info.GetValue("size",  typeof(uint));
            _capital = (City)info.GetValue("capital", typeof(City));
            _area = (uint)info.GetValue("area", typeof(uint));
            _Mparents = (Mainland)info.GetValue("Mparents", typeof(Mainland));
            _name = (string)info.GetValue("name", typeof(string));
            _board = (string)info.GetValue("board", typeof(string));
        }
```

Рисунок 2.10 – Десериализация

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. **При первом запуске программы** нужно создать базу данных, чтобы вносить туда географические единицы. Необходимо начать "заполнение" еще не существующей базы. Перед созданием базы возможно заполнение произвольным количеством элементов, т.к. программа работает с оперативной памятью до момента сохранения. Ниже представлены шаги для создания базы из одного элемента.

1.1. Нажмите вкладку «Добавить». Откроется окно добавления элементов.

1.2. Добавление элементов производится по строго определенной иерархии: материк => страна => регион => город.

1.2.1. **Вводите географические названия с прописной буквы!**

1.2.2. **Вводите реальные значения населения и площади, избегайте огромных значений!** Игнорирование этого приведет к досрочному завершению программы.

1.2.3. Учтите также то, что **некоторые параметры обязательно должны уменьшаться в указанном ранее иерархическом порядке**. Так, площадь страны не может превышать площадь материка, а площадь региона - соответственно, быть больше площади страны. Аналогично и для населения. Исключения существуют для карликовых городов-государств и городов автономного значения.

1.3. После добавления последнего т.н. «звена» (города) закройте окно добавления. Откройте вкладку «База» и выберите «Сохранить». Определите расположение и название вашей базы. Не забудьте после названия файла указать его тип (*bin). Например: UkraineCities.bin.

!Примечание: при первом запуске программы имеет место быть и такой сценарий: сохранение «пустой» базы. В таком случае после создания

незаполненной базы необходимо ее открыть, иначе добавления элементов в определенный файл *bin происходить не будет. Вкладка «База» => «Открыть».

2. После выполнения вышеуказанного в рамках функционала программы возможны следующие варианты действий: занести новую иерархическую цепочку в базу, поиск элементов в базе, просмотреть все элементы базы, создать новую(ые) базу(ы), добавить базу к существующей. **Прежде чем совершать какие-либо действия, откройте созданную базу с помощью подвкладки «Открыть» вкладки «База».**

2.1. Внесение новых элементов в базу происходит в соответствии с подпунктами 1.1. – 1.2. При добавлении элемента с каким-то общим звеном можно пропустить этот пункт. Например: при добавлении Ужгорода в базу, куда первым элементом был занесен Харьков, нет необходимости заново вводить параметры материка и страны. **Не забывайте сохранять (перезаписывать) базу после того, как вы закончите добавлять определенное вами количество элементов.**

2.2. Поиск элементов осуществляется следующим образом: в главном окне программы введите в строку поиска необходимый вам топоним (город, регион, материк, страна) и нажмите «Поиск». При наличии в базе данных топоним отобразится в области под строкой поиска, принимающей вид таблицы. Просмотреть информацию о топониме можно, нажав кнопку «Дополнительно».

2.3. Чтобы просмотреть список всех имеющихся в базе географических элементов в виде таблицы, нажмите «Показать все» под областью для представления списка.

2.4. Процесс создания новой базы аналогичен п. 1.1 – 1.3. Чтобы производить дальнейшие желаемые операции с этой базой, учтите, что она должна быть открытой.

2.5. Чтобы добавить базу к уже имеющейся открытой базе, выберите во вкладке «База» опцию «Добавить». Отметьте базу, которую хотите объединить с существующей. Дальнейшие действия (добавление нового топонима, поиск и пр.) выполняются для новой, комбинированной базы. Сохраните новую базу в той же вкладке, нажав на подвладку «Сохранить».

3. **При последующих запусках программы** перед ее эксплуатацией по прямому назначению **убедитесь, что вы открыли какую-либо из существующих баз.** Чтобы это сделать, нажмите вкладку «База», далее «Открыть» и выберите нужную вам базу. По желанию меняйте базы, открывая новые, объединяйте и сохраняйте их.

ВЫВОДЫ

Результатом выполнения данной курсовой работы стала программа – «Справочник географа». Данная программа предназначена для удобного поиска по топонимам.

В проекте представлен объектно-ориентированный подход в проектировании программного обеспечения информационного плана, дающий возможность на ранних этапах разработки учесть все нюансы будущей программы, необходимый набор функций, состав и структуру баз данных, что в дальнейшем исключает необходимость переработки уже написанных компонентов программы.

В программе присутствует простой и понятный пользовательский интерфейс, так что полностью разобраться во функционале программы не составит труда даже для неопытного пользователя.

Это приложение разработано для удобства тех, кто желает узнать справочную информацию о городах, регионах, странах и материках. С помощью этой программы можно легко найти элемент по топониму.

Планы на будущее: максимально расширить базу, создать расширенный поиск, реализовать функцию редактирования и удаления элемента из базы, функцию автоматического подставления константных параметров (например, площади материка) в соответствующее поле при добавлении элемента.

ПЕРЕЧЕНЬ ССЫЛОК

1. Бондарев, В.М. Объектно-ориентированное программирование на C# [Текст]: учеб. пособ. /В.М. Бондарев. – Х.: Компания СМИТ, 2009 – 224 с.

2. Буч, Г. Объектно-ориентированный анализ и проектирование [Текст]: пер. с англ – М.: 000 "И.Д. Вильяме", 2008 - 720 с.

3. Мандер, Т. Разработка пользовательского интерфейса [Текст]: пер. с англ. – М.: ДМК Пресс, 2008 – 412 с.

4. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд. [Текст]: пер. с англ. – М.: Вильямс, 2013 – 1312 с.

5. Шилдт, Г. C# 4.0.: Полное руководство [Текст]: пер. с англ. – М.: Вильямс, 2011 – 1056 с.

Приложение А

Частичный код программы

Класс Hesh

```
   [Serializable]
    class Hesh : ISerializable
    {
        int _size, _use;
        eNode []_obj;
        public Hesh()
        {
            _size = 50;
            _use = 0;
            _obj = new eNode[_size];
        }
        Hesh(int size)
        {
            _size = size;
            _use = 0;
            _obj = new eNode[_size];
        }

        Hesh(Hesh copy)//Конструктор, не используется (для обозримого будущего)
        {
            _size = copy._size;
            _use = copy._use;
            _obj = new eNode[_size];
            for (int t = 0; t < _size; t++)
            {
                _obj[t] = new eNode(copy._obj[t]);
            }
        }

       public bool presence(Element _new)//Проверяет присутствие элемента в хэш-таблице
        {
            eNode temp =
Program._main.GetNode(Program._main.Hash(_new.GetName()));//Создание временного элемента
для перебора элементов внутри узла
            while (temp != null)
            {
                if (temp.GetElement().GetName().Equals(_new.GetName()) &&
                    temp.GetElement().GetType().Equals(_new.GetType()) &&
                    temp.GetElement().GetCitizen().Equals(_new.GetCitizen())) return
true;
                temp = temp.GetNext();
            }


            return false;
        }

       public bool presenceCM(string _elem)//Проверяет присутствие элемента в хэш-таблице
       {
           eNode temp = Program._main.GetNode(Program._main.Hash(_elem));//Создание
временного элемента для перебора элементов внутри узла
           while (temp != null)
           {
               if (temp.GetElement().GetName().Equals(_elem)) return true;
```

```
                temp = temp.GetNext();
            }


            return false;
        }

         public int Hash(string v) //Хэш-функция, один из стандартных алгоритмов
         {
             char[]k = v.ToCharArray();
             int hashsum, i;

             for (hashsum = i = 0;i<k.LongLength; i++)
                 hashsum = (hashsum * 31) ^ k[i];
             return (hashsum & 0x7fffffff) % _size;
         }

        public int GetSize()//Возвращает текущий размер таблицы
         {
             return _size;
         }

        public int GetUse()
        {
            return _use;
        }

        public eNode GetNode(int index)//Возвращает элемент таблицы
        {
            return _obj[index];
        }

        public void AddTo(Hesh other)//Добавляет новые элементы из уже существующей хэш-
таблицы
         {
            for(int t=0;t<other._size;t++)
            {
                if(other._obj[t]!=null)
                {
                   if(other._obj[t].GetNext()==null)
Program._main.AddBase(other._obj[t].GetElement());
                   else
                   {
                       eNode temp = other._obj[t];
                       while(temp!=null)
                       {
                           Program._main.AddBase(temp.GetElement());
                           _use++;
                           temp = temp.GetNext();
                       }
                   }
                }
            }
            if (_use > (_size / 2)) resize(_obj);
         }

        public void AddTo(int size,eNode []obj)//Добавляет новые элементы из уже
существующей хэш-таблицы
        {
            for (int t = 0; t < size; t++)
            {
                if (obj[t] != null)
                {
```

```
                    if (obj[t].GetNext() == null)
Program._main.AddBase(obj[t].GetElement());
                    else
                    {
                        eNode temp = obj[t];
                        while (temp != null)
                        {
                            Program._main.AddBase(temp.GetElement());
                            _use++;
                            temp = temp.GetNext();
                        }
                    }
                }
            }
            if (_use > (_size / 2)) resize(_obj);
        }

        public void Add(Element name)//Добавляет новый элемент в хэш-таблицу
        {
            try
            {
                if (_use > (_size / 2)) resize(_obj);
                if (presence(name)) throw new MyException("Элемент уже внесен в базу!");
                if (_obj[Hash(name.GetName())] == null) _obj[Hash(name.GetName())] = new
eNode(name);
                else _obj[Hash(name.GetName())].Add(name);
                _use++;
            }
            catch(MyException exp)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show(exp.GetMessage(), "Ошибка", buttons);
            }

        }

        public void AddBase(Element name)//Добавляет новый элемент в хэш-таблицу
        {
                if (!presence(name))
                {
                    if (_use > (_size / 2)) resize(_obj);
                    if (_obj[Hash(name.GetName())] == null) _obj[Hash(name.GetName())] =
new eNode(name);
                    else _obj[Hash(name.GetName())].Add(name);
                    _use++;
                }
        }

        public Element[] find(string key)//Возвращает все элементы с заданым ключом
        {
            Element[] back;
            int size = 0;
            eNode temp = Program._main.GetNode(Program._main.Hash(key));
            while(temp!=null)
            {
                if (temp.GetElement().GetName().Equals(key)) size++;
                temp = temp.GetNext();
            }
            back = new Element[size];
            temp = Program._main.GetNode(Program._main.Hash(key));
            size = 0;
            while (temp != null)
            {
```

```
                if (temp.GetElement().GetName().Equals(key)) back[size++] =
temp.GetElement();
                temp = temp.GetNext();
            }
            return back;
        }

        public Country findCountry(string key,Mainland Mparents)
        {
                Country back;
                eNode temp = Program._main.GetNode(Program._main.Hash(key));
                while (temp != null)
                {
                    if (temp.GetElement().GetName().Equals(key))
                        if (temp.GetElement().GetType().Equals("Страна"))
                        {
                            back = (Country)temp.GetElement();
                            if (back.GetMparents() == Mparents) return back;
                        }
                    temp = temp.GetNext();
                }
                return null;
        }

        public Region findRegion(string key,Country Cparents)
        {
            Region back;
            eNode temp = Program._main.GetNode(Program._main.Hash(key));
            while (temp != null)
            {
                if (temp.GetElement().GetName().Equals(key))
                    if (temp.GetElement().GetType().Equals("Область") ||
                        temp.GetElement().GetType().Equals("Штат") ||
                        temp.GetElement().GetType().Equals("Провинция"))
                    {
                        back = (Region)temp.GetElement();
                        if (back.GetCParents() == Cparents) return back;
                    }
                temp = temp.GetNext();
            }
            return null;
        }

        public State findState(string key, Country Cparents)
        {
            State back;
            eNode temp = Program._main.GetNode(Program._main.Hash(key));
            while (temp != null)
            {
                if (temp.GetElement().GetName().Equals(key))
                    if (temp.GetElement().GetType().Equals("Штат"))
                    {
                        back = (State)temp.GetElement();
                        if (back.GetCParents() == Cparents) return back;
                    }
                temp = temp.GetNext();
            }
            return null;
        }

        public City findCity(string key, Region Rparents)
        {
            City back;
```

```
            eNode temp = Program._main.GetNode(Program._main.Hash(key));
            while (temp != null)
            {
                if (temp.GetElement().GetName().Equals(key))
                    if (temp.GetElement().GetType().Equals("Город"))
                    {
                        back = (City)temp.GetElement();
                        if (back.GetRparents() == Rparents) return back;
                    }
                temp = temp.GetNext();
            }
            return null;
        }

        public Element onefind(string _key)//Поиск единичного элемента||заменить на поиск
материка
         {
             if (_obj[Hash(_key)].GetElement().GetName().Equals(_key))
             return _obj[Hash(_key)].GetElement();
             else
             {
                 eNode temp = _obj[Hash(_key)];
                 while(temp.GetNext()!=null)
                 {
                     temp = temp.GetNext();
                     if (temp.GetElement().GetName().Equals(_key))
                         return temp.GetElement();
                 }
             }
             return null;
         }
        private void resize(eNode []obj)//Увеличение размера для уменьшения коллизии
         {
             int oldsize = _size;
             eNode[] oldobj = _obj;
             _size = oldsize * 2;
             _obj = new eNode[_size];
             Program._main.AddTo(oldsize, oldobj);
         }

        protected Hesh(SerializationInfo info, StreamingContext context)//Десериализация
         {
             if (info == null)
                 throw new System.ArgumentNullException("info");

             _size   = (int)info.GetValue("size",  typeof(int));
             _obj     = (eNode[])info.GetValue("obj",    typeof(eNode[]));
             _use  = (int)    info.GetValue("use", typeof(int));
         }

         [SecurityPermission(SecurityAction.LinkDemand, Flags =
SecurityPermissionFlag.SerializationFormatter)]
        public virtual void GetObjectData(SerializationInfo info, StreamingContext
context)//Сериализация
         {
             if (info == null)
                 throw new System.ArgumentNullException("info");

             info.AddValue("size", _size);
             info.AddValue("use", _use);
             info.AddValue("obj", _obj);
         }
    }
```

Класс eNode

```
    [Serializable]
    class eNode : ISerializable
    {
        private Element _name;
        private eNode _next;
       public  eNode(Element name)
        {
            _name = name;
            _next = null;
        }

       public eNode(Element name,eNode next)
       {
           _name = name;
           _next = next;
       }

        public eNode(eNode old)
       {
           if (old._next == null)
           {
               _name = old._name;
               _next = null;
           }
           else
           {
                   _name = old._name;
                   _next = new eNode(old._next);
           }
       }

        public Element GetElement()
        {
            return _name;
        }

        public eNode GetNext()
        {
            return _next;
        }
        public void Add(Element name)
        {
            _next = new eNode(name,_next);
        }

        protected eNode(SerializationInfo info, StreamingContext context)//Десериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            _name     = (Element)info.GetValue("name",    typeof(Element));
            _next    = (eNode)info.GetValue("next",   typeof(eNode));
        }

        [SecurityPermission(SecurityAction.LinkDemand, Flags =
SecurityPermissionFlag.SerializationFormatter)]
        public virtual void GetObjectData(SerializationInfo info, StreamingContext
context)//Сериализация
        {
```

```
            if (info == null)
                throw new System.ArgumentNullException("info");

            info.AddValue("name", _name);
            info.AddValue("next", _next);
        }
    }
```

Класс Mainland

```
    [Serializable]
    public class Mainland : Element
    {
       private uint   _area;
       private ulong  _citizen;
       private float  _density;
       private string _name;

        public uint GetArea()
       {
           return _area;
       }

        public float GetDensity()
        {
            return _density;
        }

        public Mainland(uint area, ulong citizen, float density, string name)
        {
            _area    = area;
            _citizen = citizen;
            _density = density;
            _name    = name;
        }

        Mainland(Mainland other)//Конструктор//не используется
        {
            _area    = other._area;
            _citizen = other._citizen;
            _density = other._density;
            _name    = other._name;
        }

        public string GetType()
        {
            return  "Материк";
        }

        public string GetName()
        {
            return _name;
        }

        public ulong GetCitizen()
        {
            return _citizen;
        }

        protected Mainland(SerializationInfo info, StreamingContext context)
//Десериализация
        {
```

```
            if (info == null)
                throw new System.ArgumentNullException("info");

            _name   = (string)info.GetValue("size",  typeof(string));
            _area = (uint)info.GetValue("area", typeof(uint));
            _citizen = (ulong)info.GetValue("citizen", typeof(ulong));
            _density = (float)info.GetValue("density", typeof(float));
        }

        [SecurityPermission(SecurityAction.LinkDemand, Flags =
SecurityPermissionFlag.SerializationFormatter)]
        public virtual void GetObjectData(SerializationInfo info, StreamingContext
context)//Сериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            info.AddValue("size", _name);
            info.AddValue("area", _area);
            info.AddValue("citizen", _citizen);
            info.AddValue("density", _density);
        }
    }
```

Класс Country

```
     [Serializable]
   public class Country : Element
    {
        ulong _citizens;
        City _capital;
        uint _area;
        Mainland _Mparents;
        string _name;
        string _board;

         public City GetCapital()
        {
            return _capital;
        }
         public uint GetArea()
        {
            return _area;
        }
         public string GetBoard()
        {
            return _board;
        }
         public Mainland GetMparents()
        {
            return _Mparents;
        }

         public void SetCapital(City capital)//Задать столицу
         {
             _capital = capital;
         }
        public Country(ulong citizens, uint area, string name, string board, Mainland
Mparents)
        {
            _citizens = citizens;
            _Mparents = Mparents;
```

```
            _capital  = null;
            _area     = area;
            _name     = name;
            _board    = board;
        }

         public string GetType()
         {
             return "Страна";
         }

         public string GetName()
         {
             return _name;
         }

         public ulong GetCitizen()
         {
             return _citizens;
         }

         protected Country(SerializationInfo info, StreamingContext context)//
Десериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            _citizens   = (uint)info.GetValue("size",  typeof(uint));
            _capital = (City)info.GetValue("capital", typeof(City));
            _area = (uint)info.GetValue("area", typeof(uint));
            _Mparents = (Mainland)info.GetValue("Mparents", typeof(Mainland));
            _name = (string)info.GetValue("name", typeof(string));
            _board = (string)info.GetValue("board", typeof(string));
        }

        [SecurityPermission(SecurityAction.LinkDemand, Flags =
SecurityPermissionFlag.SerializationFormatter)]
        public virtual void GetObjectData(SerializationInfo info, StreamingContext
context)//Сериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            info.AddValue("size", _citizens);
            info.AddValue("capital", _capital);
            info.AddValue("area", _area);
            info.AddValue("Mparents", _Mparents);
            info.AddValue("name", _name);
            info.AddValue("board", _board);
        }
    }
```

Класс Region

```
    public interface Region : Element
    {
        Country GetCParents();
        uint GetArea();
    }
    [Serializable]
   public class Oblast : Region     {
        string _name;
```

```
        uint _citizen;
        Country _cparents;
        uint _area;

        public uint GetArea()
        {
            return _area;
        }
        public Country GetCParents()
        {
            return _cparents;
        }

       public Oblast(string name,uint citizen,uint area,Country cparents)
        {
            _name = name;
            _citizen = citizen;
            _area = area;
            _cparents = cparents;
        }

        public string GetType()
        {
            return "Область";
        }

        public string GetName()
        {
            return _name;
        }

        public ulong GetCitizen()
        {
            return _citizen;
        }

        protected Oblast(SerializationInfo info, StreamingContext context)//
Десериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            _citizen   = (uint)info.GetValue("citizen",  typeof(uint));
            _area = (uint)info.GetValue("area", typeof(uint));
            _name = (string)info.GetValue("name", typeof(string));
            _cparents = (Country)info.GetValue("cparents", typeof(Country));
        }

        [SecurityPermission(SecurityAction.LinkDemand, Flags =
SecurityPermissionFlag.SerializationFormatter)]
        public virtual void GetObjectData(SerializationInfo info, StreamingContext
context)//Сериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            info.AddValue("citizen", _citizen);
            info.AddValue("area", _area);
            info.AddValue("name", _name);
            info.AddValue("cparents", _cparents);

        }
    }
```

```
    [Serializable]
   public  class State : Region
    {
        string _name;
        uint _citizen;
        City _capital;
        uint _area;
        Country _cparents;//Страна-родитель

        public void SetCapital(City capital)//Назначить столицу
        {
            _capital = capital;
        }

        public City GetCapital()
        {
            return _capital;
        }
       public State(string name, uint citizen, uint area, Country cparents)
        {
            _name = name;
            _citizen = citizen;
            _capital = null;
            _area = area;
            _cparents = cparents;
        }

       public Country GetCParents()
       {
           return _cparents;
       }

        public string GetType()
        {
            return "Штат";
        }

        public string GetName()
        {
            return _name;
        }

        public uint GetArea()
        {
            return _area;
        }

        public ulong GetCitizen()
        {
            return _citizen;
        }

        protected State(SerializationInfo info, StreamingContext context)//
Десериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            _citizen   = (uint)info.GetValue("citizen",  typeof(uint));
            _area = (uint)info.GetValue("area", typeof(uint));
            _name = (string)info.GetValue("name", typeof(string));
            _cparents = (Country)info.GetValue("cparents", typeof(Country));
            _capital = (City)info.GetValue("capital", typeof(City));
```

```
        }

        [SecurityPermission(SecurityAction.LinkDemand, Flags =
SecurityPermissionFlag.SerializationFormatter)]
         public virtual void GetObjectData(SerializationInfo info, StreamingContext
context)// Сериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            info.AddValue("citizen", _citizen);
            info.AddValue("area", _area);
            info.AddValue("name", _name);
            info.AddValue("cparents", _cparents);
            info.AddValue("capital", _capital);

        }
    }
    [Serializable]
  public class Provinces : Region
    {
        string _name;
        uint _citizen;
        Country _cparents;//Страна-родитель
        uint _area;

       public Provinces(string name, uint citizen, uint area, Country cparents)
        {
            _name = name;
            _citizen = citizen;
            _area = area;
            _cparents = cparents;
        }

        public uint GetArea()
       {
           return _area;
       }
        public string GetType()
        {
            return "Провинция";
        }

        public Country GetCParents()
        {
            return _cparents;
        }

        public string GetName()
        {
            return _name;
        }

        public ulong GetCitizen()
        {
            return _citizen;
        }

        protected Provinces(SerializationInfo info, StreamingContext context)//
Десериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");
```

```
            _citizen   = (uint)info.GetValue("citizen",  typeof(uint));
            _area = (uint)info.GetValue("area", typeof(uint));
            _name = (string)info.GetValue("name", typeof(string));
            _cparents = (Country)info.GetValue("cparents", typeof(Country));
        }

        [SecurityPermission(SecurityAction.LinkDemand, Flags =
SecurityPermissionFlag.SerializationFormatter)]
         public virtual void GetObjectData(SerializationInfo info, StreamingContext
context)//Сериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            info.AddValue("citizen", _citizen);
            info.AddValue("area", _area);
            info.AddValue("name", _name);
            info.AddValue("cparents", _cparents);

        }
    }
```

Класс City

```
    [Serializable]
   public  struct Coordinates// Структура для хранения координат
    {
        public char _polarity;// Полярность
        public byte _degrees;// Градусы
        public byte _minutes;// Минуты
        public byte _seconds;// Секунды
    }
    [Serializable]
    public class City: Element
    {
        Coordinates []_adress;
        uint _citizens;
        string _name;
        Region _Rparents;// Ссылка на регион
        Country _Cparents;// Ссылка на страну
        Mainland _Mparents;// Ссылка на материк

        public Country GetCparents()
        {
            return _Cparents;
        }
        public Mainland GetMparents()
        {
            return _Mparents;
        }
        public Coordinates[] GetAdress()
        {
            return _adress;
        }
        public Region GetRparents()
        {
            return _Rparents;
        }
        public City(char latitude_polarity, byte latitude_degrees, byte latitude_minutes,
byte latitude_seconds,
```

```
                    char longitude_polarity, byte longitude_degrees, byte
longitude_minutes, byte longitude_seconds,
                    uint citizens, string name, Region Rparents, Country Cparents,
Mainland Mparents)
        {
            _adress=new Coordinates[2];//Широта и долгота
            _adress[0]._degrees = latitude_degrees;
            _adress[0]._polarity = latitude_polarity;
            _adress[0]._minutes = latitude_minutes;
            _adress[0]._seconds = latitude_seconds;
            _adress[1]._degrees = longitude_degrees;
            _adress[1]._polarity = longitude_polarity;
            _adress[1]._minutes = longitude_minutes;
            _adress[1]._seconds = longitude_seconds;
            _citizens = citizens;
            _name = name;
            _Rparents = Rparents;
            _Cparents = Cparents;
            _Mparents = Mparents;
        }

        public string GetType()
        {
            return "Город";
        }

        public string GetName()
        {
            return _name;
        }

        public ulong GetCitizen()
        {
            return _citizens;
        }

        protected City(SerializationInfo info, StreamingContext context)//Десериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            _citizens   = (uint)info.GetValue("citizen",  typeof(uint));
            _name = (string)info.GetValue("name", typeof(string));
            _Cparents = (Country)info.GetValue("cparents", typeof(Country));
            _Rparents = (Region)info.GetValue("rparents", typeof(Region));
            _Mparents = (Mainland)info.GetValue("mparents", typeof(Mainland));
            _adress = (Coordinates[])info.GetValue("adress", typeof(Coordinates[]));
        }

        [SecurityPermission(SecurityAction.LinkDemand, Flags =
SecurityPermissionFlag.SerializationFormatter)]
        public virtual void GetObjectData(SerializationInfo info, StreamingContext
context)//Сериализация
        {
            if (info == null)
                throw new System.ArgumentNullException("info");

            info.AddValue("citizen", _citizens);
            info.AddValue("name", _name);
            info.AddValue("cparents", _Cparents);
            info.AddValue("mparents", _Mparents);
            info.AddValue("rparents", _Rparents);
            info.AddValue("adress", _adress);
```

```
        }

    }
```

Форма MainInterface

```
    public partial class MainInterface : Form
    {
        private static Element[] poisk;
        public MainInterface()
        {
            InitializeComponent();
        }

        private void добавитьToolStripMenuItem_Click(object sender, EventArgs
e)//Добавить новый элемент
        {
            AddingElement temp = new AddingElement();
            temp.ShowDialog(this);
        }

        private void открытьToolStripMenuItem_Click(object sender, EventArgs e)//Работа с
базой
        {
            Stream myStream = null;
            OpenFileDialog openFileDialog1 = new OpenFileDialog();

            openFileDialog1.InitialDirectory = "c:\\";
            openFileDialog1.Filter = "bin files (*.bin)|*.bin|All files (*.*)|*.*";
            openFileDialog1.FilterIndex = 2;
            openFileDialog1.RestoreDirectory = true;

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    if ((myStream = openFileDialog1.OpenFile()) != null)
                    {
                        using (myStream)
                        {
                            var formater = new BinaryFormatter();
                            Program._main = (Hesh)formater.Deserialize(myStream);
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: Could not read file from disk. Original
error: " + ex.Message);
                }
            }
        }

        private void сохранитьToolStripMenuItem_Click(object sender, EventArgs e)//Работа
с базой
        {
            Stream myStream;
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();

            saveFileDialog1.Filter = "bin files (*.bin)|*.bin|All files (*.*)|*.*";
            saveFileDialog1.FilterIndex = 2;
```

```
            saveFileDialog1.RestoreDirectory = true;

            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                if ((myStream = saveFileDialog1.OpenFile()) != null)
                {
                    var formater = new BinaryFormatter();
                    formater.Serialize(myStream, Program._main);
                    myStream.Close();
                }
            }
        }

        private void добавитьToolStripMenuItem1_Click(object sender, EventArgs e)//Работа
с базой
        {
            Stream myStream = null;
            OpenFileDialog openFileDialog1 = new OpenFileDialog();

            openFileDialog1.InitialDirectory = "c:\\";
            openFileDialog1.Filter = "bin files (*.bin)|*.bin|All files (*.*)|*.*";
            openFileDialog1.FilterIndex = 2;
            openFileDialog1.RestoreDirectory = true;

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    if ((myStream = openFileDialog1.OpenFile()) != null)
                    {
                        using (myStream)
                        {
                            var formater = new BinaryFormatter();
                            Hesh temp = new Hesh();
                            temp = (Hesh)formater.Deserialize(myStream);
                            Program._main.AddTo(temp);
                            temp = null;
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: Could not read file from disk. Original
error: " + ex.Message);
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)//Поиск
        {
            dataGridView1.Rows.Clear();
            poisk = Program._main.find(textBox1.Text);
            for (int x = 0; x < poisk.Length;x++ )
            {
                dataGridView1.Rows.Add();
                dataGridView1.Rows[x].Cells[0].Value = poisk[x].GetName();
                dataGridView1.Rows[x].Cells[1].Value = poisk[x].GetType();
                dataGridView1.Rows[x].Cells[2].Value = "Дополнительно";

            }
        }
```

```
         void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs
e)//Дополнительно
        {
            var senderGrid = (DataGridView)sender;

            if (senderGrid.Columns[e.ColumnIndex] is DataGridViewButtonColumn &&
                e.RowIndex >= 0)
            {
                if (poisk[e.RowIndex].GetType().Equals("Материк"))
                {
                    MainlandInfo temp = new MainlandInfo((Mainland)poisk[e.RowIndex]);
                    temp.ShowDialog(this);
                }
                else if (poisk[e.RowIndex].GetType().Equals("Страна"))
                {
                    CountryInfo temp = new CountryInfo((Country)poisk[e.RowIndex]);
                    temp.ShowDialog(this);
                }
                else if (poisk[e.RowIndex].GetType().Equals("Область") ||
                        poisk[e.RowIndex].GetType().Equals("Провинция")
                       )
                {
                    RegionInfo1 temp = new RegionInfo1((Region)poisk[e.RowIndex]);
                    temp.ShowDialog(this);
                }
                else if (poisk[e.RowIndex].GetType().Equals("Штат") )
                {
                    RegionInfo2 temp = new RegionInfo2((State)poisk[e.RowIndex]);
                    temp.ShowDialog(this);
                }
                else if (poisk[e.RowIndex].GetType().Equals("Город"))
                {
                    CityInfo temp = new CityInfo((City)poisk[e.RowIndex]);
                    temp.ShowDialog(this);
                }
            }
        }

         private void оПрограммеToolStripMenuItem_Click(object sender, EventArgs e)
         {
             kur.AboutProgram temp = new kur.AboutProgram();
             temp.ShowDialog(this);
         }

         private void ИнструкцияToolStripMenuItem_Click(object sender, EventArgs e)
         {
             MessageBoxButtons buttons = MessageBoxButtons.OK;
             MessageBox.Show("\tыфафыа", "Инструкция пользователя", buttons);
         }

         private void button2_Click(object sender, EventArgs e)
         {
             int _rows = 0;
             dataGridView1.Rows.Clear();
             poisk = new Element[Program._main.GetUse()];
             for (int x = 0; x < Program._main.GetSize(); x++)
             {
                 if(Program._main.GetNode(x)!=null)
                 {
                 dataGridView1.Rows.Add();
                 dataGridView1.Rows[_rows].Cells[0].Value =
Program._main.GetNode(x).GetElement().GetName();
```

```
                dataGridView1.Rows[_rows].Cells[1].Value =
Program._main.GetNode(x).GetElement().GetType();
                dataGridView1.Rows[_rows].Cells[2].Value = "Дополнительно";
                poisk[_rows++] = Program._main.GetNode(x).GetElement();
                }
            }
        }

        private void MainInterface_Load(object sender, EventArgs e)
        {

        }
    }
```

Форма AddingElement

```
    public partial class AddingElement : Form
    {

        public AddingElement()
        {
            InitializeComponent();
            //Подключение хэш-таблицы
            for (int t = 0; t < Program._main.GetSize(); t++)
            {
                if (Program._main.GetNode(t) != null)
                {
                    if (Program._main.GetNode(t).GetNext() == null)
                    {
                        if
(Program._main.GetNode(t).GetElement().GetType().Equals("Материк"))
                        {

comboBox1.Items.Add(Program._main.GetNode(t).GetElement().GetName());

comboBox7.Items.Add(Program._main.GetNode(t).GetElement().GetName());
                        }
                        else if
(Program._main.GetNode(t).GetElement().GetType().Equals("Страна"))
                        {

comboBox2.Items.Add(Program._main.GetNode(t).GetElement().GetName());

comboBox3.Items.Add(Program._main.GetNode(t).GetElement().GetName());

comboBox4.Items.Add(Program._main.GetNode(t).GetElement().GetName());

comboBox5.Items.Add(Program._main.GetNode(t).GetElement().GetName());
                        }
                        else if
(Program._main.GetNode(t).GetElement().GetType().Equals("Область") ||

Program._main.GetNode(t).GetElement().GetType().Equals("Штат") ||

Program._main.GetNode(t).GetElement().GetType().Equals("Провинция")
                                    )
                        {

comboBox6.Items.Add(Program._main.GetNode(t).GetElement().GetName());
                        }
                    }
                    else
```

```
                    {
                        eNode temp = Program._main.GetNode(t);
                        while (temp != null)
                        {
                            if
(Program._main.GetNode(t).GetElement().GetType().Equals("Материк"))
                            {

comboBox1.Items.Add(Program._main.GetNode(t).GetElement().GetName());

comboBox7.Items.Add(Program._main.GetNode(t).GetElement().GetName());
                            }
                            else if
(Program._main.GetNode(t).GetElement().GetType().Equals("Страна"))
                            {

comboBox2.Items.Add(Program._main.GetNode(t).GetElement().GetName());

comboBox3.Items.Add(Program._main.GetNode(t).GetElement().GetName());

comboBox4.Items.Add(Program._main.GetNode(t).GetElement().GetName());

comboBox5.Items.Add(Program._main.GetNode(t).GetElement().GetName());
                            }
                            else if
(Program._main.GetNode(t).GetElement().GetType().Equals("Область") ||

Program._main.GetNode(t).GetElement().GetType().Equals("Штат") ||

Program._main.GetNode(t).GetElement().GetType().Equals("Провинция")
                                        )
                            {

comboBox6.Items.Add(Program._main.GetNode(t).GetElement().GetName());
                            }
                            temp = temp.GetNext();
                        }
                    }
                }
            }
            //Виды форм правления
            comboBox10.Items.Add("Президентская республика");
            comboBox10.Items.Add("Парламентская республика");
            comboBox10.Items.Add("Смешанная республика");
            comboBox10.Items.Add("Абсолютная монархия");
            comboBox10.Items.Add("Конституционная монархия");
            comboBox10.Items.Add("Дуалистическая монархия");
            comboBox10.Items.Add("Однопартийная система");
        }

        private void button2_Click(object sender, EventArgs e)//Добавление страны
        {
            try
            {
                if (textBox2.TextLength == 0) throw new MyException("Неправильно введено
название!");
                if (Program._main.presenceCM(textBox2.Text.ToString())) throw new
MyException("Страна с данным названием уже добавлена!");
                if (comboBox10.SelectedItem==null) throw new MyException("Не указана
форма правления!");
                if
(((Mainland)Program._main.onefind(comboBox1.SelectedItem.ToString())).GetCitizen() <
```

```
Convert.ToUInt64(textBox8.Text.ToString())) throw new MyException("Население страны не
может быть больше населения материка!");
                if
(((Mainland)Program._main.onefind(comboBox1.SelectedItem.ToString())).GetArea() <
Convert.ToUInt32(textBox7.Text.ToString())) throw new MyException("Площадь страны не
может быть больше площади материка!");
                Program._main.Add(new Country(Convert.ToUInt64(textBox8.Text.ToString()),
Convert.ToUInt32(textBox7.Text.ToString()), textBox2.Text.ToString(),
comboBox10.SelectedItem.ToString(),
(Mainland)Program._main.onefind(comboBox1.SelectedItem.ToString())));
                comboBox2.Items.Add(textBox2.Text.ToString());
                comboBox3.Items.Add(textBox2.Text.ToString());
                comboBox4.Items.Add(textBox2.Text.ToString());
                comboBox5.Items.Add(textBox2.Text.ToString());
                textBox2.Clear();
                textBox8.Clear();
                textBox7.Clear();
                comboBox10.SelectedIndex = -1;
                comboBox1.SelectedIndex = -1;
            }
            catch (System.NullReferenceException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show("Не выбран материк!", "Ошибка", buttons);
            }
            catch (System.FormatException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show("Неправильно введены числовые данные!", "Ошибка",
buttons);
            }
            catch (MyException exp)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show(exp.GetMessage(), "Ошибка", buttons);
            }


        }

        private void button1_Click_1(object sender, EventArgs e)//Добавление материка
        {
            try
            {
                if (textBox1.TextLength == 0) throw new MyException("Неправильно введено
название!");
                if (Program._main.presenceCM(textBox1.Text.ToString())) throw new
MyException("Материк с данным названием уже добавлен!");
                Program._main.Add(new
Mainland(Convert.ToUInt32(textBox4.Text.ToString()),
Convert.ToUInt64(textBox6.Text.ToString()),
((float)Convert.ToUInt64(textBox6.Text.ToString())/Convert.ToUInt32(textBox4.Text.ToStrin
g())), textBox1.Text.ToString()));
                comboBox1.Items.Add(textBox1.Text.ToString());
                comboBox7.Items.Add(textBox1.Text.ToString());
                textBox1.Clear();
                textBox4.Clear();
                textBox6.Clear();
            }
            catch (System.FormatException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
```

```
                MessageBox.Show("Неправильно введены числовые данные!", "Ошибка",
buttons);
            }
            catch (MyException ex)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show(ex.GetMessage(), "Ошибка", buttons);
            }



        }

        private void button3_Click(object sender, EventArgs e)//Добавление области
        {
            try
            {
                if (textBox3.TextLength == 0) throw new MyException("Не введено
название!");
                if
(((Country)Program._main.onefind(comboBox2.SelectedItem.ToString())).GetCitizen() <
Convert.ToUInt32(textBox10.Text.ToString())) throw new MyException("Население области не
может быть больше населения страны!");
                if (Convert.ToUInt32(textBox11.Text.ToString()) >
((Country)Program._main.onefind(comboBox2.SelectedItem.ToString())).GetArea()) throw new
MyException("Площадь области не может быть больше площади страны!");
                Program._main.Add(new Oblast(textBox3.Text.ToString(),
Convert.ToUInt32(textBox10.Text.ToString()), Convert.ToUInt32(textBox11.Text.ToString()),
(Country)Program._main.onefind(comboBox2.SelectedItem.ToString())));
                comboBox6.Items.Add(textBox3.Text.ToString());
                textBox3.Clear();
                textBox10.Clear();
                textBox11.Clear();
                comboBox2.SelectedIndex = -1;
            }
            catch (System.NullReferenceException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show("Не выбрана страна!", "Ошибка", buttons);
            }
            catch (System.FormatException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show("Неправильно введены числовые данные!", "Ошибка",
buttons);
            }
            catch (MyException exp)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show(exp.GetMessage(), "Ошибка", buttons);
            }

        }

        private void button4_Click(object sender, EventArgs e)//Добаление штата
        {
            try
            {
                if (textBox12.TextLength == 0) throw new MyException("Не введено
название!");
                if
(((Country)Program._main.onefind(comboBox3.SelectedItem.ToString())).GetCitizen() <
```

```
Convert.ToUInt32(textBox14.Text.ToString())) throw new MyException("Население штата не
может быть больше населения страны!");
                if (Convert.ToUInt32(textBox13.Text.ToString()) >
((Country)Program._main.onefind(comboBox3.SelectedItem.ToString())).GetArea()) throw new
MyException("Площадь штата не может быть больше площади страны!");
                Program._main.Add(new State(textBox12.Text.ToString(),
Convert.ToUInt32(textBox13.Text.ToString()), Convert.ToUInt32(textBox14.Text.ToString()),
(Country)Program._main.onefind(comboBox3.SelectedItem.ToString())));
                comboBox6.Items.Add(textBox12.Text.ToString());
                textBox12.Clear();
                textBox13.Clear();
                textBox14.Clear();
                comboBox3.SelectedIndex = -1;
            }
            catch (System.NullReferenceException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show("Не выбрана страна!", "Ошибка", buttons);
            }
            catch (System.FormatException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show("Неправильно введены числовые данные!", "Ошибка",
buttons);
            }
            catch (MyException exp)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show(exp.GetMessage(), "Ошибка", buttons);
            }

        }

        private void button5_Click(object sender, EventArgs e)//Добавление провинции
        {
            try
            {
                if (textBox15.TextLength == 0) throw new MyException("Не введено
название!");
                if
(((Country)Program._main.onefind(comboBox4.SelectedItem.ToString())).GetCitizen() <
Convert.ToUInt32(textBox17.Text.ToString())) throw new MyException("Население провинции
не может быть больше населения страны!");
                if (Convert.ToUInt32(textBox16.Text.ToString()) >
((Country)Program._main.onefind(comboBox4.SelectedItem.ToString())).GetArea()) throw new
MyException("Площадь провинции не может быть больше площади страны!");
                Program._main.Add(new Provinces(textBox15.Text.ToString(),
Convert.ToUInt32(textBox17.Text.ToString()), Convert.ToUInt32(textBox16.Text.ToString()),
(Country)Program._main.onefind(comboBox4.SelectedItem.ToString())));
                comboBox6.Items.Add(textBox15.Text.ToString());
                textBox15.Clear();
                textBox16.Clear();
                textBox17.Clear();
                comboBox4.SelectedIndex = -1;
            }
            catch (System.NullReferenceException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show("Не выбрана страна!", "Ошибка", buttons);
            }
            catch (System.FormatException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
```

```
                MessageBox.Show("Неправильно введены числовые данные!", "Ошибка",
buttons);
            }
            catch (MyException exp)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show(exp.GetMessage(), "Ошибка", buttons);
            }

        }

        private void button6_Click(object sender, EventArgs e)
        {
            try
            {
                if (textBox18.TextLength == 0) throw new MyException("Не введено
название!");
                if (Convert.ToByte(textBox19.Text.ToString()) > 90 ||
                    Convert.ToByte(textBox19.Text.ToString()) < 0 ||
                     Convert.ToByte(textBox20.Text.ToString()) > 180 ||
                     Convert.ToByte(textBox20.Text.ToString()) < 0 ||
                     Convert.ToByte(textBox21.Text.ToString()) > 60 ||
                     Convert.ToByte(textBox22.Text.ToString()) > 60 ||
                     Convert.ToByte(textBox23.Text.ToString()) > 60 ||
                     Convert.ToByte(textBox24.Text.ToString()) > 60 ||
                    (
                     Convert.ToByte(textBox19.Text.ToString()) == 90 &&
                      Convert.ToByte(textBox21.Text.ToString()) != 0 &&
                       Convert.ToByte(textBox23.Text.ToString()) != 0
                    )
                    ||
                (
                Convert.ToByte(textBox20.Text.ToString()) == 180 &&
                      Convert.ToByte(textBox22.Text.ToString()) != 0 &&
                       Convert.ToByte(textBox24.Text.ToString()) != 0
                )) throw new MyException("Неправильно введены координаты!");
                if (Program._main.findRegion(comboBox6.SelectedItem.ToString(),
Program._main.findCountry(comboBox5.SelectedItem.ToString(),
(Mainland)Program._main.onefind(comboBox7.SelectedItem.ToString()))).GetCitizen() <
Convert.ToUInt32(textBox25.Text.ToString())) throw new MyException("Население города не
может быть больше населения региона!");
                Program._main.Add(new City(Convert.ToChar(comboBox8.SelectedItem),
Convert.ToByte(textBox19.Text.ToString()), Convert.ToByte(textBox21.Text.ToString()),
Convert.ToByte(textBox23.Text.ToString()),

Convert.ToChar(comboBox9.SelectedItem), Convert.ToByte(textBox20.Text.ToString()),
Convert.ToByte(textBox22.Text.ToString()), Convert.ToByte(textBox24.Text.ToString()),

Convert.ToUInt32(textBox25.Text.ToString()), textBox18.Text.ToString(),
Program._main.findRegion(comboBox6.SelectedItem.ToString(),
Program._main.findCountry(comboBox5.SelectedItem.ToString(),
(Mainland)Program._main.onefind(comboBox7.SelectedItem.ToString()))),

Program._main.findCountry(comboBox5.SelectedItem.ToString(),
(Mainland)Program._main.onefind(comboBox7.SelectedItem.ToString())),
(Mainland)Program._main.onefind(comboBox7.SelectedItem.ToString())));
                if (checkBox1.Checked)
Program._main.findCountry(comboBox5.SelectedItem.ToString(),
(Mainland)Program._main.onefind(comboBox7.SelectedItem.ToString())).SetCapital(Program._m
ain.findCity(textBox18.Text.ToString(),
                    Program._main.findRegion(comboBox6.SelectedItem.ToString(),
Program._main.findCountry(comboBox5.SelectedItem.ToString(),
(Mainland)Program._main.onefind(comboBox7.SelectedItem.ToString()))));
```

```
                if (checkBox2.Checked)
                {
                    if (Program._main.findRegion(comboBox6.SelectedItem.ToString(),
Program._main.findCountry(comboBox5.SelectedItem.ToString(),
(Mainland)Program._main.onefind(comboBox7.SelectedItem.ToString()))).GetType().Equals("Шт
ат"))

                        Program._main.findState(comboBox6.SelectedItem.ToString(),

Program._main.findCountry(comboBox5.SelectedItem.ToString(),

(Mainland)Program._main.onefind(comboBox7.SelectedItem.ToString()))).SetCapital(Program._
main.findCity(textBox18.Text.ToString(),

Program._main.findRegion(comboBox6.SelectedItem.ToString(),

Program._main.findCountry(comboBox5.SelectedItem.ToString(),

(Mainland)Program._main.onefind(comboBox7.SelectedItem.ToString()))));
                    else throw new MyException("Регион не найден, попробуйте снова");
                }
                textBox18.Clear();
                textBox19.Clear();
                textBox20.Clear();
                textBox21.Clear();
                textBox22.Clear();
                textBox23.Clear();
                textBox24.Clear();
                textBox25.Clear();
                comboBox5.SelectedIndex = -1;
                comboBox6.SelectedIndex = -1;
                comboBox7.SelectedIndex = -1;
                comboBox8.SelectedIndex = -1;
                comboBox9.SelectedIndex = -1;

            }
            catch (System.NullReferenceException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show("Не полностью введены данные о территории!", "Ошибка",
buttons);
            }
            catch (System.FormatException)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show("Неправильно введены числовые данные!", "Ошибка",
buttons);
            }
            catch (MyException exp)
            {
                MessageBoxButtons buttons = MessageBoxButtons.OK;
                MessageBox.Show(exp.GetMessage(), "Ошибка", buttons);
            }
        }

        private void tabPage5_Click(object sender, EventArgs e)
        {

        }

        private void tabPage1_Click(object sender, EventArgs e)
        {
```

```
}

private void label21_Click(object sender, EventArgs e)
{

}

private void AddingElement_Load(object sender, EventArgs e)
{

}

private void textBox23_TextChanged(object sender, EventArgs e)
{

}

private void textBox22_TextChanged(object sender, EventArgs e)
{

}

private void label24_Click(object sender, EventArgs e)
{

}

private void label23_Click(object sender, EventArgs e)
{

}

private void textBox18_TextChanged(object sender, EventArgs e)
{

}

private void label36_Click(object sender, EventArgs e)
{

}

private void textBox25_TextChanged(object sender, EventArgs e)
{

}

private void label26_Click(object sender, EventArgs e)
{

}

private void comboBox7_SelectedIndexChanged(object sender, EventArgs e)
{

}

private void label25_Click(object sender, EventArgs e)
{

}

private void comboBox6_SelectedIndexChanged(object sender, EventArgs e)
```

```
{

}

private void checkBox2_CheckedChanged(object sender, EventArgs e)
{

}

private void label27_Click(object sender, EventArgs e)
{

}

private void textBox19_TextChanged(object sender, EventArgs e)
{

}

private void textBox20_TextChanged(object sender, EventArgs e)
{

}

private void label31_Click(object sender, EventArgs e)
{

}

private void textBox21_TextChanged(object sender, EventArgs e)
{

}

private void label30_Click(object sender, EventArgs e)
{

}

private void label32_Click(object sender, EventArgs e)
{

}

private void label33_Click(object sender, EventArgs e)
{

}

private void textBox24_TextChanged(object sender, EventArgs e)
{

}

private void label34_Click(object sender, EventArgs e)
{

}

private void label35_Click(object sender, EventArgs e)
{

}
```

```
    private void comboBox8_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void comboBox9_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void label37_Click(object sender, EventArgs e)
    {

    }

    private void tabPage4_Click(object sender, EventArgs e)
    {

    }
}
```