ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ADO – Объекты данных ActiveX (от англ. ActiveX Data Objects)

DES – Стандарт шифрования данных (англ. Data Encryption Standard)

IDE - Интегрированная среда разработки (англ. Integrated Development Environment)

MS - Microsoft

WS – WorldSkills

БД – База данных

МДК – Междисциплинарный курс

ОК – Общая компетенция

ООП – Объектно-ориентированное программирование

ОС – Операционная система

ПК – Профессиональная компетенция

ПМ – Профессиональный модуль

ПО – Программное обеспечение

СПО – Среднее профессиональное обучение

УГС – Укрупнённая группа специальностей

ФГОС – Федеральные государственный образовательный стандарт

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В связи с развитием информационных технологий крайне востребованными являются высококвалифицированные специалисты, разрабатывающие программные приложения любой сложности. Первоначальная подготовка специалистов в области разработки продуктов информационных технологий осуществляется, в том числе, на базе СПО.

Для проведения лабораторных занятий по программированию, которые, собственно, и позволяют практически овладеть азами программирования, требуются задачи и задания по каждому изучаемому разделу, желательно индивидуальные. В основной части литературы для среднего профессионального образования, посвященной визуальному программированию, присутствует существенный недостаток: немногочисленны практические задания, позволяющие читателю в полной мере освоить объектно-ориентированное программирование. Поэтому возникает проблема в дидактическом наполнении дисциплин, посвященных изучению программирования, в частности, визуального. Для изучения визуального программирования написано много различных пособий и учебников, позволяющих изучить теоретическую часть среды разработки, но изучение программирования не останавливается только на теоретической части, необходимо также овладеть практическими навыками.

Таким образом, существует проблема в дидактическом наполнении дисциплин среднего профессионального образования, связанных с визуальным программированием. По этой причине выполнение настоящей работы представляется *актуальным*.

Указанная проблема определила *объект* исследования: процесс обучения программированию, и *предмет*: дидактическое обеспечение визуального программирования. Была поставлена следующая *цель*: разработать учебно-методический комплект для изучения расширенных возможностей

программирования в визуальных средах, предназначенный для обеспечения преподавания программирования.

Исходя из цели исследования, выделены следующие *задачи*, определившие структуру работы:

- проанализировать государственные стандарты среднего профессионального образования;
- проанализировать учебную и научную литературу по обучению визуальному и объектно-ориентированному программированию;
- разработать комплект дидактических материалов в поддержку курса ООП;
- разработать методику применения комплекта дидактических материалов.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, библиографического списка и приложений.

Глава 1. Объектно-ориентированный и визуальный подход к программированию (обзор)

1.1. История объектно-ориентированного программирования

В 1967 г. ведущие специалисты в области программирования выдвинули идею преобразования постулата фон Неймана: "данные и программы неразличимы в памяти машины". Их цель заключалась в максимальном сближении данных и кода программы. Решая поставленную задачу, они столкнулись с задачей, решить которую без декомпозиции оказалось невозможно, а традиционные структурные декомпозиции не сильно упрощали задачу. Усилия многих программистов и системных аналитиков, направленные на формализацию подхода, увенчались успехом. [5]

Были разработаны три основополагающих принципа того, что потом стало называться объектно-ориентированным программированием: наследование, инкапсуляция, полиморфизм.

Результатом их первого применения стал язык Simula-1, в котором был введен новый тип – объект. В описании этого типа одновременно указывались данные (поля) и процедуры, их обрабатывающие – методы. Родственные объекты объединялись в классы, описания которых оформлялись в виде блоков программы. При этом класс можно использовать в качестве префикса к другим классам, которые становятся в этом случае подклассами первого. Впоследствии Симула-1 был обобщен, и появился первый универсальный объектно-ориентированный язык программирования – Симула-67 (67 – по году создания).

Взгляд на программирование «под новым углом» (отличным от процедурного) предложили Алан Кэй и Дэн Ингаллс в языке Smalltalk. Здесь понятие класса стало основообразующей идеей для всех остальных конструкций языка. Именно он стал широко распространённым объектно-ориентированным языком программирования. [5]

Как выяснилось, ООП оказались пригодными не только для моделирования (Simula) и разработки графических приложений (SmallTalk),

но и для создания большинства других приложений, а их приближенность к человеческому мышлению и возможность многократного использования кода сделали их наиболее используемыми в программировании.

В 1974 г. Марвин Мински, основоположник теории искусственного интеллекта, предложил идею фрейма, отделившего описание класса (структуры) объекта от его конкретного представления (экземпляра) и быстро завоевавшего популярность в языках искусственного интеллекта. Фрейм стал прямым предшественником современного понятия объектов в C++.

В 1976 г. Кринстен Нюгорн создал новый язык BETA, в котором ввел концепцию шаблонов – более высокого уровня абстракций, нежели объекты.

В 1980 г. Бьерн Страуструп, дополнил язык C концепцией классов, основанной на фреймах и объектных механизмах Симулы.

В 1982 г. В Мехико прошла 8-я Международная конференция по сверхбольшим БД (VLDB), где была предложена концепция объектно-ориентированных БД и рассматривались вопросы расширения существовавших языков запросов к БД новыми, объектными типами данных.

В 1983 г. Страуструп дал своему творению окончательное название Си++.

В 1986 г. Первая всемирная конференция по объектно-ориентированным системам программирования прошла в Портленде. Возможно, именно прозвучавшие на ней доклады оказали стимулирующее влияние на Уильяма Аткинсона, инженера Apple, который через год после этого спроектировал систему HyperCard, прообраз современных визуальных сред быстрой разработки. Эффективность новой технологии оказалась столь высокой, что уже в 1989 г. одиннадцать компаний, среди которых были 3Com, American Airlines, Canon, Data General, Hewlett-Packard, Philips Telecommunications, Sun Microsystems и Unisys, основали группу OMG (Object Management Group), призванную формировать индустриальные стандарты на объектное программирование и упрощать интеграцию приложений с помощью универсальных кроссплатформенных технологий. Эта группа первым делом приступила к выработке единого стандарта компонентной модели CORBA

(Common Object Request Broker Architecture) – набора спецификаций, определяющих способы объектно-ориентированного взаимодействия компонентов промежуточного уровня в различных средах без привязки к конкретным языкам программирования. С самого начала CORBA нацеливалась на поддержку крупных, индустриальных проектов, и этот подход со временем себя полностью оправдал. Сегодня нет другого столь же распространенного независимого стандарта, поддерживающего самые разные ОС и объектные модели. [24]

В 1992 г. Вышел стандарт CORBA 1.0, определяющий ключевые аспекты функционирования CORBA-систем. В него были включены базовое описание объектной модели, наборы программных интерфейсов поддержки CORBA-систем, а также декларативный язык определения интерфейсов Interface Definition Language (IDL), созданный OMG для описания распределенных интерфейсов.

В 1993 г. Корпорация Microsoft выпустила первую версию компонентной модели COM (Component Object Model). Первоначально COM готовилась только для поддержки технологии встраивания и связывания документов OLE (Object Linking and Embedding – объектное связывание и встраивание), но быстро выделилась в самостоятельное направление. [24]

В том же 1993-м была предложена компонентная технология Microsoft ActiveX, основанная на элементах управления OLE, пришедших из Visual Basic, где они назывались VBX/OCX.

В 1994 г. Опубликован стандарт CORBA 2.0, который быстро получил массовое признание, так как представлял собой богатый и глубоко проработанный набор документов и охватывал большинство востребованных рынком задач. В нем были ликвидированы недостатки прежней версии, в результате чего CORBA 2.0 начал поддерживать транзакции и понимать универсальную кодировку Unicode, а также появился набор средств обеспечения безопасности и взаимодействия COM- и CORBA-объектов.

Гради Буч и Джеймс Румбах из Rational Software решили объединить две методологии визуального моделирования Booch и OMT (Object Modeling Technique) и создать на их основе новый язык UML (Unified Modeling Language).

В 1995 г. Sun Microsystems свободно распространяет в Интернете элемент технологии Java – среду HotJava, поддерживающую мобильный код, разработку проекта Green, которая к тому времени считалась в Sun практически пропащей, если бы не развитие Сети. Новинку сразу же лицензирует Netscape Communication, а следом за ней к Java проявляют коммерческий интерес десятки компаний, в том числе Microsoft, IBM, Adobe, Borland, Lotus, Oracle.

Корпорация Borland выпустила первую версию среды быстрой визуальной разработки Delphi 1, основанную на концепции библиотек стандартных компонентов.

Microsoft сообщает о новой технологии DCOM – распределенной версии COM, позволившей собирать приложения из компонентов, выполнявшихся на разных компьютерах. Первоначально эта технология называлась Network OLE (сетевое OLE), однако по мере выделения COM в самостоятельное направление решено было отказаться от упоминания OLE в ее названии.

В 1996 г. Microsoft называет ActiveX новой объектной стратегией, направленной на поддержку Интернета.

В 1997 г. Sun Microsystems предлагает концепцию Enterprise JavaBeans – технологию создания корпоративных Java-компонентов, которые можно исполнять на серверах приложений, реализуя логику крупных, хорошо масштабируемых и защищенных систем на платформенно-независимой основе.

Эксперты OMG осознают не только важность объектных технологий программирования, но и острую потребность в универсальных методологических концепциях проектирования крупных систем. Секретом стабильности системы и высокой отдачи инвестиций специалисты OMG

называют независимую UML-модель и приступают к созданию концепции "Архитектура, управляемая моделью" (Model Driven Architecture, MDA). В ее основу закладывается базовая платформенно-независимая UML-модель системы, несколько платформенно-зависимых моделей и коллекция определений программных интерфейсов. Первую реализацию этой универсальной концепции (так называемое "отображение в объектный стандарт") OMG выполнила для CORBA. [24]

В 2000 г. Microsoft анонсирует новую объектную платформу .NET и новый язык программирования C#, сочетающий лучшие свойства C++ и Java. Он был предложен Microsoft во многом в противовес Java. [13]

В 2001 г. OMG выпускает спецификацию CORBA 3.0. Она дополнена возможностями асинхронного обмена сообщениями, разработки систем реального времени и создания встраиваемых систем. В ней появились подключаемые компоненты, поддержка XML и средства интеграции различных Интернет-технологий. Была продумана модель сборки системы из компонентов JavaBeans и ActiveX. Стало допустимым в рамках одного компонента описывать множество интерфейсов, а также использовать язык сценариев. Особый акцент в третьей версии CORBA сделан на эффективном взаимодействии с Java.

На рис.1.1 представлена генеалогия наиболее значимых и распространенных объектно-ориентированных языков программирования, где интенсивность разработки языка показана длиной прямоугольника, а стрелки отображают влияние одних языков на другие.

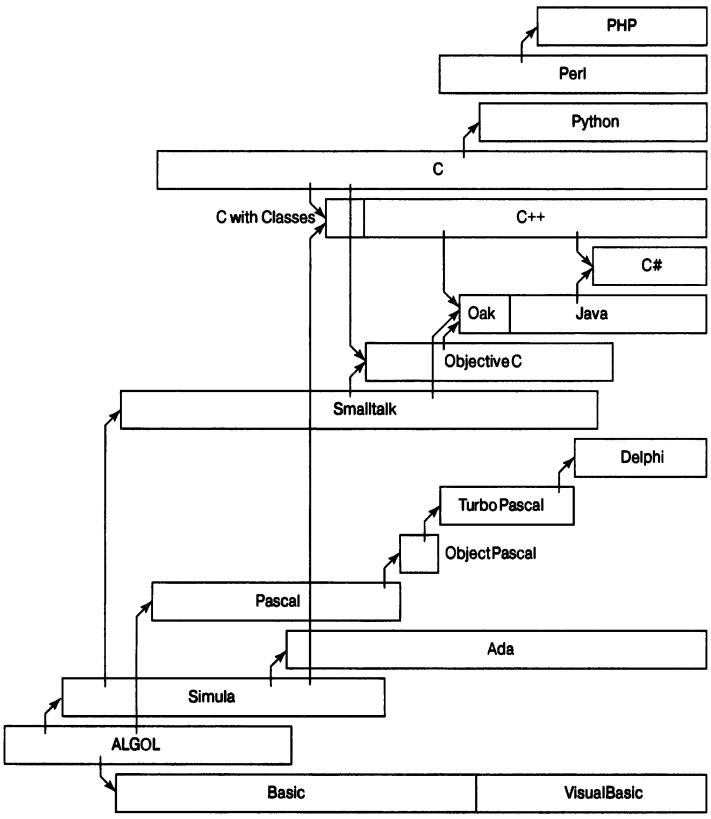


Рис.1.1. Генеалогия языков программирования [5]

Несмотря на многомиллионные вложения, сделанные в 1970-1980 годах коммерческими компаниями и государственными структурами разных стран в универсальные языки программирования (такие, как Алгол, PL/1, C) и языки логического программирования (прежде всего Prolog), самой распространенной в мире программной технологией остается ООП. Наиболее известным событием нового тысячелетия в этой сфере стал быстро набравший популярность язык программирования C#. Его можно смело считать лучшим на сегодня объектно-ориентированным средством создания графических приложений.

В ближайшее десятилетие развитие ООП будет проходить под влиянием трех концепций: Microsoft .NET (прежде всего .NET Framework и ее подмножества и соответственно реализации C#), Java (все входящие в это понятие технологии) и CORBA. Причем важнейшей особенностью CORBA останется независимость от ОС и языка программирования CORBA-компонентов. [5]

1.2. Обзор языков объектно-ориентированного программирования

Объектно-ориентированные языки программирования пользуются в последнее время большой популярностью среди программистов, так как они позволяют использовать преимущества объектно-ориентированного подхода не только на этапах проектирования и конструирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

Объектно-ориентированное программирование – это представление программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования. Объектно-ориентированное программирование сводится к выстраиванию иерархий классов, описаний объектов и их взаимодействий, программной реализации различных действий над объектами. [8]

Все языки ООП, основаны на трёх основополагающих концепциях, называемых инкапсуляцией, полиморфизмом и наследованием. Рассмотрим эти концепции.

1. *Инкапсуляция* — это механизм программирования, объединяющий вместе код и данные, которыми он манипулирует, исключая как вмешательство извне, так и неправильное использование данных. В объектно-ориентированном языке данные и код могут быть объединены в совершенно автономный «черный ящик». Внутри такого «ящика» находятся все необходимые данные и код. Когда код и данные связываются вместе подобным образом, создается объект. Иными словами, объект — это элемент, поддерживающий инкапсуляцию. [10]

В объекте код, данные или же и то, и другое могут быть закрытыми или же открытыми. Закрытые данные или код известны и доступны только остальной части объекта. Это означает, что закрытые данные или код недоступны части программы, находящейся за пределами объекта. Если же данные или код оказываются открытыми, то они доступны другим частям программы, хотя и определены внутри объекта. Как правило, открытые части

объекта служат для организации управляемого интерфейса с закрытыми частями.

Основной единицей инкапсуляции является класс, который определяет форму объекта. Он описывает данные, а также код, который будет ими оперировать. Описание класса служит для построения объектов, которые являются экземплярами класса. Следовательно, класс, по существу, представляет собой ряд схематических описаний способа построения объекта.

Код и данные, составляющие вместе класс, называют членами. Данные, определяемые классом, называют полями, или переменными экземпляра. А код, оперирующий данными, содержится в функциях-членах, самым типичным представителем которых является метод. Метод служит в качестве аналога подпрограммы. (К числу других функций-членов относятся свойства, события и конструкторы.) Таким образом, методы класса содержат код, воздействующий на поля, определяемые этим классом.

2. *Наследование* подразумевает то, что дочерний класс содержит все атрибуты родительского класса, при этом некоторые из них могут быть переопределены или добавлены в дочернем. [8]

Наследование представляет собой процесс, в ходе которого один объект приобретает свойства другого объекта. Это очень важный процесс, поскольку он обеспечивает принцип иерархической классификации.

Если не пользоваться иерархиями, то для каждого объекта пришлось бы явно определять все его свойства. А если воспользоваться наследованием, то достаточно определить лишь те свойства, которые делают объект особенным в его классе. Он может также наследовать общие свойства своего родителя. Следовательно, благодаря механизму наследования один объект становится отдельным экземпляром более общего класса.

Класс может наследовать функциональность от нескольких классов. Это называется множественным наследованием. Множественное наследование – это, когда класс наследуется от нескольких классов-посредников, которые в свою очередь наследуются от одного класса.

3. *Полиморфизм* – возможность объектов с одинаковой спецификацией иметь различную реализацию. Например, можно сложить два числа, и можно сложить две строки. При этом получим разный результат, так как числа и строки являются разными классами. [8]

Полиморфизм реализуется с помощью наследования классов и виртуальных функций. Класс-потомок наследует сигнатуры методов класса-родителя, а реализация, в результате переопределения метода, этих методов может быть другой, соответствующей специфике класса-потомка. Другие функции могут работать с объектом класса-родителя, но при этом вместо него во время исполнения будет подставляться один из классов-потомков. Это называется поздним связыванием.

Класс-потомок сам может быть родителем. Это позволяет строить сложные схемы наследования – древовидные или сетевидные.

Рассмотрим для примера стек, т.е. область памяти, функционирующую по принципу "последним пришел — первым обслужен". Допустим, что в программе требуются три разных типа стеков: один — для целых значений, другой — для значений с плавающей точкой, третий — для символьных значений. В данном примере алгоритм, реализующий все эти стеки, остается неизменным, несмотря на то, что в них сохраняются разнотипные данные. В языке, не являющемся объектно-ориентированным, для этой цели пришлось бы создать три разных набора стековых подпрограмм с разными именами. Но благодаря полиморфизму для реализации всех трех типов стеков достаточно создать лишь один общий набор подпрограмм. Зная, как пользоваться одним стеком, вы сумеете воспользоваться и остальными.

В более общем смысле понятие полиморфизма нередко выражается следующим образом: "один интерфейс — множество методов". Это означает, что для группы взаимосвязанных действий можно разработать общий интерфейс. Полиморфизм помогает упростить программу, позволяя использовать один и тот же интерфейс для описания общего класса действий. Выбрать конкретное действие (т.е. метод) в каждом отдельном случае — это

задача компилятора. Программисту не нужно делать это самому. Ему достаточно запомнить и правильно использовать общий интерфейс.

Объектно-ориентированные языки программирования

В настоящее время существует множество языков программирования. Однако среди этого многообразия присутствуют самые распространённые и востребованные языки программирования.

1) Delphi – интегрированная среда разработки компании Borland. Она предназначена для разработки приложений на языке программирования, названном Object Pascal, который в 2003г. получил такое же название, как и среда, т. е. Delphi. Назначение Delphi - быстрая разработка приложений. С ее помощью можно быстро и качественно создавать любые программы. Не секрет, что лучшим языком для изучения и освоения программирования является Паскаль, а лучшей системой программирования для MS-DOS - Turbo Pascal. Delphi продолжает серию Паскаль-ориентированных средств программирования и является наиболее удобным инструментом для Windows-программирования.

2) C++ – это язык программирования, который представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений.

C++ является мощным языком, унаследовав богатые возможности по работе с памятью. Поэтому нередко C++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. Только системным программированием применение данного языка не ограничивается. C++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где C++ тоже

нашел свое применение. И даже в веб-разработке также можно использовать C++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения. C++ - это язык широкого пользования, на котором можно создавать практически любые виды программ.

3) C# и связанную с ним среду .NET Framework можно без преувеличения назвать самой значительной из предлагаемых в настоящее время технологий для разработчиков. Среда .NET является такой средой, которая была создана для того, чтобы в ней можно было разрабатывать практически любое приложение для запуска в Windows, а C# является языком программирования, который был специально создан для использования в .NET Framework. Например, с применением C# и .NET Framework можно создавать динамические веб-страницы, приложения Windows Presentation Foundation, веб-службы XML, компоненты для распределенных приложений, компоненты для доступа к базам данных, классические настольные приложения Windows и даже клиентские приложения нового интеллектуального типа, обладающие возможностями для работы в оперативном и автономном режимах. [18]

4) Java представляет собой язык программирования и платформу вычислений, которая была впервые выпущена Sun Microsystems в 1995 г. Существует множество приложений и веб-сайтов, которые не работают при отсутствии установленной Java, и с каждым днем число таких веб-сайтов и приложений увеличивается. Java отличается быстротой, высоким уровнем защиты и надежностью.

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем этот байт-код выполняется виртуальной машиной JVM (Java Virtual Machine). В этом плане Java отличается от стандартных интерпретируемых языков как PHP или Perl, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как C или C++.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные

программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Mac OS и т.д.

Еще одной ключевой особенностью Java является то, что он поддерживает автоматическую сборку мусора. А это значит, что не надо освобождать вручную память от ранее использовавшихся объектов, как в C++, так как сборщик мусора это сделает автоматически.

Язык поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

5) JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Язык обычно используется как встраиваемый язык для программного доступа к объектам приложений.

JavaScript — это относительно простой объектно-ориентированный язык, предназначенный для создания небольших клиентских и серверных приложений для Internet. Программы, написанные на языке JavaScript, включаются в состав HTML-документов и распространяются вместе с ними. Программы просмотра (браузеры) типа Google Chrome и Internet Explorer распознают встроенные в текст документа программы-вставки (script-коды) и выполняют их. Таким образом, JavaScript — интерпретируемый язык программирования. Примерами программ на JavaScript могут служить программы, проверяющие введенные пользователем данные или выполняющие какие-то действия при открытии или закрытии документа. Такие программы могут реагировать на действия пользователя — нажатие кнопок "мыши", ввод данных в экранной форме или перемещение "мыши" по странице. Более того, JavaScript-программы могут управлять самим браузером и атрибутами документа.

6) Python – высокоуровневый язык программирования, который предназначен для создания приложений различных типов. Это и веб-

приложения, и игры, и настольные программы, и работа с базами данных. Довольно большое распространение язык получил в области машинного обучения и исследований искусственного интеллекта.

Python интерпретируемый, объектно-ориентированный высокоуровневый язык программирования с динамической семантикой. Встроенные высокоуровневые структуры данных в сочетании с динамической типизацией и связыванием делают язык привлекательным для быстрой разработки приложений. Кроме того, его можно использовать в качестве сценарного языка для связи программных компонентов. Синтаксис Python прост в изучении, в нем придается особое значение читаемости кода, а это сокращает затраты на сопровождение программных продуктов. Python поддерживает модули и пакеты, поощряя модульность и повторное использование кода. Интерпретатор Python и большая стандартная библиотека доступны бесплатно в виде исходных и исполняемых кодов для всех основных платформ и могут свободно распространяться.

7) Swift новый язык программирования, разработанный Apple для создания приложений iOS и OS X. Он построен на основе C и Objective-C, но более простой и гибкий. Кроме того, создатели Swift позаимствовали лучшие элементы языков программирования JavaScript и Python.

Особенностью языка является возможность писать код и видеть результаты в режиме реального времени. До этого на протяжении долгого времени процесс создания продукта и сам продукт были разделены, из-за этого программисты должны были тратить много времени на проверку и оптимизацию кода. В Swift они могут вносить поправки и сразу видеть результат. Это значит, что разработчики смогут быстрее проверять в деле свои концепты и в целом быстрее создавать приложения.

Swift – мощный и интуитивно понятный язык программирования для, но отличается от Objective-C более широким набором возможностей, которые ограничивались связью с C. Кроме того, Swift можно назвать безопасным, поскольку язык имеет строгую типизацию: в любой момент времени известно

с объектом какого типа ведется работа. Главным его преимуществом является скорость: как в работе над проектом, так и при запуске готового приложения. Именно за это преимущество перед другими языками (его и назвали Swift (в переводе — быстрый).

Сравнение C# с другими языками программирования

C# являясь одним из широко распространенных языков программирования, должен впитать в себя весь имеющийся опыт и вобрать лучшие стороны существующих языков программирования, при этом являясь специально созданным для работы в .NET. Сама архитектура .NET продиктовала ему (как и многим другим языкам, на которых можно писать под .NET) объектно-ориентированную направленность. [22]

Свой синтаксис C# во многом унаследовал от C++ и Java. Разработчики, имеющие опыт написания приложений на этих языках, найдут в C# много знакомых черт. Но вместе с тем он является во многом новаторским - атрибуты, делегаты и события, прекрасно вписаны в общую идеологию языка. Их введение позволило применять принципиально новые приемы программирования.

Самым распространенным объектом для сравнения с C# является Java. Также разработанный для работы в виртуальной среде выполнения, имеющей объектно-ориентированную архитектуру и сборщик мусора, основанный на механизме ссылок. При сравнении с этим языком сразу выделаются такие особенности, как возможность объявлять несколько классов в одном файле, из чего следует синтаксическая поддержка иерархической системы пространств имен. Из реализации ООП-концепций сходство в механизме наследования и реализации (и в Java и в C# возможно единичное наследование). Но в Java отсутствуют свойства и индексаторы. Также есть возможность перечисления контейнеров. [25]

Из вещей, включенных в спецификацию языка, необходимо отметить возможность использование комментариев в формате XML. Если

комментарии отвечают специально описанной структуре, компилятор по ним может сгенерировать единый XML-файл документации.

C# внес и свои уникальные черты, которые уже были упомянуты - это события, индексаторы, атрибуты и делегаты. Они представляют собой полезные возможности, которые не останутся невостребованными.

1.3. Визуальное программирование

Визуальное программирование – это технология программирования, предусматривающая создание приложений с помощью наглядных средств.

К визуальному программированию можно отнести также Rapid Application Development (RAD) – быструю разработку программ. RAD – технология программирования, обеспечивающая ускоренную разработку и модификацию приложений за счет использования объектно-ориентированного и визуального программирования.

Средствами визуального программирования обычно решают задачи построения пользовательского интерфейса и упрощения разработки приложения путем замены метода «написания программы» на метод конструирования.

Визуальное программирование, бесспорно, обладает достоинством наглядного представления информации и гораздо лучше соответствует природе человеческого восприятия, чем методы традиционного, текстового программирования. Однако практически все визуальные средства нуждаются в дополнении функциями, которые не могут быть представлены в виде графических конструкций и требуют текстового выражения. Визуальные средства дополняются специальными программами – «скриптами», написанными на различных языках программирования.

Концепция визуального программирования реализована во многих современных средах разработки программных систем. Все ведущие фирмы, создающие средства для программирования и конструирования имеют системы, поддерживающие технологию визуального программирования.

Фирма Microsoft, разрабатывая концепцию .NET Framework, создала Visual Studio .NET Enterprise Architect 2003, в которой реализовала все последние достижения в области программирования и в частности, технологии визуального программирования. [18]

Visual Studio .NET - полная многоязычная среда разработки для платформы Microsoft .NET. Visual Studio .NET предоставляет набор технологий, упрощающих создание, развертывание и последующее усовершенствование безопасных, масштабируемых и высокодоступных веб-приложений и веб-служб XML.

Фирма Borland Software Corporation традиционно развивает интегрированную среду программирования, имеющую несомненный успех на рынке профессиональных программных продуктов. Эта среда всегда поддерживала методы визуального программирования, предлагала разработчику мощные библиотеки, интерактивные мастера-построители, готовые элементы графического интерфейса.

Основы визуального программирования

Визуализация – это процесс графического отображения сложных процессов или понятий на экране компьютера в виде графических примитивов. Визуализировать можно многие процессы: управления, построения, рисования и т.д.

Пользователи приложений привыкли к графическому интерфейсу приложений и зачастую даже не задумываются, что привычные элементы интерфейса представляют собой визуальные графические примитивы. Например, простейший вариант визуализации – линейка прогресса (прямоугольник, процент заполнения которого прямо пропорционален времени выполнения какой-либо операции). Глядя на нее, вы можете приблизительно оценить время окончания операции. Но, если бы было выведено значение времени исполнения в виде числа или процента без линейки прогресса, то такой вывод явился бы лишь отображением текущего значения, но не визуализацией процесса.

Сегодня принято визуализировать интерфейсы программного обеспечения. Визуализация снимает проблемы "общения" пользователем с программным продуктом. Графические изображения на элементах управления позволяют пользователю интуитивно разбираться в назначении этих элементов.

Для визуализации интерфейсов программного обеспечения существует целый ряд специально разработанных элементов интерфейса – визуальный компонент, позволяющих отображать различную информацию и осуществлять управление программой в целом. Простейший пример - визуальная кнопка на экране компьютера. Программная кнопка имитирует поведение обычной кнопки на пульте управления любого прибора. Кнопку можно "нажимать" как настоящую.

В интегрированной среде разработки выполняется визуальное построение интерфейса программы, но не самого кода. Определяющими элементами процесса визуализации являются:

- визуализируемая модель – модель, которая подвергается отображению с целью возможности изменения ее структуры или ее параметров (либо параметров ее отдельных частей);
- панель инструментов (ToolBox) – окно, содержащее набор элементов, из которых строится визуальная модель. Обычно элементы разделяются по их назначению на отдельные группы, размещающиеся на отдельных закладках окна инструментов;
- окно свойств (PropertyBox) – окно, в котором отображаются параметры (свойства) выбранного элемента визуальной модели. Термин "свойство" пришел из объектно-ориентированного программирования и обозначает параметр объекта (элемента).

Визуализируемой моделью является окно (форма, диалог) Windows, а не код программы.

Обычной практикой является визуализация работы с элементами интерфейса, когда в качестве объектов визуализации рассматриваются визуальные компоненты, из которых состоят формы (окна и диалоги) интерфейса программы. Но и операторы программы можно рассматривать как объекты визуализации. В этом случае параметры операторов и функций программы можно настраивать при помощи окна свойств (PropertyBox), а сами операторы и функции хранится в списковой форме (табличной форме).

Интегрированные среды разработки программ

Интегрированная среда разработки — система программных средств, используемая программистами для разработки программного обеспечения (ПО).

Обычно среда разработки включает в себя:

- текстовый редактор;
- компилятор и / или интерпретатор;
- средства автоматизации сборки;
- отладчик.

Содержит средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. Многие современные среды разработки также включают браузер классов, инспектор объектов и диаграмму иерархии классов — для использования при объектно-ориентированной разработке ПО. Хотя и существуют IDE, предназначенные для нескольких языков программирования — такие, как Eclipse, NetBeans, Embarcadero RAD Studio, Qt Creator или Microsoft Visual Studio, но обычно IDE предназначается для одного определённого языка программирования - как, например, Visual Basic, PureBasic, Delphi, Dev-C++. [18]

Интегрированные среды разработки были созданы для того, чтобы максимизировать производительность программиста благодаря тесно связанным компонентам с простыми пользовательскими интерфейсами. Это позволит разработчику делать меньше действий для переключения различных режимов, в отличие от дискретных программ разработки. Однако, так как IDE является сложным программным комплексом, то лишь после долгого процесса обучения среда разработки сможет качественно ускорить процесс разработки ПО.

Обычно IDE ориентирована на определенный язык программирования, предоставляя набор функций, который наиболее близко соответствует парадигмам этого языка программирования. Однако, есть некоторые IDE с поддержкой нескольких языков, такие как Eclipse, ActiveState Komodo, последние версии NetBeans, Microsoft Visual Studio, WinDev и Xcode. [11]

IDE представляет из себя единственную программу, в которой проводилась вся разработка. Она обычно содержит много функций для создания, изменения, компилирования, развертывания и отладки программного обеспечения. Цель среды разработки заключается в том, чтобы абстрагировать конфигурацию, необходимую, чтобы объединить утилиты командной строки в одном модуле, который позволит уменьшить время, чтобы изучить язык, и повысить производительность разработчика. Также считается, что трудная интеграция задач разработки может далее повысить производительность. Например, среда позволяет проанализировать код и тем самым обеспечить мгновенную обратную связь и уведомить о синтаксических ошибках.

Интегрированная среда разработки - это совокупность программных средств, поддерживающая все этапы разработки программного обеспечения от написания исходного текста программы до ее компиляции и отладки, и обеспечивающая простое и быстрое взаимодействие с другими инструментальными средствами (программным отладчиком-симулятором, внутрисхемным эмулятором и программатором).

Работа в интегрированной среде дает программисту:

- возможность использования встроенного многофайлового текстового редактора, специально ориентированного на работу с исходными текстами программ;
- иметь автоматическую диагностику выявленных при компиляции ошибок, когда исходный текст программы, доступный редактированию, выводится одновременно с диагностикой в многооконном режиме;
- возможность параллельной работы над несколькими проектами. Менеджер проектов позволяет использовать любой проект в качестве шаблона для вновь создаваемого проекта;
- минимум перекомпиляции. Ей подвергаются только редактировавшиеся модули;
- возможность загрузки отлаживаемой программы в имеющиеся средства отладки, и возможность работы с ними без выхода из оболочки;
- возможность подключения к оболочке практически любых программных средств.

В последнее время, функции интегрированных сред разработки становятся стандартной принадлежностью программных интерфейсов эмуляторов и отладчиков-симуляторов.

Подобные функциональные возможности, в сочетании с дружественным интерфейсом, в состоянии существенно увеличить скорость разработки программ, особенно для микроконтроллеров и процессоров цифровой обработки сигналов, являющихся очень трудоемкими и труднообозримыми процессами.

1.4. Реализация объектно-ориентированного подхода к программированию в C#

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ–отрасли. В настоящий момент на нем создаются самые различные приложения: от небольших программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

Основным понятием C# является объектно-ориентированное программирование. Методика ООП неотделима от C#, и поэтому все программы на C# являются объектно-ориентированными хотя бы в самой малой степени. В связи с этим очень важно и полезно усвоить основополагающие принципы ООП, прежде чем приступать к написанию самой простой программы на C#.

Платформа .NET Framework

Несмотря на то что C# является самодостаточным языком программирования, у него имеется особая взаимосвязь со средой выполнения .NET Framework. Наличие такой взаимосвязи объясняется двумя причинами. Во-первых, C# первоначально предназначался для создания кода, который должен выполняться в среде .NET Framework. И во-вторых, используемые в C# библиотеки определены в среде .NET Framework. На практике это означает, что C# и .NET Framework тесно связаны друг с другом, хотя теоретически C# можно отделить от среды .NET Framework. В связи с этим очень важно иметь хотя бы самое общее представление о среде .NET Framework и ее значении для C#. [22]

Назначение .NET Framework — служить средой для поддержки разработки и выполнения сильно распределенных компонентных приложений. Можно выделить следующие ее основные черты:

- поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает

несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках;

- кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Framework поддерживается на большинстве современных ОС Windows (Windows 10/8.1/8/7/Vista). А также можно создавать приложения, которые будут работать и на других ОС, например, семейства Linux или на мобильных платформах Android и iOS;
- .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение не было написано на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

Разнообразие технологий, общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF. Для создания веб-сайтов - ASP.NET и т.д.

Также еще следует отметить такую особенность языка C# и .NET Framework, как автоматическая сборка мусора (служба, которая автоматически высвобождает неиспользуемую память).

1.5. Интегрированная среда разработки Visual Studio

Visual Studio представляет собой интегрированную среду разработки программного обеспечения от компании Microsoft. С помощью нее можно создавать приложения для Windows, iOS, Android и других платформ. В Visual Studio включены инструменты не только для создания desktop приложений, но и web, мобильные и облачные инструменты разработки. Она позволяет быстро и эффективно писать код, не теряя из виду контекст текущего файла. Можно легко углубиться в подробности, такие как структура вызова, связанные функции, возвраты и состояние тестирования. Также доступны перепроектирование (рефакторинг) кода, нахождение и устранение ошибок в коде. [12]

Интерфейс Visual Studio

После открытия Visual Studio можно увидеть три основные части интегрированной среды разработки: окна инструментов, меню с панелями инструментов и область главного окна (рис.1.2). Окна инструментов закреплены в левой и правой частях окна приложения, а панель быстрый запуск, строка меню и стандартная панель инструментов закреплены в его верхней части. В центре окна приложения находится Начальная страница. При открытии решения или проекта редакторы и конструкторы отображаются в этом пространстве. При разработке приложения чаще всего используется именно эта область.

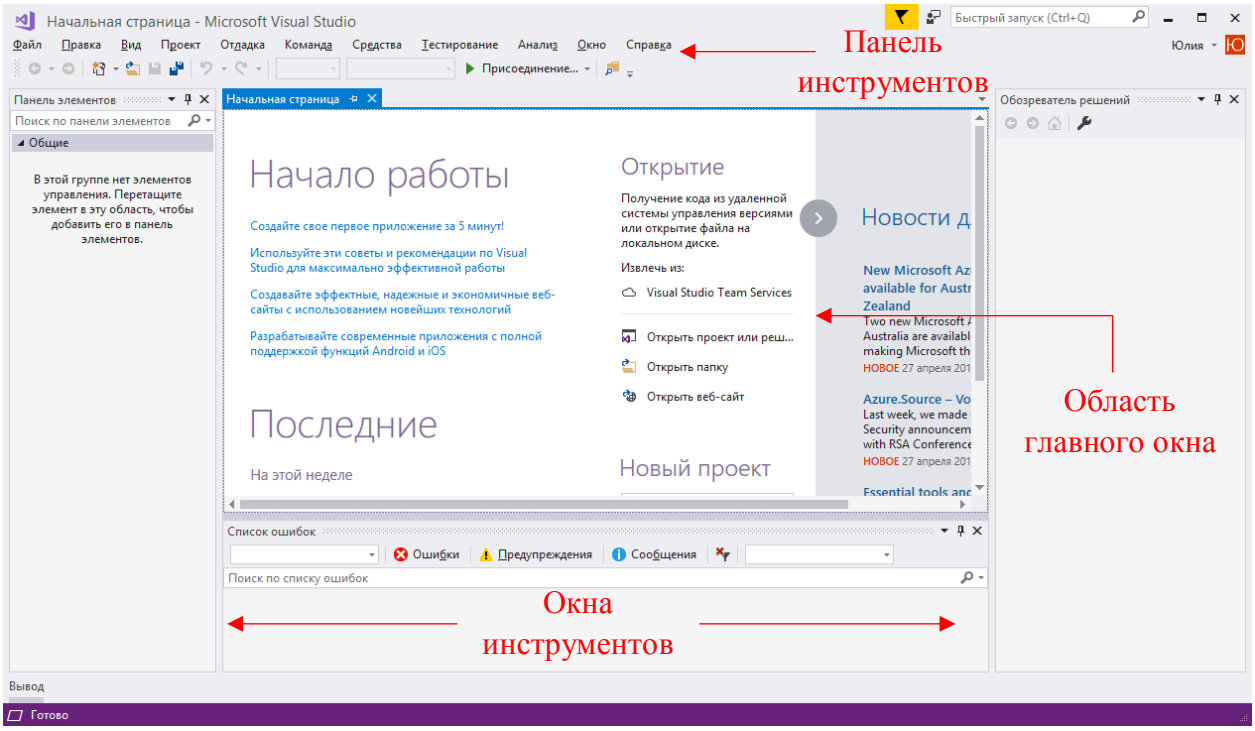


Рис.1.2. Начальная страница

Чтобы создать решение и проект, необходимо последовательно выбрать элементы Файл, Создать, Проект (File, New, Project).

В диалоговом окне Создать проект (New Project) отображаются несколько шаблонов проекта. Они упорядочены по языку программирования и типу проекта. Необходимо ввести имя нового проекта в поле Имя (Name). Можно сохранить проект в расположении по умолчанию в системе или нажать кнопку Обзор (Location) и выбрать другое место. Для начала работы, необходимо нажать кнопку ОК (рис.1.3).

Проект в Visual Studio – это логический контейнер, который содержит объекты, необходимые для создания приложения, например, файлы исходного кода, растровые изображения, значки, а также ссылки на компоненты и службы. При создании проекта, Visual Studio создает решение, в котором будут содержаться проекты. После этого в решение при необходимости можно добавить другие новые или существующие проекты. Решения также могут содержать файлы, не связанные с определенным проектом.

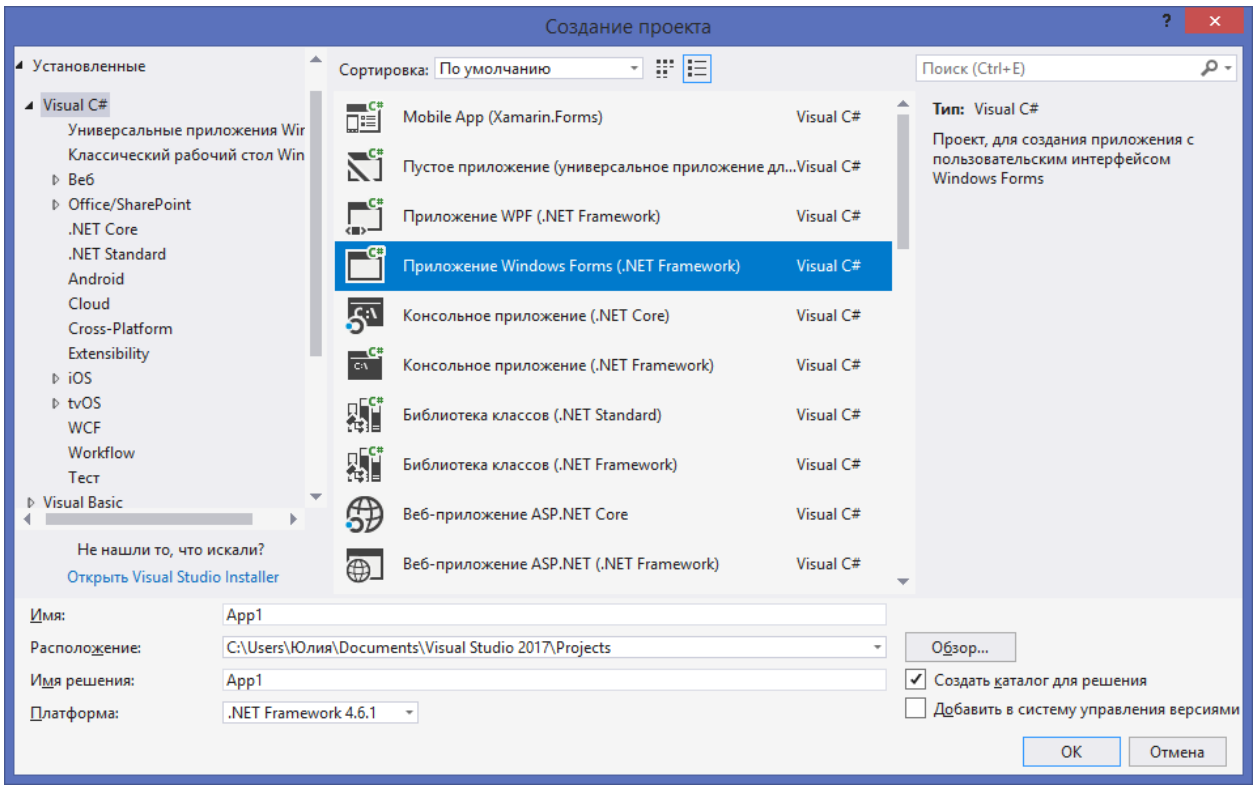


Рис.1.3. Создание проекта

Для создания графических интерфейсов применяются разные технологии – Window Forms, WPF, приложения для магазина Windows Store (для ОС Windows 8/8.1/10). Однако наиболее простой и удобной платформой до сих пор остается Window Forms или формы.

Windows.Forms – это набор различных управляемых библиотек, с помощью которых можно выполнить все необходимые для оконного приложения действия, начиная от обмена сообщениями с операционной системой для отслеживания любых событий клиентского окна, заканчивая диалоговыми системами, связью с другими компьютерами по сети и многими другими возможностями. В данном случае под формой понимается видимая поверхность окна, включающая информацию для конечного пользователя, а также содержащую в себе набор инструментов (элементов управления) для работы с представленными данными или взаимодействия с пользователем.

Большую часть пространства Visual Studio занимает графический дизайнер, который содержит форму будущего приложения. Пока она пуста и имеет только заголовок Form1. Справа находится окно файлов

решения/проекта – Обозреватель решений (Solution Explorer). Там и находятся все связанные с приложением файлы, в том числе файлы формы Form1.cs (рис.1.4).

Окно панели элементов (Toolbox) отображает элементы управления, которые можно добавлять в проекты Visual Studio. По умолчанию панель элементов свернута в левой части Visual Studio (рис.1.4). Чтобы отобразить ее, необходимо навести на нее курсор. Можно закрепить панель элементов, щелкнув на панели инструментов пиктограмму Закрепить, чтобы она оставалась открытой. Также можно открепить окно панели элементов и перетащить его в любое место на экране.

Для обеспечения обмена данными и диалога между пользователем и программой существует панель элементов, данные компоненты представлены на вкладке «Стандартные элементы управления».

Внизу справа находится окно свойств (Properties). Так как в данный момент выбрана форма как элемент управления, то в этом поле отображаются свойства, связанные с формой (рис.1.4).

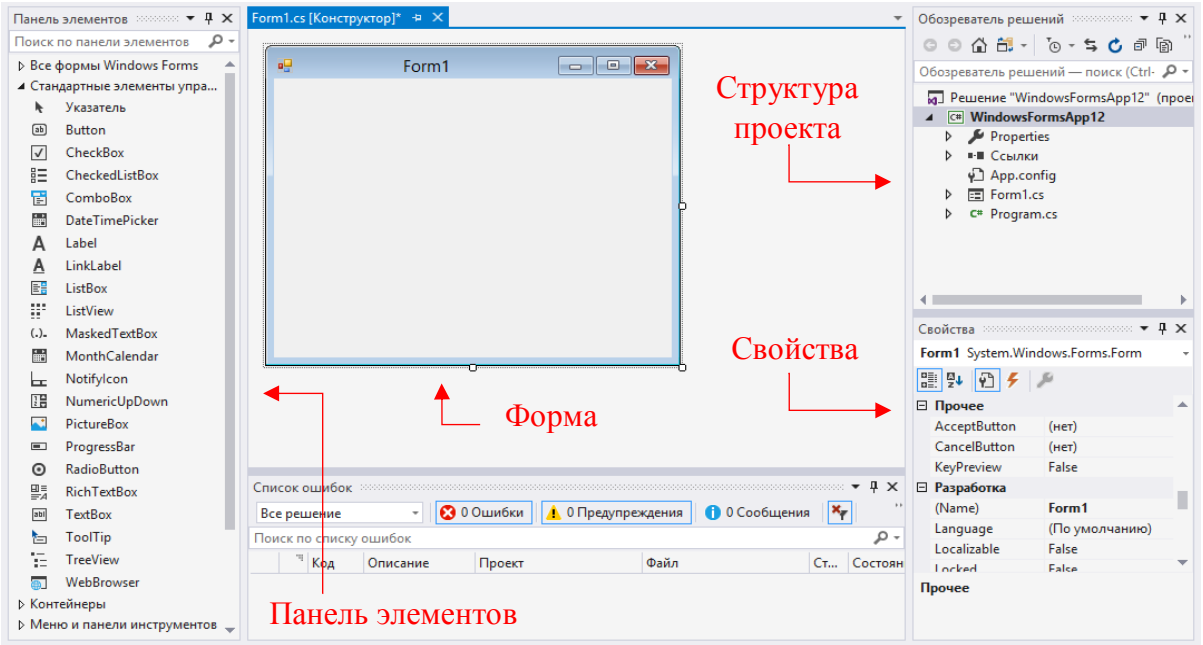


Рис.1.4. Form1.cs

Окно свойств  служит для установки нужных свойств компонента, вкладка  События (Events) позволяет определить реакцию компонента на то или иное событие. Совокупность свойств отображает видимую сторону компонента: положение относительно левого верхнего угла рабочей области формы, его размеры и цвет, шрифт и текст надписи на нем и т. д; совокупность событий – его поведенческую сторону: будет ли компонент реагировать на щелчок мыши или на нажатие клавиш, как он будет вести себя в момент появления на экране или в момент изменения размеров окна.

Глава 2. Курсы объектно-ориентированного и визуального программирования (обзор)

2.1. Анализ стандартов

В условиях динамично меняющегося мира, глобальной взаимозависимости и конкуренции, необходимости широкого использования, постоянного развития и усложнения технологий фундаментальное значение имеет информатизация сферы образования. Содержание и качество образования, его доступность, соответствие потребностям конкретной личности в решающей степени определяют состояние интеллектуального потенциала современного общества. Интенсивное развитие сферы образования на основе использования информационных и телекоммуникационных технологий становится важнейшим национальным приоритетом.

Основная образовательная программа подготовки специалистов среднего звена разрабатывается на основе государственного образовательного стандарта и включает в себя учебный план, программы учебных дисциплин, программы учебно-исследовательской и педагогических практик, программы итоговой аттестации.

Обязательный минимум содержания основной образовательной программы формируется из дисциплин федерального компонента, дисциплин национально–регионального компонента, дисциплин по выбору студента, а также факультативных дисциплин. Дисциплины и курсы по выбору студента в каждом цикле содержательно должны дополнять дисциплины, указанные в федеральном компоненте цикла.

Основная образовательная программа подготовки специалистов среднего звена должна предусматривать изучение студентами следующих циклов дисциплин и итоговую государственную аттестацию:

- общие гуманитарные и социально–экономические дисциплины;
- общие математические и естественнонаучные дисциплины;

- общепрофессиональные дисциплины направления;
- дисциплины профильной подготовки;
- факультативные дисциплины.

Данный методический комплект предназначен для наполнения курсов, связанных с визуальным, объектно-ориентированным программированием. Для рассмотрения возьмем следующие направления:

- 09.02.03 Программирование в компьютерных системах;
- 09.02.04 Информационные системы (по отраслям);
- 09.02.07 Информационные системы и программирование.

В направлении «Информационные системы и программирование (09.02.07)» в разделе дисциплин профильной подготовки обозначен профессиональный модуль «Разработка модулей программного обеспечения для компьютерных систем».

Таблица. 1. Выдержка из ФГОС СПО направление 09.02.07 Информационные системы и программирование.

ПМ.01	***Разработка модулей программного обеспечения для компьютерных систем*** знать: основные этапы разработки программного обеспечения; основные принципы технологии структурного и объектно-ориентированного программирования; способы оптимизации и приемы рефакторинга; основные принципы отладки и тестирования программных продуктов, уметь: осуществлять разработку кода программного модуля на высокого уровня; создавать программу по разработанному алгоритму как отдельный модуль; выполнять отладку и тестирование программы на уровне модуля; осуществлять разработку кода программного модуля на современных языках программирования; уметь выполнять оптимизацию и рефакторинг программного кода; оформлять документацию на программные средства. иметь практический опыт в: разработке кода программного продукта на основе готовой спецификации на уровне модуля; использовании	524 ч.

	инструментальных средств на этапе отладки программного продукта; проведении тестирования программного модуля по определенному сценарию; использовании инструментальных средств на этапе отладки программного продукта.	

В содержании профессионального модуля изучается раздел объектно-ориентированное программирование. На весь профессиональный модуль отводится 524 часа, частичное наполнение модуля разработанным комплектом достаточно для рассмотрения ООП на примере языка программирования высокого уровня C#.

Следующее направление – «Информационные системы (по отраслям) (09.02.04)», в разделе профессиональных дисциплин есть предмет «Основы алгоритмизации и программирования».

Таблица. 2. Выдержка из ФГОС СПО направление 09.02.04 Информационные системы (по отраслям).

ОП.06	***Основы алгоритмизации и программирования*** уметь: использовать языки программирования, строить логически правильные и эффективные программы; знать: общие принципы построения алгоритмов, основные алгоритмические конструкции; понятие системы программирования; основные элементы процедурного языка программирования, структуру программы, операции, управляющие структуры, структуры данных, файлы, кассы памяти; подпрограммы, составление библиотек программ; объектно-ориентированную модель программирования, понятие классов и объектов, их свойств и методов	138 ч.

В стандарте определено изучение объектно-ориентированного программирования, частичное наполнение этого курса может обеспечить разработанный комплект.

«Программирование в компьютерных системах (09.02.03)» в разделе профессиональные модули есть дисциплина «Прикладное программирование»

Таблица. 3. Выдержка из ФГОС СПО направление 09.02.03 Программирование в компьютерных системах.

МДК.01.02	В результате изучения профессионального модуля обучающийся должен: иметь практический опыт: разработки алгоритма поставленной задачи и реализации его средствами автоматизированного проектирования; разработки кода программного продукта на основе готовой спецификации на уровне модуля; использования инструментальных средств на этапе отладки программного продукта; проведения тестирования программного модуля по определенному сценарию; уметь: осуществлять разработку кода программного модуля на современных языках программирования; создавать программу по разработанному алгоритму как отдельный модуль; выполнять отладку и тестирование программы на уровне модуля;; использовать инструментальные средства для автоматизации оформления документации; знать: основные принципы технологии структурного и объектно-ориентированного программирования; основные принципы отладки и тестирования программных продуктов.	254 ч.

В данной дисциплине изучение программирования можно основывать на визуальных средах поддерживающих объектно-ориентированную парадигму программирования.

Были рассмотрены федеральные государственные стандарты УГС 09 среднего профессионального образования. Во всех стандартах присутствуют дисциплины, связанные с визуальным и объектно-ориентированным программированием. Особое внимание заслуживает то, что все предметы, в которых изучается визуальное программирование, располагаются в федеральном компоненте, который является обязательным для изучения. Выделяется достаточное количество учебного времени. Таким образом, одной из проблем является поддержка таких дисциплин дидактическими материалами.

2.2. Компетентностный подход в изучении объектно-ориентированного программирования

Внедрение компетентностного подхода в систему среднего профессионального образования направлено на улучшение взаимодействия с рынком труда, повышение конкурентоспособности специалистов, обновление содержания, методологии и соответствующей среды обучения.

Среднее профессиональное образование имеет целью подготовку специалистов соответствующего уровня и профиля, конкурентоспособного на рынке труда, компетентного, свободно владеющего своей профессией, готового к постоянному профессиональному росту, социальной и профессиональной мобильности.

Компетентностный подход – это совокупность общих принципов определения целей образования, отбора содержания образования, организации образовательного процесса и оценки образовательных результатов. К числу таких принципов относятся следующие положения. [6]

- Смысл образования заключается в развитии у обучаемых способности самостоятельно решать проблемы в различных сферах и видах деятельности на основе использования социального опыта, элементом которого является и собственный опыт учащихся.
- Содержание образования представляет собой дидактически адаптированный социальный опыт решения познавательных, мировоззренческих, нравственных, политических и иных проблем.
- Смысл организации образовательного процесса заключается в создании условий для формирования у обучаемых опыта самостоятельного решения познавательных, коммуникативных, организационных, нравственных и иных проблем, составляющих содержание образования.

- Оценка образовательных результатов основывается на анализе уровней образованности, достигнутых учащимися на определенном этапе обучения.

Под компетентностью в области объектно-ориентированного программирования понимается способность специалиста применять знания из области методологии объектно-ориентированного программирования (понятия, принципы, механизмы) и умения выделять и структурировать объекты изучаемой предметной области, определять назначение объектов и взаимодействие между ними, разрабатывать алгоритмы обработки объектов, реализовывать их с помощью объектно-ориентированного языка программирования и среды визуального программирования для решения учебных задач с целью создания приложений образовательного назначения.

Для примера, рассмотрим направление 09.02.04 Информационные системы (по отраслям). Специалисты по данному направлению должны уметь разрабатывать, внедрять и сопровождать технологические процессы и компоненты автоматизированной обработки информации и управления в организациях различной отраслевой направленности и организационно-правовых норм. Дисциплина «Основы алгоритмизации и программирования» частично реализует следующие компетенции. [2]

- Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес (ОК 1).
- Организовывать собственную деятельность, определять методы и способы выполнения профессиональных задач, оценивать их эффективность и качество (ОК 2).
- Решать проблемы, оценивать риски и принимать решения в нестандартных ситуациях (ОК 3).
- Осуществлять поиск, анализ и оценку информации, необходимой для постановки и решения профессиональных задач, профессионального и личностного развития (ОК 4).

- Использовать информационно-коммуникационные технологии для совершенствования профессиональной деятельности (ОК 5).
- Работать в коллективе и команде, обеспечивать ее сплочение, эффективно общаться с коллегами, руководством, потребителями (ОК 6).
- Ставить цели, мотивировать деятельность подчиненных, организовывать и контролировать их работу с принятием на себя ответственности за результат выполнения заданий (ОК 7).
- Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации (ОК 8).
- Быть готовым к смене технологий в профессиональной деятельности (ОК 9).
- Взаимодействовать со специалистами смежного профиля при разработке методов, средств и технологий применения объектов профессиональной деятельности (ПК 1.2).
- Производить модификацию отдельных модулей информационной системы в соответствии с рабочим заданием, находить ошибки кодирования в разрабатываемых модулях информационной системы, документировать выполняемые работы (ПК 1.3).
- Программировать в соответствии с требованиями технического задания (ПК 2.2).
- Применять методики тестирования разрабатываемых приложений (ПК 2.3).

2.3. Цели и задачи обучения объектно-ориентированному программированию

В процессе развития информатики как прикладной науки появились разные подходы к программированию. Курсы для изучения визуального программирования призваны содействовать знакомству студентов с различными парадигмами проектирования и разработке программного обеспечения. Они важны с той точки зрения, что, являются составной частью подготовки специалиста и бакалавра соответствующего профиля, способствуют развитию алгоритмического мышления, навыков программирования.

В курсе, призванном способствовать формированию достаточно четкого представления об основах высокоуровневого программирования, рассматриваются фундаментальные вопросы, связанные с современными технологиями программирования, эволюцией программного обеспечения, парадигмами объектно-ориентированного программирования.

Основное внимание уделяется объектно-ориентированному программированию. Рассматриваются такие основополагающие понятия и конструкции ООП как классы и объекты, инкапсуляция; наследование и полиморфизм, события и компоненты, визуальные технологии проектирования.

Курс призван содействовать формированию и развитию логической, алгоритмической и программистской культуры будущего специалиста в сфере информатики и информационных систем.

Цель дисциплин: изучение методов программирования для овладения знаниями в области технологии программирования; подготовка к осознанному использованию как языков программирования, так и методов программирования, развитие таких умений, как решать образовательные и исследовательские задачи, анализ научной и учебно-научной литературы, использования современных технологий сбора и обработки информации, осуществлять обучение и воспитание обучающихся.

Воспитательной целью дисциплин является формирование у студентов научного, творческого подхода к освоению технологий, методов и средств производства программного обеспечения.

Основные задачи курса программирования на основе объектно-ориентированного подхода:

- знакомство с методами объектно-ориентированного программирования как наиболее распространенными и эффективными методами разработки программных продуктов;
- обучение разработке алгоритмов на основе объектно-ориентированного подхода;
- закрепление навыков алгоритмизации и программирования на основе изучения C#;
- знакомство с основными объектами, классами, возможностями объектно-ориентированного программирования на основе C#.

Отбор материала основывается на необходимости ознакомить студентов со следующей современной научной информацией:

- парадигмы программирования (императивной, функциональной, логической);
- технологии программирования (структурной, модульной, объектно-ориентированной);
- аспекты формализации синтаксиса и семантики языков программирования.

Изучение визуального программирования базируется на знании математических дисциплин и общего курса информатики.

Концепция дисциплины основана на том, что она имеет общеобразовательный и в определенной степени мировоззренческий характер и предназначена для формирования человека с широким научным кругозором.

В результате изучения дисциплины обучающийся должен:

иметь представление:

- о объектно-ориентированном анализе, проектирования и программирования;
- методах и технологиях программирования в объектно-ориентированных программных средах;
- о применении методологии высокоуровневого программирования для решения широкого круга управленческих и иных задач.

уметь:

- разрабатывать новые компоненты на основе своих либо существующих классов;
- разрабатывать приложения, работающие под управлением Windows;
- разрабатывать приложения работы с базами данных с использованием технологии ADO.

приобрести навыки:

- в разработке новых компонентов;
- разработке приложений, работающих под Windows;
- разработке приложений, для управления работой баз данных.

владеть, иметь опыт:

- работы в средах программирования;
- в разработке новых компонентов;
- разработке приложений, работающих под Windows;
- разработке приложений, для управления работой баз данных.

2.4. Конкурсы профессионального мастерства WorldSkills и обучение программированию

Движение WorldSkills International (WSI) зародилось в послевоенные годы в Испании (1947 год), когда миру катастрофически не хватало квалифицированных рабочих рук. Первые чемпионаты проводились с целью

популяризации рабочих профессий и повышения их престижа. Сегодня это эффективный инструмент подготовки кадров в соответствии с мировыми стандартами и потребностями новых высокотехнологичных производств.

Под эгидой WorldSkills проводятся региональные, национальные и мировые чемпионаты, континентальные первенства. Участники совершенствуют свои навыки, соревнуясь по шести блокам профессий: строительной отрасли, информационных и коммуникационных технологий, творчества и дизайна, промышленного производства, сферы услуг и обслуживания гражданского транспорта.

За полувековую историю международного движения к WorldSkills присоединились 79 стран. Россия это сделала в 2012 году.

Сегодня в нашей стране движение WorldSkills набирает все большую силу. Популярность растет, а воздействие на институты профессионального образования, национальной системы квалификаций усиливается. То, что Россия выиграла право на проведение мирового первенства WorldSkills Competition в 2019 в Казани только подогревает интерес общества к этому явлению.

В основе, конечно же, лежат соревнования по определенным компетенциям. Обязательно Соревнования сопровождаются сильной конгрессно-выставочной программой. Сопровождает Соревнования много социально-общественных проектов, которые рассматриваются в течение двухлетнего периода подготовки Дирекцией WSI, согласовываются Советом Директоров WSI и окончательно утверждаются на текущей Генеральной Ассамблее, которые проходят ежегодно.

Условия соревнований просты. Участник должен выполнить работу быстрее соперников и сделать это в соответствии с установленным стандартом качества. Вроде бы всё просто. Однако стандарты WS соответствуют требованиям ведущих мировых экономик. Пройдя подготовку по программе WS, студент станет профессионалом своего дела. А уж если он победит в национальном, и тем более международном чемпионате, то его с радостью

возьмёт на работу предприятие самого высокого уровня. То есть это не виртуальный чемпионат по выявлению лучшего в своём деле, а соревнование молодых специалистов в самых актуальных компетенциях, которые востребованы уже сегодня. И борьба за лучшие предложения работодателей.

В списке компетенций WorldSkills больше двухсот пятидесяти позиций, и каждый может выбрать себе профессию по вкусу. Методический комплекс по изучению объектно-ориентированного и визуального программирования подходит для развития навыков программирования и возможности дальнейшего участия в таких компетенции, как разработка компьютерных игр и мультимедийных приложений (Video Games and Multimedia Applications Development) и программирование (Programming).

На вопрос «Кем быть» каждый ответит по-своему, но обучение по программам WorldSkills, участие в региональных, национальных и международных чемпионатах — это отличная возможность обрести себя в любимом деле.

Глава 3. Методические особенности визуального программирования на C#

При изучении программирования требуется овладеть прежде всего практическими навыками. Поэтому в настоящей работе обсуждается разработанный комплекс практических заданий для освоения объектно-ориентированного и визуального программирования средствами ЯПВУ C#.

Каждый раздел состоит из двух блоков.

Первый блок – это задачи с готовым решением, демонстрирующие студентам примеры и стиль написания программ в среде Visual Studio. Задания первого блока направлены на освоение визуальной среды Visual Studio, а также закреплению навыков программирования в объектно-ориентированного стиле. Задания распределены по уровням сложности.

Второй блок – это подборка задач для самостоятельного решения, с помощью которых студент сможет сформировать соответствующие компетенции. Второй блок состоит из практических заданий, которые предусмотрены для проверки знаний объектно-ориентированного и визуального программирования C#.

Данный комплекс дидактического обеспечения предназначен для студентов среднего профессионального образования, обучающихся по технологическим и физико-математическим профилям (УГС 09). Комплекс рассчитан на следующие начальные знания обучающихся:

- базовые знания информатики и математики;
- знание основ логики;
- знание алгоритмизации и программирования C-подобных структурных языков;
- умение программировать в объектно-ориентированном стиле;
- умение разрабатывать программы в консольном режиме C#.

Изучение происходит в два этапа.

- Первый этап — ознакомление, с каким-либо разделом программирования C#, в этот этап входят задачи с приведенной идеей решения (алгоритма решения), и кодом с подробными комментариями, рассмотрев которые, читатель ознакомится с принципами ООП и средой программирования.
- Второй этап — закрепление полученных знаний, в этот этап входят нерешенные задачи, которые предназначены для самостоятельного решения.

Комплекс разделен на восемь тем:

- базовые компоненты;
- графика;
- файлы;
- базы данных;
- мультимедиа;
- работа с COM-объектами;
- работа с сетевыми компонентами;
- создание собственного компонента.

Рассмотрим подробнее данный комплект дидактического обеспечения.

3.1. Базовые компоненты

Задачи данного раздела демонстрируют примеры оформления интерфейса, внешнего вида программы, они предназначены для изучения свойств компонентов, обеспечивающих ввод–вывод информации. Задача первого блока сопровождаются изображениями и таблицами позволяющие наглядно продемонстрировать, варианты оформления интерфейса. В этом разделе представлена одна задача с полным решением и 20 задач для самостоятельного решения. Задача с полным решением содержит основные компоненты обеспечивающих интерфейс программы, что обеспечивает

изучение компонентов одновременно, показывает их взаимодействие друг с другом.

Первоначально предлагается к рассмотрению задача с готовым решением. Данные задача демонстрирует работу стандартных элементов управления, которые обеспечивают обмен данными и диалог между пользователем и программой.

Следующий тип заданий – это задачи для самостоятельного решения, т.е. приводится только условие задачи. Читателю необходимо сформировать интерфейс, вид программы и реализовать модель задачи.

Изучение следует начинать с рассмотрения задачи с использованием стандартных элементов управления, которая приведена с кодом решения, пояснением и комментариями основных действий, что обеспечивает наглядность изучения строения среды Visual Studio. В задаче рассматриваются элементы и их свойства, события и методы. Это обеспечит закрепление знаний основных свойств ООП, таких как инкапсуляция (свойства и методы компонента объединены), полиморфизм (свойства, и методы имеют общее имя, но выполняются по-разному), наследственность (обнаружение общих свойств и методов компонента).

Для закрепления полученных навыков необходимо перейти к следующему разделу – это задачи для самостоятельной работы. Самостоятельная работа является одним из эффективных средств развития и активизации творческой, познавательной и интеллектуальной деятельности студентов. Решение задач позволяет закрепить знания, полученные при разборе задач на первом этапе, и развить навыки по работе с элементами управления.

Существует и другой подход к изучению данной темы. Это решение задач, разделенных на группы по конкретным темам, т.е. рассматриваем первую задачу на компоненты Label, Edit, Form, Button, которая приведена с подробным решением. Следующие задачи – с частичным решением, потом нерешенные задача. У такого способа изучения присутствует недостаток,

нельзя в полной мере рассмотреть взаимное применение компонентов, т.е. разработать удобный, красивый интерфейс программы.

На этом изучение основных компонентов среды программирования Visual Studio не заканчивается, эти компоненты будут использоваться во всех остальных задачах.

3.2. Графика

Следующим разделом будет графика. Любое современное программное приложение уделяет большое место внешнему виду, графическому дизайну. Одним из способов представления, может быть создание на его форме каких-либо рисунков с помощью анимации. Рассмотрение данного раздела предлагается начать с примитивов графики: первая задача предлагает студенту нарисовать флаг олимпийских игр. Составляющие этого флага будет пять колец, которые выводятся на элемент PictureBox при использовании функции DrawEllipse. Следующие предложенные задачи немного сложнее, они посвящены выводу на форму различных графиков и диаграмм.

Вторая задача – это вывод на форму графика с помощью примитивов, здесь приведен код программы с автоматическим подбором масштаба (также автоматически подбирается цена деления осей). Для того чтобы построить график определяется функция, которая по заданному аргументу x вычисляет y. Разделяется форма по полам, по оси x и по оси y. В цикле по всей оси x вычисляем координату по оси y, и соединяем с предыдущей точкой обычной линией.

Третья задача – это использование компонента Chart для вывода диаграммы, так же здесь впервые используется элемент DataGridView, он отображает строки и столбцы данных в сетке, это необходимо для ввода и хранения данных, которые используются для построения диаграммы.

Однако, C# не ограничивается только графическими примитивами. C# предоставляет большую возможность для создания различной анимации. Рассмотрим задачу на создания мультипликационных эффектов. Идея задачи

заключается в следующем: мы рисуем простой рисунок с помощью графических примитивов, по команде его стираем и рисуем уже в другом месте, тем самым создается иллюзия движения нашего рисунка.

Также C# предоставляет возможность любому пользователю нарисовать свой рисунок. Задается два режима первое произвольное, второе какой-либо фигуры. При произвольном рисовании если пользователь нажал на кнопку мыши мы каждый раз вычисляем координаты нажатия и соединяем с предыдущей координатой линией. Если режим рисования объектов, то при нажатии запоминаем начальную точку, при повторном нажатии конечную точку, между ними рисуется необходимый нам объект. Данная тема подробно рассмотрена в четвертой задаче.

Данный раздел насчитывает 5 текстовых задач, распределенных по темам:

- рисование на холсте
- график функции
- диаграммы
- графический редактор
- анимация

Следующий раздел темы «Графика» включает задачи для самостоятельной работы. В нем насчитывается 22 задачи.

Изучение необходимо начать с рассмотрения задач на линейную графику, что способствует формированию способов построения изображения. Далее рассматриваются задачи на построение графиков и диаграмм. При построении диаграмм, студентов необходимо познакомить с новыми компонентами Chart и DataGridView, и показать их функционал и особенности.

После рассмотрения построения диаграмм, знакомим студентов со способами построения графического редактора и возможностях рисования

различными инструментами. Также применим диалоговые окна для открытия, сохранения файлов.

Далее рассматриваем задачи на создание простой анимации. Особое внимание здесь необходимо обратить на способ построения иллюзии движения, а также использование и создание градиентной заливки.

3.3. Файловая система и исключения

Этот раздел, один из наиболее важных в программировании, так как рано или поздно придется столкнуться с необходимостью долговременного хранения полученной информации. Сохранение информации происходит в файлах. Этот раздел посвящен работе с файлами, каталогами и обработке исключительных ситуаций.

Изучению файлов предшествует тема, связанная с обработкой исключительных ситуаций. Для этого студентам предложена задача, которая помещена в отдельный раздел, потому что последующие задачи должны сопровождаться перехватом исключительных ситуаций, возникающих при написании программ.

После полного освоения способов перехвата исключительных ситуаций, можно приступить к непосредственной работе с каталогами.

Наиболее частыми действиями над каталогами является создание, удаление, переход в каталог и указание текущей папки.

Далее рассматриваем основные действия над файлами, такие как поиск файла, создание и запись во временный файл, копирование файла.

Поиск файла осуществляется рекурсивно, функция просматривает иерархию папок, начиная с папки, полный путь к которой передан ей, обходит все вложенные подпапки и заполняет элемент ListBox списком файлов, которые соответствуют маске поиска.

При работе с временным файлом необходимо показать создание, заполнение и расположение файла. А также генерацию имени, которое не будет совпадать с другими файлами. Заполнять файл можно вручную, а также

с помощью генерации текста, который подключается из Интернет-ресурса, подробно данная функция рассмотрена в четвертой задаче.

Следующим шагом будет рассмотрение одной из наиболее часто применяемых операций над файлами – копирование. В C# существует несколько способов организации копирования файла, в разобранной задачи было описано два типа: первый — с помощью встроенной функции C# CopyFile; и второй — создание своего собственного метода копирования посредством файловых потоков.

Задачи для самостоятельной работы насчитывают 29 текстовых задач, которые распределены следующим образом:

- исключительные ситуации
- работа с каталогами (папками)
- работа с файлами
- работа с временными файлами

Хотя все остальные разделы можно изучать в произвольном порядке, но так как в разделах графика и базы данных идет прямое применение файлов, то рекомендуем остановиться на разделе «Файловая система и исключения». При решении задач раздела «Файловая система и исключения» могут допускаться ошибки записи, чтения файла, существования файла, поэтому первоначально необходимо полностью изучить соответствующие исключительные ситуации, что позволит в дальнейшем уменьшить вероятность некорректного завершения приложения. Остальные блоки изучаются аналогично первому разделу. Особое внимание здесь следует обратить на такие задачи, как поиск файла, копирование файла и работа с каталогами.

3.4. Базы данных

Хранение информации в файлах можно организовать по–разному, одним из способов является хранение в виде баз данных. C# предоставляет широкие возможности для создания и работы с базами данных. В изучении баз данных, задачи разделяют на следующие группы:

- основные компоненты для работы с базами данных;
- основы языка SQL;
- создание базы данных MySQL;
- организация работы с базами данных посредствам C#.

Чтобы начать изучение работы с базами данных, необходимо ее создать. Для начала создадим реляционную базу данных MySQL, которая будет использоваться во всех последующих задачах.

В первой задаче рассматриваем возможность подключения и изменения созданной базы данных. Важно подчеркнуть внимание не только на отображение данных в приложении, но и сохранение всех манипуляций над ней в реальной базе данных.

Следующая задача – это фильтрация данных по определенным диапазонам. Для быстрого доступа к данным их необходимо представить в порядочном виде, по какому-либо параметру. Создается специальная панель, которая фильтрует данные с помощью SQL запросов – И, ИЛИ.

Далее рассматриваются задачи на основные действия с базами данных: поиск, сортировка, вставка и удаление записей.

Последняя задача посвящена работе со связными таблицами, которая показывает связь отношений в базе данных.

Задачи для самостоятельной работы насчитывают 20 текстовых задач.

Изучение данного раздела рекомендуется строго по выделенным блокам, т.к. организация базы данных сопровождается сложными программными действиями, и для качественного овладения, нужно изучить сначала один из блоков, а потом приступать к изучению другого по аналогии с предыдущим, что позволит качественно и быстро изучить ООП подход к программированию баз данных.

3.5. Мультимедиа

Иногда возникает необходимость разнообразить программы, сделать более привлекательными. Один из способов — это мультимедиа, анимационные эффекты, сопровождаемые или не сопровождаемые звуком. В разделе мультимедиа рассматривается два типа анимации: простая анимация, не сопровождаемая звуком и анимационные эффекты со звуком (AVI клипы).

А также работа только со звуком (MP3 плеер). Соответственно, рассмотрены такие компоненты:

- Метод ImageAnimator (анимация без звука);
- WMP (Windows Media Player) (подключаемый элемент для просмотра видео);
- Winmm (DLL библиотека для работы с аудиофайлами).

Первый компонент рассчитан на простые анимационные ролики, которые выводят анимированное изображение на экран. Изображение создается из анимированного GIF-файла, находящегося в той же папке, что и приложение или в ресурсах проекта. Такие анимации не сопровождаются звуком. Они могут украшать приложение при запуске (например, заставка программы), вызове справки и т.п.

Подключаемый элемент Windows Media Player. С его помощью можно проигрывать файлы любых форматов, поддерживаемых данной системой.

Существует единый набор команд для воспроизведения аудиофайлов с помощью любой звуковой карты. В ОС Windows функции, предназначенные для выполнения мультимедийных операций, находятся в библиотеке WinMM.DLL. Соответственно, чтобы MP3-плеер мог воспроизводить аудиофайлы, необходимо подключить данную библиотеку и работать с ней, описание представлено в третьей задаче.

Раздел заканчивается заданиями для самостоятельной работы, которые насчитывают 25 задач.

Для изучения данного раздела необходимо иметь представление о форматах графических файлов. Раздел следует начинать с первой задачи, которая очень проста в обращении, что позволит, во-первых, понять, как устроены программы, работающие с анимацией, во-вторых, заинтересовать студентов данной темой, так как результат получить очень просто и быстро. После изучения, можно переходить к Windows MediaPlayer и библиотеке Winmm. Таким образом, можно добиться хороших программных результатов.

3.6. COM-объекты

Написать все приложения, удовлетворяющее требованию пользователя, невозможно, иногда хочется использовать сторонние приложения в собственной разработке. C# предоставляет такую возможность благодаря COM-объектам.

Объектная модель программных компонентов (Component Object Model, COM) проектировалась с целью дать возможность создавать компоненты на любом языке/платформе, обладающем поддержкой этой модели, и использовать их в любом языке/платформе (другом), так же обладающем поддержкой COM. Платформа .NET не исключение и позволяет легко использовать сторонние COM-объекты и экспортировать типы .NET в виде COM-объектов.

COM-объекты используется для обеспечения межпроцессного взаимодействия и создания динамических объектов на широком спектре языков программирования. В C# есть свой набор стандартных COM-объектов. В данном разделе показано, как подключать, использовать и взаимодействовать с такими объектами.

Рассмотрено три задачи, все они направлены на изучение использования сторонних объектов.

Чтобы подключить какой-либо объект, необходимо в обозревателе решений добавить ссылку на него, после добавить уже сам компонент (при необходимости) на панель элементов.

В первой задаче рассматривается работа с PDF Adobe Reader, в ней детально акцентировано внимание на подключение COM-объектов.

Вторая и третья задача рассматривает работу с компонентами Microsoft – Excel и Word.

Задания для самостоятельной работы насчитывают 30 задач, которые распределены по следующим объектам:

- PDF Adobe Reader

- MS Excel
- MS Word
- MS PowerPoint
- MS Outlook
- MS OneNote

Один из самых интересных и увлекательных разделов, так с его помощью при небольших затратах, можно получить быстро качественный продукт. Изучение необходимо начинать с первой задачи, где присутствует небольшая теоретическая информация, которая облегчит работу не только со следующими задачами раздела, но и работу с C# и Visual Studio в общем.

Порядок изучения заданий согласно нумерации. Особое внимание надо уделить подключению объекта.

3.7. Сетевые компоненты

Компьютер не представлял бы большой ценности, если не было бы возможности обмена информации между компьютерами. Данную проблему можно легко решить, используя сетевые приложения. Данный раздел показывает, как быстро и легко создать небольшие сетевые приложения.

В разделе всего три задачи, ввиду их сложности разработки, рассмотрены только самые простейшие компоненты.

Первая задача описывает протокол SMTP, который осуществляет отправку почты в среде Интернет. Данный протокол указывает, как почтовые сервера взаимодействуют при передаче электронной почты.

Вторая задача рассматривает работу с очередью сообщений, которая является компонентом Windows. Администратор домена создает очередь сообщений, и пользователи могут прислать и получать сообщения используя эту очередь. Особенность заключается в том, что пользователю не нужно быть в сети в тот момент, когда ему отправляется сообщение, эти сообщения будут хранится на сервере и когда пользователь подключится к домену он сможет их прочитать.

Также в разделе рассмотрена работа с сокетами TCP, дана интересная задача обмена сообщения между Клиентом и Сервером, которые зашифрованы с помощью технологии DES.

Раздел заканчивается заданиями для самостоятельной работы, которые насчитывают 12 задач.

Данный раздел необходимо начинать с теоретической части, так как он содержит много специфических разделов. Если студенты не могут овладеть с помощью справочной подборки и подробно разобранных задач, которые даны в комплекте, необходимо использовать дополнительную литературу. После разбора студентами первой и второй задачи, рекомендуется на практическом занятии подробно разобрать вместе со студентами третью задачу, которая представлена с полным решением. После этого можно дать на самостоятельное решение задачи подобной тематики, что позволит студентам понять принцип работы программы. После этого можно переходить на другие задачи, так как все задачи данного раздела разобраны, то порядок их изучения особо не важен, но рекомендуется в порядке нумерации.

3.8. Собственные компоненты

При программировании в визуальных средах можно встретиться с такой проблемой, как нехватка какого-либо компонента. Поэтому возникает потребность разработать свой компонент. Для этого существует специальная тема «собственные компоненты». В нем подробно рассматривается, как создается компонент, с чего необходимо начать.

Раздел разбит на две темы – создание визуального компонента и создание не визуального компонента.

Собственные элементы управления предоставляют средства для создания и повторного использования настраиваемых графических интерфейсов. Собственный элемент управления — это компонент, имеющий визуальное представление. Таким образом, он может состоять из одного или нескольких элементов управления, компонентов или блоков кода Windows

Forms, позволяющих расширить функциональные возможности за счет проверки введенных пользователем данных, изменения свойств отображения или выполнения других предусмотренных разработчиком действий. Собственные элементы управления можно вставлять в Windows Forms точно так же, как другие элементы управления.

Данная тема очень важна, так как она определяет алгоритм создания компонента или класса. Ниже рассмотрены вопросы, которым необходимо уделить особое внимание.

Визуальный компонент обычно разрабатывается не с нуля, а на основе уже существующего класса, функциональность которого специализируется или расширяется путем изменения имеющихся и добавления новых свойств и методов. Программирование компонента, в отличие от визуального конструирования классического приложения, сводится в основном к написанию исходного кода, для чего совершенно необходимо более тесное знакомство с объектно-ориентированным программированием.

Цикл разработки нового компонента состоит из следующих этапов:

1. *Анализ требований к компоненту.* Необходимо тщательно продумать, какой функциональности мы от него ожидаем, и какие свойства и методы окажутся лишними.

2. *Выбор подходящего родительского класса.* По результатам предыдущего этапа нужно выбрать элемент, функциональность которого ближе всего подходит к требуемой, и произвести новый класс от него.

3. *Программирование компонента* – реализация его свойств, методов и событий.

4. Запуск приложения для к*омпилирования библиотеки DLL.*

5. *Тестирование компонента* в специально разработанном тестовом приложении. Для этого подключаем собственный компонент и добавляем его на палитру элементов.

Вторая задача посвящена созданию не визуального компонента, назначение которых – инкапсулировать некоторый часто используемый код.

На этапе проектирования приложения предоставляется доступ до неких функций, методов, полей, которые объединены общим целым. Необходимо добавить сборку, которая содержит компонент. После добавления он становится доступным, и можно объявить какую-либо переменную или поле класса этого компонента. Также мы получаем все необходимые доступы до методов и полей этого компонента. Основная задача, выполняемая не визуальными компонентами – инкапсуляция участков кода, выполняющих какие-либо функции и имеющих набор входных параметров, влияющий на их поведение.

В пределах темы построения не визуальных компонентов рассматривается пример определение треугольника по трем сторонам, по двум сторонам и углу между ними, либо по одной стороне и двум прилежащим углам, а также его использование в тестовом приложении.

Задачи для самостоятельной работы насчитывают 26 задач, которые распределены по следующим объектам:

- создание визуального компонента;
- создание не визуального компонента.

Из всех представленных разделов этот самый сложный для понимания, но в тоже время является ключевым в объектно-ориентированной методологии программирования. Изучение должно начинаться исключительно с ознакомительной части, притом вся теория должна сопровождаться практическими примерами, что позволит обеспечить наглядность в обучении. После полного освоения можно приступать к практическим заданиям, под четким руководством преподавателя, так как у студентов могут возникать ошибки, это связанно с тем, что большую часть кода они должны прописывать руками. Когда студенты усвоят простые задачи, то можно переходить к более сложным задачам. Порядок изучения можно начинать с любой темы. При изучении данной темы необходим контроль преподавателя, после успешного выполнения нескольких заданий контроль можно ослабить вплоть до самостоятельного изучения.

ЗАКЛЮЧЕНИЕ

В связи с быстрым развитием вычислительной техники, обществу требуется большое количество специалистов, умеющих разрабатывать программные приложения любой сложности. Одним из наиболее распространённых способов разработки программных приложений является визуальное программирование. Для обучения визуальному программированию был разработан дидактический комплекс, предназначенный для обучения студентов СПО, обучающихся по направлению «Информатика и вычислительная техника» (УГС 09). Основной акцент делается на практическую составляющую обучения, т.к. в большинстве литературы по программированию описана только теоретическая часть, а практических заданий немного, или они не имеют определенной тематики.

Таким образом частично решается проблема дидактического наполнения дисциплин, связанных с визуальным программированием.

В данной работе рассматривается изучение объектно-ориентированного подхода к программированию на основе ЯПВУ C# на практических примерах ориентированных на использование основных свойств ООП. Разработаны задания, позволяющие получить навыки программирования C#, приведена справочная информация и практические рекомендации по каждой теме.

В процессе выполнения ВКР были поставлены и решены следующие задачи:

- анализ государственных стандартов среднего профессионального образования;
- анализ учебной и научной литературы по обучению визуальному программированию;
- разработан комплект дидактических материалов в поддержку курса ООП на основе ЯПВУ C# в среде Visual Studio;

- разработана методика применения разработанного комплекта дидактических материалов в поддержку визуального программирования.

Комплект дидактических материалов состоит из практической части и заданий для самостоятельной работы. В процессе разработки дидактического комплекта были выделены следующие разделы среды программирования C#.

- Базовые компоненты, для его изучения представлена одна задача с решением и 20 задач для самостоятельной работы.
- Графика, для его изучения представлено 5 задач с решением и 22 задачи для самостоятельной работы.
- Файлы – представлено 5 задач с решением и 29 задач для самостоятельной работы.
- Базы данных – представлено 6 задач с решением и 20 задач для самостоятельной работы.
- Мультимедиа – представлено 3 задачи с решением и 25 задач для самостоятельной работы.
- Com-объекты – представлено 3 задачи с решением и 30 задач для самостоятельной работы.
- Сетевые компоненты – представлено 3 задачи с решением и 13 задач для самостоятельной работы.
- Создание собственных компонентов – представлено 2 задачи с решением и 26 задач для самостоятельной работы.

Таким образом, дидактический комплект в поддержку изучения C# в интегрированной среде разработки Visual Studio насчитывает 18 задач с решением и 185 задач для самостоятельной работы, что позволяет в полной мере изучить визуальное программирование на основе ЯПВУ C#.

Библиографический список

1. ФГОС СПО 09.02.07 Информационные системы и программирование. – Введ. 09.12.2016г. – www.edu.ru.

2. ФГОС 09.02.04 Информационные системы (по отраслям). – Введ. 14.05.2014г. – www.edu.ru.

3. ФГОС 09.02.03 Программирование в компьютерных системах. – Введ. 28.07.2014г. – www.edu.ru.

4. Винникова О.Н. Анализ соотнесения ведущих педагогических категорий «компетенции» и «знания и умения» в профессиональном образовании // Вестник Томского государственного педагогического университета – 2012. -№11 -126с.

5. Романов С. С. Ключевые понятия и особенности объектно-ориентированного программирования // Таврический научный обозреватель. – 2016. -№12(17) -159с.

6. Лебедев О.Е. Компетентностный подход в образовании // Школьные технологии. – 2004. -№3 -101с.

7. Цветкова М.С. Информатика и ИКТ: учебник для нач. и сред. проф. образования / М.С. Цветкова, Л.В. Великович – 3-е изд., стер. – М.: Издательский центр «Академия», 2012. – 352с.

8. Иванова Г. С. Объектно-ориентированное программирование: учебник / Г. С. Иванова, Т. Н. Ничушкина ; под общ. ред. Г. С. Ивановой. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2014. — 455с.

9. Шестаков А.П. Дидактические материалы по программированию [Электронный ресурс] / А.П. Шестаков, - URL: http://comp-science.narod.ru/p.htm (дата обращения 15.02.2018).

10. Создание Windows-приложений на основе Visual C#: Лекция [Электронный ресурс] / НОУ «ИНТУИТ» – URL: http://www.intuit.ru/studies/courses/106/106/info (дата обращения 16.03.2018).

11. Microsoft. C#. Спецификация языка Версия 3.0 / – М. : Москва, 2007. – 535 с.

12. Абрамян М.Э. Visual C# на примерах. – СПб: БХВ-Петербург, 2008. – 496 с.: ил. + CD-ROM.

13. Албахари Дж. C# 5.0. Справочник: Полное описание языка: Пер. с англ. / Албахари Дж., Албахари Б. – М.: Издательский дом «Вильямс», 2013. – 1008 с.

14. Ватсон Б. C# 4.0 на примерах. – СПб.: БХВ-Петербург, 2011. – 608 с.

15. Гросс К. C# 2008 и платформа NET 3.5 Framework: базовое руководство. – 2-е изд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2009. – 480 с.

16. Зиборов В.В. Visual C#2010 на примерах. – СПб.: БХВ-Петербург, 2011. – 432 с.

17. Климов А.П. C#. Советы программистам. – СПб.: БХВ-Петербург, 2008. – 544 с.: ил. + CD-ROM.

18. Культин Н.Б. Microsoft Visual C# в задачах и примерах. СПб.: БХВ-Петербург, 2009. – 320 с.: ил. + CD-ROM.

19. Нэш Т. C# 2008. Ускоренный курс для профессионалов. – М.: ООО Издательский дом «Вильямс», 2008. – 576 с.

20. Павловская Т.А. C#. Программирование на языке высокого уровня. Учебник для вузов. – СПб.: Питер. 2009. – 432 с.

21. Подбельский В.В. Язык C# Базовый курс: учеб. пособие. – 2-е изд. – М.: Финансы и статистика, 2013. – 408 с.

22. Половина И.П. Программирование на C#: лабораторный практикум / сост. И.П. Половина; Перм. гос. гуманит.-пед ун-т. Пермь, - 1 электрон. опт. диск (CD ROM).

23. Троелсен Э. Язык программирования C# и платформа .NET 3.5. – М.: ООО «И.Д. Вильямс», 2011. – 1344 с.

24. Фаронов В.В. Создание приложений с помощью C#. Руководство программиста. – М.: Эксмо, 2008. – 576 с.

25. Фленов М.Е. Библия C#. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2011. – 560 с.: ил. + CD-ROM.

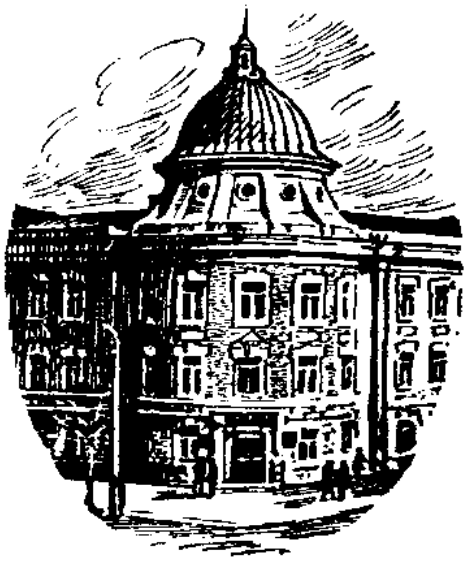
26. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#. – М.: КУДИЦ-ОБРАЗ, 2003. – 512 с.

ПРИЛОЖЕНИЕ №1.

Министерство образования и науки Российской Федерации

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ПЕРМСКИЙ ГОСУДАРСТВЕННЫЙ ГУМАНИТАРНО-
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ И ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ НА C#



Пермь
ПГПУ
2018

1. Базовые компоненты

Среда визуального программирования C# .NET включает в себя Windows.Forms.Designer (конструктор или дизайнер форм Windows) – инструмент, позволяющий в интерактивном режиме выполнять визуальное проектирование формы, размещая на ней необходимые элементы управления.

Преимущества Windows.Forms.Designer в том, что можно размещать элементы управления на форме в соответствии с представлением красоты и при этом не думать о конкретных значениях многих свойств этих элементов, например, о свойствах (точнее числовых значениях этих свойств), определяющих их местоположение или размер. Значения большинства свойств задаются автоматически конструктором формы.

В Windows приложении (в отличие от консольного приложения) конструктором формы автоматически создаются несколько классов с расширением .cs, например, класс с именем Form1.cs и класс с именем Program.cs.

Файл "Form1.cs", предназначен для разработчика – именно в ней располагаются автоматически создаваемые обработчики событий, происходящих с элементами управления, код которых создается самим разработчиком. Такая технология программирования, основанная на работе с формами, называется визуальной, событийно управляемой технологией программирования.

Элементы управления, или компоненты, помещают на форму из Панели элементов. Они представляют собой визуальные классы, которые получают введенные пользователем данные и могут инициировать различные события.

В основе визуального программирования лежат следующие базовые понятия: свойство, событие, метод.

Свойство – это атрибут объекта, определяющий то, как объект выглядит или как он может себя вести. Свойства объекта легко установить или изменить помощью панели свойств.

Событие – это действие, требующее реагирования или "обработки" в коде. События могут генерироваться действиями пользователя (например, нажатием кнопки мыши или клавиши на клавиатуре), программным кодом или системой.

Метод – это процедура или функция, которая определена как часть класса и инкапсулирована (содержится) в нем. Методы манипулируют полями и свойствами классов, хотя могут работать и с любыми переменными, и имеют автоматический доступ к любым полям и свойствам своего класса.

1.1. Стандартные элементы управления

Постановка задачи: Разработать приложение для формирования подсчета стоимости туристической путевки.

1. Создайте новый проект WindowsForms. Назовите проект travelCompany.

Для создания графических интерфейсов применяются разные технологии – Window Forms, WPF, приложения для магазина Windows Store (для ОС Windows 8/8.1/10). Для создания приложений на C# будем использовать Windows Forms. Формы являются наиболее простой и удобной платформой.

2. Активизируйте форму и измените ее имя. Для этого необходимо перейти в вкладку **Свойства** и заменить стандартное значение **Text** – Form 1 на новое, например, **Оформление путевки**. Имя сразу отобразиться в заголовке (см. рис. 1.1).

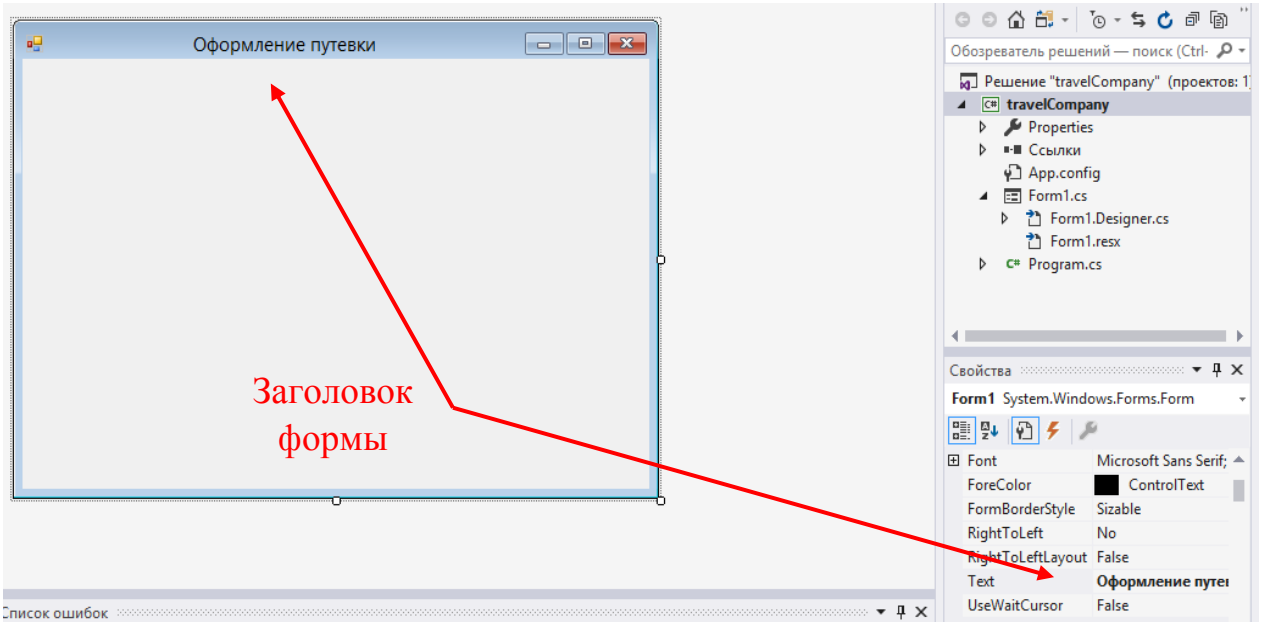


Рис.1.1. Изменение заголовка формы

3. Установите в свойствах формы значение **BackColor** (фоновый цвет компонента) на White (Белый).

4. Установите значение **Size** (Размер данного элемента управления в пикселях) на 758;740.

5. Для значения **Icon** (Указывает значок для формы) загрузите любое изображение, например, изображение Земли. Изображения загружаются только в формате .ico (рис.1.2).

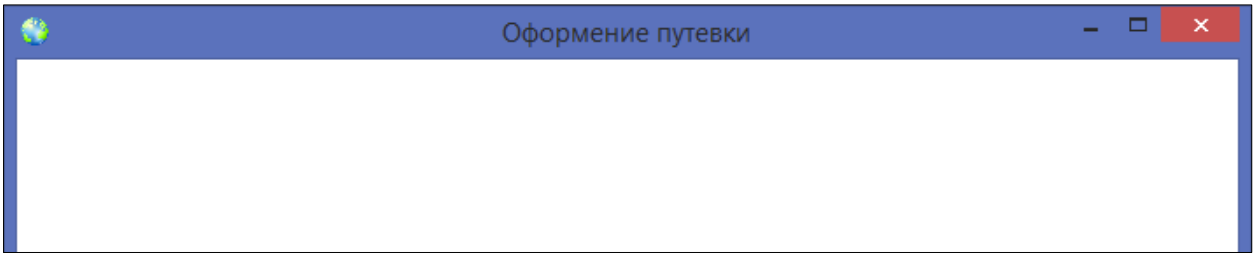


Рис.1.2. Отображение значка для формы

⇨ Начнем с разработки оформления названия нашей туристической фирмы.

Label

Для отображения простого текста на форме, доступного только для чтения, служит элемент Label. Чтобы задать отображаемый текст метки, надо установить свойство *Text* элемента.

6. Разместите на форме компонент Label. Придумайте название туристической фирмы или возьмите уже существующее, например, центр тур.

7. В свойствах компонента измените значение **Font** (Шрифт, используемый для отображения текста на элементе управления) на шрифт – Monte-Carlo, начертание – обычный, размер – 28.

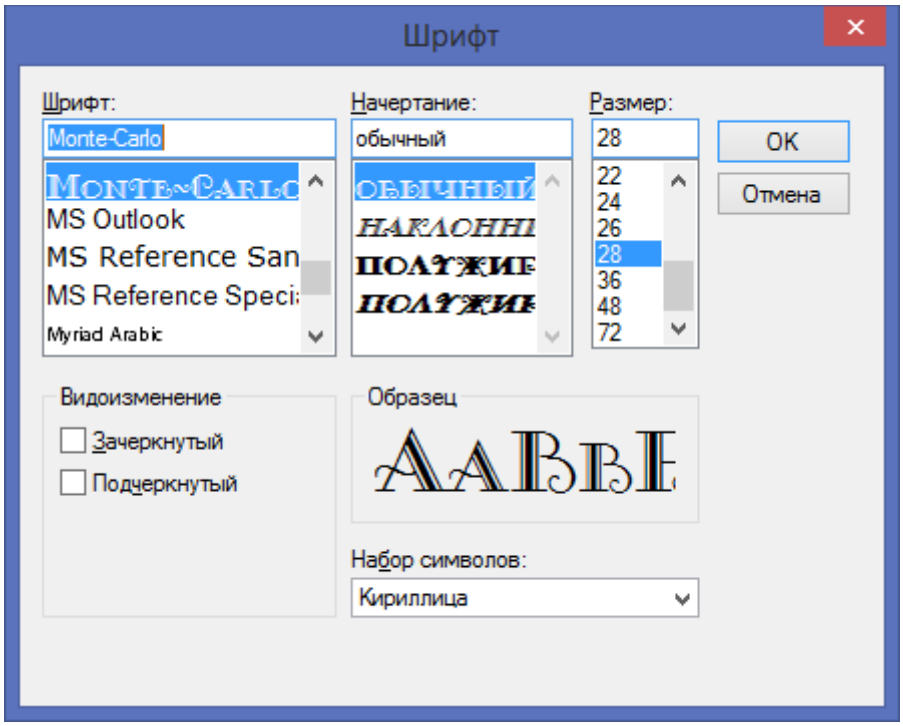


Рис. 1.3. Свойство Font

8. Установите в свойствах компонента значение **ForeColor** (Цвет переднего плана для отображения текста в данном элементе управления) на MidnightBlue (Полуночно-синий).

9. В свойстве **Location** (Координаты левого верхнего угла элемента управления относительно левого верхнего угла его контейнера) измените значение на 237; 12 (рис. 1.4).

10. Добавьте еще один компонент Label. Задайте текст – Туристическое агентство. В свойстве **Location** укажите 246; 2 (рис. 1.4).

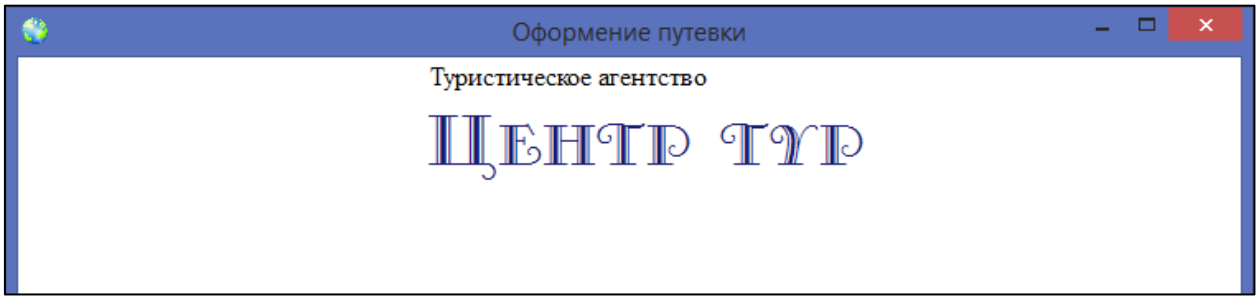


Рис. 1.4. Оформление названия

⇨ Добавим логотип нашей туристической фирмы.

PictureBox

PictureBox предназначен для показа изображений. Он позволяет отобразить файлы в формате bmp, jpg, gif, а также метафайлы изображений и иконки.

11. Разместите на форме компонент PictureBox. В свойствах укажите размер 221; 228 и координаты 12; 12.

12. Для значения **Image** (Изображение, отображаемое в PictureBox) загрузите изображение, которое будет являться логотипом туристической фирмы (рис.1.5).

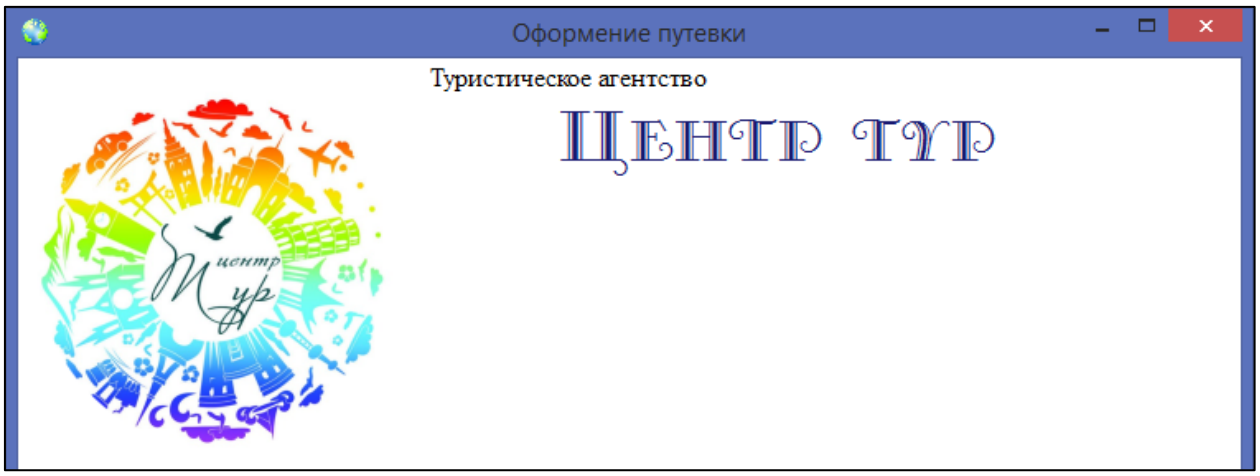


Рис. 1.5. Добавление компонента PictureBox

⇨ Заполним страны и города, и присвоим каждому наименованию изображение.

GroupBox

GroupBox представляет собой специальный контейнер, который ограничен от остальной формы границей. Он имеет заголовок, который устанавливается через свойство Text. Чтобы сделать GroupBox без заголовка, в качестве значения свойства Text просто устанавливается пустая строка. Компонент находится во вкладке «Контейнеры» панели элементов.

13. Разместите на форме компонент GroupBox, измените стандартное значение **Text** – groupBox1 на Страны и города.

14. Укажите в свойствах компонента размер 497; 162 и координаты 239; 75 (рис.1.6).

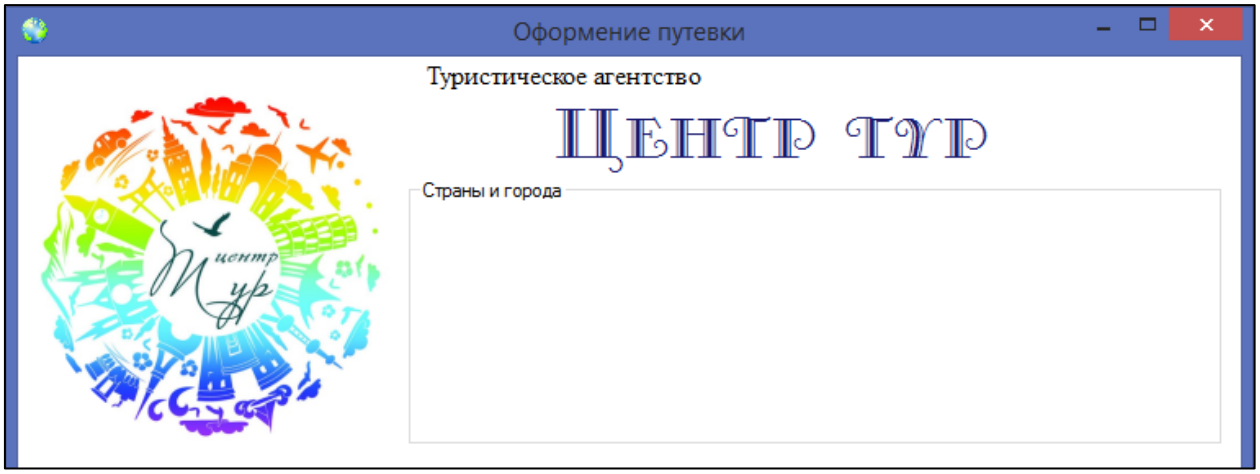


Рис. 1.6. Добавление компонента GroupBox

TreeView

TreeView представляет визуальный элемент в виде дерева. Дерево содержит узлы, которые представляют объекты TreeNode. Узлы могут содержать другие подузлы и могут находиться как скрытом, так и в раскрытом состоянии. Все узлы содержатся в свойстве Nodes.

15. Разместите на форме компонент TreeView. В нем мы укажем наши Страны и города.

16. В свойствах компонента укажите размер 346; 137. Перетащите компонент в GroupBox (рис.1.7).

17. Добавьте компонент PictureBox. Укажите размер 133; 137 и перетащите компонент в GroupBox (рис.1.7).

Установите в свойствах компонента значение **BorderStyle** (Определяет тип границы для PictureBox) на FixedSingle (Фиксированная однострочная граница)

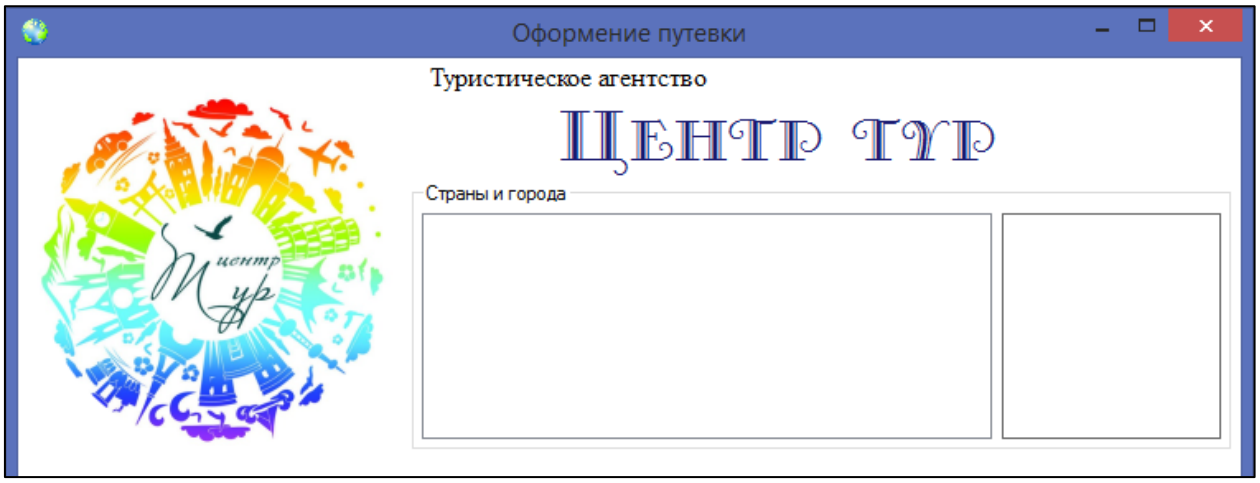


Рис. 1.7. Добавление компонентов TreeView и PictureBox в GroupBox

18. Чтобы добавить объекты в TreeView активизируйте компонент, нажмите на стрелочку и выберете **Изменить узлы.**

19. В редакторе TreeNode добавьте корень и в свойстве **Text** (Текст, отображаемый в подписи узла дерева) измените название на Россия (рис.1.8).

20. Замените стандартное значение в свойстве **Name** (Имя объекта) – Узел0 на nodeRussia (рис.1.8).

21. В свойстве **Tag** (Определяемые пользователе данные, связанные с этим объектом) можно добавить изображения, которые при выборе объекта будут отображаться на компоненте PictureBox2. Необходимо указать только название и расширение изображения (рис.1.8).

Изображения необходимо хранить в папке проекта …\travelCompany\travelCompany\bin\Debug.

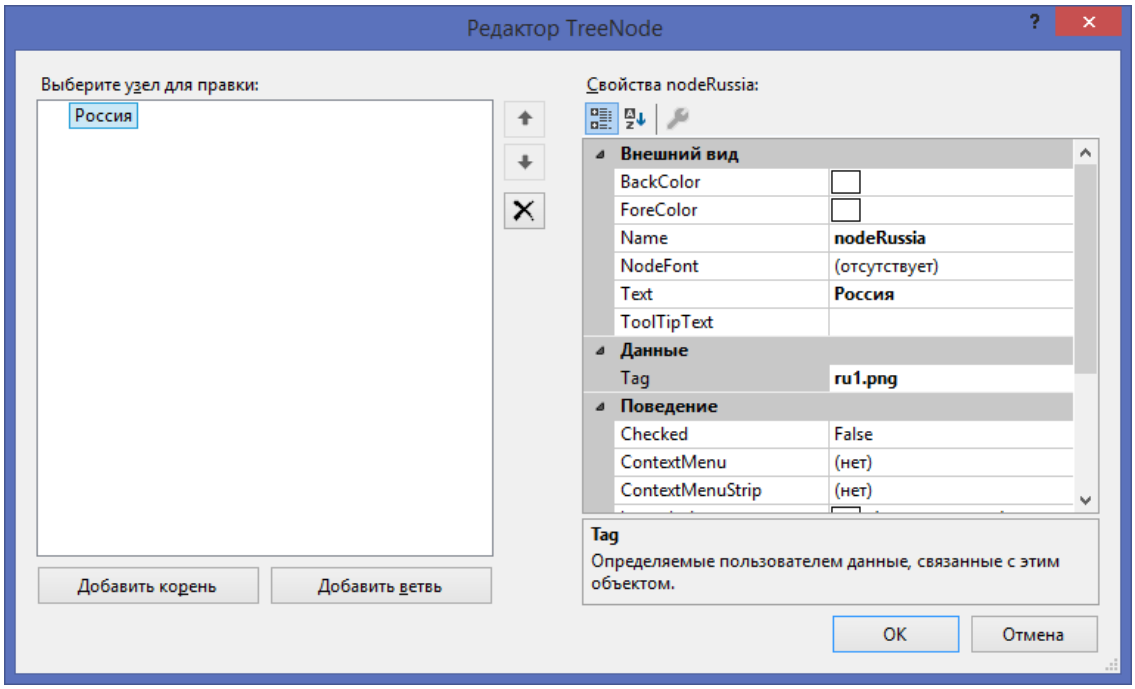


Рис. 1.8. Работа с редактором TreeNode

22. Добавьте ветвь и укажите в свойстве **Text** – Москва. Измените стандартное значение **Name** на nodeMoscow. В **Tag** добавьте картинку.

В нашем случае добавление корня является страной, а добавление ветви городом.

23. Самостоятельно добавьте другие страны и города, аналогично изменяя свойства (Таб.1).

Таблица 1.

Text	**Name**
Россия	nodeRussia
Москва	nodeMoscow
Санкт-Петербург	nodeStPetersburg
Анапа	nodeAnapa
Сочи	nodeSochi
Танзания	nodeTanzania
Дар-эс-Салам	nodeDaresSalaam
Занзибар	nodeZanzibar
ОАЭ	nodeUAE
Дубай	nodeDubai
Шарджа	nodeSharjah
Абу-Даби	nodeAbuDhabi
Италия	nodeItaly
Катания	nodeCatania
Рим	nodeRome
Милан	nodeMilan
Китай	nodeChina
Пекин	nodeBeijing
Гонконг	nodeHongKong
Шанхай	nodeShanghai
Сейшелы	nodeSeychelles
Маэ	nodeMahe
Сёрф	nodeCerf

⇨ Создадим информационную панель, на которой будет содержаться необходимые компоненты для формирования путевки по требованиям пользователя.

Panel

Элемент Panel представляет панель и также, как и GroupBox, объединяет элементы в группы. Она может визуально сливаться с остальной формой, если она имеет, то же значение цвета фона в свойстве BackColor, что и форма. Чтобы ее выделить можно кроме цвета указать для элемента границы с помощью свойства BorderStyle, которое по умолчанию имеет значение None, то есть отсутствие границ. Компонент находится во вкладке «Контейнеры» панели элементов.

24. Разместите на форме компонент Panel, укажите в свойствах компонента размер 724; 386 и координаты 12; 246.

25. В свойствах компонента установите значение **BorderStyle** на FixedSingle.

⇨ Добавим компоненты для расчета прибытия и размещения в любом из городов.

DateTimePicker и MonthCalendar

DateTimePicker представляет раскрывающийся по нажатию календарь, в котором можно выбрать дату. собой элемент, который с помощью перемещения ползунка позволяет вводить числовые значения.

С помощью *MonthCalendar* также можно выбрать дату, только в данном случае этот элемент представляет сам календарь, который не надо раскрывать.

Для выбора даты заезда и даты отъезда можно использовать как компонент DateTimePicker, так и MonthCalendar.

26. Разместите на форме два компонента Label. Измените свойство **Text** на Дата заезда и Даты отъезда (рис.1.9).

27. Добавьте на форму два компонента DateTimePicker и разместите их согласно рис.1.9.

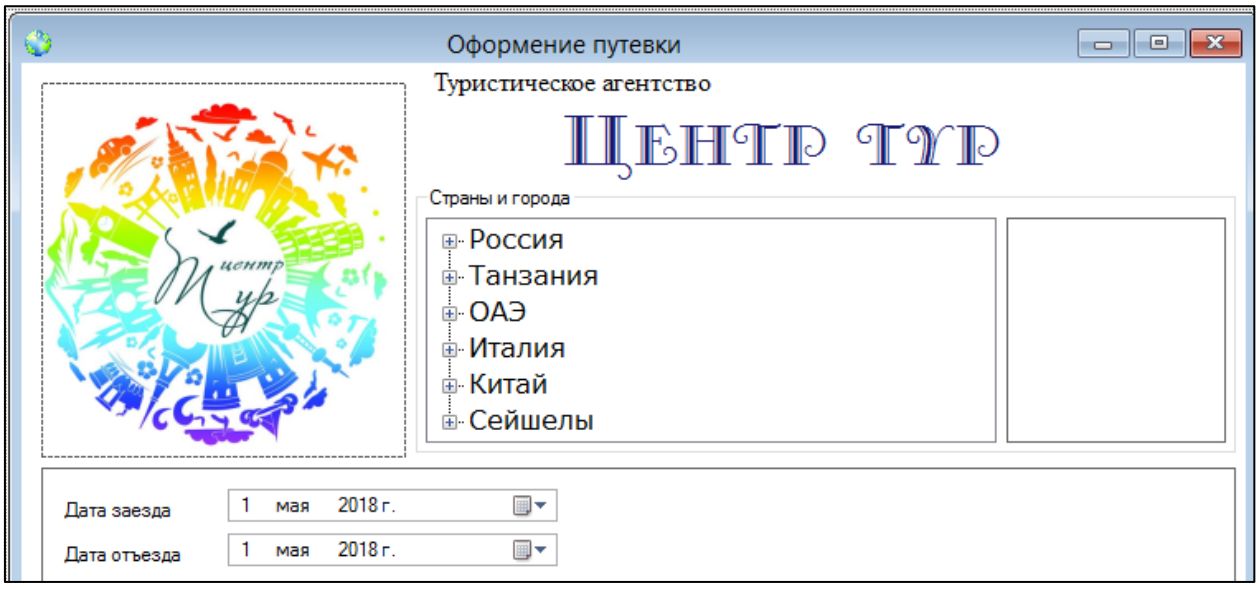


Рис. 1.9. Добавление компонентов DateTimePicker

ComboBox

Элемент ComboBox образует выпадающий список. Для хранения элементов списка в ComboBox также предназначено свойство Items.

28. Разместите на форме два компонента Label. Измените свойство Text на Отель и Транспорт (рис.1.10).

29. Добавьте на форму два компонента ComboBox. Для ComboBox1 установите размер 289; 21, а для ComboBox2 – 147; 21. Разместите компоненты согласно рис.1.10. Выпадающий список прописывается в свойстве Items, но мы с вами будем использовать другой способ, распишем его программно.

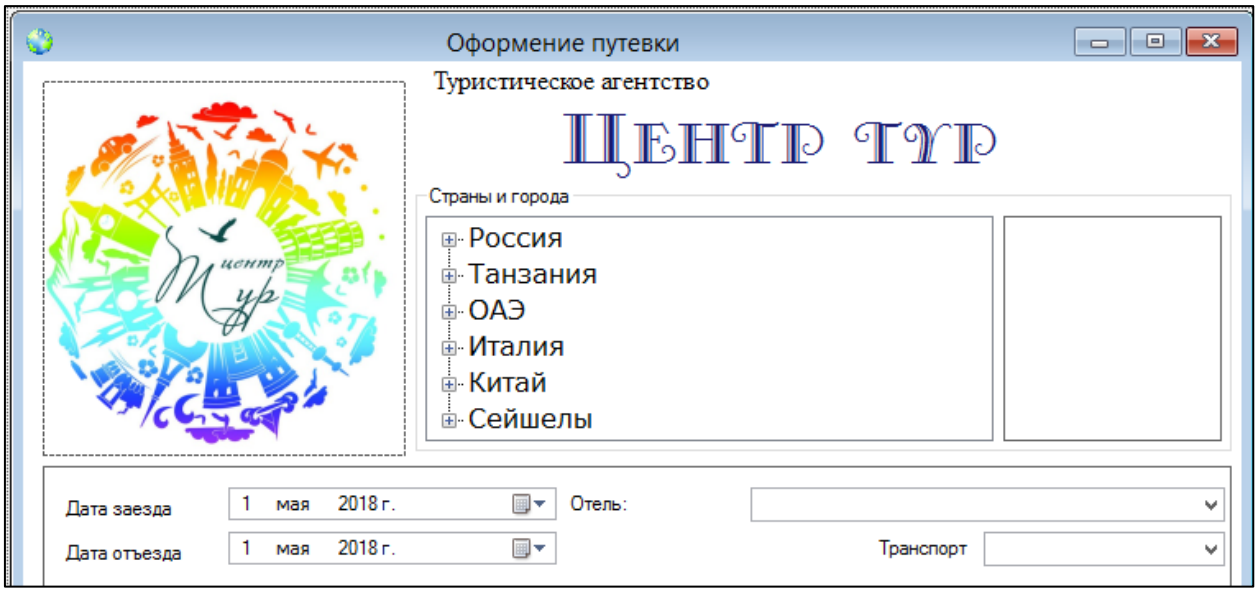


Рис. 1.10. Добавление компонентов ComboBox

NumericUpDown

Элемент NumericUpDown представляет пользователю выбор числа из определенного диапазона. Для определения диапазона чисел для выбора NumericUpDown имеет два свойства: Minimum (задает минимальное число) и Maximum (задает максимальное число). Само значение элемента хранится в свойстве Value.

По умолчанию элемент отображает десятичные числа. Однако если мы установим его свойство Hexadecimal равным true, то элемент будет отображать все числа в шестнадцатеричной системе.

30. Разместите на форме компонент Label. Измените свойство **Text** на Количество человек (рис.1.11).

31. Добавьте на форму компонент NumericUpDown. Установите размер 69; 20. В свойствах компонента укажите: **Minimum** – 1, **Maximum** – 10, **Value** – 1. Разместите компоненты согласно рис.1.11.

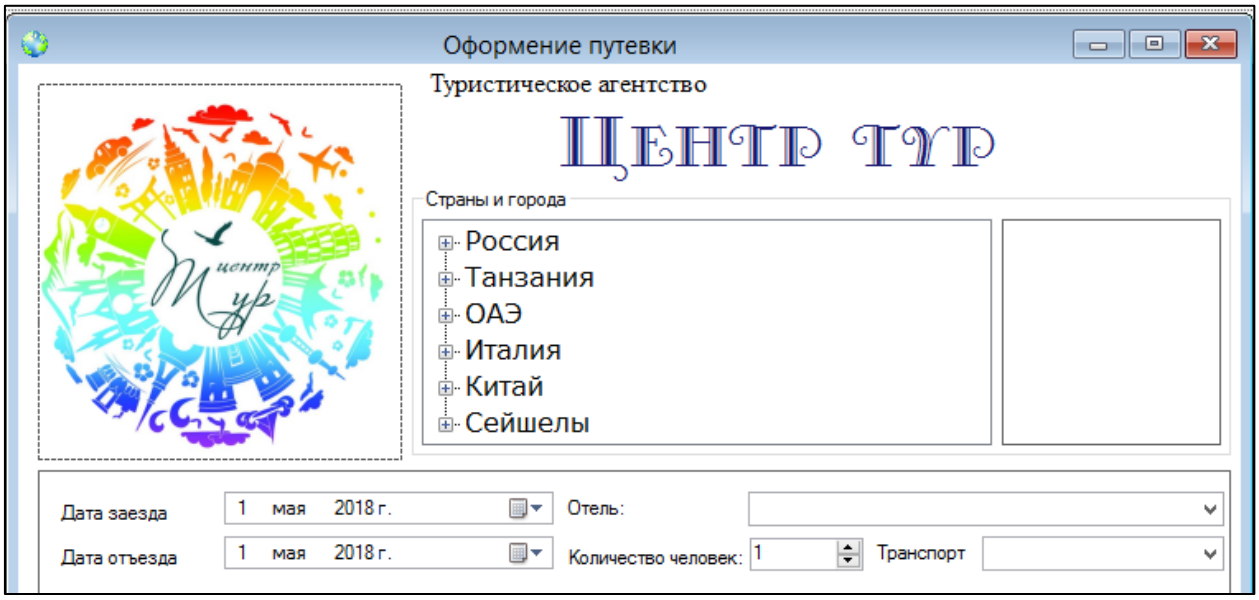


Рис. 1.11. Добавление компонента NumericUpDown

⇨ Добавим компоненты для расчета питания, страховки и удобств.

32. Добавьте три компонента GroupBox на форму. В свойстве **Text** задайте значения Питание, Страховка, Удобства для каждого компонента соответственно. Разместите компоненты согласно рис.1.12.

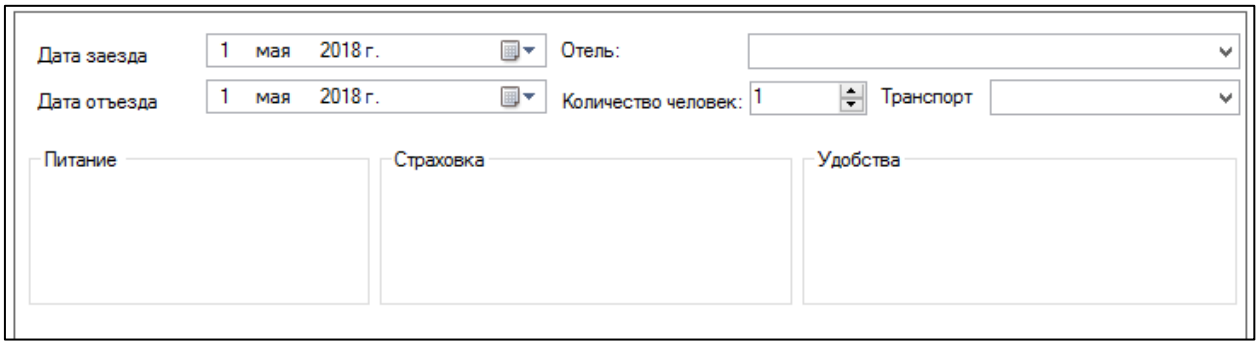


Рис. 1.12. Добавление компонентов GroupBox

RadioButton

RadioButton или переключатель. Переключатели располагаются группами, и включение одного переключателя означает отключение всех остальных.

Чтобы установить у переключателя включенное состояние, надо присвоить его свойству *Checked* значение true.

Для создания группы переключателей, из которых можно бы было выбирать, надо поместить несколько переключателей в какой-нибудь контейнер, например, в элемент GroupBox. Переключатели, находящиеся в разных контейнерах, будут относиться к разным группам.

33. В первый контейнер GroupBox добавьте три компонента RadioButton. В свойстве **Text** задайте значения Завтрак, Все включено, Нет, для каждого компонента соответственно (рис.1.13).

34. Для компонента RadioButton (Нет) в свойстве **Checked** (Указывает состояние переключателя) установите True.

CheckBox

Элемент CheckBox или флажок предназначен для установки одного из двух значений: отмечен или не отмечен. Чтобы отметить флажок, надо установить у его свойства *Checked* значение true.

Кроме свойства Checked у элемента CheckBox имеется свойство *CheckState*, которое позволяет задать для флажка одно из трех состояний – Checked (отмечен), Indeterminate (флажок не определен – отмечен, но находится в неактивном состоянии) и Unchecked (не отмечен)

Также следует отметить свойство *AutoCheck* – если оно имеет значение false, то мы не можем изменять состояние флажка. По умолчанию оно имеет значение true.

35. Во второй контейнер GroupBox добавьте три компонента CheckBox. В свойстве **Text** задайте значения Страхование багажа, Страхование жизни, Страхование от невыезда, для каждого компонента соответственно (рис.1.13).

CheckedListBox

Для каждого элемента такого списка определено специальное поле CheckBox, которое можно отметить.

Все элементы задаются в CheckedListBox задаются в свойстве *Items*. Также, как и для элемента ComboBox, мы можем задать набор элементов. По умолчанию для каждого добавляемого нового элемента флажок не отмечен

36. В третий контейнер GroupBox добавьте компонента CheckedListBox (рис.1.13). В свойстве **Items** введите значения:

Трансфер от/до аэропорта
Размещение домашних животных
Кондиционер
Балкон

Каждый элемент вводится с новой строки.

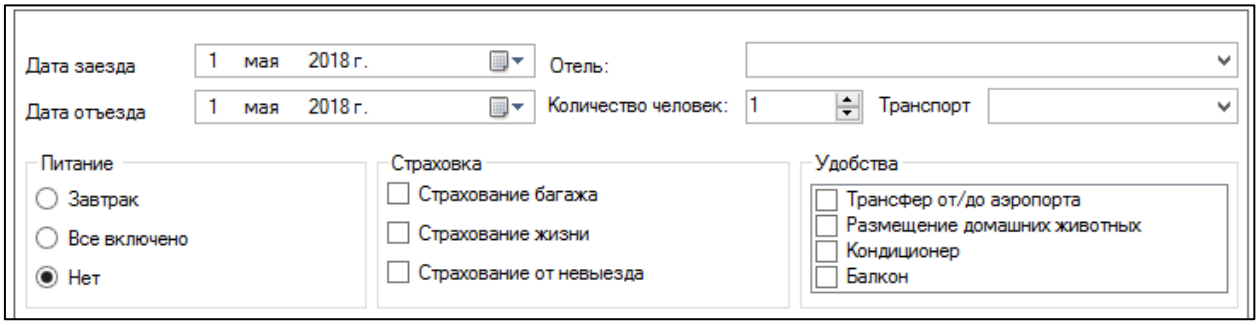


Рис. 1.13. Добавление компонентов RadioButton, CheckBox и CheckedListBox

⇨ Добавим компоненты для заполнения пользователями информации о себе.

37. Добавьте компонент GroupBox на форму. В свойстве **Text** задайте значение Информация о себе. Установите размер 711; 100. Разместите компоненты согласно рис.1.14.

TextBox

Для ввода и редактирования текста предназначены текстовые поля - элемент TextBox. Так же, как и у элемента Label текст элемента TextBox можно установить или получить с помощью свойства *Text.*

По умолчанию при переносе элемента с панели инструментов создается однострочное текстовое поле. Для отображения больших объемов информации в текстовом поле нужно использовать его свойства *Multiline* и *ScrollBars*. При установке для свойства Multiline значения true, все избыточные символы, которые выходят за границы поля, будут переноситься на новую строку. Кроме того, можно сделать прокрутку текстового поля, установив для его свойства ScrollBars одно из значений.

38. Разместите на форме пять компонентов Label. В свойстве **Text** задайте значения Фамилия, Имя, Отчество, E-mail, Паспорт, для каждого компонента соответственно. Разместите компоненты согласно рис.1.14.

39. Добавьте на форму пять компонентов TextBox. Разместите компоненты согласно рис.1.14.

MaskedTextBox

Элемент MaskedTextBox по сути представляет обычное текстовое поле. Однако данные элемент позволяет контролировать ввод пользователя и проверять его автоматически на наличие ошибок.

Чтобы задать маску, надо установить свойство *Mask* элемента. Найдя это свойство в окне свойств(Porperties), нажмем на него и нам отобразится окно для задания одного из стандартных шаблонов маски.

40. Добавьте еще один компонент Label. В свойстве **Text** задайте значение Телефон. Разместите компонент согласно рис.1.14.

41. Разместите на форме компонент MaskedTextBox согласно рис.1.14. В свойстве **Mask** выберете маску **Phone number** и нажмите ОК.

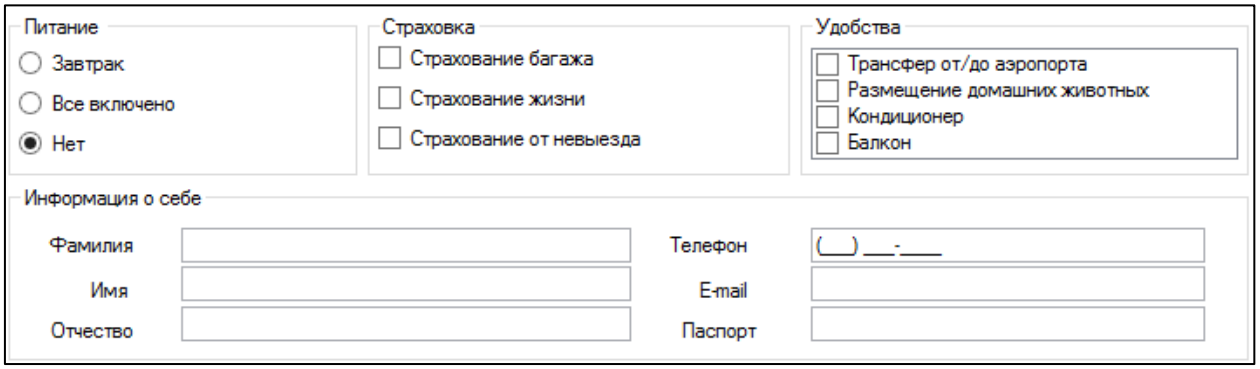


Рис. 1.14. Добавление компонентов Text и MaskedTextBox

RichTextBox

Обеспечивает дополнительные возможности ввода и редактирования текста (например, форматирование символов и абзацев).

42. Добавьте компонент GroupBox на форму. В свойстве **Text** задайте значение Дополнительные сведения. Установите размер 431; 95. Разместите компонент согласно рис.1.15.

43. Перетащите компонент RichTextBox в контейнер. Установите размер 418; 70 (рис.1.15).

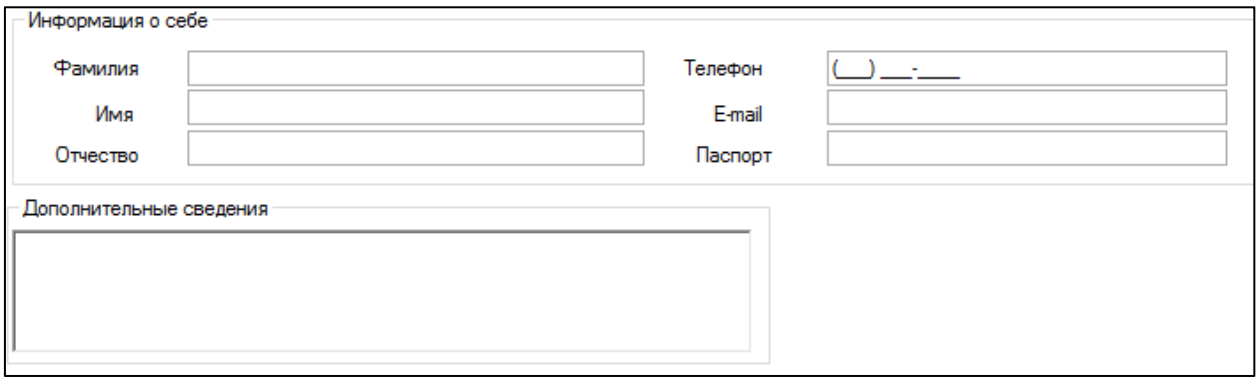


Рис. 1.15. Добавление компонента RichTextBox

Button

Наиболее часто используемым элементом управления является компонент Button. Он позволяет разместить на форме кнопку. Кнопкой называется элемент управления, все взаимодействие пользователя с которым ограничивается одним действием – нажатием. Все, что вам необходимо сделать при работе с кнопкой, – это поместить ее в нужном месте формы и назначить ей соответствующий обработчик.

44. Добавьте компонент GroupBox на форму. В свойстве Text задайте значение Цена путевки. Установите размер 271; 95. Разместите компонент согласно рис.1.16.

45. Добавьте компонент Label. В данном случае, он будет служить нам для отображения подсчета стоимости. Для этого в свойстве **Text** оставьте пустое значение. Установите размер 265; 53. Перетащите компонент в контейнере GroupBox согласно рис.1.16.

46. Разместите на форме компонент Button. В свойстве Text задайте значение Пересчитать цену путевки. Установите размер 271; 23. Перетащите компонент в контейнер GroupBox согласно рис.1.16.

47. Добавьте на форму еще один компонент Button. В свойстве Text задайте значение Выбрать путевку. Установите размер 724; 37. Сделайте фоновой цвет – MidnightBlue, а шрифт – White. Разместите компонент согласно рис.1.16.

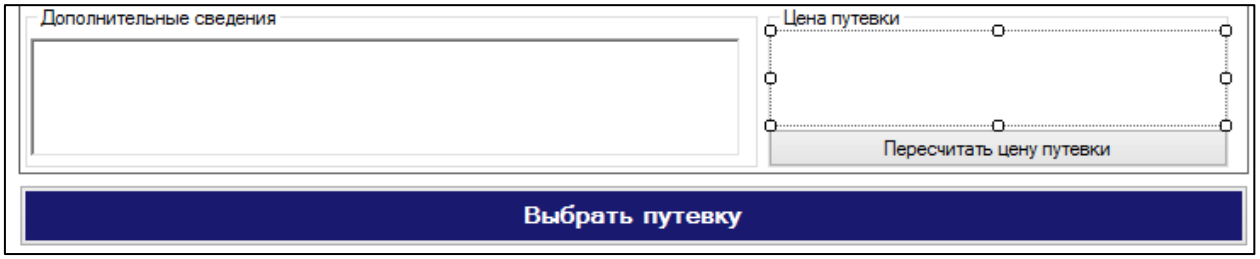


Рис. 1.16. Добавление компонентов Button

⇨ Добавим компоненты для более удобного использования приложения.

LinkLabel

Особый тип меток представляют элементы LinkLabel, которые предназначены для вывода ссылок, которые аналогичны ссылкам, размещенным на стандартных веб-станиц.

48. Разместите на форме компонент LinkLabel под кнопкой «Выбрать путевку». В свойстве Text задайте значение ссылки на сайт, например, c-tur.ru

NotifyIcon

Элемент NotifyIcon позволяет задать значок, который будет отображаться при запуске приложения в панели задач.

49. Перенесите элемент NotifyIcon на форму с панели инструментов. После этого добавленный компонент NotifyIcon отобразится внизу дизайнера формы. Затем задайте для свойства **Icon** какую-нибудь иконку в формате .ico. И также установите для свойства **Visible** значение true. При запуске иконка будет отображаться в панели задач.

ToolTip

Отображает информацию при наведении указателя мыши на соответствующий элемент управления.

50. Перенесите элемент ToolTip на форму с панели инструментов. После этого добавленный компонент отобразится внизу дизайнера формы. Чтобы добавить подсказку к какому-либо элементу, необходимо активизировать данный элемент, например, LinkLabel и в свойстве **ToolTip на toolTip1** указать подсказку – **Перейти на сайт.**

51. Сохраните приложение.

Окончательный дизайн формы изображен на рис. 1.17.

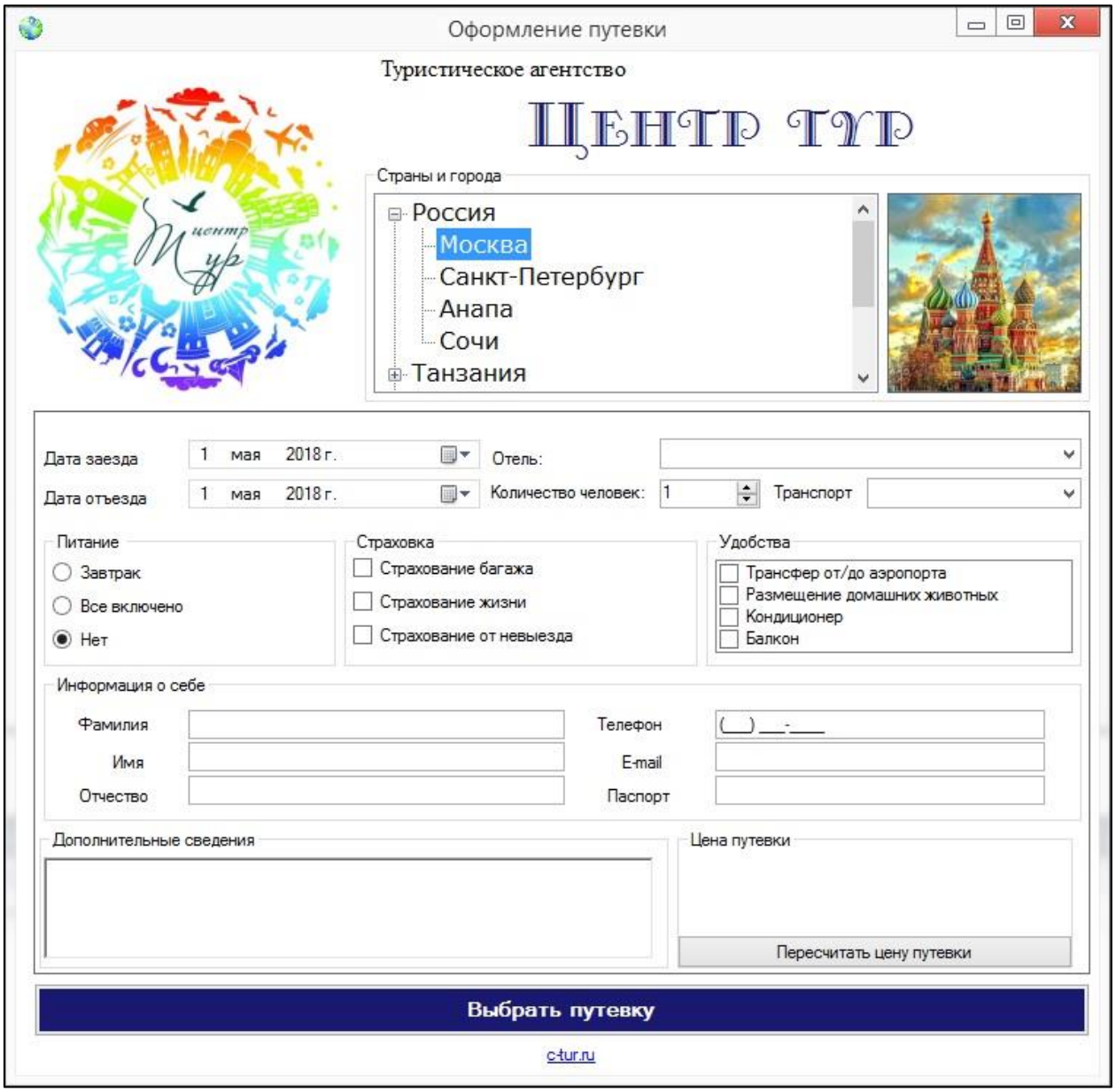


Рис. 1.17. Дизайн формы приложения «Оформление путевки»

⇨ Добавим программный код и обработчики событий для нашего приложения.

52. Для начала определим классы:

1) Класс Отель (характеризуется наименованием отеля, количеством звезд и ценой за одну ночь).

2) Класс Транспорт (характеризуется типом транспорта и ценой).

3) Класс Путешествие (характеризуется страной, городом, отелями, транспортом, доплатой за каждого дополнительного человека (в процентах), стоимостью визы).

53. Добавьте описание класса Отель, а также функцию, которая реализует полное наименование отеля, включающее в себя наименование и количество звезд (именно в таком формате данные будут отображаться в

элементе ComboBox при выборе отеля на форме) и конструктор класса для присваивания значений полям класса.

```
class Hottel
        {
            //  Класс описания отеля
            public string Name { set; get; }   //  Наименование отеля
            public byte Stars { set; get; }    //  Количество звезд
            public float Price { set; get; }   //  Цена за одну ночь
            public string Fullname
            {
                get
                {
                    return Name + " " + Stars.ToString() + "*";
                }
            }
            public Hottel(string HottelName, byte HottelStars, float HotelPrice)
            {
                Name = HottelName;
                Stars = HottelStars;
                Price = HotelPrice;
            }
               }
```

54. Добавьте описание класса Транспорт, определите перечисления типа транспорт и конструктор класса для присваивания значений полям класса.

```
class Transport
        {
            //  Класс описания транспорта
            public enum TypeTransport {Самолет, Поезд, Автобус}
            public TypeTransport typeTransport { set; get; }   //  Тип транспорта
            public float Price { set; get; }               //  Цена на билет
            public Transport(TypeTransport transp, float price)
            {
                typeTransport = transp;
                Price = price;
            }
               }
```

55. Добавьте описание класса Путешествие и конструктор класса для присваивания значений полям класса.

В каждом городе есть свой список отелей и до каждого города мы можем добраться определенным типом транспорта (или несколькими), для этого мы вводим Класс List (список).

```
class travel
        {   //  Определяем класс описания возможностей для путешествия
            //  в определенный город, определенной страны
            public string Country { get; set; }    //  Страна
            public string City { set; get; }       //  Город
            public List <Hottel> Hottels;          //  Отели
            public List<Transport> Transports;     //  Транспорт
            public int SurchargePerson { get; set; }  //  Доплата за каждого дополнительного
человека (в процентах)
            public int Visa { get; set; }      //  Стоимость визы в страну
            public travel(string country, string city, List<Hottel> hottel, List<Transport>
transport, int surchargePerson, int visa)
            {
                Country = country;
```

```
            City = city;
            Hottels = hottel;
            Transports = transport;
            SurchargePerson = surchargePerson;
            Visa = visa;
        }
           }
```

56. Задайте список всех туров Travel.

```
List<travel> Travel = new List<travel>();
```

57. Добавьте функцию, которая по наименованию страны и города выдает значения доплаты за каждого дополнительного человека.

Примечание. Лямбда-выражения позволяют создать емкие лаконичные методы, которые могут возвращать некоторое значение и которые можно передать в качестве параметров в другие методы.

```
public int GetSurchargePerson(string country, string city)
        {
            travel item = Travel.Find(x => (x.Country == country) && (x.City == city));
            return item.SurchargePerson;
               }
```

58. Добавьте функцию, которая по наименованию страны и города выдает стоимость визы.

```
public int GetVisa(string country, string city)
        {
            travel item = Travel.Find(x => (x.Country == country) && (x.City == city));
            return item.Visa;
               }
```

59. Добавьте программный обработчик событий в конструктор формы.

Заполняем весь наш список путешествий, на примере для города Москва.

Сформируйте список отелей:

```
public Form1()
        {
                   InitializeComponent();

                   List<Hottel> MoscowHottels = new List<Hottel>();
```

Заполните данный список через конструктора класса.

Функция Add добавляет элемент в список new Hotel, создает экземпляр класса и вызывает конструктор этого класса, т.е. вводимые в скобках значения присваиваются полям класса.

```
MoscowHottels.Add(new Hottel("Гостиница Интерконтиненталь", 5, 28880));
MoscowHottels.Add(new Hottel("Вилла Кадаши Бутик", 4, 13000));
MoscowHottels.Add(new Hottel("Меркюр", 3, 5500));
MoscowHottels.Add(new Hottel("Отель на Цветном Бульваре", 2, 1890));
MoscowHottels.Add(new Hottel("Отель Post", 1, 1500));
```

60. Подобным образом создайте список транспорта, которым можно добраться до Москвы.

```
List<Transport> MoscowTransports = new List<Transport>();
           MoscowTransports.Add(new Transport(Transport.TypeTransport.Самолет, 4000));
           MoscowTransports.Add(new Transport(Transport.TypeTransport.Поезд, 4864));
           MoscowTransports.Add(new Transport(Transport.TypeTransport.Автобус, 2688));
```

61. В список всех туров поместите путешествие Москва.

```
Travel.Add(new travel("Россия", "Москва", MoscowHottels, MoscowTransports, 60, 0));
```

62. Самостоятельно добавьте в список остальные туры.

63. Перейдите обратно на форму, активизируйте элемент TreeView и в окне свойств выберете обработчик событий **AfterSelect** (Происходит при изменении выбора).

```
private void treeView1_AfterSelect(object sender, TreeViewEventArgs e)
        {
           //  событие, которое срабатывает после выбора узла в компоненте TreeView
           // (изначально элементы неактивны)
            panel1.Enabled = treeView1.SelectedNode.Level == 1;

            //  Покажем картинку для выбранного node
            pictureBox2.ImageLocation     =     AppDomain.CurrentDomain.BaseDirectory     +
(string)e.Node.Tag;
         //  Доступным для редактирования сделаем если выбран node второго уровня т.к.
     // у нас в узлах первого уровня расположены наименования стран, наша задача выбрать
     // тур только в определённый город страны, а они у нас расположены на втором уровне
            if (treeView1.SelectedNode.Level == 1)
            {

                //  Заполним все необходимые компоненты (ComboBox)
                //  Отели, транспорт доступный для этого города
                foreach (travel tr in Travel)
                {
                    if ((tr.Country == treeView1.SelectedNode.Parent.Text) && (tr.City ==
treeView1.SelectedNode.Text))
                    {
                        cbHotel.DataSource = tr.Hottels;
                        cbHotel.DisplayMember = "FullName";
                        cbHotel.ValueMember = "Price";

                        cbTransport.DataSource = tr.Transports;
                        cbTransport.DisplayMember = "typeTransport";
                        cbTransport.ValueMember = "Price";
                    }
                }
            }
        }
```

64. Добавьте обработчик событий **Click** (Возникает при щелчке элемента управления) для кнопки «Пересчитать цену путевки».

```
  private void button1_Click(object sender, EventArgs e)
        {
            int PriceTour = 0;
            //Определим, количество дней
            int CountDay = (dateTimePicker2.Value.Date - dateTimePicker1.Value.Date).Days;
            //Начнем с оплаты визы в страну
```

```
            PriceTour            =            GetVisa(treeView1.SelectedNode.Parent.Text,
treeView1.SelectedNode.Text);
            // Подсчитаем, сколько будет стоить проживание в данной гостинице для одного
человека
            PriceTour += Convert.ToInt32(cbHotel.SelectedValue.ToString()) * CountDay;
            //Прибавим цену за проезд к месту отдыха
            PriceTour += Convert.ToInt32(cbTransport.SelectedValue.ToString());
            // Учтем питание (Если без питания, то не прибавляем)
            PriceTour += (rbBreakfast.Checked) ? 755 * CountDay : 0;
            PriceTour += (rbAllInclusive.Checked) ? 2100 * CountDay : 0;
            //  Страховка
            PriceTour += (cbInsuranceBag.Checked) ? 5000 : 0;
            PriceTour += (cbInsuranceLive.Checked) ? 5000 : 0;
            PriceTour += (cbInsuranceDesertir.Checked) ? 5000 : 0;
            //  Удобства в номере
            foreach (int item in clConveniences.CheckedIndices)
            {
                PriceTour += (item == 0) ? 500 : 0;  //  Трансфер до аэропорта
                PriceTour += (item == 1) ? 1000 : 0; //  Размещение домашних животных
                PriceTour += (item == 2) ? 100 : 0;  //  Кондиционер
                PriceTour += (item == 3) ? 500 : 0;  //  Балкон
            }
            //  Доплата за каждого дополнительного человека
            PriceTour  +=  PriceTour  *  ((int)numericUpDown1.Value  -  1)  /  100  *
GetSurchargePerson(treeView1.SelectedNode.Parent.Text, treeView1.SelectedNode.Text);
            label13.Text = PriceTour.ToString();
               }
```

65. Добавьте обработчик событий **Click** для кнопки «Выбрать путевку».

```
private void button2_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Спасибо!\nЗаказ                отправлен                на
обработку.","",MessageBoxButtons.OK,MessageBoxIcon.Information);
        }
```

66. Перейдите обратно на форму и добавьте элемент ErrorProvider (Предоставляет интерфейс пользователя, указывающий пользователю о наличии ошибки в элементе управления формы).

Активизируйте элемент textBox4 (форма для заполнения e-mail) и добавьте обработчик событий **Validating** (Возникает при проверке действительности элемента управления).

```
private void textBox4_Validating(object sender, System.ComponentModel.CancelEventArgs e)
        {
            errorProvider1.Clear();
            if (textBox4.Text == string.Empty) return;
            try
            {
                var mail = new System.Net.Mail.MailAddress(textBox4.Text);
                e.Cancel = false;

            }
            catch
            {
                e.Cancel = true;
                errorProvider1.SetError(textBox4, "E-Mail адрес введен некорректно");
            }
               }
```

67. Чтобы ссылку на сайт сделать рабочей, необходимо для элемента linkLabel добавить обработчик событий **LinkClicked** (Происходит при выборе данной ссылки).

```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            System.Diagnostics.Process.Start("IExplore.exe", "http://www.c-tur.ru");
               }
```

68. Для возможности открывать приложение из панели задач, необходимо для элемента NotifyIcon добавить обработчик событий MouseDoubleClick (Происходит при двойном щелчке данного компонента).

```
private void notifyIcon1_MouseDoubleClick(object sender, MouseEventArgs e)
        {
            this.WindowState = FormWindowState.Normal;
               }
```

69. Сохраните проект и запустите программу для проверки работоспособности, попробуйте протестировать различные виды туров. Результат показан на рис. 1.18.

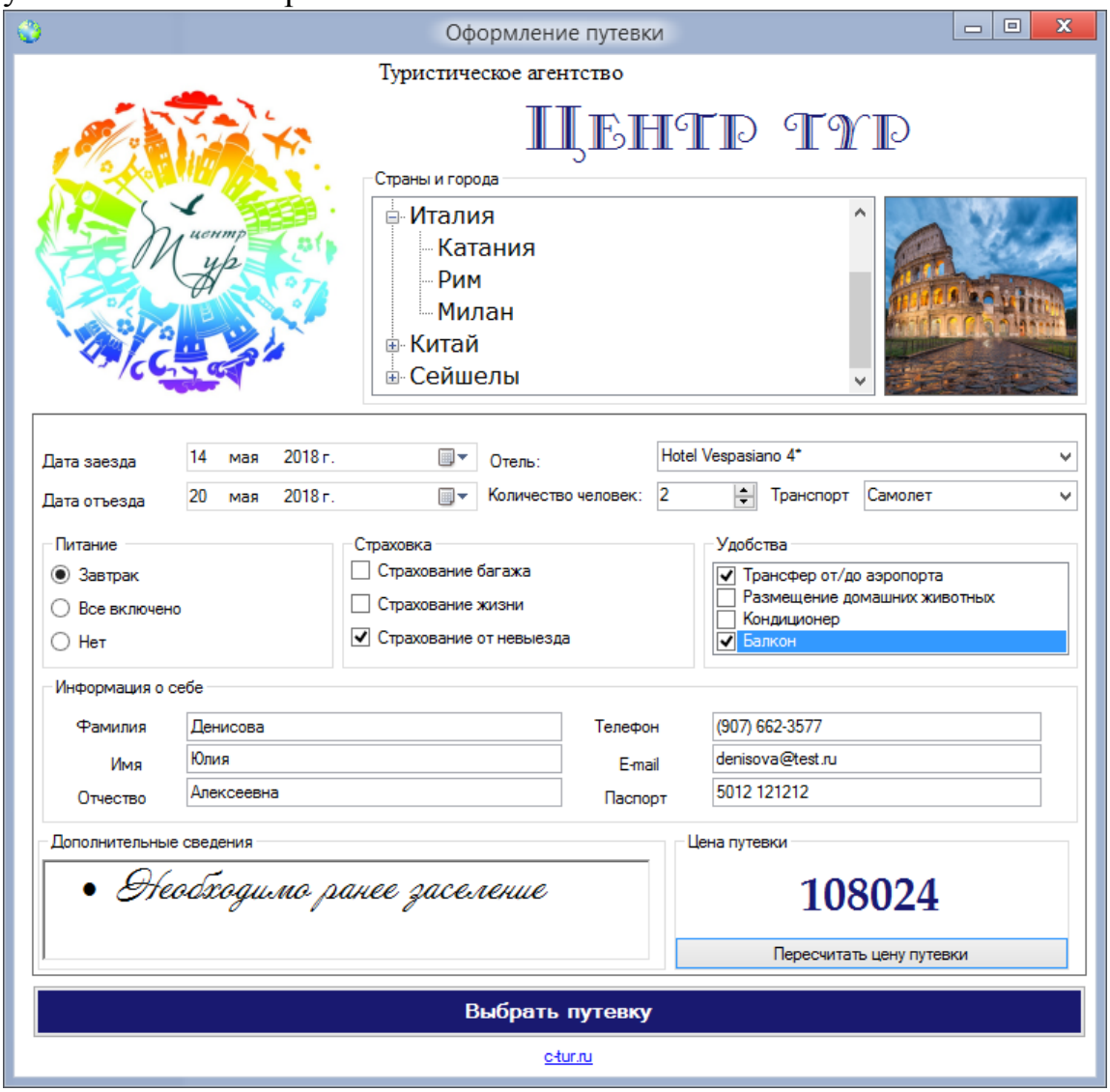


Рис. 1.18. Приложение для формирования подсчета стоимости туристической путевки

1.2. Задачи для самостоятельной работы

1. Напишите программу, которая пересчитывает скорость ветра из «метров в секунду» в «километров в час». Программа должна быть спроектирована таким образом, чтобы пользователь мог ввести в поле Скорость только целое положительное число.

2. Напишите программу, которая вычисляет сопротивление электрической цепи, состоящей из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Если величина сопротивления превышает 1 000 Ом, то результат должен быть выведен в килоомах.

3. Напишите программу, которая вычисляет стоимость поездки на автомобиле, например, от города до дачного участка.

4. Разработать приложение анкета для студентов со следующими вопросами: 1.Сколько вам лет? 2. На каком факультете вы учитесь? 3. в какой группе вы учитесь? 4. Нравиться ли вам учиться? Результат анкетирования должен быть представлен тут же, содержащий всю Информацию, полученную при опросе.

5. Напишите программу, которая в зависимости от введенного возраста выдавала соответствующую надпись (менее 20 лет – Почему вы не в школе? от 20 до 40 – Молодым везде дорога! от 40 до 60 – Главное – побольше здоровья! от 60 – Почетный возраст).

6. Напишите программу, которая пересчитывает массу из фунтов в килограммы (1 фунт=409,5 грамм). Программа должна быть спроектирована таким образом, чтобы кнопка <Пересчет> была доступна только в том случае, если пользователь ввел исходные данные.

7. Напишите программу, которая вычисляет доход по вкладу. Программа должна обеспечить расчет простых и сложных процентов. Простые проценты начисляются в конце срока вклада, сложные начисляются ежемесячно и прибавляются к первоначальной сумме вклада, в следующем месяце проценты начисляются на новую сумму.

8. Разработать программу для перевода чисел из одной меры весов в другую.

9. Разработать приложение, определяющую скидку на покупку автомобиля по следующим данным: возраст (старше 10 лет или нет) – 10%, аварийная (да или нет) – 30%, таможня (уплачены таможенные пошлины или нет) – 10%, пробег по РФ (есть или нет пробега по РФ) – 5%.

10. Усовершенствовать программу Стоимость машины: есть список, в который можно добавить или удалить автомобиль (с его характеристикой), а также проверить, какая скидка предоставляется на него.

11. Напишите программу, которая вычисляет стоимость покупки. Пользователь должен вводить код товара и количество единиц. Программа должна формировать список товаров.

12. Напишите программу, которая вычисляет стоимость с учетом скидки. Скидка 1% предоставляется, если сумма покупки больше 300р., 2% – если сумма больше 500р., 3% – если сумма больше 1000. Информация о предоставлении скидки (процент и величина) должна быть выведена в диалоговом окне.

13. Напишите программу, которая по коэффициентам квадратного уравнения вычисляет корни уравнения.

14. Напишите программу, вычисляющую площадь треугольника по длине сторон. Рассмотреть, когда треугольник прямоугольный, равнобедренный, равносторонний, произвольный.

15. Напишите программу, вычисляющую сумму покупки. Пользователь вводит название товара и его цену, а также должна быть возможность предоставления скидки и для сотрудника компании, и для постоянного клиента, и для покупателя представившего дисконтную карту. Программа должна формировать список товаров.

16. Напишите программу, вычисляющую пройденный путь, свободно падающим телом через t секунд. Программу составить так, чтобы можно было вычислить пройденный путь, если оно имело начальный путь.

17. Усовершенствовать задачу «Поездка на дачу» № 3. Сформировать поле с выпадающим списком автомобилей (программно, для каждого автомобиля известен расход топлива), при выборе автомобиля автоматически подсчитывается стоимость поездки.

18. Напишите программу, которая вычисляет объем правильных пространственных фигур (тетраэдр, куб, 4-х угольная пирамида).

19. Напишите программу, которая формирует туристический маршрут, состоящий из 5 пунктов, названия пунктов выбираются из выпадающего списка.

20. Напишите программу, вычисляющую время, которое нужно лодки, чтобы проплыть определенное расстояние. Программа должна быть сконструирована так, чтобы была возможность вычисления времени полного пути, т.е. пути туда и обратно.

2. Графика

Многие программы не ограничиваются указанными выше компонентами для организации интерфейса программы. Для улучшения вида программы можно использовать графические возможности среды.

Нарисованные изображения отображаются на форме при выполнении программы с помощью различных инструментов.

Изображение при этом представляет собой комбинацию простейших фигур – графических примитивов (точка, линия, круг или прямоугольник).

Каждая точка на форме имеет координаты X и Y. Текущая позиция при рисовании определяется горизонтальной (X) и вертикальной (Y) координатами, заданными в пикселях. Координата X возрастает при перемещении указателя слева на право, а координата Y– при перемещении его сверху вниз.

Объект **Graphics** – это указатель на место, где будут рисоваться примитивы. Пусть мы хотим рисовать в форме Windows. Синтаксис задания ссылки на нее: `Graphics g = Graphics.FromHwnd(this.Handle);`

В C# инструменты рисования определены в пространстве имен System.Drawing. Там находятся классы:

- Pen (перо). Объекты пера используются в методах рисования линий и контуров геометрических фигур.
- Brush (кисть). Объекты кисти используются в методах заливки областей, ограниченных контурами.

В C# определены методы рисования линий, фигур и заливки. Все методы перегружаемые, то есть выполняются по-разному с разными аргументами.

- Pen – при рисовании можно использовать перо с разными стилями линий LineStyle. Например, solid (сплошная), Dash (штрих), Dot (пунктир), DashDot (штрих-пунктир), DashDotDot (штрих-пунктир-пунктир).
- DrawLine – прямая линия между двумя точками.
- DrawRectangle – прямоугольник.
- DrawEllipse – эллипс.
- Заливка разных фигур осуществляется простой кистью с Brush.Cyan и кистью HatchBrush с разными стилями заливки DashStyle.

2.1. Линейная графика

Постановка задачи: Разработать приложение, которое рисует олимпийский флаг.

1. Создайте новый проект WindowsForms. Назовите проект OlympicFlag.

2. Активизируйте форму и измените ее имя на Олимпийский флаг. Укажите размер 385; 290 (рис.2.1).

3. Измените стандартное значение свойства формы **AutoScaleMode** (Определяет изменение масштаба формы или элемента управления при изменении разрешения экрана или шрифтов) – None на Font.

4. Добавьте компонент PictureBox на форму (рис.2.1).

5. Установите свойства компонента согласно таб.2.

Таблица 2.

Свойство	Значение
BackColor	White
BorderStyle	FixedSingle
Size	345; 226
Name	picture

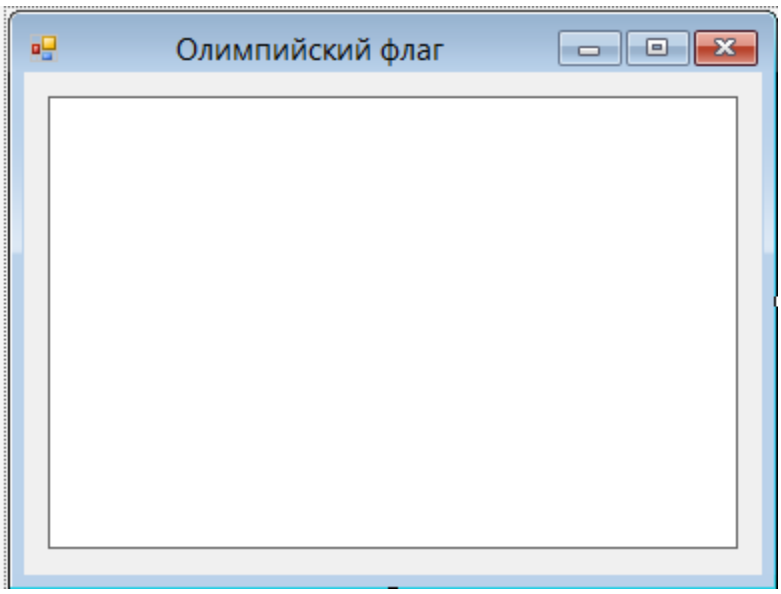


Рис 2.1. Форма для рисунка

6. Перейдите в редактор кода и опишите функцию Draw, которая будет рисовать наш флаг.

```
public partial class Form1 : Form
    {
        public Form1()
```

```
        {
            InitializeComponent();
            Draw();
               }
```

7. Рисование в C# происходит в объекте Graphics, используя встроенные функции DrawEllipse, DrawLine и другие.

Для рисования Олимпийского флага нам понадобится только DrawEllipse, где new Pen – это цвет и толщина линии, а new Rectangle – это координаты куда будет вписана наша окружность.

Для того чтобы увидеть результат рисования, свойству pictureBox необходимо присвоить нашему растровому объекту рисования формат bmp.

Запишите функцию рисования в редакторе кода.

```
private void Draw()
        {
            //  Объект Bitmap хранит в себе любой растровый объект
            //  Длину и ширину зададим по PicterBox
            Bitmap bmp = new Bitmap(picture.Width, picture.Height);

            Graphics g = Graphics.FromImage(bmp);

            g.DrawEllipse(new Pen(Color.Blue, 8), new Rectangle(40, 40, 80, 80));
            g.DrawEllipse(new Pen(Color.Black, 8), new Rectangle(130, 40, 80, 80));
            g.DrawEllipse(new Pen(Color.Red, 8), new Rectangle(220, 40, 80, 80));
            g.DrawEllipse(new Pen(Color.Yellow, 8), new Rectangle(85, 80, 80, 80));
            g.DrawEllipse(new Pen(Color.Green, 8), new Rectangle(175, 80, 80, 80));
            picture.Image = bmp;
        }
  }
```

8. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис. 2.2.

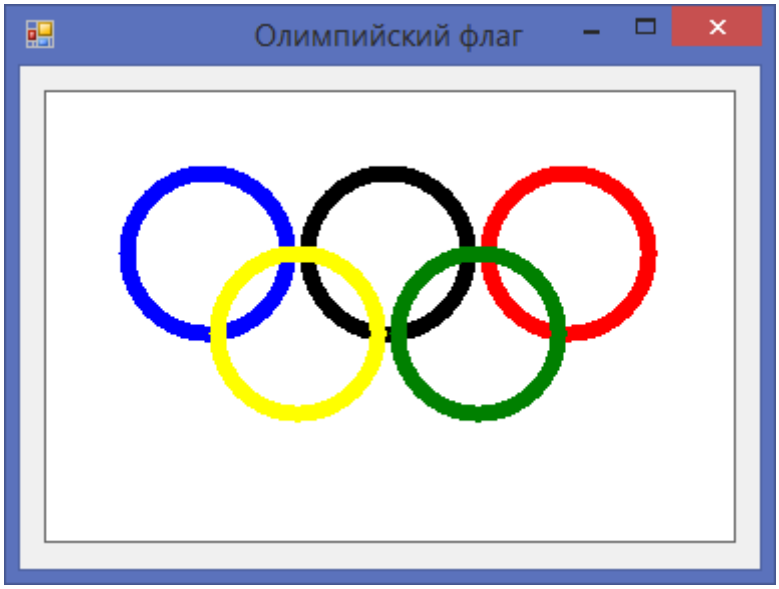


Рис. 2.2. Олимпийский флаг

2.2. График функции

Постановка задачи: Разработать приложение, которое на поверхности формы вычерчивает график функции, например, $2\sin(x)\,e^{x}$.

1. Создайте новый проект WindowsForms. Назовите проект GraphicFunction.

2. Активизируйте форму и измените ее имя на Построение графика функции. Укажите размер 812; 497.

3. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

4. Добавьте компонент PictureBox на форму.

5. Установите свойства компонента согласно таб.3.

Таблица 3.

Свойство	Значение
Dock	Fill
Modifiers	Public
Name	pict

Примечание. Свойство Dock указывает, какие границы элемента управления привязаны к контейнеру.

Свойство Modifiers указывает уровень видимости объекта.

6. Перейдите в редактор кода и опишите функцию Draw, которая будет рисовать наш график функции.

```
public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            Draw();
               }
```

7. Так же опишите процедуру вычисления функции, по заданному значению $(2\sin(x)\,e^{x})$.

```
private double Func(double x)
        {
            return 2*Math.Sin(x)*Math.Exp(x);
               }
```

8. Нарисуем оси ординат и абсцисс с помощью фикции DrawLine, а также нарисуем деления, которые подпишем при помощи функции DrawString.

```
public void Draw()
```

```
        {
            //  Зададим все необходимые переменные

            //  Зададим шаг
            //  По оси X и оси Y
            //  От -scaleX до scaleX и от -scaleY до scaleY
            int scaleX =1, scaleY = 1;
            //  Количество пикселей между делениями
            int interval = 50;
            //  Объект Bitmap хранит в себе любой растровый объект
            //  Длину и ширину зададим по PicterBox
            Bitmap bmp = new Bitmap(pict.Width, pict.Height);

            Graphics g = Graphics.FromImage(bmp);

            //  Определим линии для рисования
            //  1. для рисования оси координат
            //  2. для рисования непосредственно самого графика
            Pen pen_asix = new Pen(Color.Black, 2);
            Pen pen_func = new Pen(Color.Aquamarine, 3);

            //  Нарисуем оси координат
            //  Вычислим серидины нашего холста и отступим по 10 пикселей
            g.DrawLine(pen_asix, bmp.Width / 2, 10, bmp.Width / 2, bmp.Height - 10);
            g.DrawLine(pen_asix, 10, bmp.Height/2, bmp.Width-10, bmp.Height/2);

            //  Вычислим точку пересечения осей
            Point pointZero = new Point(bmp.Width / 2, bmp.Height / 2);

            //  Проставим деления на осях и подпишем их
            //  деления будем ставить на расстоянии 20 пикселей, чтобы надписи не //налезали
            друг на друга
            //  деления длиной 10 пикселей. Проставляем в обе стороны от нулевой отметки
            //  Во время рисования делений подсчитаем их количество
            int countX1 = 0, countX2 = 0, countY1 = 0, countY2 = 0;
            for (int i = pointZero.X; i < bmp.Width-10; i+= interval)
            {
                g.DrawLine(pen_asix, i, bmp.Height / 2-5, i, bmp.Height / 2 + 5);
                countX1++;
            }
            for (int i = pointZero.X; i > 10; i -= interval)
            {
                g.DrawLine(pen_asix, i, bmp.Height / 2 - 5, i, bmp.Height / 2 + 5);
                countX2++;
            }
            //  Тоже самое по оси Y
            for (int i = pointZero.Y; i < bmp.Height - 10; i += interval)
            {
                g.DrawLine(pen_asix, bmp.Width/2 - 5, i, bmp.Width / 2 + 5, i);
                countY1++;
            }
            for (int i = pointZero.Y; i > 10; i -= interval)
            {
                g.DrawLine(pen_asix, bmp.Width / 2 - 5, i, bmp.Width / 2 + 5, i);
                countY2++;
            }


//  Подпишем наши оси и деления
            //  Оси
            Font fontAxis = new Font("Segoe UI", 11, FontStyle.Bold);
            SolidBrush sbAxis = new SolidBrush(Color.DodgerBlue);
            g.DrawString("Y", fontAxis, sbAxis, pointZero.X+5, 5);
            g.DrawString("X", fontAxis, sbAxis, pointZero.X*2-15, pointZero.Y-25);
```

```
            g.DrawString("0", fontAxis, sbAxis, pointZero.X, pointZero.Y-2);
            //  Деления
            Font fontLine = new Font("Vivaldi", 8, FontStyle.Regular);
            SolidBrush sbLine = new SolidBrush(Color.Black);
            for (int i = pointZero.X + interval, j = scaleX; i < bmp.Width - 10; i +=
interval, j+=scaleX)
            {
                g.DrawString(j.ToString(), fontLine, sbLine, i-8, pointZero.Y + 8);
            }
            for (int i = pointZero.X - interval, j = scaleX; i > 10; i -= interval, j +=
scaleX)
            {
                g.DrawString("-" + j.ToString(), fontLine, sbLine, i - 8, pointZero.Y +
8);
            }
            for (int i = pointZero.Y + interval, j = scaleY; i < bmp.Height - 10; i +=
interval, j += scaleY)
            {
                g.DrawString("-"+j.ToString(), fontLine, sbLine, pointZero.X - 25, i - 8);
            }
            for (int i = pointZero.Y - interval, j = scaleY; i > 10; i -= interval, j +=
scaleY)
            {
                g.DrawString(j.ToString(), fontLine, sbLine, pointZero.X - 20, i - 8);
            }

            //  Нарисуем сам график
            double stepX=10.0000f, stepY=Func(-scaleX*countX1);
            //  Рассчитаем первую точку
            Point oldPoint = new Point( (int)(pointZero.X+(-countX2 * scaleX) * (bmp.Width
- 20f) / 2f / scaleX / countX1),
                                        (int)(pointZero.Y  -  (Func(-countX2  *  scaleX)  *
(bmp.Height) / 2f / scaleY / countY1)));
            Point newPoint = new Point();
            for    (float    i    =    -countX2    *    scaleX;    i    <    countX1*scaleX;
i+=(countX1*scaleX*2f/bmp.Width))
            {
                //  Вычислим координаты на экране
                stepX = pointZero.X + i * (bmp.Width-20f) / 2f / scaleX / countX1;
                stepY = pointZero.Y - (Func(i) * (bmp.Height) / 2f / scaleY / countY1);
                newPoint.X = (int)stepX;
                newPoint.Y = (int)stepY;
                if (oldPoint == null)
                    oldPoint = newPoint;
                //  Рисуем линию от предыдущей точки к новой
                g.DrawLine(pen_func, oldPoint, newPoint);
                oldPoint = newPoint;
            }
            //  Выведем в PictureBox наш растровый объект и все, что мы нарисовали
            pict.Image = bmp;
               }
```

9. Добавьте обработчик событий для формы **SizeChanged** (событие возникает, когда в Control изменяется значение свойства Size).

```
private void Form1_SizeChanged(object sender, EventArgs e)
        {
            Draw();
               }
```

10. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис. 2.3.

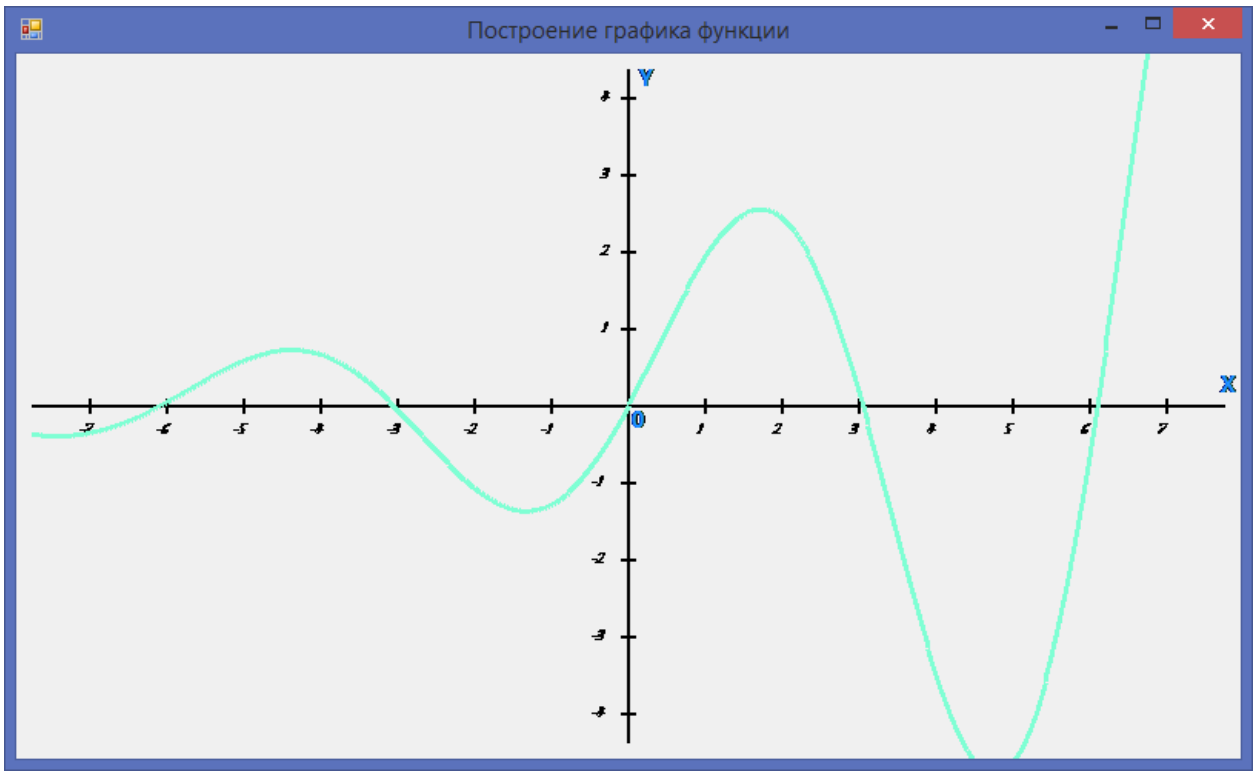


Рис. 2.3. Построение графика функции

2.3. Диаграммы

Постановка задачи: Разработать приложение, которое выводит на поверхности формы круговую, столбчатую и в форме воронки диаграмму, отражающую, например, товарооборот книжного магазина. Сделайте возможность отображения в 2D и 3D.

1. Создайте новый проект WindowsForms. Назовите проект Chart.

2. Активизируйте форму и измените ее имя на Построение диаграмм. Укажите размер 871; 411.

3. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

4. Разместите компоненты на форме согласно рис.2.4.

Примечание. DataGridView отображает строки и столбцы данных в сетке, которые пользователь может изменить.

Chart служит элементом управления диаграмм в Windows Forms.

Оба компонента находятся на панели элементов, вкладка данные.

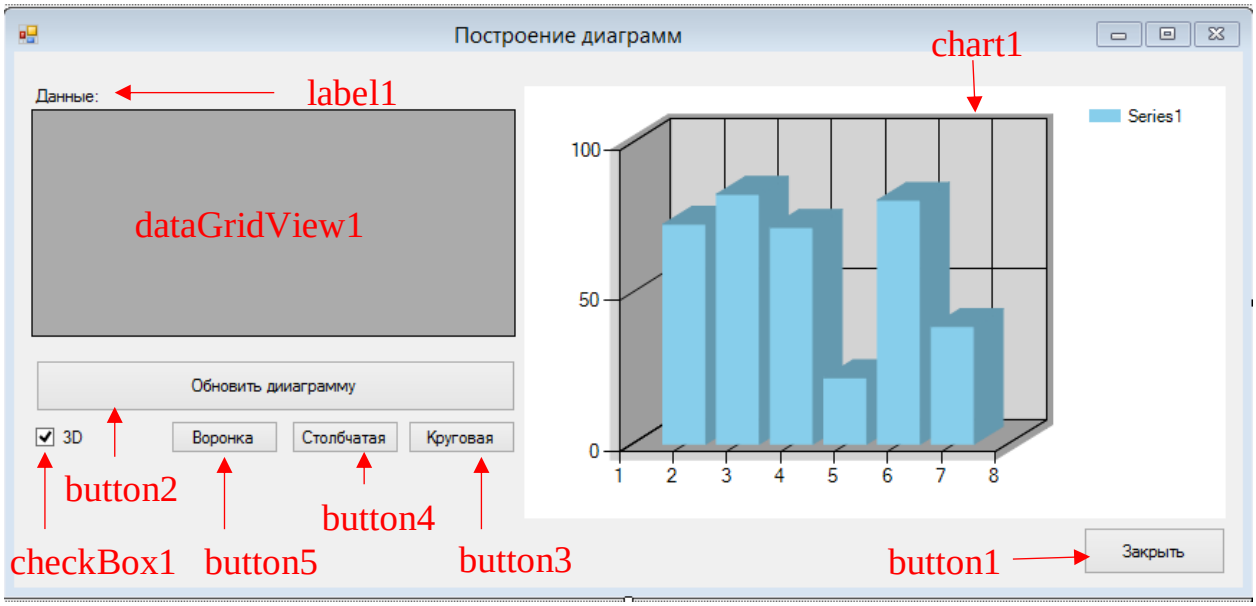


Рис. 2.4. Компоненты приложения Построение диаграмм

5. Для компонента checkBox1 измените стандартное значение свойства **CheckState** – Unchecked на Checked.

6. Опишите функцию chartRefresh которая обновляет данные в chart.

```
public Form1()
        {
            InitializeComponent();
        }

        private void chartRefresh()
        {
            chart1.Series[0].Points.Clear();
```

```
            // пройдемся циклом по всем ячейкам
            for (int i = 0; i < dataGridView1.Rows.Count - 1; ++i)
            {
                chart1.Series[0].Points.AddXY(dataGridView1.Rows[i].Cells[0].Value,
dataGridView1.Rows[i].Cells[1].Value);
            }

               }
```

7. Добавьте обработчик событий для кнопки Закрыть.

```
private void button1_Click(object sender, EventArgs e)
        {
            Close();
               }
```

8. Задайте первоначальное значение элементов и провидите инициализацию.

```
private void Form1_Load(object sender, EventArgs e)
        {
            //создадим таблицу вывода товаров с колонками
            //Категория, количество

            var column1 = new DataGridViewColumn();
            column1.HeaderText = "Категория"; //текст в шапке
            column1.Width = 100; //ширина колонки
            column1.Name = "name"; //текстовое имя колонки, его можно использовать вместо
обращений по индексу
            column1.Frozen = true; // данная колонка всегда отображается на своем месте
            column1.CellTemplate = new DataGridViewTextBoxCell(); //тип нашей колонки

            var column2 = new DataGridViewColumn();
            column2.HeaderText = "Количество";
            column2.Name = "count";
            column2.CellTemplate = new DataGridViewTextBoxCell();

            dataGridView1.Columns.Add(column1);
            dataGridView1.Columns.Add(column2);

            //dataGridView1.AllowUserToAddRows = false; //запрещаем пользователю самому
добавлять строки

            //Создание объекта для генерации чисел
            Random rnd = new Random();

            //Добавляем строку, указывая значения каждой ячейки по имени (можно
использовать индекс 0, 1, 2 вместо имен)
            dataGridView1.Rows.Add(4);
            dataGridView1["name", 0].Value = "Книги";
            dataGridView1["count", 0].Value = rnd.Next(10,200);
            dataGridView1["name", 1].Value = "Журналы";
            dataGridView1["count", 1].Value = rnd.Next(10, 200);
            dataGridView1["name", 2].Value = "Газеты";
            dataGridView1["count", 2].Value = rnd.Next(10, 200);
            dataGridView1["name", 3].Value = "Учебники";
            dataGridView1["count", 3].Value = rnd.Next(10, 200);

            //  Обновим данные в диаграмме
            chartRefresh();
               }
```

9. Напишите обработчики событий для button2 (Обновить диаграмму), button3 (Круговая), button4(Столбчатая) и button5(Воронка).

Примечание. Тип диаграммы должен соответствовать значению системного перечисления:

```
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
```

После точки зажмите комбинацию клавиш ctrl+space и Initialize покажет все возможные типы диаграмм.

```
private void button3_Click(object sender, EventArgs e)
        {
            chart1.Series[0].ChartType                                          =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            chart1.Series[0].ChartType                                          =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;
        }

        private void button5_Click(object sender, EventArgs e)
        {
            chart1.Series[0].ChartType                                          =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Funnel;
            }
```

10. Напишите обработчик событий для checkBox1 (3D).

График сможет отображаться как в 3D, так и в 2D формате.

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            chart1.ChartAreas[0].Area3DStyle.Enable3D = checkBox1.Checked;
            }
```

11. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис. 2.5.

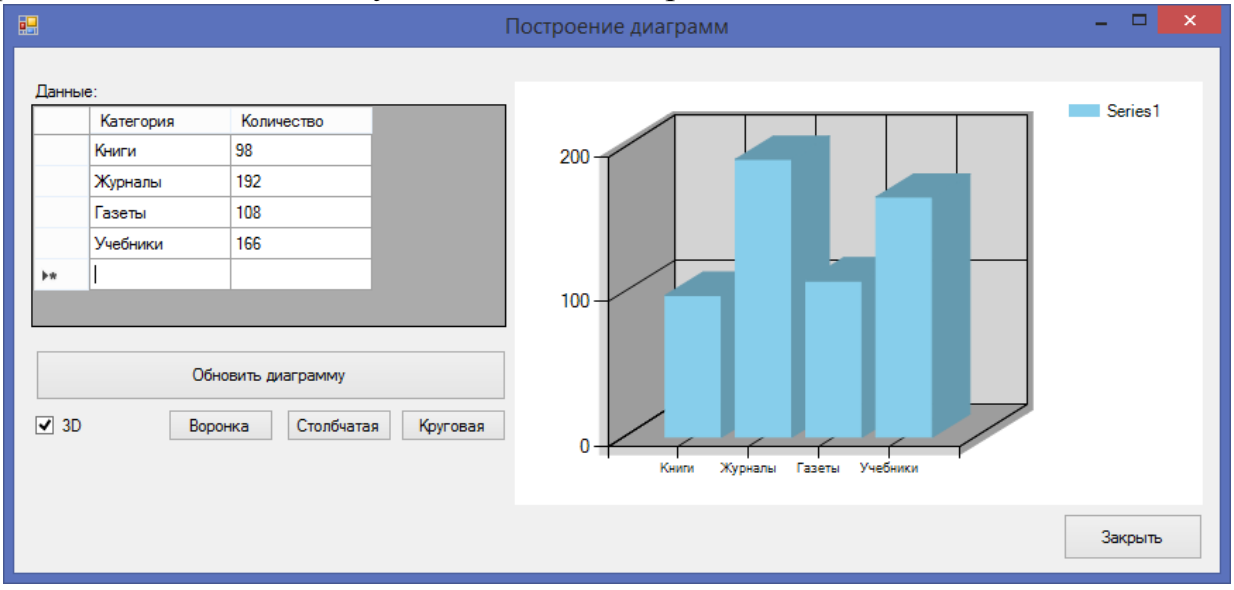


Рис. 2.5. Построение диаграмм

2.4. Графический редактор

Постановка задачи: Разработать приложение, позволяющее выполнять простейшие функции графического редактора (рисовать линией и фигурой, выбор цвета, толщины линии, сохранение рисунка).

1. Создайте новый проект WindowsForms. Назовите проект Painte.

2. Активизируйте форму и измените ее имя на Графический редактор. Укажите размер 866; 579.

3. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

4. Добавьте на форму два элемента Panel. Установите свойства согласно таб. 4.

Таблица 4.

Свойство	**Элемент**	
	Panel1	***Panel2***
Dock	Left	Bottom
BorderStyle	Fixed3D	None
Size	850; 43	143; 497

5. Разместите на форме элемент PictureBox, в свойстве **Dock** укажите Fill.

6. Добавьте на форму семь кнопок: button1 – Загрузить, button2 – Сохранить, button3 – Очистить, button4 – Цвет, button5 – Линия, button6 – Прямоугольник, button7 – Рисование. Разместите компоненты на форме согласно рис.2.6.

7. Для кнопок Линия, Прямоугольник и Рисование, в свойстве **FlatStyle** (Определяет внешний вид элемента управления при наведении указателя мыши и щелчке элементе управления) измените стандартное значение – Standard на Flat (Плоский).

8. Добавьте еще на форму три компонента: label1 – Толщина, PictureBox2 и numericUpDown1 (для выбора цвета и толщины карандаша).

Измените размер элемента PictureBox2 на 12; 55.

В свойстве Maximum и Minimum элемента numericUpDown1 установите значения 100 и 0 соответственно.

Разместите компоненты на форме согласно рис.2.6.

9. Перенесите на форму элементы ColorDialog, openFileDialog, saveFileDialog с панели элементов, вкладка диалоговые окна.

Примечание. ColorDialog отображает доступные цвета и элементы управления, позволяющие пользователю задать другие цвета.

openFileDialog отображает диалоговое окно, позволяющее пользователю открыть файл.

saveFileDialog отображает диалоговое окно, позволяющее пользователю выбрать местоположение для сохранение файла.

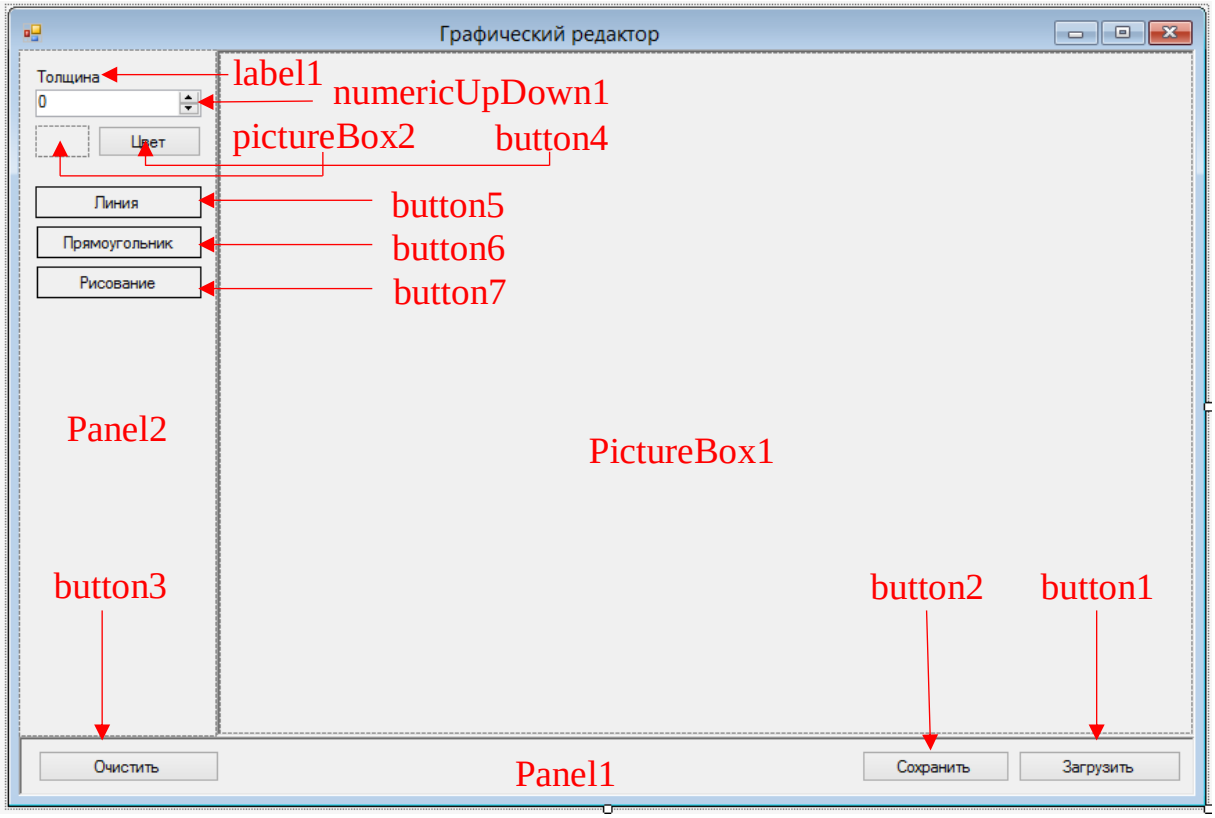


Рис. 2.6. Компоненты приложения Графический редактор

10. В начале задайте и проинициализируйте необходимые переменные.

```
public partial class Form1 : Form
    {
        //  Перечисление типа карандаша - линя, прямоугольник, произвольное рисование
        enum paint {line, rect, custom}
        private paint paintType;
        //  Толщина карандаша
        private int paintWidth;
        //  Цвет карандаша
        private Color paintColor;
        //  Параметр, определяющий, нажата ли кнопка мыши
        private bool isPressed;
        //  Координаты точек по которым мы будем рисовать
        private Point oldPoint;
        private Point newPoint;
        //  Объект Graphics с помощью которого мы и будем рисовать
        private Bitmap bmp;
        public Graphics g;
```

11. Для рисования линии и прямоугольника опишите функцию painter.

Когда мы нажимаем на левую кнопку мыши, начинается рисование, для этого запоминаем точку где мы нажали и точку где мы ее отпустили. Если выбран тип карандаша линия или прямоугольник, то рисуем между этими двумя точками соответствующую фигуру, если выбрано произвольное рисование, то при каждом движении мыши (левая кнопка зажата) мы соединяем предыдущую точку с новой точкой, которая находится у нас на позиции курсора.

```
public Form1()
        {
            InitializeComponent();
        }
        private void painter()
        {
            Pen pen = new Pen(paintColor, paintWidth);
            int X1, Y1;
            if (paintType != paint.rect)
                g.DrawLine(pen, oldPoint, newPoint);
            else
            {
                if (oldPoint.X > newPoint.X) X1 = newPoint.X; else X1 = oldPoint.X;
                if (oldPoint.Y > newPoint.Y) Y1 = newPoint.Y; else Y1 = oldPoint.Y;
                g.DrawRectangle(pen,   X1,   Y1,   Math.Abs(newPoint.X   -   oldPoint.X),
                  Math.Abs(newPoint.Y - oldPoint.Y));
                pictureBox1.Image = bmp;
              }
```

12. В обработчике события для формы **Load** задайте первоначальное значение.

```
private void Form1_Load(object sender, EventArgs e)
        {
            //  Задаем первоначальную толщину
            paintWidth = 3;
            numericUpDown1.Value = paintWidth;

            //  Кнопка мыши не нажата
            isPressed = false;

            //  Цвет карандаша - черный
            paintColor = Color.Black;
            pictureBox2.BackColor = paintColor;
            //  Изначально, у нас произвольный карандаш
            button7.BackColor = Color.Bisque;
            paintType = paint.custom;

            bmp = new Bitmap(pictureBox1.Width, pictureBox1.Height);
            g = Graphics.FromImage(bmp);

        }
```

13. Добавьте обработчик событий для кнопки button4 – Цвет.

Используем системное диалоговое окно, для выбора цвета и в зависимости от выбора наш карандаш начинает рисовать этим цветом.

В PictureBox2 покажем текущий выбранный цвет.

```
private void button4_Click(object sender, EventArgs e)
        {
            DialogResult result;
            result = colorDialog1.ShowDialog();
            if (result == DialogResult.OK)
            {
                paintColor = colorDialog1.Color;
                pictureBox2.BackColor = paintColor;
            }
           }
```

14. Задайте обработчики событий для выбора типа рисования.

В зависимости от выбора подсвечиваем кнопку.

```
       //  Линия
private void button5_Click(object sender, EventArgs e)
        {
            button5.BackColor = Color.Bisque;
            button6.BackColor = Color.WhiteSmoke;
            button7.BackColor = Color.WhiteSmoke;
            paintType = paint.line;
        }
       //  Прямоугольник
        private void button6_Click(object sender, EventArgs e)
        {
            button5.BackColor = Color.WhiteSmoke;
            button6.BackColor = Color.Bisque;
            button7.BackColor = Color.WhiteSmoke;
            paintType = paint.rect;
        }
       //  Свободное рисование
        private void button7_Click(object sender, EventArgs e)
        {
            button5.BackColor = Color.WhiteSmoke;
            button6.BackColor = Color.WhiteSmoke;
            button7.BackColor = Color.Bisque;
            paintType = paint.custom;

        }
```

15. В обработчике событий PictureBox1 опишите события **MouseDown** (Возникает в момент нажатия кнопки мыши, когда указатель мыши находится над компонентом) и **MouseUp** (Возникает в момент отпускания кнопки мыши, когда указатель мыши находится над компонентом).

При отпускании кнопки мыши, и если у нас выбран тип рисование линии, то вызовем функцию painter.

```
private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
        {
            isPressed = true;
            newPoint = e.Location;
        }

        private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
        {
            if (isPressed == true)
                if (paintType != paint.custom)
```

```
            {
                oldPoint = newPoint;
                newPoint = e.Location;
                painter();
            }
        isPressed = false;
    }
```

16. Добавьте обработчик событий для кнопки button3 – Очистить.

```
private void button3_Click(object sender, EventArgs e)
    {
        g.Clear(Color.White);
        pictureBox1.Image = bmp;
       }
```

17. Добавьте еще один обработчик событий для элемента PictureBox1 – **MouseMove** (Возникает при наведении указателя мыши на компонент).

Опишем обработчик событий для перемещения мыши, если у нас нажата кнопка и выбран произвольный тип рисования, соединяем предыдущую точку с текущей точкой положения курсора мыши, и запоминаем ее.

```
private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
    {
        if (isPressed == true)
        {
            if (paintType == paint.custom)
            {
                oldPoint = newPoint;
                newPoint = e.Location;
                painter();
            }
        }
    }
```

18. Добавьте обработчик события для элемента numericUpDown1 – **ValueChanged** (Происходит при изменении значения данного поля со стрелками ‘вверх/вниз’) для выбора толщины линии.

```
private void numericUpDown1_ValueChanged(object sender, EventArgs e)
    {
        paintWidth = (int)numericUpDown1.Value;
       }
```

19. Добавьте обработчики событий для кнопок button1 – Загрузить и button2 – Сохранить.

```
private void button1_Click(object sender, EventArgs e)
    {
        DialogResult result;
        result = openFileDialog1.ShowDialog();
        if (result == DialogResult.OK)
        {
            Image image = Image.FromFile(openFileDialog1.FileName);
            bmp = (Bitmap)image;
            g = Graphics.FromImage(bmp);
            pictureBox1.Image = bmp;
        }
    }
```

```
        private void button2_Click(object sender, EventArgs e)
        {
            Bitmap bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);
            pictureBox1.DrawToBitmap(bitmap, pictureBox1.ClientRectangle);
            if (pictureBox1.Image != null) //если в pictureBox есть изображение
            {
                saveFileDialog1.Title = "Сохранить картинку как...";
                //отображать ли предупреждение, если пользователь указывает имя уже
               // существующего файла
                saveFileDialog1.OverwritePrompt = true;
                //отображать ли предупреждение, если пользователь указывает
               // несуществующий путь
                saveFileDialog1.CheckPathExists = true;
                //список форматов файла, отображаемый в поле "Тип файла"
                saveFileDialog1.Filter       =       "Image       Files(*.BMP)|*.BMP|Image
Files(*.JPG)|*.JPG|Image Files(*.GIF)|*.GIF|Image Files(*.PNG)|*.PNG|All files (*.*)|*.*";
                //отображается ли кнопка "Справка" в диалоговом окне
                saveFileDialog1.ShowHelp = true;
                if (saveFileDialog1.ShowDialog() == DialogResult.OK) //если в диалоговом
               //окне нажата кнопка "OK"
                {
                    try
                    {
                        bitmap.Save(saveFileDialog1.FileName,
System.Drawing.Imaging.ImageFormat.Jpeg);
                    }
                    catch
                    {
                        MessageBox.Show("Невозможно сохранить изображение", "Ошибка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
                bitmap.Dispose();
            }
```

20. Сохраните проект и запустите программу для проверки работоспособности, нарисуйте картинку, а также сохраните и загрузите ее. Результат показан на рис. 2.7.

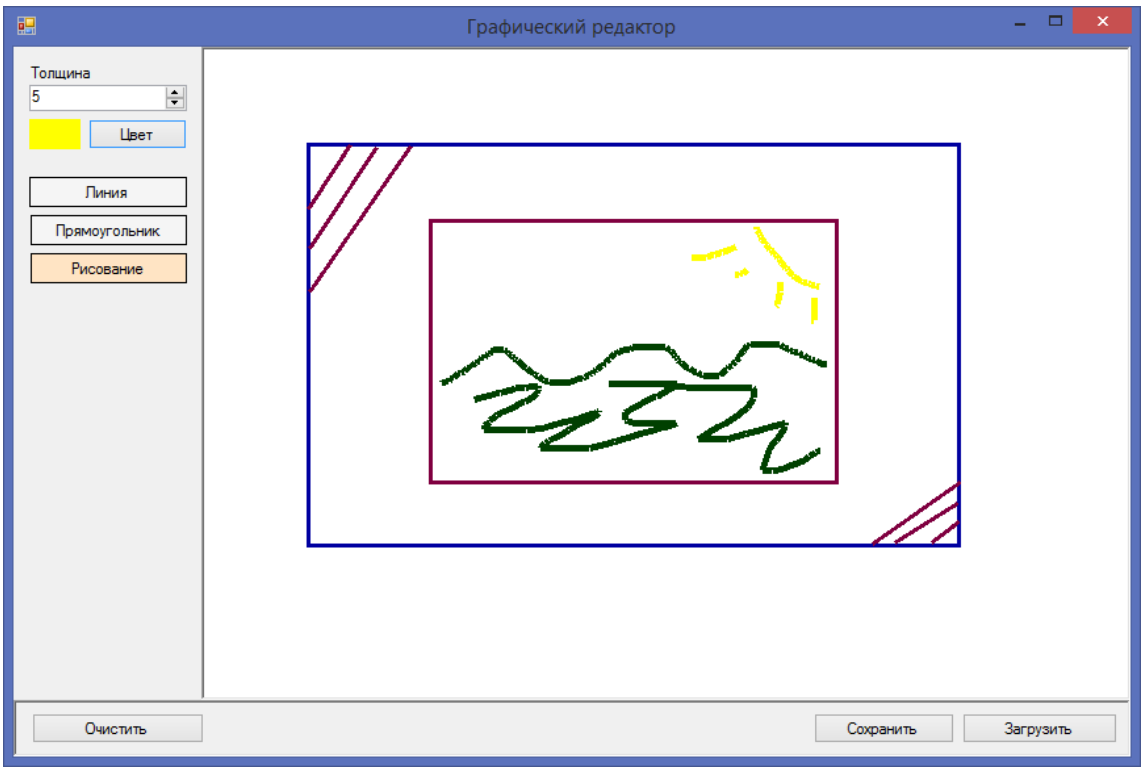


Рис. 2.7. Графический редактор.

2.5. Анимация

Постановка задачи: Разработать приложение, по поверхности которого перемещается графический объект, например, кораблик.

1. Создайте новый проект WindowsForms. Назовите проект Ship.

2. Активизируйте форму и измените ее имя на Кораблик. Укажите размер 886; 545.

3. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

4. Добавьте на форму элемент PictureBox. В свойствах компонента измените значение **Dock** на Fill. Так же измените свойство **Name** на picture.

5. Перенесите на форму элемент Timer (Компонент, создающий событие с интервалами, определенными пользователем).

Примечание. Элемент будет предназначаться для изменения скорости нашего кораблика.

Чтобы изменить скорость объекта, нужно задать в свойствах компонента – **Interval** (Интервал между событиями в миллисекундах) необходимое значение.

6. Перейдите в редактор кода и объявите необходимые переменные.

```
public partial class Form1 : Form
    {
        Bitmap bmp;             //  Фон
        Bitmap oldbmp;          //  Здесь будем хранить фон за корабликом
        Bitmap btmShip;         //  Растр для рисования кораблика
        Graphics gShip;         //  Инструмент для графического рисования кораблика на
                                 //растре
        Point coordinateShip;   //  Координаты корабля
               bool direction = true;         //  Направление
```

7. Опишем принцип работы:

1) Зададим обработчик событий при загрузке формы, где нарисуем небо и море.

2) Зальем градиентом верхнюю и нижнюю половину экрана, которое соответствует небу и морю.

3) Создадим основу для нашего кораблика, это будет пространство размером 100 × 100 пиксел.

4) Зададим координаты корабля по x и по y.

Добавьте обработчик события для формы – **Load**.

```
private void Form1_Load(object sender, EventArgs e)
        {
            bmp = new Bitmap(picture.Width, picture.Height);
```

```
            Graphics g = Graphics.FromImage(bmp);
            //  Рисуем море
            LinearGradientBrush gradBrush = new LinearGradientBrush(
                new Rectangle(0, picture.Height / 2, picture.Width, picture.Height),
                Color.LightSkyBlue,
                Color.DarkBlue,
                LinearGradientMode.Vertical);
            g.FillRectangle(gradBrush,    0,    picture.Height    /    2,    picture.Width,
picture.Height); //с градиентной заливкой
            //  Рисуем небо
            gradBrush = new LinearGradientBrush(
                new Rectangle(0, 0, picture.Width, picture.Height / 2 + 1),
                Color.LightSkyBlue,
                Color.White,
                LinearGradientMode.Vertical);
            g.FillRectangle(gradBrush, 0, 0, picture.Width, picture.Height / 2 + 1);
            //с градиентной заливкой

            //  Выводим наши рисунки на экран - это будет фон
            picture.Image = bmp;
            g.Dispose();
            //  Создадим основу для нашего кораблика
            btmShip = new Bitmap(100, 100);
            gShip = Graphics.FromImage(btmShip);

            coordinateShip.X = 10;
            coordinateShip.Y = picture.Height/2;

            oldbmp = new Bitmap(100, 100);
            drawShip();
               }
```

8. Нам необходимо рассмотреть схему анимации нашего корабля и по текущим координатам запомнить фон.

Для этого опишите функцию drawShip.

```
private void drawShip()
        {
            //  Запомним фон, который был на месте, где мы сейчас нарисуем кораблик
            oldbmp  =  bmp.Clone((new  Rectangle(coordinateShip.X,  coordinateShip.Y,  100,
100)), System.Drawing.Imaging.PixelFormat.DontCare);

            // Нарисуем кораблик
            gShip.Clear(Color.Transparent);
            Pen p = new Pen(Color.Black, 1);
            Pen pW = new Pen(Color.White, 2);


            //  Рисуем корпус корабля и заливаем его белым цветом
            Point[] myArray =
            {
                 new Point(20, 70), new Point(20, 60), new Point(60, 60),
                 new Point(30, 60), new Point(35, 55), new Point(35, 47),
                 new Point(85, 47), new Point(85, 55), new Point(95, 55),
                 new Point(80, 70), new Point(20, 70)
             };
            GraphicsPath myPath = new GraphicsPath();
            myPath.AddLines(myArray);

            gShip.DrawPath(p, myPath);
            gShip.FillPath(Brushes.White, myPath);
```

```
            gShip.DrawLine(p, 20, 70, 20, 60);
            gShip.DrawLine(p, 20, 60, 60, 60);
            gShip.DrawLine(p, 60, 60, 65, 55);
            gShip.DrawLine(p, 65, 55, 95, 55);
            gShip.DrawLine(p, 95, 55, 80, 70);
            gShip.DrawLine(p, 80, 70, 20, 70);

            gShip.DrawLine(p, 30, 60, 35, 55);
            gShip.DrawLine(p, 35, 55, 35, 47);
            gShip.DrawLine(p, 35, 47, 85, 47);
            gShip.DrawLine(p, 85, 47, 85, 55);

            gShip.DrawLine(p, 42, 53, 58, 53);

            // Рисуем белые трубы и обводим контур черным цветом
            gShip.FillRectangle(Brushes.White, 48, 28, 5, 19);
            gShip.DrawRectangle(p, 48, 28, 5, 19);
            gShip.FillRectangle(Brushes.White, 55, 42, 13, 5);
            gShip.DrawRectangle(p, 55, 42, 13, 5);

            gShip.DrawLine(p, 60, 42, 60, 10);
            gShip.DrawLine(pW, 20, 60, 60, 10);
            gShip.DrawLine(pW, 95, 55, 60, 10);

            gShip.DrawEllipse(p, 67, 57, 6, 6);
            gShip.DrawEllipse(p, 77, 57, 6, 6);

            //  Если движемся в обратную сторону, то кораблик разворачиваем
            if (!direction) btmShip.RotateFlip(RotateFlipType.Rotate180FlipY);

            //  Совместим оба изображения фон+коралбик
            Graphics g = Graphics.FromImage(bmp);
            g.DrawImage(btmShip,coordinateShip);
            picture.Image = bmp;
               }
```

9. Добавьте обработчик события для элемента Timer – Tick.

```
private void timer1_Tick(object sender, EventArgs e)
        {
            //  Сотрем кораблик
            Graphics g = Graphics.FromImage(bmp);
            g.DrawImage(oldbmp, coordinateShip);
            picture.Image = bmp;

            //  передвинем координаты
            if (direction)
                coordinateShip.X++;
            else
                coordinateShip.X--;

            // Если достигли края экрана, меняем направление
            if (coordinateShip.X > picture.Width - 110)
                direction = false;
            if (coordinateShip.X < 10)
                direction = true;

            //  Нарисум кораблик на новом месте
            drawShip();
               }
```

10. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис. 2.8.

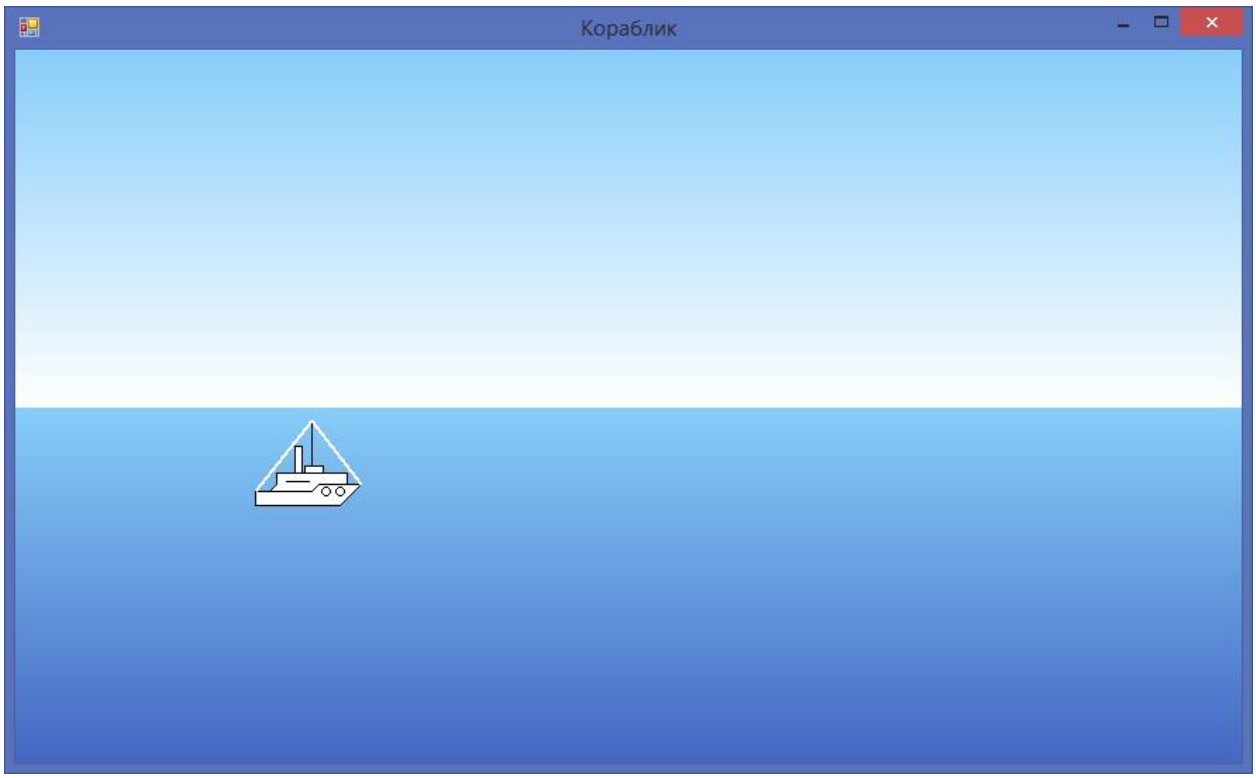


Рис. 2.8. Анимация «Кораблик»

2.6. Задачи для самостоятельной работы

1. Напишите программу, которая в элементе PictureBox рисует российский флаг (три горизонтальные полосы белого, красного и синего цветов).

2. Напишите программу, которая в элементе PictureBox рисует норвежский флаг (красный фон с синим крестом).

3. Напишите программу, которая в элементе PictureBox рисует японский флаг (белый фон с красным кругом).

4. Напишите программу, которая в элементе PictureBox рисует чешский флаг (синий треугольник и две горизонтальные полосы белый и красный цветов).

5. Напишите программу, которая в элементе PictureBox рисует великобританский флаг (синий фон, два пересекающихся креста).

6. Напишите программу, которая на поверхности формы выводит изображение оцифрованной координатной оси (каждое деление подписано цифрой).

7. Напишите программу, которая на поверхности формы вычерчивает график функции, который задается таблицей, первая строка которой содержит значения аргументов, вторая – соответствующие значения функции.

8. Напишите программу, которая на поверхности формы вычерчивает график функции $\sin x$.

9. Напишите программу, которая на поверхности формы вычерчивает график функции $\tan x$.

10. Напишите программу, которая на поверхность формы выводит изображение идущих часов с часовой, минутной и секундной стрелками.

11. Напишите программу, которая на поверхность формы выводит изображение снеговика с морковкой, пуговками, ведром и ветками вместо рук.

12. Разработать анимационное приложение, состоящее из рисунка дома, предусмотреть смену суток (восход и заход солнца, ночью светится луна, а также включаться свет в доме).

13. Разработать анимационное приложение, состоящее из моря, неба и солнца. Лучи солнца должны играть. Предусмотреть смену суток, ночью небо с морем становятся почти черными, а когда солнце в зените все становится ярко голубым. Применить градиентную заливку, чтобы небо и море не сливалось.

14. Разработать анимационное приложение звездное небо, звезды должны мерцать разными цветами, предусмотреть возможность звездопада.

15. Дополнить разобранную программу №5, нарисуйте две рыбки, которые плавают в море по разным траекториям ($\sin x$ и $\cos x$).

16. Напишите программу, которая выводит столбчатую диаграмму по изменению курсов валют (USD и EUR) по отношению к рублю за месяц.

17. С помощью компонента Chart построить график функции. Функция задается таблицей, первая строка которой содержит значения аргументов, вторая – соответствующие значения функции.

18. С помощью компонента Chart построить анимационную модель движения солнца ($x = r\cos\alpha,\ y = r\ \sin\alpha$). Угол α меняется в зависимости от времени.

Примечание. 24 раза в секунду очищаем элемент Chart и рисуем солнце по новым координатам.

19. Разработать анимационное приложение. Шарик движется под произвольным углом. При достижении краев формы, шарик зеркально отражается.

20. Дополните предыдущую задачу так, что шарик постепенно, то увеличивается в размере, то уменьшается, при этом изменяется его цвет от светлого до темного и обратно.

21. Дополнить разобранную программу №4, добавьте в графический редактор возможность рисовать окружность.

22. Дополнить разобранную программу №4, добавьте в графический редактор возможность закрашивать замкнутые фигуры (цвет заливки пользователь выбирает сам).

3. Файловая система и исключения

После организации интерфейса программы, разработчик практически любого приложения рано или поздно столкнется с необходимостью сохранить информацию для того, чтобы была возможность с ней работать в дальнейшем.

Классы работы с файловой системой находятся в пространстве имен System.IO. IO – сокращение от английского Input/Output (ввод/вывод, запись/чтение). Пространство имен System.IO содержит:

1. Классы для работы с каталогами, такие как – Directory, DirectoryInfo;

DirectoryInfo содержит набор членов, используемых для создания, перемещения, удаления и перечисления каталогов и подкаталогов. В дополнение к функциональности, предоставленной базовым классом (FileSystemInfo), DirectoryInfo предлагает ключевые члены, перечисленные ниже:

– Create(), CreateSubdirectory() – создает каталог (или набор подкаталогов) по заданному путевому имени

– Delete() – удаляет каталог и все его содержимое

– GetDirectories() – возвращает массив объектов DirectoryInfo, представляющих все подкаталоги в текущем каталоге

– GetFiles() – извлекает массив объектов FileInfo, представляющий множество файлов в заданном каталоге

– MoveTo() – Перемещает каталог со всем содержимым по новому пути

Что касается вспомогательного класса **Directory**, то он годится для выполнения простых операций над директориями, хотя в принципе имеет все те же возможности, что и класс DirectoryInfo. Класс Directory удобно использовать для проверки существования каталога, если больше никаких действий над этим каталогом проводить не требуется. При помощи этого класса можно легко создать новую папку методом *CreateDirectory*, переместить методом *Move* или удалить - *Delete*.

2. Классы для работы с файлами – File, FileInfo;

Класс **FileInfo** – основной, у него нет статичных методов, в отличие от вспомогательного класса File, который кроме таковых ничего больше не имеет.

Класс FileInfo очень похож на класс DirectoryInfo, более того у обоих классов общий родитель – FileSystemInfo. Имеет свойство **Exists**, для проверки существования файла. У FileInfo есть метод **MoveTo**, для перемещения файла,

и метод **Delete** для удаления файла, и даже метод **Create** для создания файла, хотя последний в реальных условиях используется редко. Установка прав доступа к файлу осуществляется методом **SetAccessControl**, принцип такой же, как и у папок. Получить список разрешений можно методом **GetAccessControl**.

Но помимо этого, у класса FileInfo есть свойство, содержащее размер файла в байтах – **Length**, а также свойство **IsReadOnly**, которое указывает на возможность записи данных в файл. Из других особенностей, FileInfo имеет ссылку на директорию, в которой находится файл – **Directory**, ссылка представляет из себя экземпляр класса **DirectoryInfo**. Для удобства получения имени директории, существует вспомогательное свойство **DirectoryName**.

Для записи и чтения данных, класс FileInfo содержит функции: **Open, OpenRead, OpenText** и **OpenWrite.** Большинство функций возвращают FileStream – поток записи или чтения. **FileStream** позволяет записывать и считывать данные из/в файла по байтам.

- Класс для работы с путями – Path.

Класс **Path** позволяет формировать пути к каталогам и файлам, имеет методы для анализа имен файлов. Это очень удобный и полезный класс. Класс Path работает исключительно с путями (со строками) и не производит ни каких реальных манипуляций с файлами и папками, он лишь позволяет получить все необходимое для осуществления этих действий.

Чаще всего, используется функция Combine, которая позволяет объединить фрагменты путей в один путь.

Диалоговые окна для работы с файлами

Многим приложениям нужно выполнять одни и те же стандартные задачи выбирать файл для открытия или сохранения и др. Библиотека компонентов Visual Studio содержит несколько так называемых стандартных диалогов. Эти компоненты реализуют не только внешний вид диалогового окна, но и необходимую функциональность.

В таб.3.1 перечислены компоненты, реализующие стандартные диалоги, которые можно найти в панели элементов на вкладке Диалоговые окна.

Таблица 3.1.

Элемент	Описание
ColorDialog	Отображает доступные цвета и элементы управления, позволяющие пользователю задать другие цвета.
FolderBrowserDialog	Отображает диалоговое окно, позволяющие пользователю выбрать папку.

FontDialog	Отображает диалоговое окно, позволяющие пользователю выбрать шрифт из списка шрифтов, установленных на локальном компьютере.
OpenFileDialog	Отображает диалоговое окно, позволяющие пользователю открыть файл.
SaveFileDialog	Отображает диалоговое окно, позволяющие пользователю выбрать местоположение для сохранения файла.

Обработка исключительных ситуаций

Иногда при выполнении программы возникают ошибки, которые трудно предусмотреть или предвидеть, а иногда и вовсе невозможно. Например, при передаче файла по сети может неожиданно оборваться сетевое подключение, такие ситуации называются исключениями. Язык C# предоставляет разработчикам возможности для обработки таких ситуаций. Для этого в C# предназначена конструкция **try...catch...finally**. При возникновении исключения среда ищет блок catch, который может обработать данное исключение. Если такого блока не найдено, то пользователю отображается сообщение о необработанном исключении, а дальнейшее выполнение программы останавливается.

3.1. Исключительные ситуации

Постановка задачи: Разработать приложение, которое демонстрирует работу с исключительными ситуациями: деление на ноль, переполнение массива, открытие несуществующего файла и создание собственного исключения.

1. Опишем принцип работы.

Данная программа показывает, как необходимо обрабатывать исключительные ситуации. Мы генерируем ошибки деление на ноль, переполнение массива и открытие не существующего файла. Для того чтобы программа не прекратила свою работу, мы перехватываем исключительную ситуацию и обрабатываем ее.

2. Создайте новый проект WindowsForms. Назовите проект Exception.

3. Активизируйте форму и измените ее имя на Обработка исключений. Укажите размер 676; 194. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

4. Разместите компоненты на форме согласно рис.3.1.

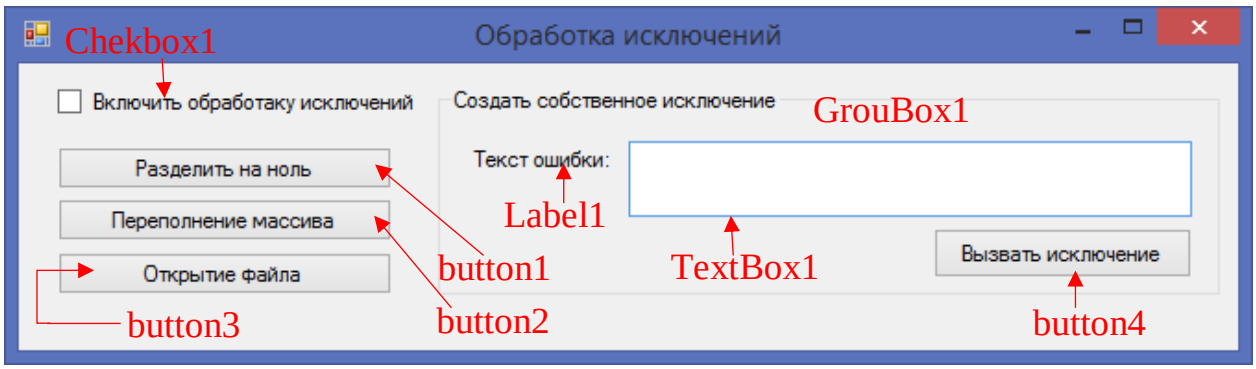


Рис. 3.1. Компоненты приложения Графический редактор

5. В свойстве **Name** элемента ChekBox измените стандартное значение – ChekBox1 на cbIncludeException.

6. В свойстве **Text** элемента TextBox добавьте сообщение – Произошел запланированный сбой, вызвавший исключительную ситуацию.

7. Перейдите в редактор кода и добавьте класс определения собственного исключения.

```
public partial class Form1 : Form
    {
        private bool IsIncludeException = false;

        //  Класс определения собственного исключения
        class MyException : System.Exception
        {
            public MyException(string message)
                : base(message)
            { }
        }
```

8. Добавьте обработчик событий кнопки «Разделить на ноль».

Примечание. Все что находится в try – это безопасный код и даже если в нем произойдет ошибка мы сможем ее перехватить и обработать.

Все что находится в catch – это код обработки исключительной ситуации.

Все что находится в finally – это код который выполнится в любом случае произошла ли у нас исключительная ситуация или нет.

```
public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int x = 2, y = 0;
            MessageBox.Show("В следующей строке попытаемся 2 разделить на 0");
            if (IsIncludeException)
            {
                try
                {
                    x = x / y;
```

```
                }
                catch (System.Exception ex)
                {
                    MessageBox.Show("Ошибка: " + ex.Message, "Произошла исключительная
ситуация", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    MessageBox.Show("Программа не прервала работу, несмотря на
возникновение исключительной операции", "Информация", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                }
            }
            else
                x = x / y;
        }
```

9. По подобной схеме опишите обработчики событий для кнопки «Переполнение массива» и «Открытие файла».

```
private void button2_Click(object sender, EventArgs e)
        {
            int[] x = new int[2];
            int y = 2, z=0;
            MessageBox.Show("У нас определен массив из 2 элементов. \nВ следующей строке
попытаемся обратиться к 3 элементу массива");
            if (IsIncludeException)
            {
                try
                {
                x[3] = 5;
                }
                catch (System.IndexOutOfRangeException ex)
                {
                    MessageBox.Show("Ошибка: " + ex.Message, "Произошла исключительная
ситуация", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    MessageBox.Show("Программа работает нормально!", "Информация",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
            else
                x[3] = 5;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            FileStream fstream = null;
               //  Сгенерируем временный файл
            string fileName = Path.GetTempPath() + Path.GetRandomFileName();
            MessageBox.Show("Попытаемся открыть случайно сгенерированный файл: " +
fileName);
            if (IsIncludeException)
            {
                try
                {
                    fstream = new FileStream(fileName, FileMode.Open);
                    // операции с потоком
                }
                catch (System.Exception ex)
                {
```

```
                    //  Создадим файл
                    fstream = new FileStream(fileName, FileMode.Create);
                }
                //  Пример показывает, для чего обычно используется инструкция finally:
                //  добавляет в программу прогнозируемость, позволяя выполнить определенный
код при любых обстоятельствах.
                //   Это может быть полезно для выполнения операций очистки, например,
закрытия сетевого подключения и т.д.
                finally
                {
                        if (fstream != null)
                        fstream.Close();
                    MessageBox.Show("Программа   работает   нормально!",   "Информация",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
            else
            {
                fstream = new FileStream(fileName, FileMode.Open);
                if (fstream != null)
                    fstream.Close();
            }
        }
```

10. Добавьте обработчик события элемента Chekbox.

В нем будем задавать значение в IsIncludeException, в зависимости от того стоит галочка в нем или нет.

```
private void cbIncludeException_CheckedChanged(object sender, EventArgs e)
        {
            IsIncludeException = cbIncludeException.Checked;
               }
```

11. Добавьте обработчик событий для кнопки «Вызвать исключение».

Здесь вызываем описанную ранее, нашу собственную исключительную ситуацию.

```
private void button4_Click(object sender, EventArgs e)
        {
            if (IsIncludeException)
            {
                try
                {
                    throw new MyException(textBox1.Text);
                }
                //  Пример того, что в инструкции catch
                //  обрабатываются все исключительные ситуации
                catch (System.Exception ex)
                {
                    MessageBox.Show("Ошибка:  " + ex.Message, "Произошла исключительная
ситуация", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally
                {
                    MessageBox.Show("Программа   работает   нормально!",   "Информация",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
            else throw new MyException(textBox1.Text);
        }
```

12. Сохраните проект и запустите программу для проверки работоспособности, протестируйте все исключительные ситуации. Результат показан на рис. 3.2.

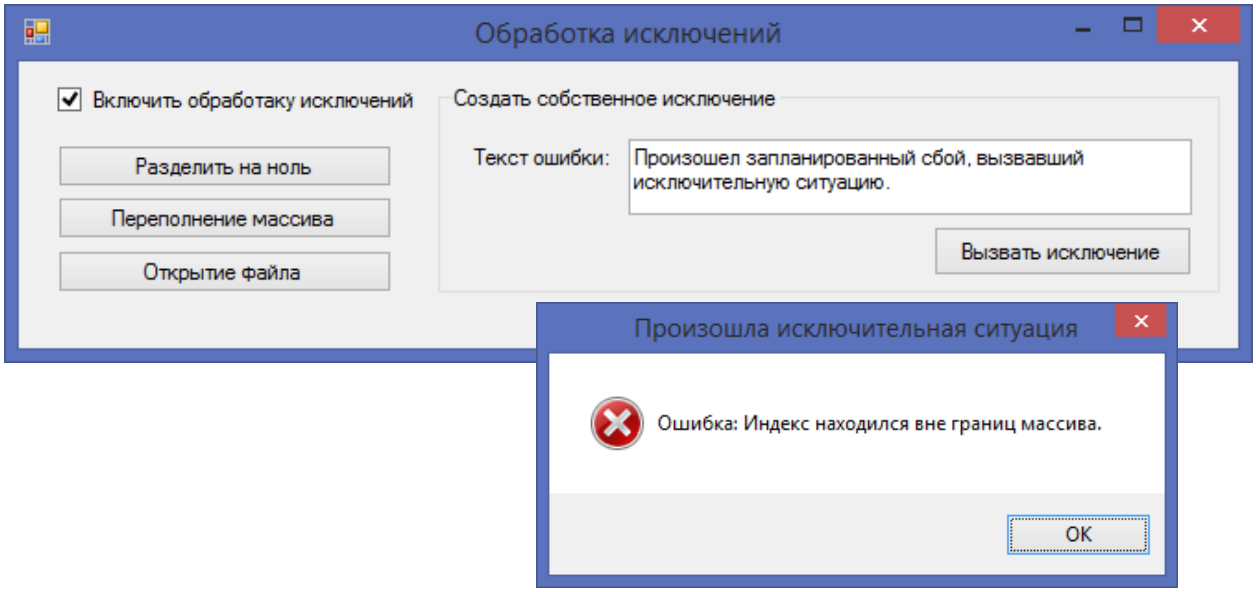


Рис.3.2. Обработка исключений

3.2. Работа с каталогами (папками)

Постановка задачи: Разработать приложение, которое будет выполнять типичные действия над папками: определение и переход в текущую папку, создание и удаление папок.

1. Создайте новый проект WindowsForms. Назовите проект DirectoriesWork.

2. Активизируйте форму и измените ее имя на Работа с папками. Укажите размер 457; 87. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

3. Разместите компоненты на форме четыре кнопки – Button1(Текущая папка), Button2 (Перейти в папку), Button3 (Создать папку), Button4 (Удалить папку) согласно рис.3.3.

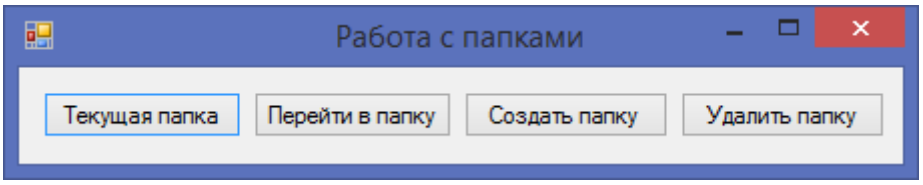


Рис.3.3. Компоненты формы «Работа с папками»

4. Добавьте вторую форму.

Чтобы добавить еще одну форму в проект, нажмем на имя проекта в окне Обозреватель решений (Solution Explorer) правой кнопкой мыши и выберем Добавить (Add), Формы Windows (Form Windows).

5. Разместите элементы на второй форме согласно рис.3.4.

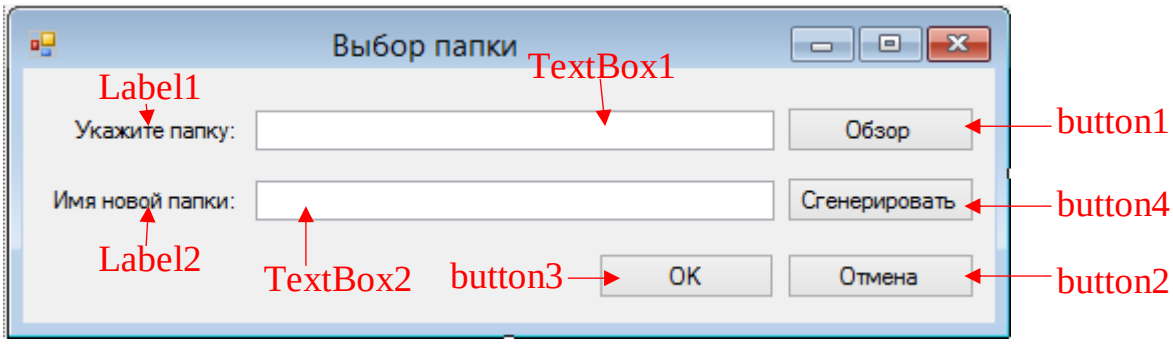


Рис.3.4. Компоненты формы «Выбор папки»

6. Перетащите элемент FolderBrowserDialog (Отображает диалоговое окно, позволяющее пользователю выбрать папку) на форму.

7. Перейдите в редактор кода первой формы.

Подключим к первой форме вторую. Во всех обработчиках событий четырех кнопок будет использоваться вторая форма.

```
public partial class Form1 : Form
```

```
    {
        Form2 inputFolder = new Form2();
        public Form1()
        {
            InitializeComponent();
               }
```

8. Добавьте обработчик событий для кнопки Текущая папка.

Здесь мы вызываем диалоговое окно где показываем текущую папку.

```
private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show(Directory.GetCurrentDirectory(),       "Текущая       папка",
MessageBoxButtons.OK, MessageBoxIcon.Information);
               }
```

9. Добавьте обработчик событий для кнопки Перейти в папку.

Вызываем диалоговое окно по выбору папки, при чем не разрешаем через это окно создавать новую папку, если пользователь нажал на кнопку ОК, то устанавливаем выбранную папку в качестве текущей.

```
private void button2_Click(object sender, EventArgs e)
        {
            inputFolder.IsNewFolderForm = false;
            if (inputFolder.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    //  Устанавливаем текущую директорию
                    Directory.SetCurrentDirectory(inputFolder.DirName);

                    // Выведем сообщение, что все у нас все прошло хорошо
                    MessageBox.Show("Текущая папка успешно изменена.", "Перейти в папку",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (DirectoryNotFoundException ex)
                {
                    MessageBox.Show("Ошибка.\nУказанная  папка  не  существует.\n"  +  ex,
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
```

10. Добавьте обработчик событий для кнопки Создать папку.

Вызываем диалоговое окно по выбору папки, при чем разрешаем через это окно создавать новую, если пользователь нажал на кнопку ОК, то создаем новую папку и устанавливаем ее в качестве текущей.

```
private void button3_Click(object sender, EventArgs e)
        {
            string newFolder;
            inputFolder.IsNewFolderForm = true;
            if (inputFolder.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    //  Создаём новую папку
                    newFolder = inputFolder.DirName + "\\" + inputFolder.newDirName;
```

```
                    Directory.CreateDirectory(newFolder);
                    //  Устанавливаем текущую директорию
                    Directory.SetCurrentDirectory(newFolder);

                    // Выведем сообщение, что все у нас все прошло хорошо
                    MessageBox.Show("Папка успешно Создана.", "Создание новой папки",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Ошибка при создании новой папки.\n" + ex, "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
               }
```

11. Добавьте обработчик событий для кнопки Удалить папку.

Вызываем диалоговое окно по выбору папки, затем спрашиваем у пользователя уверен ли он в своих действиях, и при положительном ответе удалям папку.

```
private void button4_Click(object sender, EventArgs e)
        {
            inputFolder.IsNewFolderForm = false;
            if (inputFolder.ShowDialog() == DialogResult.OK)
            {
                try
                {
                   if(MessageBox.Show(   "Вы уверены, что хотите удалить эту папку:\n"+
inputFolder.DirName,
                                         "Удаление папки",
                                         MessageBoxButtons.YesNo,
                                         MessageBoxIcon.Question)
                        == DialogResult.Yes)
                    //  Удаляем текущую директорию
                    Directory.Delete(inputFolder.DirName);

                    // Выведем сообщение, что все у нас ок
                    MessageBox.Show("Текущая папка успешно удалена.", "Перейти в папку",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Ошибка  при  удалении  папки.\n"  +  ex,  "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
         }
```

12. Перейдите в редактор кода второй формы. Добавьте описание классов.

В классе объявляются поля DirName, которое показывает выбранную папку и newDirName, которая показывает имя новой папки, при этом пользователь может ввести вручную, либо воспользоваться генерацией случайного имени нажав на кнопку сгенерировать.

Значение isNewFolderForm указывает пользователю существует ли возможность указать новое имя папки или нет делая соответствующие элементы формы видимыми в зависимости от своего значения.

```
public partial class Form2 : Form
    {
        public string DirName { get; set; }
        public string newDirName { get; set; }
        private bool isNewFolderForm;
        public bool IsNewFolderForm
        {
            set
            {
                label2.Visible = value;
                textBox2.Visible = value;
                button4.Visible = value;
                isNewFolderForm = value;
            }
            get
            {
                return isNewFolderForm;
            }

        }
```

13. Добавьте обработчик событий для кнопки Обзор.

Вызываем стандартный диалог folderBrowserDialog и если пользователь нажал ОК, то в textBox для выбранной папки помещаем выбранный путь.

```
private void button1_Click(object sender, EventArgs e)
        {
            if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
                textBox1.Text = folderBrowserDialog1.SelectedPath;
           }
```

14. Добавьте обработчик событий для textBox1.

Обработчик событий для textBox1 отвечает за имя выбранной папки. Изначально кнопка ОК имеет свойство Enabled – False и если textBox1 у нас не пустой, то кнопку разблокируем и при этом учитываем значение параметра из isNewFolderForm.

```
private void textBox1_TextChanged(object sender, EventArgs e)
        {
            //  Кнопка ОК становится доступной, если указана какая-то директория
            button3.Enabled = textBox1.Text != string.Empty;
            if (IsNewFolderForm)
            {
                button3.Enabled = button3.Enabled && (textBox2.Text != string.Empty);
            }
        }
```

15. Добавьте обработчик событий для кнопки ОК.

```
private void button3_Click(object sender, EventArgs e)
        {
            //  При выходе присвоим нашему свойству DirName и newDirName значение введенной
папки,
            //  чтобы это значение было доступно из вызываемой формы
```

```
            DirName = textBox1.Text;
            newDirName = textBox2.Text;
               }
```

16. Добавьте обработчик событий для кнопки Сгенерировать.

Помещаем в textBox2 случайным образом сгенерированное имя файла.

```
private void button4_Click(object sender, EventArgs e)
        {
            textBox2.Text = Path.GetFileNameWithoutExtension(Path.GetRandomFileName());
        }
```

17. Сохраните проект и запустите программу для проверки работоспособности, протестируйте все возможности работы с каталогами (папками). Результат создания папки показан на рис. 3.5.

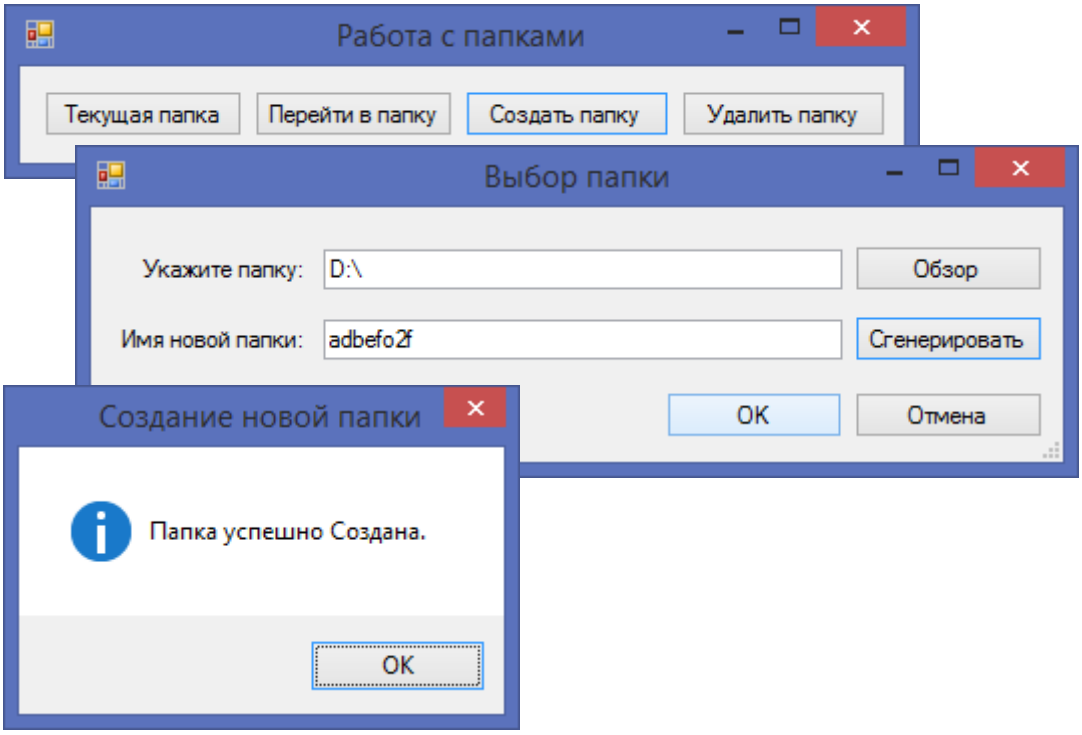


Рис.3.5. Работа с каталогами (папками)

3.3. Поиск файлов

Постановка задачи: Разработать приложение, которое выполняет поиск файла. (Поиск должен осуществляться в указанном пользователем каталоге и его подкаталогах).

1. Создайте новый проект WindowsForms. Назовите проект SearchFiles.

2. Активизируйте форму и измените ее имя на Поиск файлов. Укажите размер 835; 497. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

3. Добавьте на форму Panel. Установите размер 819; 120 и в свойстве **Dock** укажите Top.

4. Разместите элемент label1 на Panel1 и в свойстве **Text** укажите – Выберите каталог, укажите имя или маску файла и нажмите на кнопку Найти (рис.3.6).

5. Добавьте еще на Panel1 элементы – label2 (Каталог:), label3 (Файл:), textBox1, textBox2, кнопки Обзор и Найти согласно рис.3.6.

TextBox1 отвечает за каталог, где будем производить поиск. По кнопке Обзор вызываем стандартный диалог выбора и если пользователь нажал ОК, то помещаем имя в TextBox1.

TextBox2 отвечает за маску файлов, которые будем искать в указанном каталоге. Для этого в свойстве **Text** укажите *.*.

Кнопка Найти служит для функции поиска.

6. Добавьте на форму еще один элемент Panel. В свойстве **Dock** укажите Fill.

7. Добавьте на Panel2 элемент ListView (Отображает коллекцию элементов в одном из пяти представлений).

Укажите свойства элемента согласно таб. 4.

Примечание. FullRowSelect указывает, выделяются ли при выборе элемента все подэлементы.

GridLines отображает линии сетки вокруг элементов и подэлементов только в представлении Details.

View выбирает одно из пяти различных представлений, в которых могут отображаться элементы.

Таблица 4.

Свойство	Значение
FullRowSelect	True
GridLines	True
View	Details
Dock	Fill

Все объекты представления Details, задаются с помощью свойства **Columns**. Элемент будет содержать найденные нами файлы. Определите в нем поля – Имя, Папка, Дата создания, Размер.

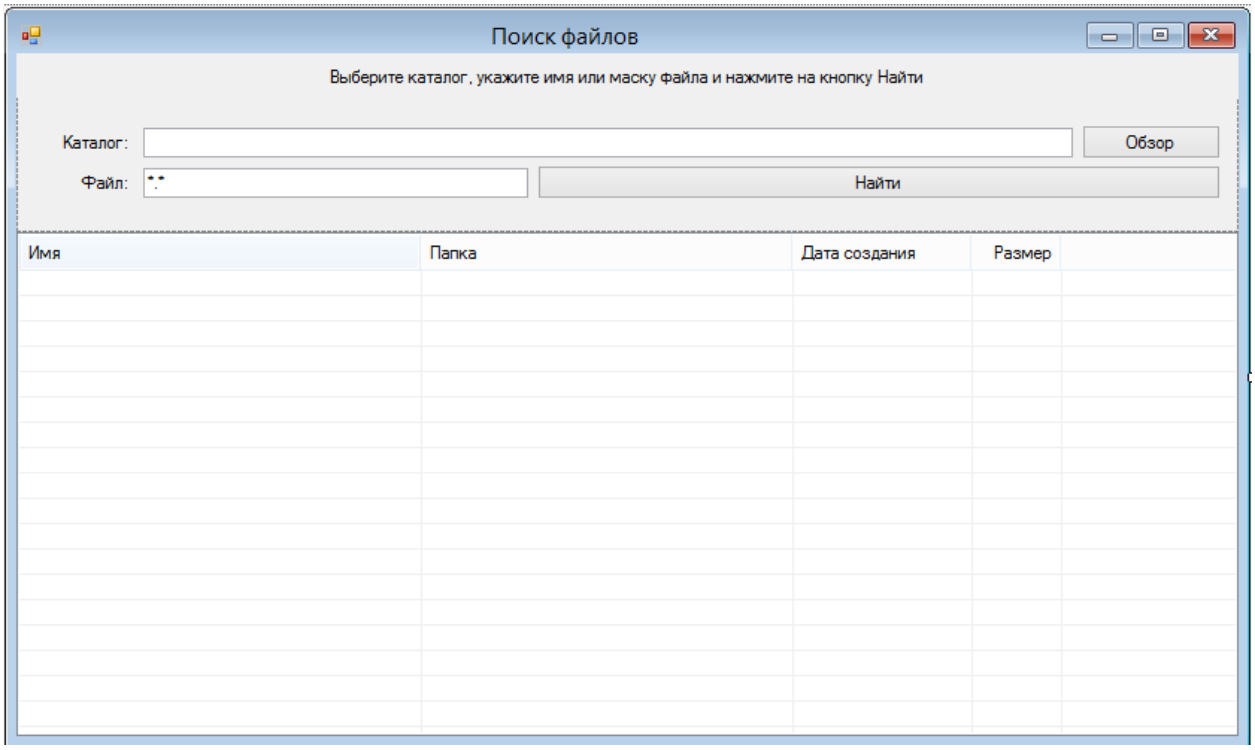


Рис.3.6. Компоненты формы «Поиск файлов»

8. Перетащите элемент FolderBrowserDialog на форму.

9. Перейдите в редактор кода и добавьте рекурсивную функцию, которая обходит текущую папку и все вложенные и при этом заполняет ListView.

```
public Form1()
        {
            InitializeComponent();
        }
        private void findFiles(string folder)
        {
            if (Directory.Exists(folder))
            {
                //  Получим список всех вложенных папок
                string[] dirs = Directory.GetDirectories(folder);
                //  Будем каждую папку перебирать и вызывать для нее поиск файлов
                foreach (string s in dirs)
                {
                    findFiles(s);
                }
                //  Получим список всех вложенных файлов
```

```
                string[] files = Directory.GetFiles(folder,textBox2.Text);
                foreach (string s in files)
                {
                    //  Получим всю информацию о файле
                    FileInfo fileInf = new FileInfo(s);
                    //  Занесем нужную нам информацию в ListView
                    ListViewItem lvItem1 = new ListViewItem(new string[] { fileInf.Name,
fileInf.Directory.ToString(),                              fileInf.CreationTime.ToString(),
fileInf.Length.ToString()});
                    listView1.Items.Add(lvItem1);

                }
            }
        }
```

10. Добавьте обработчик события для кнопки Обзор.

Если директория выбрана и нажата кнопка ОК, то заносим имя этого каталога в TextBox.

```
private void button1_Click(object sender, EventArgs e)
        {
            if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
                textBox1.Text = folderBrowserDialog1.SelectedPath;
        }
```

11. Добавьте обработчик события для textBox1.

Обработаем событие на изменение директории. Кнопка Найти будет доступна, если оба TextBox не пустые.

```
private void textBox1_TextChanged(object sender, EventArgs e)
        {
            button2.Enabled  =  (textBox1.Text  !=  string.Empty)  &&  (textBox2.Text  !=
string.Empty);
              }
```

12. Добавьте обработчик события для textBox2. Обработаем событие на изменение имени или маски файла.

```
private void textBox2_TextChanged(object sender, EventArgs e)
        {
           button2.Enabled  =  (textBox1.Text  !=  string.Empty)  &&  (textBox2.Text  !=
string.Empty);
              }
```

13. Добавьте обработчик события для кнопки Найти.

```
private void button2_Click(object sender, EventArgs e)
        {
            //  На время поиска обе кнопки заблокируем
            //  Запустим поиск
            listView1.Items.Clear();
            findFiles(textBox1.Text);
              }
```

14. Сохраните проект и запустите программу для проверки работоспособности, протестируйте возможность поиска файлов. Результат создания папки показан на рис. 3.7.

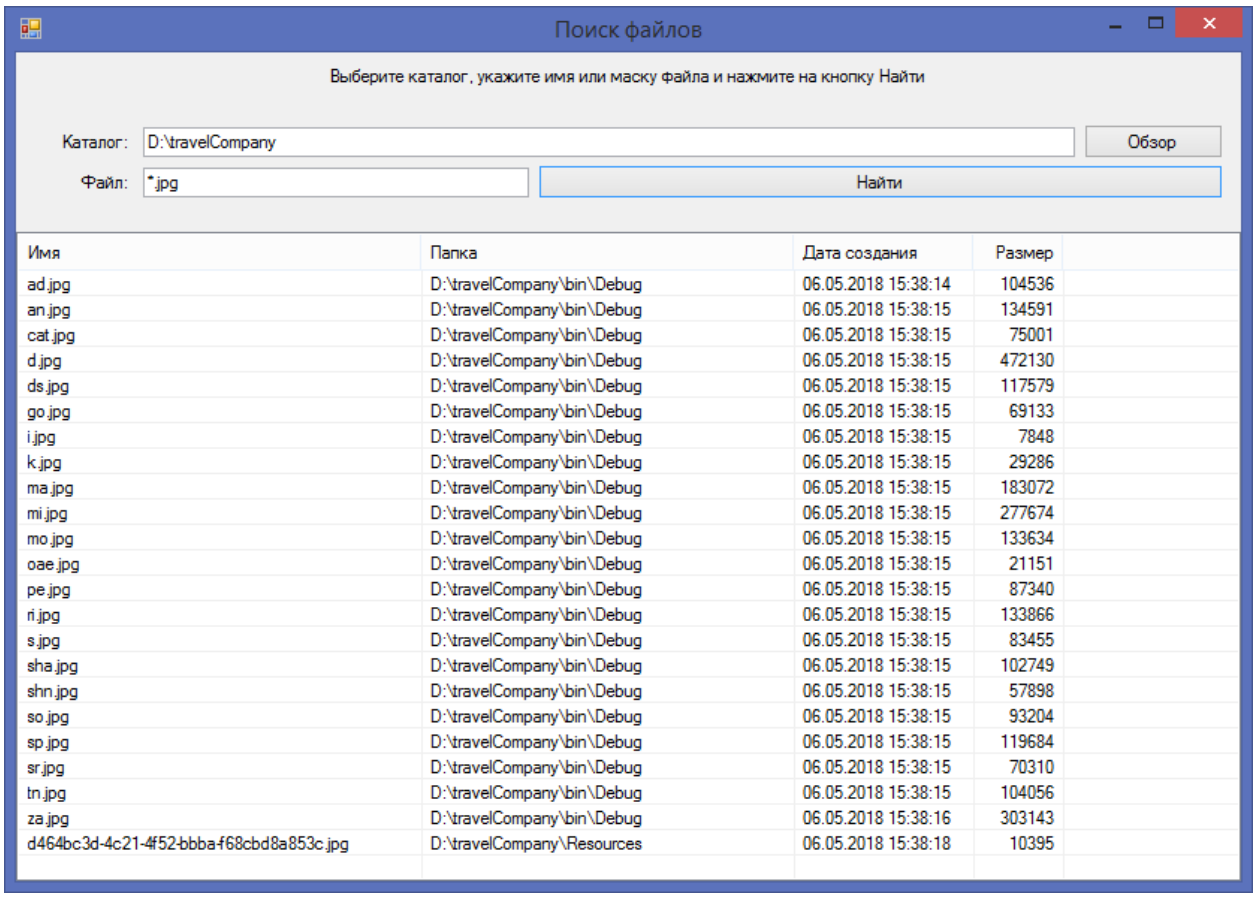


Рис.3.7. Поиск файлов

3.4. Создание и запись во временный файл

Постановка задачи: Разработать приложение, которое создает временный файл и заполнят его.

1. Создайте новый проект WindowsForms. Назовите проект TemporaryFile.

2. Активизируйте форму и измените ее имя на Создание и запись во временный файл. Укажите размер 776; 530. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

3. Добавьте на форму три элемента Panel и в свойстве **Dock** укажите Left, Bottom и Fill соответственно (рис.3.8).

4. Разместите на Panel3 элемент RichTextBox и в свойстве Dock укажите Fill.

RichTextBox отвечает за наполнение файла какой-либо информацией.

5. Добавьте на Panel1 элементы – button1(Очистить), button2(Сгенерировать), label1(Абзацев:), numericUpDown1 согласно рис.3.8.

6. В свойстве **Value** элемента numericUpDown1 установите значение 3.

7. Добавьте на Panel2 элементы – button3(Сохранить), button4(Удалить), label2 согласно рис.3.8.

В label2 показываем полный путь до файла.

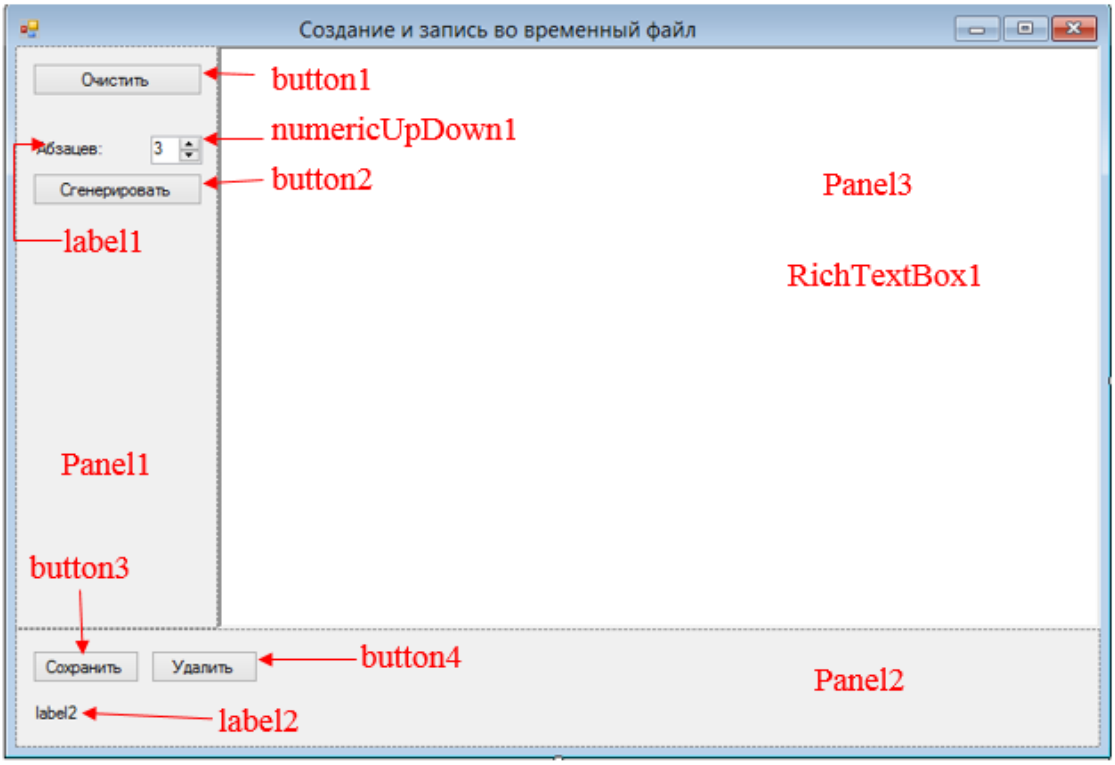


Рис.3.8. Компоненты формы «Создание и запись во временный файл»

8. Перейдите в редактор кода и добавьте генерацию имени временного файла.

Во время запуска программы формируем случайное имя файла в каталоге для хранения временных файлов.

```
public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            label2.Text = Path.GetTempFileName();
               }
```

9. Напишите обработчик событий для кнопки Сгенерировать.

Для этого воспользуемся возможностью API на сайте fish-text.ru. Создавая get запрос, мы получаем ответ с случайно сформированным осознанным текстом, после этого вставляем его в RichTextBox. Ответ приходит в формате html, для этого заменим тэги <p> на символ перехода на новую строку.

```
private void button2_Click(object sender, EventArgs e)
        {
            string str;
            using (StreamReader strr = new StreamReader(
                HttpWebRequest.Create(@"https://fish-
text.ru/get?type=paragraph&number="+numericUpDown1.Value.ToString()+
"&format=html").GetResponse().GetResponseStream()))
                str = strr.ReadToEnd();
            //  Ответ с сайта придет в формате html
            //  Абзацы заключены в теги <p></p>
            //  Преобразуем наш текст перед выводом
            richTextBox1.Text=                                          str.Replace("<p>","\n
").Replace("</p>","\n").Remove(0,1);
               }
```

10. Напишите обработчик событий для кнопки Очистить.

```
private void button1_Click(object sender, EventArgs e)
        {
            richTextBox1.Text = "";
               }
```

11. Напишите обработчик событий для кнопки Сохранить.

Воспользуемся методом SaveFile RichTextBox, который запишет нам все содержимое поле текст в созданный выше временный файл. После чего выведем сообщение о том, что файл создан успешно.

```
private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                richTextBox1.SaveFile(label2.Text, RichTextBoxStreamType.PlainText);
                MessageBox.Show("Файл "+label2.Text+" сохранен успешно", "Создание файла",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }
```

```
            catch (System.Exception ex)
            {
                MessageBox.Show("Ошибка:  " + ex.Message, "Произошла исключительная
ситуация", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
               }
```

12. Напишите обработчик событий для кнопки Удалить.

Проверяем если уже существует такой файл, то по запросу пользователя его удаляем. Если нет, то сообщим, что такого файла не существует.

```
private void button4_Click(object sender, EventArgs e)
        {
            try
            {
                if (File.Exists(label2.Text))
                {
                    if (MessageBox.Show("Вы уверены, что хотите удалить файл:\n" +
label2.Text,
                                        "Удаление папки",
                                        MessageBoxButtons.YesNo,
                                        MessageBoxIcon.Warning)
                         == DialogResult.Yes)
                    {
                        File.Delete(label2.Text);
                        MessageBox.Show("Файл " + label2.Text + " удален успешно",
"Удаление файла", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                    }
                }
                else
                {
                    MessageBox.Show("Файл " + label2.Text + " не существует.\n Удалять
нечего.", "Удаление файла", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                }
            }
            catch (System.Exception ex)
            {
                MessageBox.Show("Ошибка:  " + ex.Message, "Произошла исключительная
ситуация", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

13. Сохраните проект и запустите программу для проверки работоспособности, протестируйте создание временного файла. Результат создания временного файла показан на рис. 3.9.

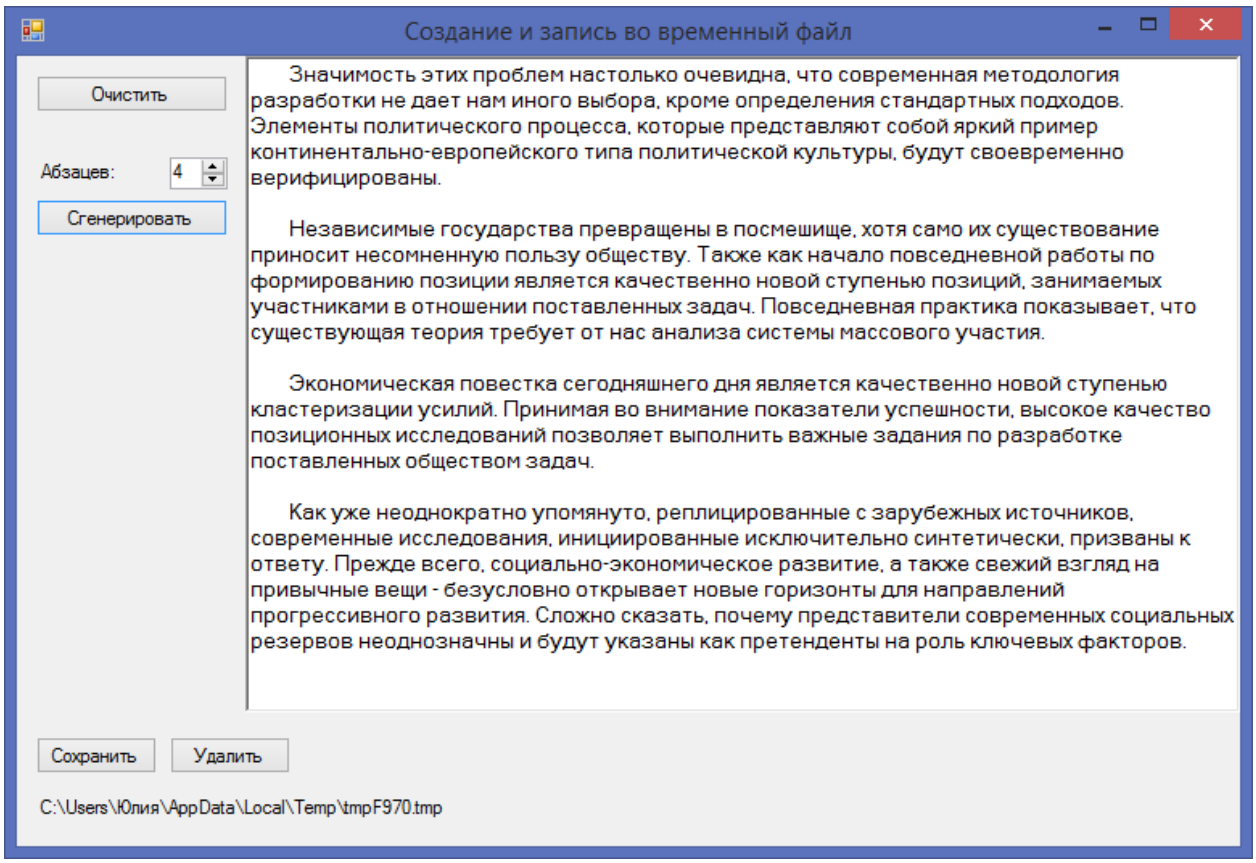


Рис.3.9. Создание и запись во временный файл

3.5. Копирование файлов

Постановка задачи: Разработать приложение, которое копирует указанный файл в указанный каталог (для указания файла и папки использовать поле ввода).

1. Создайте новый проект WindowsForms. Назовите проект CopyFile.

2. Активизируйте форму и измените ее имя на Копирование. Укажите размер 510; 248. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

3. Добавьте на форму элементы – label1(Укажите файл который необходимо скопировать:), label2(Укажите каталог в который необходимо скопировать:), label3, textBox1, textBox2, progressBar1, кнопки Закрыть, Быстрое копирование, Стандартное копирование и Обзор согласно рис.3.10.

Примечание. ProgressBar отображает индикатор выполнения операции.

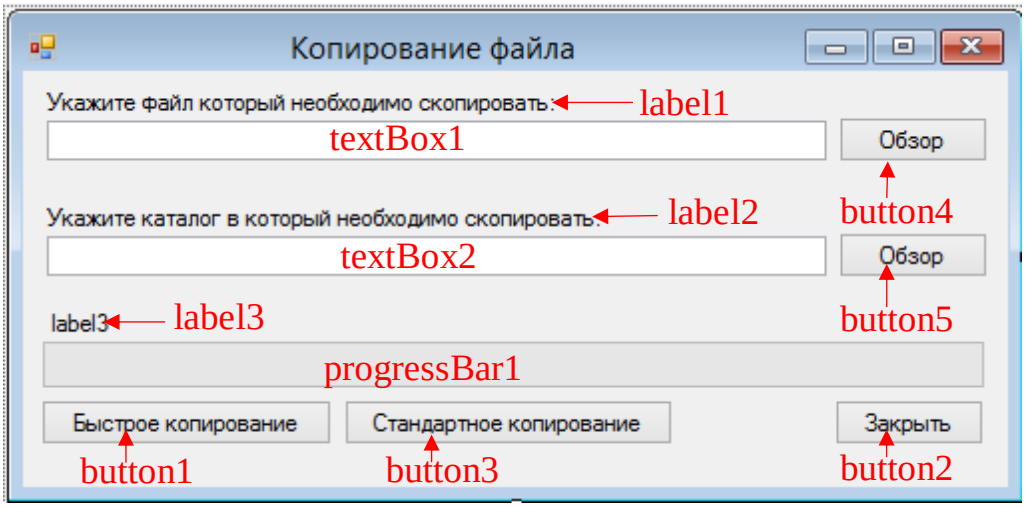


Рис.3.10. Компоненты формы «Копирование»

В textBox1 помещаем полный путь до файла, который нам необходимо скопировать.

В textBox2 отображается имя папки куда мы копируем.

По кнопкам обзор вызываются соответствующие стандартные диалоговые окна.

В ProgressBar1 будем показывать прогресс копирования.

Копирование можно будет осуществить одним из двух способов – стандартным средством C# и через файловый поток (быстрое копирование).

4. При инициализации формы label3 очищаем. Элемент будет отвечать за вывод процентов копирования файла.

```
public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            label3.Text = "";
        }
```

5. Создайте метод который делает доступным, либо недоступным кнопки и textBox в зависимости от значения.

```
private void enabledControls(bool isEnabled)
        {
            foreach (var control in this.Controls)
            {
                if (control is Button) (control as Button).Enabled = isEnabled;
                if (control is TextBox) (control as TextBox).Enabled = isEnabled;
            }
           }
```

6. Создайте метод быстрого копирования файла.

Принцип работы метода заключается в открытии двух файловых потоков, один на чтение второй на запись, в некий буфер считываем из файла определенный объем информации и записываем его.

```
public void MyCopyFile(string sourceFile, string destFile)
        {
            int BuffSize = 3 * 4 * 4096;
            int countRead = 0;
            int readStop = 1000;
            long total=0, transferred=0;
           //  В буфер мы будем считывать нашу информацию, а потом из него же и сохранять
            Byte[] Buffer = new Byte[BuffSize];
            try
            {
                //  Открываем поток для чтения
                //  Используем инструкцию using (упрощенный try-finally)
                using (FileStream sourceStream = new FileStream(sourceFile, FileMode.Open,
FileAccess.Read))
                {
                    //  Получим размер файла
                    total = sourceStream.Length;
                    readStop = (int)(total / 100 / BuffSize);
                    //  Открываем поток для записи
                    using  (FileStream  destinationStream  =  new  FileStream(destFile,
FileMode.Create, FileAccess.Write))
                    {
                        while (total > transferred)
                        {
                            //  Считаем очередную информацию
                            int bytesRead = sourceStream.Read(Buffer, 0, BuffSize);
                            //  Запишем нашу информацию в новый файл
                            destinationStream.Write(Buffer, 0, bytesRead);

                            transferred += bytesRead;

                            countRead++;
                            if (countRead == readStop)
                            {
//  Что бы при копировании большого файла не было зависаний периодически будем делать
прерывания.
```

```
                                countRead = 0;
                                //  Заполняем ProgressBar
                                progressBar1.Maximum = (int)(total / 1024);
                                progressBar1.Value = (int)(transferred / 1024);
                                label3.Text = "Скопировано:    " + ((int)(transferred /
(total / 100))).ToString() + "%";
                                Application.DoEvents();
                            }
                        }
                        //  Покажем, что у нас скопировалось 100%
                        progressBar1.Maximum = (int)(total / 1024);
                        progressBar1.Value = (int)(transferred / 1024);
                        label3.Text = "Скопировано:    " + ((int)(transferred / (total /
100))).ToString() + "%";
                    }

                }

            }
            catch (System.Exception ex)
            {
                MessageBox.Show("Ошибка:   "   +   ex.Message,   "Произошла   исключительная
ситуация", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
               }
```

7. Добавьте обработчик событий для кнопки Быстрое копирование.

При копировании большого файла, сделаем все кнопки недоступными, чтобы пользователь не нажал на какую-нибудь из них. Вызовем метод копирования после чего разблокируем кнопки, очистим label3 и ProgressBar, чтобы они приняли исходное состояние.

```
private void button1_Click(object sender, EventArgs e)
        {
            //  Копирование файла вручную.
            //  Открываем два потока: один на чтение, другой на запись
            //  и перекидываем информацию из одного потока в другой
            enabledControls(false);
            MyCopyFile(textBox1.Text,                          Path.Combine(textBox2.Text,
Path.GetFileName(textBox1.Text)));
            MessageBox.Show("Файл     скопирован     успешно",     "Копирование     файла",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            progressBar1.Maximum = 0;
            progressBar1.Value = 0;
            label3.Text = "";
            enabledControls(true);
               }
```

8. Добавьте обработчик событий для кнопки Закрыть.

```
private void button2_Click_1(object sender, EventArgs e)
        {
            Close();
               }
```

9. Добавьте обработчик событий для кнопки Стандартное копирование.

Будем копировать файл стандартным методом C#. Процесс копирования не подразумевает callback (передачу функции в качестве параметра другой функции) поэтому не можем отразить процесс копирования в ProgressBar.

```
private void button3_Click(object sender, EventArgs e)
        {
            //  Копирование файла стандартным методом c#
            try
            {
                //  Сделаем форму недоступной на время копирования
                enabledControls(false);
                label3.Text = "Идет копирование файла...";
                label3.Refresh();
                //  Скопируем файл. Установим признак, перезаписать файл, если он существует
                File.Copy(textBox1.Text,                    Path.Combine(textBox2.Text,
Path.GetFileName(textBox1.Text)), true);
                //  Выведем сообщение, что файл скопирован успешно
                MessageBox.Show("Файл   скопирован   успешно",   "Копирование   файла",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }
            catch (System.Exception ex)
            {
                MessageBox.Show("Ошибка:   "   +   ex.Message,   "Произошла   исключительная
ситуация", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
               enabledControls(true);
                label3.Text = "";
            }
        }
```

10. Добавьте обработчики событий для кнопок Обзор.

```
private void button4_Click(object sender, EventArgs e)
        {
            OpenFileDialog dialog = new OpenFileDialog();
            if (dialog.ShowDialog() == DialogResult.OK) textBox1.Text = dialog.FileName;

        }

        private void button5_Click(object sender, EventArgs e)
        {
            FolderBrowserDialog dialog = new FolderBrowserDialog();
            if    (dialog.ShowDialog()    ==    DialogResult.OK)    textBox2.Text    =
dialog.SelectedPath;
               }
```

11. Добавьте обработчики событий для textBox1 и textBox2.

Если в textBox1 и textBox2 ничего не введено, то делаем кнопки по копированию не доступными.

```
private void textBox1_TextChanged(object sender, EventArgs e)
        {
            button1.Enabled  =  button3.Enabled  =  button3.Enabled  =  ((textBox1.Text  !=
string.Empty) && (textBox2.Text != string.Empty));
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
```

```
            button1.Enabled = button3.Enabled = button3.Enabled = ((textBox1.Text != string.Empty) && (textBox2.Text != string.Empty));

        }
```

12. Сохраните проект и запустите программу для проверки работоспособности, протестируйте копирование файлов разными способами. Процесс копирования файла показан на рис. 3.11.

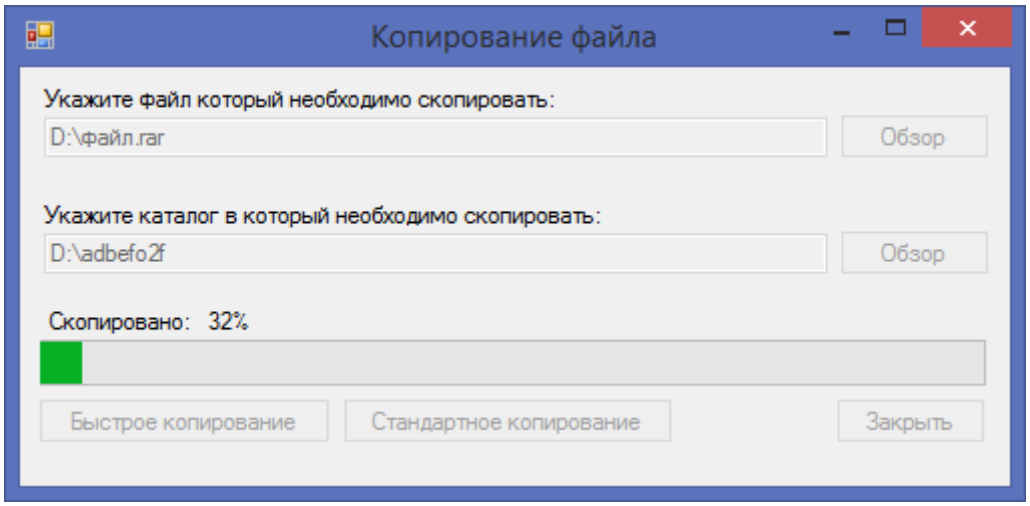


Рис.3.11. Копирование файла

3.6. Задачи для самостоятельной работы

1. Напишите программу, которая продемонстрирует работу с исключительными ситуациями в C# (на примере удаления папки, которой не существует).

2. Напишите программу, которая округляет вещественное число до указанного количества цифр после запятой. (Это вещественное число должно быть введено пользователем в поле редактирования.). Задачу нужно решить, с помощью исключительных ситуаций.

3. Напишите программу, которая вычисляет корень из числа. (Это вещественное число должно быть введено пользователем в поле редактирования.) Задачу нужно решить, с помощью исключительных ситуаций.

4. Напишите программу, в которой задается собственная исключительная ситуация при копировании файла. Ситуация возникает при копировании файла в самого себя. Продемонстрировать обработку данной исключительной ситуации.

5. Напишите программу, в которой задается собственная исключительная ситуация при задании неправильного имени файла. Ситуация возникает при создании или переименовании файла, когда имя содержит недопустимые символы, либо превышает 255 символов. Продемонстрировать обработку данной исключительной ситуации.

6. Напишите программу, которая открывает и выводит содержимое текстового файла на печать.

7. Напишите программу, которая считывает содержимое текстового файла, хранящегося на диске в корневой папке каталога, в многострочное поле ввода, а также иметь возможность сохранить этот файл в другой папке.

8. Напишите программу, которая осуществляющая транслитерацию русских букв в английские.

9. Напишите программу, которая выводит статистику содержимого файла: количество строк и символов, самое длинное слово, количество использованных знаков препинания.

10. Напишите программу, которая считывает файл, в котором находятся простые арифметические выражения. (Можно использовать только целые числа и операции +-*/). В каждой строке только одно выражение. Вычислить итоговую сумму всех выражений.

11. Напишите программу, которая "кодирует" файл, записывая его наоборот. Предусмотреть, что файл перезаписывает сам себя.

12. Напишите программу, которая ищет в файле определенный текст и заменяет его на новый. Расположение файла, заменяемый текст и новый вводит пользователь.

13. Напишите программу, которая выполняет поиск файла, в указанном пользователем каталоге. Поиск должен осуществляться по маске, который пользователь вводит в поле редактирования. Список найденных файлов со свойствами разместить в элементе ListBox.

14. Напишите программу, которая выполняет поиск файла, в указанном пользователем каталоге. Поиск должен осуществляться по диапазону размера файла и даты создания, которые пользователь вводит в поле редактирования. Список найденных файлов со свойствами разместить в элементе ListView.

15. Напишите программу, которая очищает папку для временных файлов. Учесть, что некоторые файлы могут быть заняты системой и удалены не могут.

16. Напишите программу, которая выполняет поиск файлов с одинаковыми именами, в указанном пользователем каталоге. Список найденных файлов с указанием папки, где они расположены разместить в элементе ListBox. После нахождения файлов программа должна предложить пользователю выполнить следующие действия: переименовать, удалить данный файл.

17. Напишите программу, которая выполняет поиск файлов с одинаковым содержимым, в указанном пользователем каталоге. Список найденных файлов с указанием папки, где они расположены разместить в элементе ListBox. После нахождения файлов программа должна предложить пользователю выполнить следующие действия: удалить данный файл.

18. Напишите программу, которая собирает сведения указанной папки: количество подпапок и файлов, дату последнего изменения.

19. Напишите программу, которая в элемент TreeView помещает дерево папок. Начальную папку необходимо выбрать, используя элемент BrowserFolder.

20. Напишите программу, которая, в указанной пользователем папке, создает случайное число подпапок, в каждой из которых создается случайное число временных файлов нулевой длины. Предельные числа количества вложенных папок, файлов и вложенности задается пользователем.

21. Напишите программу, которая создает некоторое количество файлов (число должно быть введено пользователем в поле редактирования), в указанном пользователем каталоге, определенного размера. Заполнять файлы случайным содержимым, включающее в себя буквы русского и латинского алфавита и цифры. Размер пользователь задает с точностью до байта.

22. Напишите программу, которая перемещает файл из папки, адрес которой вводит пользователь, в указанную папку.

23. Напишите программу, которая меняет файлы, расположенные в разных папках. Расположение папок задается пользователем.

24. Напишите программу, которая синхронизирует две папки. Файлы считаются одинаковыми, если имеют одинаковое имя и содержимое (остальные атрибуты игнорировать). Расположение папок задается пользователем.

25. Напишите программу, умеющую переименовывать файлы в папке определённым образом: в начале имени файла дописывать счетчик, начинающийся с 0 и с шагом 1, а в расширение файла в конце дописать обратный счетчик.

26. Напишите программу, которая умеет перемещать или копировать папку на новое место со всем ее содержимым. Имя копируемой папки и новое расположение задается пользователем.

27. Напишите программу, которая умеет разбивать и собирать файл. Расположение файла, размер, на который будет разбит файл, вводит пользователь.

28. Напишите программу, которая отслеживает изменения внутри папки (изменения количества файлов, изменения размеров файла, даты открытия или атрибута), указанной пользователем. Использовать элемент Timer, который через определенные промежутки времени считывает содержимое папки и анализирует происходящие изменения. Вывести сообщение, если изменения произошли.

29. Напишите программу, которая отслеживает изменения внутри папки (изменения количества файлов, изменения размеров файла, даты открытия или атрибута), указанной пользователем. Использовать элемент FileStreamWatcher, который анализирует происходящие изменения. Все происходящие изменения записываются в лог-файл.

4. Базы данных

Работа с базами данных повсеместно встречается в современных приложениях, будь то настольное приложение, веб-сайт или веб-служба. Естественно платформа .NET Framework предлагает множество способов взаимодействия с базами данных из управляемого кода. Основной системой управления базами данных для Windows является MySQL.

Можно быстро и легко отображать данные из локального файла базы данных в приложении. Для этого требуется создать набор данных и добавить в приложение элементы управления с привязкой к данным.

Для работы с MySQL необходимо подключить ссылку MySql.Data и в исходном коде добавить пространство имён MySql.Data.MySqlClient

1. В обозревателе решений нажмите правой кнопкой мыши в разделе «Ссылки» → «Добавить ссылку» (рис.4.1).

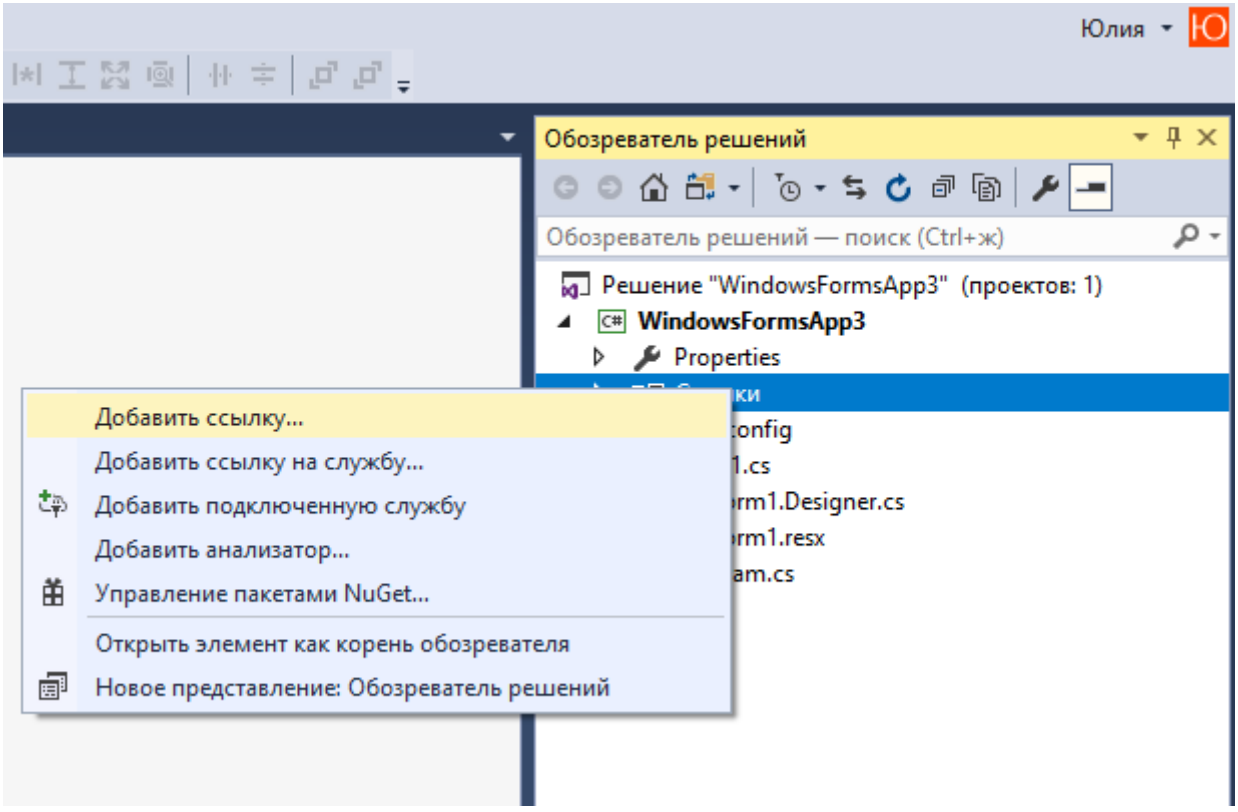


Рис.4.1. Обозреватель решений

2. В открывшемся окне, найдите и выделите компонент «MySql.Data» и нажмите "ОК" (рис.4.2).

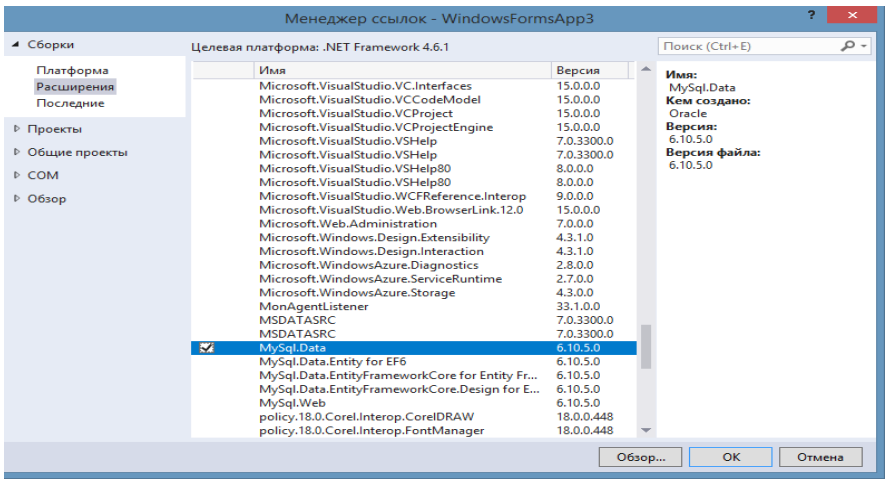


Рис.4.2. Менеджер ссылок

3. Далее в редакторе кода, добавьте пространство имён:

```
using MySql.Data.MySqlClient;
```

4. Что бы подключить базу данных, можно использовать несколько способов.

Первый способ:

Можно просто создать строковую переменную и указать в ней все данные для подключения, например:

```
string myConnectionString = "Database=DBNAME;
DataSource=IPADRESS;User Id=DBUSER;Password=DBPASSWORD";

cn = new MySqlConnection(myConnectionString);
```

Где:
DBNAME - это имя базы данных MySQL;
IPADRESS - IP адрес базы;
DBUSER - Имя пользователя базы данных;
DBPASSWORD - Пароль пользователя БД.

Второй способ:

```
MySqlConnectionStringBuilder mysqlCSB = new
MySqlConnectionStringBuilder();
mysqlCSB.Server = "192.168.0.1";  // IP адрес БД
mysqlCSB.Database = "test_db";    // Имя БД
mysqlCSB.UserID = "root";        // Имя пользователя БД
mysqlCSB.Password = "rootpass";   // Пароль пользователя БД
mysqlCSB.CharacterSet = "cp1251"; // Кодировка БД
```

4.1. Создание базы данных в MySQL

Для работы с MySQL (сохранять, изменять, удалять, получать данные), необходимо воспользоваться phpMyAdmin. Локальные серверы Denwer, XAMPP, AppServ, Open Server включают данное приложение в сборку и при установке портативного сервера phpMyAdmin уже доступен.

Рассмотрим работу по созданию полноценной базы данных.

1. Откройте интерфейс phpMyAdmin. Перейдите на вкладку Базы данных (рис.4.3). Под меткой Создать базу данных введите имя для новой Базы данных, TravelAgency (Туристическое агенство) и нажмите на кнопку **Создать.**

Примечание. Имя базы данных может быть любое, но состоять должно из латинских букв и нижнего подчёркивания.

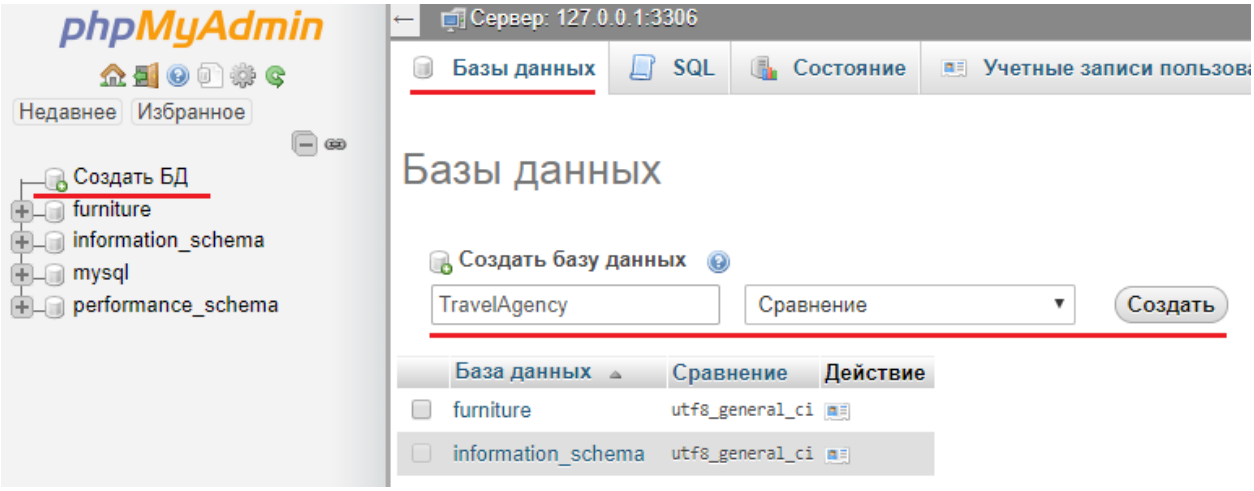


Рис.4.3. Создание базы данных MySQL в phpMyAdmin

2. Новая база данных пока пуста и не содержит ничего. Добавьте в нее таблицу, которая будет хранить данные. Для этого нажмите на название базы данных, откроется вкладка "Структура", где будут предложены опции новой таблицы. В поле "Имя" введите название новой таблицы. Таблица будет хранить данные о клиентах туристического агентства, поэтому введите название "Client" (Клиент), нам нужны такие данные как Фамилия, Имя, Отчество, Паспорт, Номер телефона, Адрес проживания и Фото, для этого в качестве количества столбцов введите цифру 8 (рис.4.4). Для создания таблицы нажмите на кнопку "Вперед".

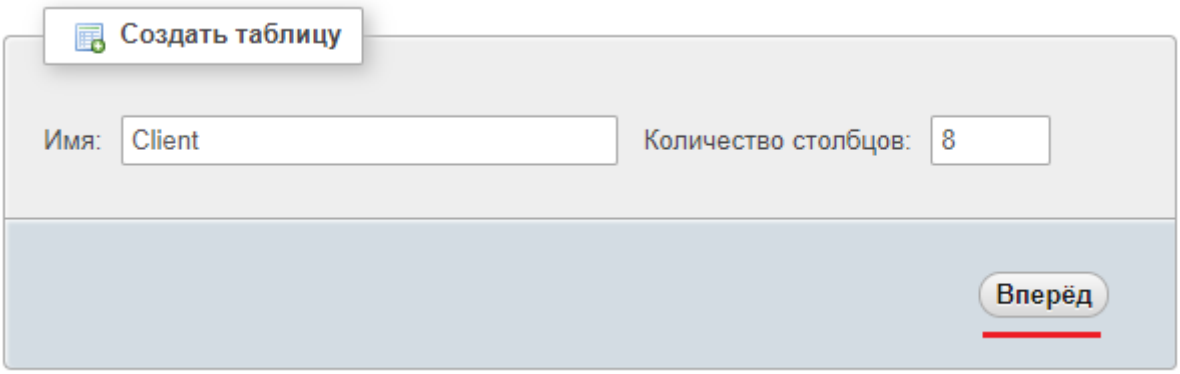


Рис.4.4. Создание таблицы базы данных MySQL в phpMyAdmin

3. У нас появится набор ячеек для установки параметров столбцов. Укажите последовательно для имен столбцов следующие: id, surname, name, patronymic, passport, phone, address, photo. В качестве типа укажите для столбца id тип INT, а для столбцов surname, name, patronymic, passport, phone, address и photo - тип TEXT. Для столбца id укажите в поле "Индекс" PRIMARY, а в поле "A_I" (Auto_Increment) поставим галочку (рис 4.5). И затем нажмите внизу на кнопку **Сохранить.**

Примечание. Auto_Increment генерирует новое порядковое значение для строк.

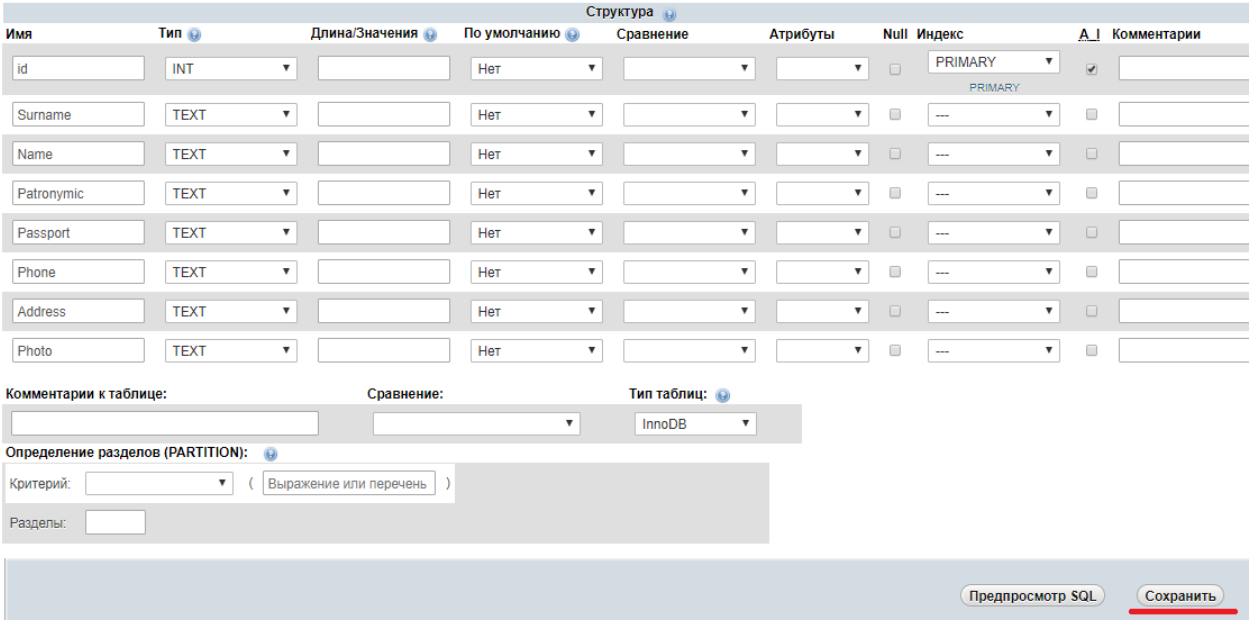


Рис. 4.5. Создание столбцов базы данных MySQL в phpMyAdmin

После создания таблицы мы сможем увидеть в колонке баз данных таблицу и ее столбцы.

4. Аналогично создайте еще три таблицы: тур, поездка и поездка клиентов (таб.4.1).

Таблица 4.1.

Таблица	Имя столбца	Тип	Индекс
Tour	id	INT	PRIMARYKEY
	Country	TEXT	
	City	TEXT	
	Season	TEXT	
	Description	TEXT	
	Price_one_day	INT	
	Hotel	TEXT	
	Visa	TEXT	
	Photo	TEXT	
Trip	id	INT	PRIMARYKEY
	Departure	DATE	
	Arrival	DATE	
	Transport	TEXT	
	Tour	INT	
TripClient	id	INT	PRIMARYKEY
	Client	INT	
	Tour	INT	

5. Чтобы работать с базой данной ее необходимо заполнить. Выберите таблицу для заполнения и перейдите во вкладку **Вставить** (рис. 4.6), после заполнения нажмите Вперед. Заполните все таблицы. Обратите внимание, что в таблице Trip, столбец Tour заполняется id таблицы Tour и в таблице TripClient столбцы Client и Tour заполняются id из таблиц Client и Tour соответственно.

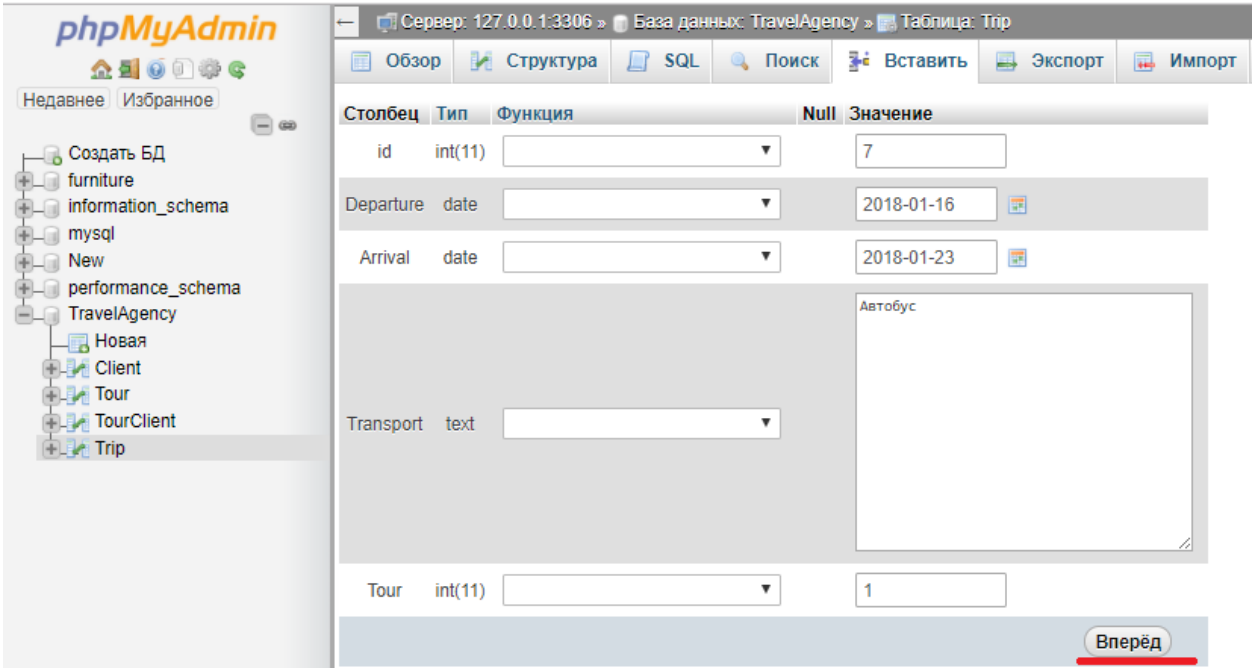


Рис. 4.6. Добавление данных MySQL

В итоге должна получится реляционная база данных MySQL туристической фирмы с четырьмя таблицами (рис. 4.7-4.9).

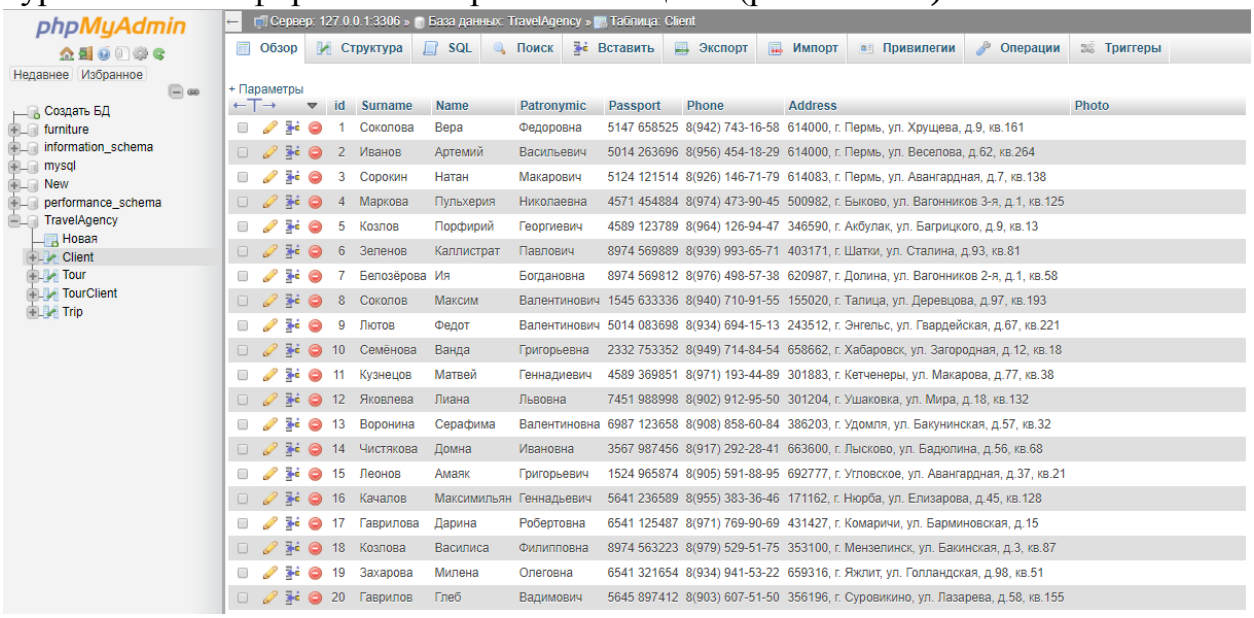


Рис.4.7. Таблица Client БД TravelAgency

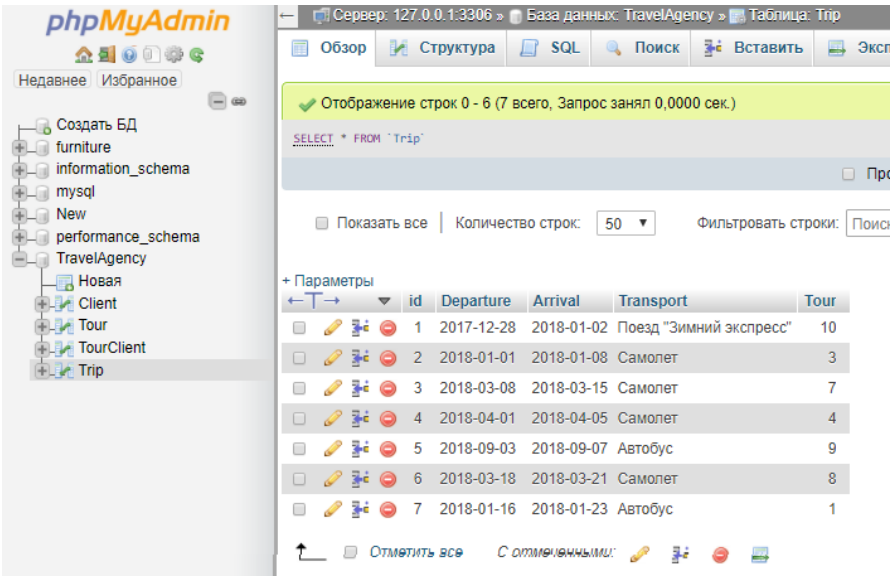


Рис. 4.8. Таблица Tour БД TravelAgency

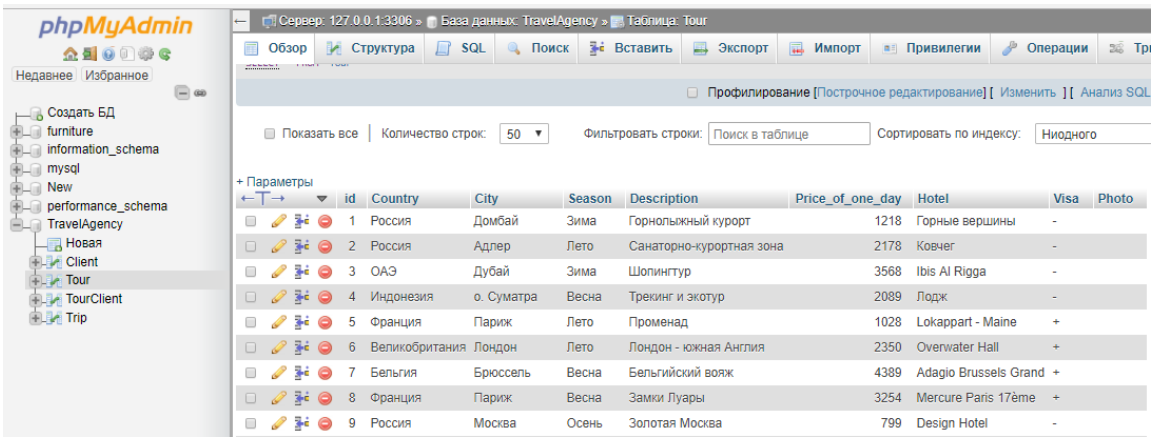


Рис.4.9. Таблица Trip БД TravelAgency

4.2. Просмотр и изменение содержимого базы данных

Постановка задачи: Разработать приложение, которое позволяет просматривать и изменять содержимое базы данных, на примере базы данных “TravelAgency”, таблица “Client”.

1. Создайте новый проект WindowsForms. Назовите проект ViewEdit.

2. Подключите ссылку MySql.Data.

3. Активизируйте форму и измените ее имя на Просмотр и изменение.

4. Расположите на форме элемент BindingNavigator (Указывает интерфейс пользователя для навигации и управления данными, привязанными к управлению формы) панели элементов, вкладка данные (рис. 4.11).

Примечание. Наша задача предусматривает только просмотр и изменение данных в таблице “Client” базы данных “TravelAgency”, поэтому можно удалить ненужные компоненты “Добавить” и “Удалить” элемента BindingNavigator**,** для простоты и удобства использования**.** Для этого необходимо щелкнуть правой кнопкой мыши по компоненту и нажать **Удалить** (рис.4.10)**.**

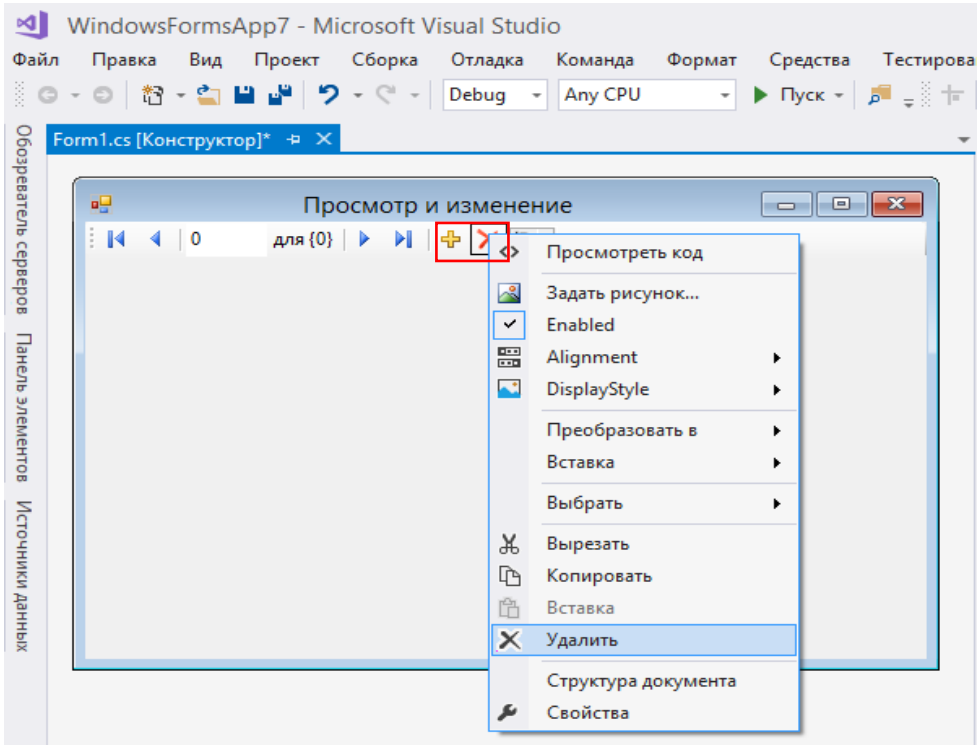


Рис.4.10. Удаление компонентов BindingNavigator

4. В свойстве элемента **AddNewItem** (Вызывает действие «Добавить новый элемент») и **DeleteItem** (Вызывает действие «Удалить») укажите значение **False.**

5. Расположите на форме две кнопки Отмена и ОК (рис.4.11).

6. Разместите кнопки в правом нижнем углу формы. Выберете кнопку Отмена и зажав клавишу *Ctrl* выбрать кнопку ОК, в свойстве **Anchor** (Определяет грани контейнера, к которому привязан определенный элемент управления. Если элемент управления привязан к определенной грани, расстояние между ближним краем элемента управления и указанной гранью постоянно) указать Top, Right.

7. Расположите на форме элемент Panel и в свойстве **Dock** заменить стандартное значение, на новое значение Bottom.

8. Добавьте на форму элемент DataGriedView. Данный элемент будет отображать данные в настраиваемой сетке. В свойствах элемента **Dock** укажите значение **Fill**, для заполнения свободного пространства.

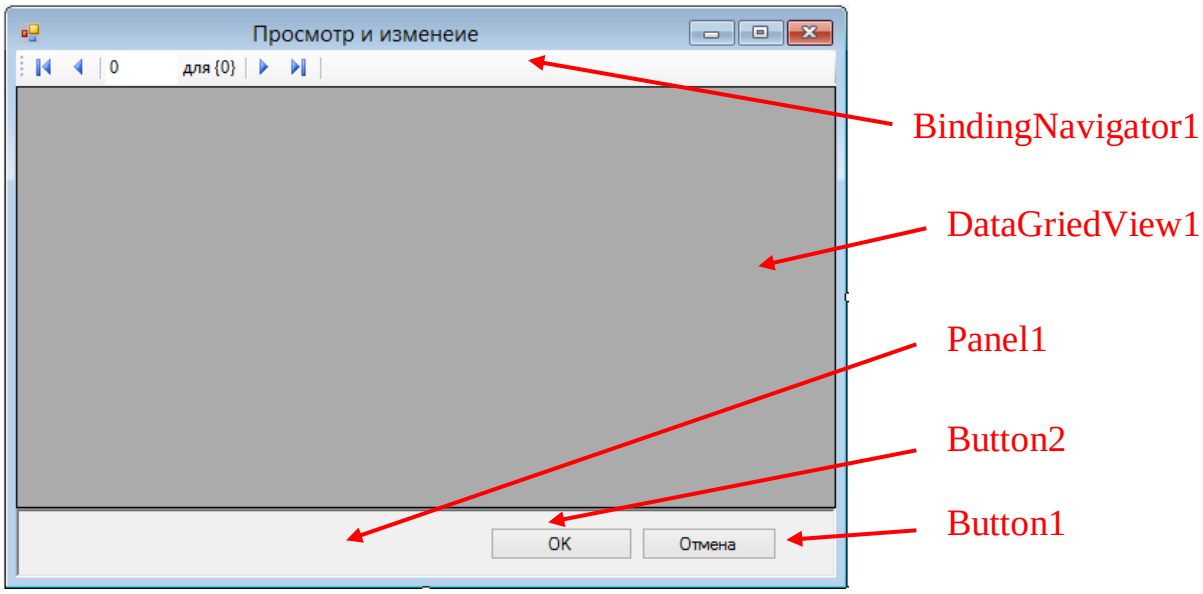


Рис.4.11. Элементы формы “Просмотр и изменение”

9. Перейдите в редактор кода и добавьте подключаемый модуль базы данных MySQL.

```
using MySql.Data.MySqlClient;
```

10. Объявите переменные для работы с базой данных.

```
public partial class Form1 : Form
{
   MySqlDataAdapter da;
   DataSet ds;
   MySqlConnection cn;
   BindingSource bs;
```

11. Добавьте обработчик события **Load** и опишите процедуру для выполнения подключения базы данных.

```
private void Form1_Load(object sender, EventArgs e)
```

```
        {
            string host = "localhost"; // Имя хоста
            string database = "TravelAgency"; // Имя базы данных
            string user = "root"; // Имя пользователя
            string password = ""; // Пароль пользователя
            string Connect = "Database=" + database + ";Datasource=" + host + ";User="
 + user + ";Password=" + password;
            cn = new MySqlConnection(Connect);
```

Примечание. Работать будем с таблицей “Client” базы данных “TravelAgency”, переименовываем поля для удобства использования, а также отключаем ненужные поля.

```
da = new MySqlDataAdapter("SELECT * FROM Client", cn.ConnectionString);
            ds = new DataSet();
            bs = new BindingSource();

            da.Fill(ds, "Client");

            bs.DataSource = ds;
            bs.DataMember = "Client";

            dataGridView1.DataSource = bs;
            dataGridView1.Columns["id"].Visible = false;
            dataGridView1.Columns["Surname"].HeaderText = "Фамилия";
            dataGridView1.Columns["Name"].HeaderText = "Имя";
            dataGridView1.Columns["Patronymic"].HeaderText = "Отчество";
            dataGridView1.Columns["Passport"].HeaderText = "Паспорт";
            dataGridView1.Columns["Phone"].HeaderText = "Телефон";
            dataGridView1.Columns["Address"].HeaderText = "Адрес";
            dataGridView1.Columns["Photo"].Visible = false;

            bindingNavigator1.BindingSource = bs;
          }
```

12. Добавьте обработчик событий для кнопок Button1 – Отмена и Button2 – ОК, закрытие приложения без сохранения и сохранение всех изменений в реальной базе данных соответственно.

```
private void button2_Click(object sender, EventArgs e)
        {
            MySqlCommandBuilder commandBuilder = new MySqlCommandBuilder(da);
            da.Update(ds.Tables[0]);
            Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();

        }
```

13. Сохраните проект и запустите программу, для проверки работоспособности, попробуйте изменить существующих клиентов в таблице. Результат показан на рис.4.12.

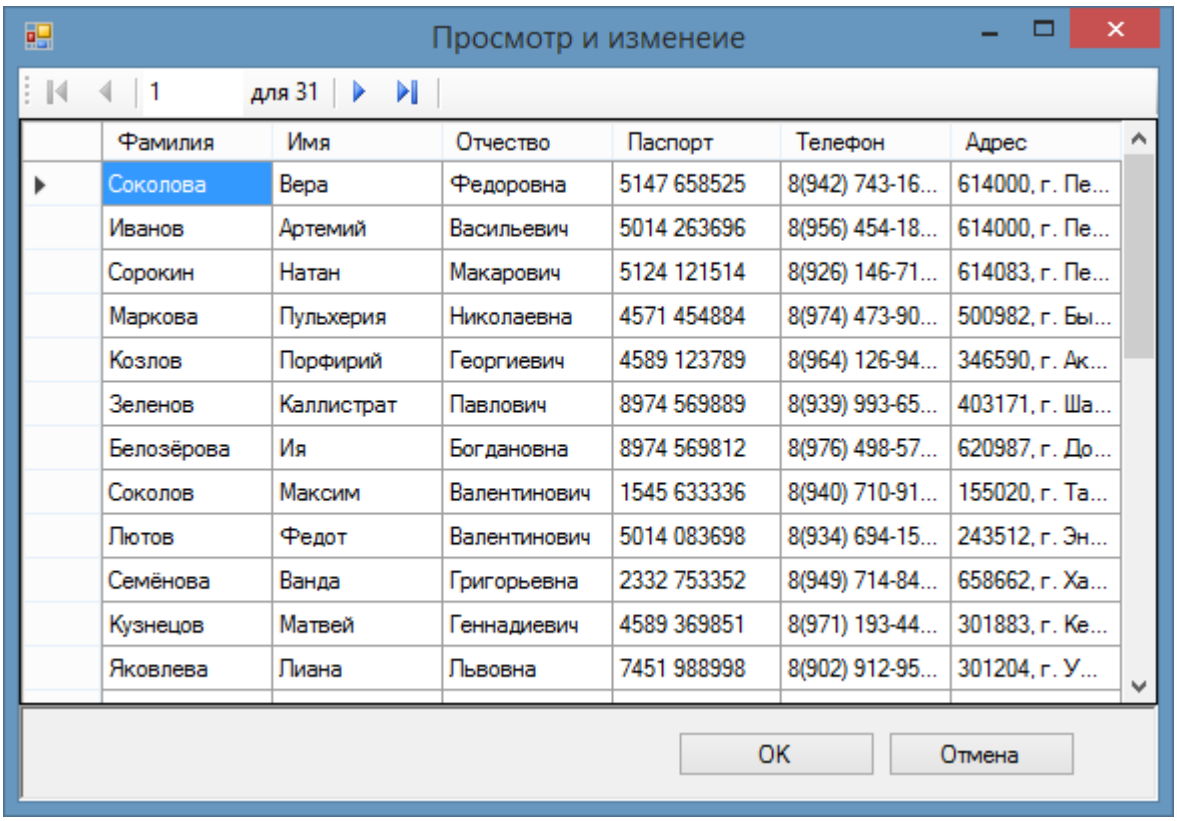


Рис.4.12. Просмотр и изменение таблицы “Clients” базы данных “TravelAgency”

4.3. Фильтрация данных по определенным диапазонам

Постановка задачи: Разработать приложение, которое фильтрует данные по определенным диапазонам (изменяемые пользователем), на примере базы данных “TravelAgency”, таблица “Client”.

1. Создайте новый проект WindowsForms. Назовите проект Filter.

2. Подключите ссылку MySql.Data.

3. Активизируйте форму и измените ее имя на Фильтрация данных.

4. Расположите на форме элемент BindingNavigator (рис. 4.13)**.** Удалите ненужные компоненты “Добавить” и “Удалить”.

5. Добавьте на форму элементы и установите их свойства, как указано в таб.4.2 (рис. 4.13).

Таблица 4.2.

Элемент / Свойство	**Button1**	**Button2**	**Panel1**	**Panel2**	**DataGriedView1**
Text	Отмена	Ок			
Dock			Bottom	Left	Fill

6. Разместить кнопки в правом нижнем углу формы. Выбрать кнопку Отмена и зажав клавишу *Ctrl* выбрать кнопку ОК, в свойстве **Anchor** указать Top, Right (см. рис 4.13).

7. Добавить на форму **CheckBox1,** в свойстве **Text** значение **CheckBox1** заменить на **Панель фильтра**.

8. Добавить на форму **lebel1…lebel6**, **textBox1…textBox6, comboBox1…comboBox6** (см. рис 4.13)**.**

Изменить значения **label**, как указано в таб.4.3.

Таблица 4.3.

Свойство / Элемент	**Text**
lebel1	Фамилия
lebel1	Имя
lebel1	Отчество
lebel1	Паспорт
lebel1	Телефон
lebel1	Адрес

9. Выделить все элементы **comboBox1…comboBox6.**

В свойстве **Items** добавить коллекцию:

И
ИЛИ

В свойстве **Text** указать значение И.

В свойстве **Enabled** (Указывает, включен ли элемент управления) указать False.

10. Добавить кнопки Button3 и Button4, в свойстве **Text** указать Фильтр и Очистить, соответственно (рис.4.13).

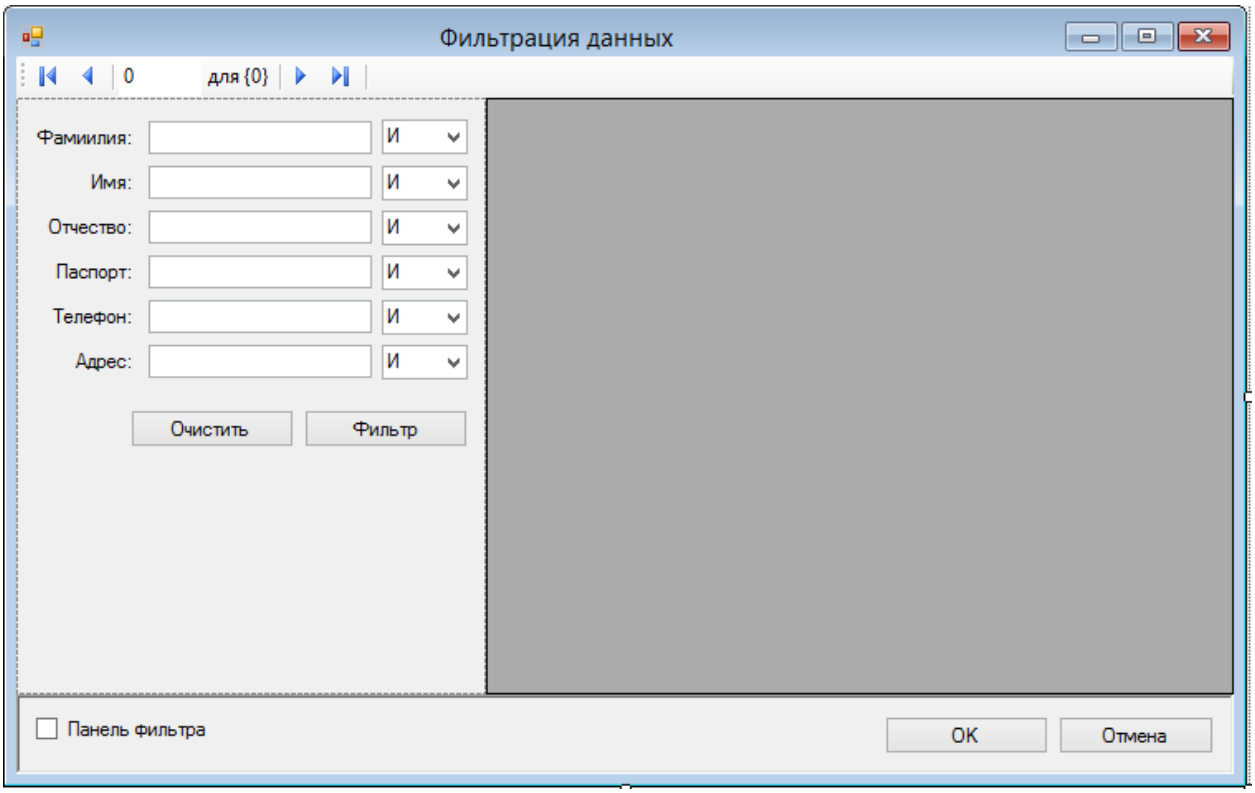


Рис. 4.13 Форма “Фильтрация данных”

11. Перейдите в редактор кода и добавьте подключаемый модуль базы данных MySQL.

```
using MySql.Data.MySqlClient;
```

12. Объявите переменные для работы с базой данных.

```
public partial class Form1 : Form
    {
        MySqlDataAdapter da;
        DataSet ds;
        DataTable dt;
        MySqlConnection cn;
        BindingSource bs;
```

13. Добавьте обработчик событий **Load** (процедура подключения базы данных) для Form1**.**

Выберете таблицу “Client” базы данных “TravelAgency”, переименуйте поля для удобства использования, а также отключите ненужные поля.

```
private void Form1_Load(object sender, EventArgs e)
        {
            string host = "localhost"; // Имя хоста
            string database = "TravelAgency"; // Имя базы данных
            string user = "root"; // Имя пользователя
            string password = ""; // Пароль пользователя
            string Connect = "Database=" + database + ";Datasource=" + host + ";User="
+ user + ";Password=" + password;
            cn = new MySqlConnection(Connect);
            cn.Open();
            da = new MySqlDataAdapter("SELECT * FROM Client", cn.ConnectionString);
            ds = new DataSet();
            dt = new DataTable("Client");
            bs = new BindingSource();
            da.Fill(dt);
            ds.Tables.Add(dt);

            bs.DataSource = ds;
            bs.DataMember = "Client";

            dataGridView1.DataSource = bs;
            dataGridView1.Columns["id"].Visible = false;
            dataGridView1.Columns["Surname"].HeaderText = "Фамилия";
            dataGridView1.Columns["Name"].HeaderText = "Имя";
            dataGridView1.Columns["Patronymic"].HeaderText = "Отчество";
            dataGridView1.Columns["Passport"].HeaderText = "Паспорт";
            dataGridView1.Columns["Phone"].HeaderText = "Телефон";
            dataGridView1.Columns["Address"].HeaderText = "Адрес";
            dataGridView1.Columns["Photo"].Visible = false;

            bindingNavigator1.BindingSource = bs;
            cn.Close();

        }
```

14. Добавьте обработчик событий для кнопок Button1 – Отмена и Button2 – ОК.

```
private void button2_Click(object sender, EventArgs e)
        {
            MySqlCommandBuilder commandBuilder = new MySqlCommandBuilder(da);
            da.Update(ds.Tables[0]);
            Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();

        }
```

15. Для CheckBox1 добавьте обработчик события **CheckedChanged** (при нажатии откроется панель фильтров).

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            panel2.Visible = checkBox1.Checked;

        }
```

16. Добавьте функцию, которая конвертирует русские “И” “ИЛИ” в английские “AND” и “OR” для использования их в SQL-запросе.

```
string ConvertLogic(string Operator)
        {
            switch (Operator)
            {
                case "И": { return "AND"; }
                case "ИЛИ": { return "OR"; }
                default: { return ""; }

            }
```

17. Добавьте обработчик событий для кнопки Button3 – Фильтр.

Примечание. Сформируем строку для SQL-запроса, применяем SQL-запрос и обновляем DataGriedView1.

```
private void button3_Click(object sender, EventArgs e)
        {
            string textFiltr = "", textLogOp ="";
            if (textBox1.Text != "") { textFiltr += ConvertLogic(textLogOp) + " Surname
like '%" + textBox1.Text + "%' "; textLogOp = comboBox1.Text;}
            if (textBox2.Text != "") { textFiltr += ConvertLogic(textLogOp) + " Name
like  '%" + textBox2.Text + "%' "; textLogOp = comboBox2.Text; }
            if (textBox3.Text != "") { textFiltr += ConvertLogic(textLogOp) + "
Patronymic like  '%" + textBox3.Text + "%' "; textLogOp = comboBox3.Text; }
            if (textBox4.Text != "") { textFiltr += ConvertLogic(textLogOp) + "
Passport like  '%" + textBox4.Text + "%' "; textLogOp = comboBox4.Text; }
            if (textBox5.Text != "") { textFiltr += ConvertLogic(textLogOp) + " Phone
like  '%" + textBox5.Text + "%' "; textLogOp = comboBox5.Text; }
            if (textBox6.Text != "") { textFiltr += ConvertLogic(textLogOp) + " Address
like  '%" + textBox6.Text + "%' "; textLogOp = comboBox6.Text; }
            textFiltr = "SELECT * FROM Client where " + textFiltr;
            da.SelectCommand.CommandText = textFiltr;
            dt.Clear();
            da.Fill(dt);
            dataGridView1.Refresh();

            }
```

18. Для каждого TextBox пропишите обработчик событий **TextChanged** (Событие возникает, когда в Control изменяется значение свойства Text).

```
 private void textBox1_TextChanged(object sender, EventArgs e)
        {
            comboBox1.Enabled = !(textBox1.Text == "");
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            comboBox2.Enabled = !(textBox2.Text == "");
        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {
            comboBox3.Enabled = !(textBox3.Text == "");
        }

        private void textBox4_TextChanged(object sender, EventArgs e)
        {
```

```
    comboBox4.Enabled = !(textBox4.Text == "");
}

private void textBox5_TextChanged(object sender, EventArgs e)
{
    comboBox5.Enabled = !(textBox5.Text == "");
}

private void textBox6_TextChanged(object sender, EventArgs e)
{
    comboBox6.Enabled = !(textBox6.Text == "");

}
```

19. Добавьте обработчик событий для кнопки Button4 – Очистить, очищаем Textbox и убираем фильтр с базы данных.

```
private void button4_Click(object sender, EventArgs e)
        {
            foreach (Control c in panel2.Controls)
            {
                if (c is TextBox) c.Text = "";
            }
            da.SelectCommand.CommandText = "SELECT * FROM Client";
            dt.Clear();
            da.Fill(dt);
            dataGridView1.Refresh();

             }
```

20. Сохраните проект и запустите программу для проверки работоспособности, попробуйте фильтр с разными комбинациями на существующих клиентов в таблице. Результат показан на рис. 4.14-4.15.

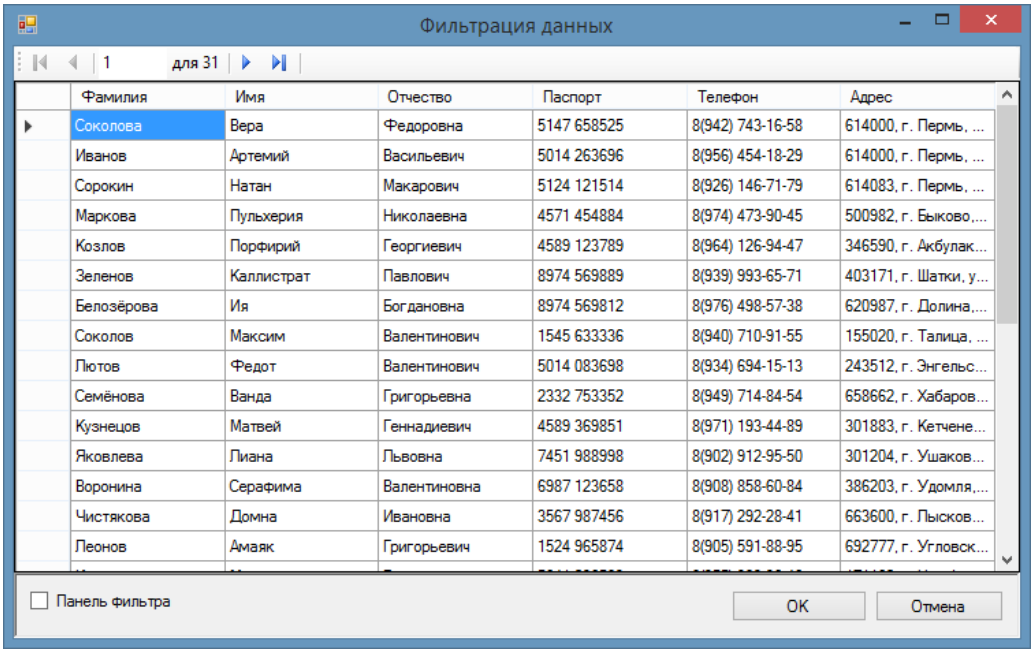


Рис. 4.14. Фильтрация данных №1

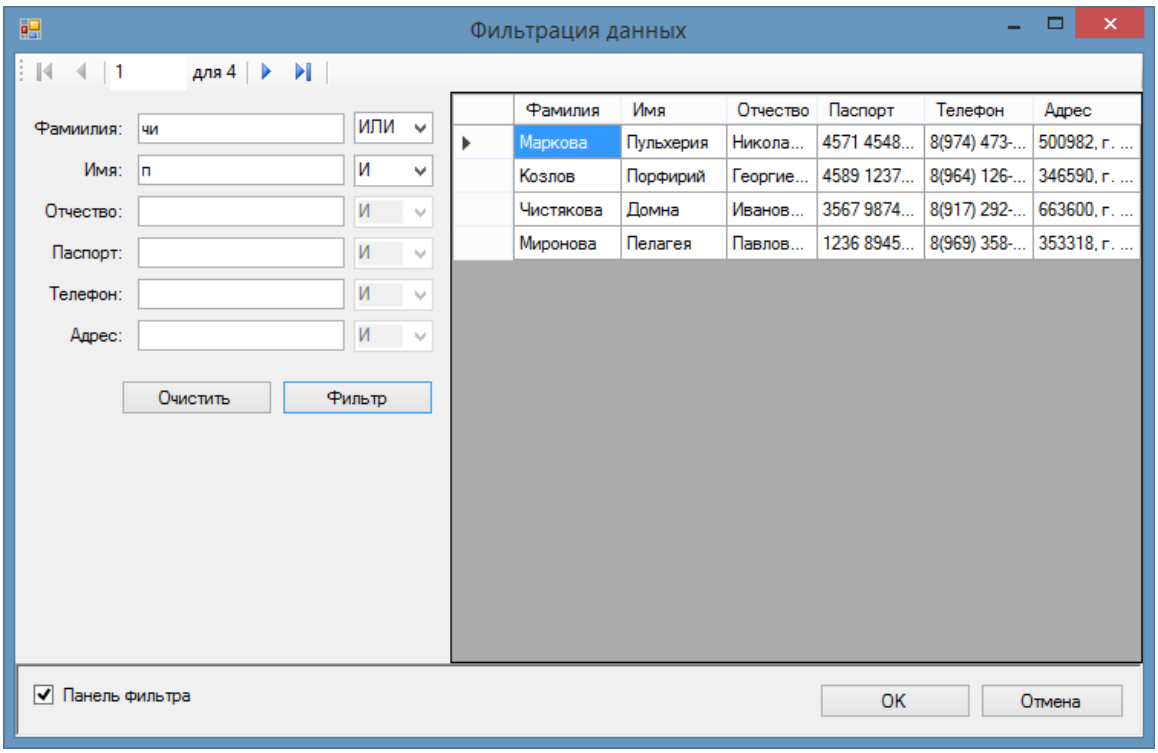


Рис.4.15. Фильтрация данных №2

4.4. Поиск в базе данных

Постановка задачи: Разработать приложение, реализующее поиск в базе данных, на примере базы данных “TravelAgency”, таблица “Client”.

1. Создайте новый проект WindowsForms. Назовите проект Search.

2. Подключите ссылку MySql.Data.

3. Активизируйте форму и измените ее имя на Поиск.

4. Расположите на форме элемент **BindingNavigator** (рис.4.16)**.** Удалите ненужные компоненты “Добавить” и “Удалить”.

5. Добавить на форму элементы и установить их свойства, как указано в таб.4.4 (рис.4.16).

Таблица 4.4.

Элемент / Свойство	**Button1**	**Button2**	**Panel1**	**DataGriedView1**
Text	Отмена	Ок		
Dock			Bottom	Fill

6. Разместить кнопки в правом нижнем углу формы. Выбрать кнопку Отмена и зажав клавишу *Ctrl* выбрать кнопку ОК, в свойстве **Anchor** указать Top, Right (рис.4.16).

7. Добавить на форму CheckBox заметить в свойстве **Text** значение на Поиск (рис.4.16).

8. Добавить элемент TextBox (рис.4.16). В свойстве **Enabled** указать False. Свойство указывает, на то, что элемент не активен при первой загрузке.

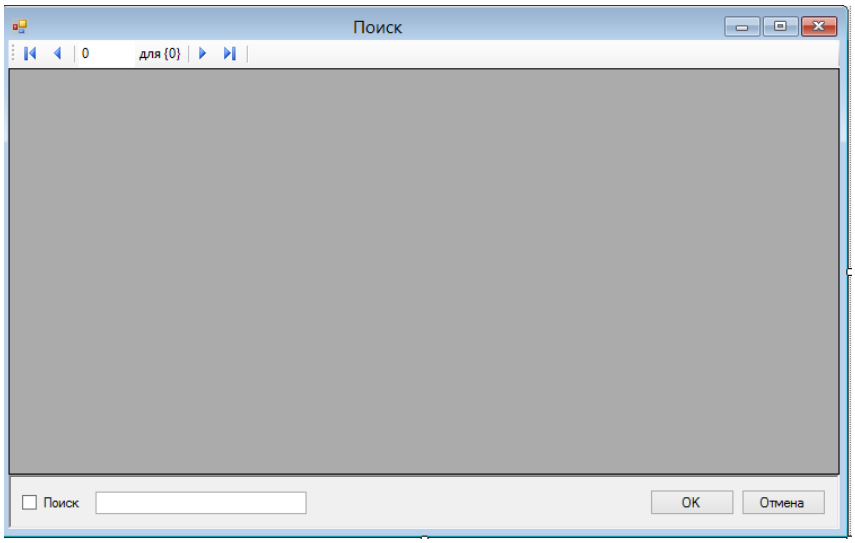


Рис.4.16. Форма “Поиск”

9. Перейдите в редактор кода и добавьте подключаемый модуль базы данных MySQL.

```
using MySql.Data.MySqlClient;
```

10. Объявите переменные для работы с базой данных.

```
public partial class Form1 : Form
    {
        MySqlDataAdapter da;
        DataSet ds;
        DataTable dt;
        MySqlConnection cn;
        BindingSource bs;
```

11. Добавьте обработчик событий **Load** (процедура подключения базы данных) для Form1**.**

Выберете таблицу “Client” базы данных “TravelAgency”, переименуйте поля для удобства использования, а также отключите ненужные поля.

```
private void Form1_Load(object sender, EventArgs e)
        {
            string host = "localhost"; // Имя хоста
            string database = "TravelAgency"; // Имя базы данных
            string user = "root"; // Имя пользователя
            string password = ""; // Пароль пользователя
            string Connect = "Database=" + database + ";Datasource=" + host + ";User="
+ user + ";Password=" + password;
            cn = new MySqlConnection(Connect);
            cn.Open();
            da = new MySqlDataAdapter("SELECT * FROM Client", cn.ConnectionString);
            ds = new DataSet();
            dt = new DataTable("Client");
            bs = new BindingSource();
            da.Fill(dt);
            ds.Tables.Add(dt);

            bs.DataSource = dt;
            dataGridView1.DataSource = bs;
            dataGridView1.Columns["id"].Visible = false;
            dataGridView1.Columns["Surname"].HeaderText = "Фамилия";
            dataGridView1.Columns["Name"].HeaderText = "Имя";
            dataGridView1.Columns["Patronymic"].HeaderText = "Отчество";
            dataGridView1.Columns["Passport"].HeaderText = "Паспорт";
            dataGridView1.Columns["Phone"].HeaderText = "Телефон";
            dataGridView1.Columns["Address"].HeaderText = "Адрес";
            dataGridView1.Columns["Photo"].Visible = false;

            bindingNavigator1.BindingSource = bs;
            cn.Close();

             }
```

12. Для CheckBox1 добавьте обработчик события **CheckedChanged** (при нажатии откроется доступ к поиску).

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            textBox1.Enabled = checkBox1.Checked;
        }
```

13. Добавьте обработчик событий для кнопок Button1 – Отмена и Button2 – ОК.

```
private void button2_Click(object sender, EventArgs e)
        {
            MySqlCommandBuilder commandBuilder = new MySqlCommandBuilder(da);
            da.Update(ds.Tables[0]);
            Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();
        }
```

14. Для TextBox1 пропишите обработчик события **TextChanged**. Пробегаем по всем столбцам и пытаемся найти в них искомый текст.

```
private void textBox1_TextChanged(object sender, EventArgs e)
        {
            for (int col=1; col < dataGridView1.Columns.Count-1; col++)
            {
                int    itemFound    =    bs.Find(dataGridView1.Columns[col].Name,
textBox1.Text);

                if (itemFound > -1)
                {
                    bs.Position = itemFound;
                    dataGridView1.ClearSelection();
                    dataGridView1.Rows[itemFound].Selected = true;
                    dataGridView1.CurrentCell                                    =
dataGridView1[dataGridView1.Columns[col].Name, itemFound];
                }
            }

        }
```

15. Сохраните проект и запустите программу для проверки работоспособности, протестируйте поиск для каждого столбца. Результат показан на рис. 4.17.

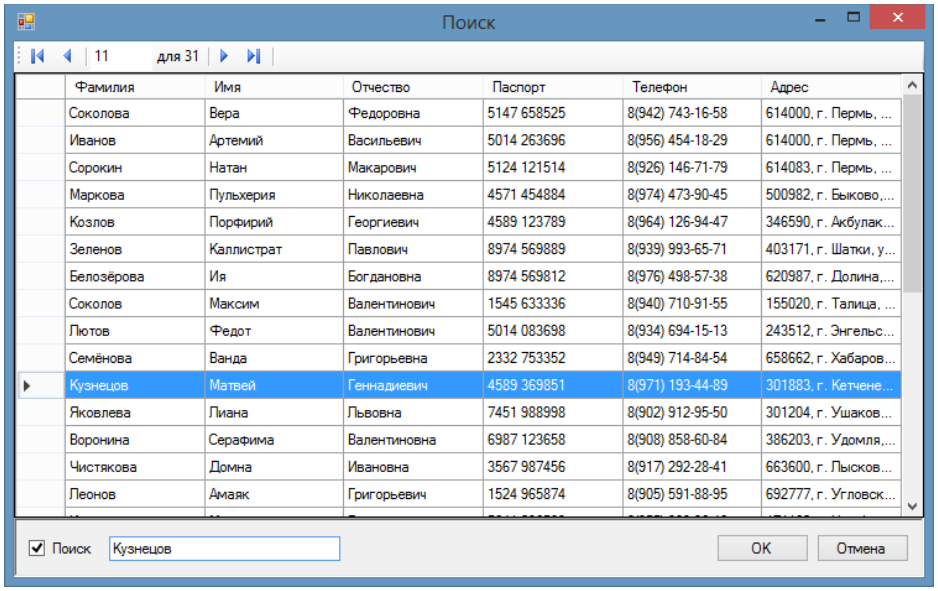


Рис. 4.17. Поиск

4.5. Сортировка записей по выбранному полю

Постановка задачи: Разработать приложение, которое сортирует записи в таблице по выбранному полю, на примере базы данных “TravelAgency”, таблица “Client”.

1. Создайте новый проект WindowsForms. Назовите проект Sorting.

2. Подключите ссылку MySql.Data.

3. Активизируйте форму и измените ее имя на Сортировка.

4. Расположите на форме элемент **BindingNavigator** (рис.4.18)**.** Удалить ненужные компоненты “Добавить” и “Удалить”.

5. Добавить на форму элементы и установить их свойства, как указано в таб.4.5 (рис.4.18).

Таблица 4.5.

Элемент / Свойство	**Button1**	**Button2**	**Panel1**	**DataGriedView1**
Text	Отмена	Ок		
Dock			Bottom	Fill

6. Разместить кнопки в правом нижнем углу формы. Выбрать кнопку Отмена и зажав клавишу *Ctrl* выбрать кнопку ОК, в свойстве **Anchor** указать **Top, Right** (см. рис 6.1).

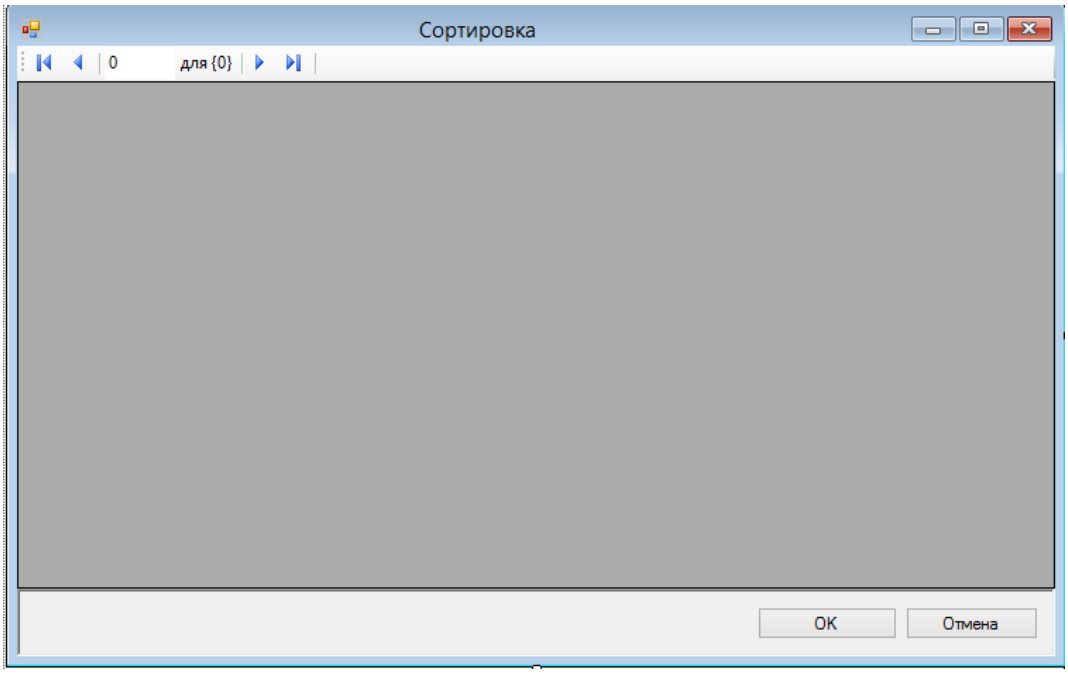


Рис.4.18. Форма “Сортировка”

7. Перейдите в редактор кода и добавьте подключаемый модуль базы данных MySQL.

```
using MySql.Data.MySqlClient;
```

8. Объявите переменные для работы с базой данных.

```
public partial class Form1 : Form
    {
        MySqlDataAdapter da;
        DataSet ds;
        DataTable dt;
        MySqlConnection cn;
        BindingSource bs;
```

9. Добавьте функцию вычисления максимального числа поля id таблицы “Client”.

```
public int max()
     {   int ret = 0;
         string query = "SELECT * FROM Client";
         MySqlCommand com = new MySqlCommand(query, cn);
         MySqlDataReader rd = com.ExecuteReader();
         while (rd.Read())
         {
             ret = rd.GetInt32(0);
         }
         return ret;
     }
```

10. Добавьте обработчик событий **Load** (процедура подключения базы данных) для Form1**.**

Выберете таблицу “Client” базы данных “TravelAgency”, переименуйте поля для удобства использования, а также отключите ненужные поля.

```
private void Form1_Load(object sender, EventArgs e)
        {
            string host = "localhost"; // Имя хоста
            string database = "TravelAgency"; // Имя базы данных
            string user = "root"; // Имя пользователя
            string password = ""; // Пароль пользователя
            string Connect = "Database=" + database + ";Datasource=" + host + ";User="
+ user + ";Password=" + password;
            cn = new MySqlConnection(Connect);
            cn.Open();
            da = new MySqlDataAdapter("SELECT * FROM Client", cn.ConnectionString);
            ds = new DataSet();
            dt = new DataTable("Client");
            bs = new BindingSource();
            da.Fill(dt);
            ds.Tables.Add(dt);

            bs.DataSource = dt;
            dataGridView1.DataSource = bs;
            dataGridView1.Columns["id"].Visible = false;
            dataGridView1.Columns["Surname"].HeaderText = "Фамилия";
            dataGridView1.Columns["Name"].HeaderText = "Имя";
            dataGridView1.Columns["Patronymic"].HeaderText = "Отчество";
            dataGridView1.Columns["Passport"].HeaderText = "Паспорт";
            dataGridView1.Columns["Phone"].HeaderText = "Телефон";
```

```
dataGridView1.Columns["Address"].HeaderText = "Адрес";
dataGridView1.Columns["Photo"].Visible = false;

bindingNavigator1.BindingSource = bs;
cn.Close();

 }
```

11. Добавьте обработчик событий для кнопок Button1 – Отмена и Button2 – ОК, закрытие приложения без сохранения и сохранение всех изменений в реальной базе данных соответственно.

```
private void button2_Click(object sender, EventArgs e)
        {
            MySqlCommandBuilder commandBuilder = new MySqlCommandBuilder(da);
            da.Update(ds.Tables[0]);
            Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();
             }
```

12. Добавьте обработчик событий **ColumnHeaderMouseClick** (Возникает при щелчке заголовка столбца) для элемента DataGriedView1**.**

```
private      void      dataGridView1_ColumnHeaderMouseClick(object      sender,
DataGridViewCellMouseEventArgs e)
       {
           DataGridViewColumn newColumn = dataGridView1.Columns[e.ColumnIndex];
           DataGridViewColumn oldColumn = dataGridView1.SortedColumn;
           ListSortDirection direction;

           // Если oldColumn равно null, то DataGridView не сортируется.
           if (oldColumn != null)
           {

               // Отсортируйте столбец снова, изменив значение SortOrder.
               if (oldColumn == newColumn &&
                   dataGridView1.SortOrder == SortOrder.Ascending)
               {
                   direction = ListSortDirection.Descending;
               }
               else
               {
                   // Сортировка нового столбца и удаление старого SortGlyph.
                   direction = ListSortDirection.Ascending;
                   oldColumn.HeaderCell.SortGlyphDirection = SortOrder.None;
               }
           }
           else
           {
               direction = ListSortDirection.Ascending;
           }

           // Сортировка выбранного столбца.
           dataGridView1.Sort(newColumn, direction);
           newColumn.HeaderCell.SortGlyphDirection =
               direction == ListSortDirection.Ascending ?
               SortOrder.Ascending : SortOrder.Descending;
       }
```

```
}
```

13. Сохраните проект и запустите программу для проверки работоспособности, протестируйте сортировку для каждого столбца. Результат показан на рис. 4.19.

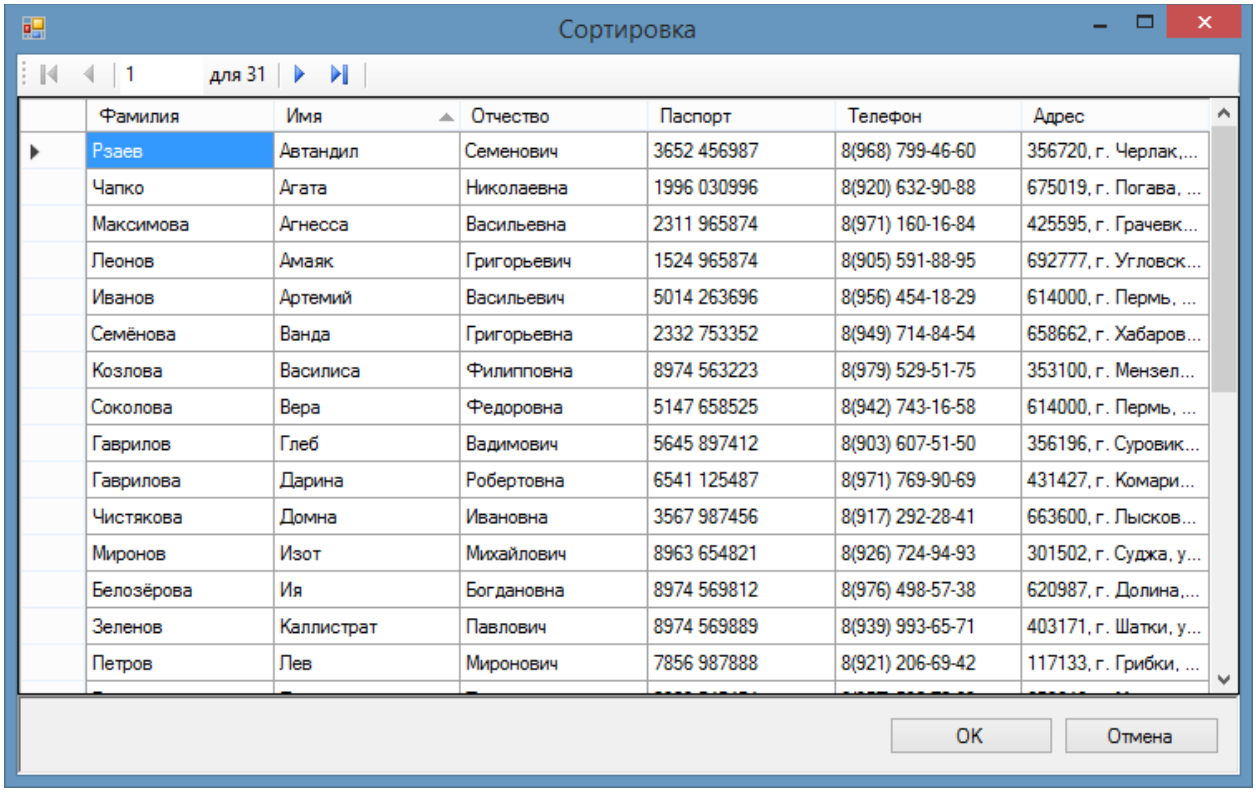


Рис.4.19. Сортировка

4.6. Вставка и удаление записей в поле

Постановка задачи: Разработать приложение, которое вставляет и удаляет запись в поле, на примере базы данных “TravelAgency”, таблица “Client”.

1. Создайте новый проект WindowsForms. Назовите проект InsertDelete.

2. Подключите ссылку MySql.Data.

3. Активизируйте форму и измените ее имя на Вставка и удаление.

4. Расположите на форме элемент BindingNavigator (рис.4.20).

5. Добавьте на форму элементы и установите их свойства, как указано в таб.4.6 (рис.4.20).

Таблица 4.6.

Элемент / Свойство	**Button1**	**Button2**	**Panel1**	**DataGriedView1**
Text	Отмена	Ок		
Dock			Bottom	Fill

6. Разместите кнопки в правом нижнем углу формы. Выбрать кнопку Отмена и зажав клавишу *Ctrl* выбрать кнопку ОК, в свойстве **Anchor** указать Top, Right (рис.4.20).

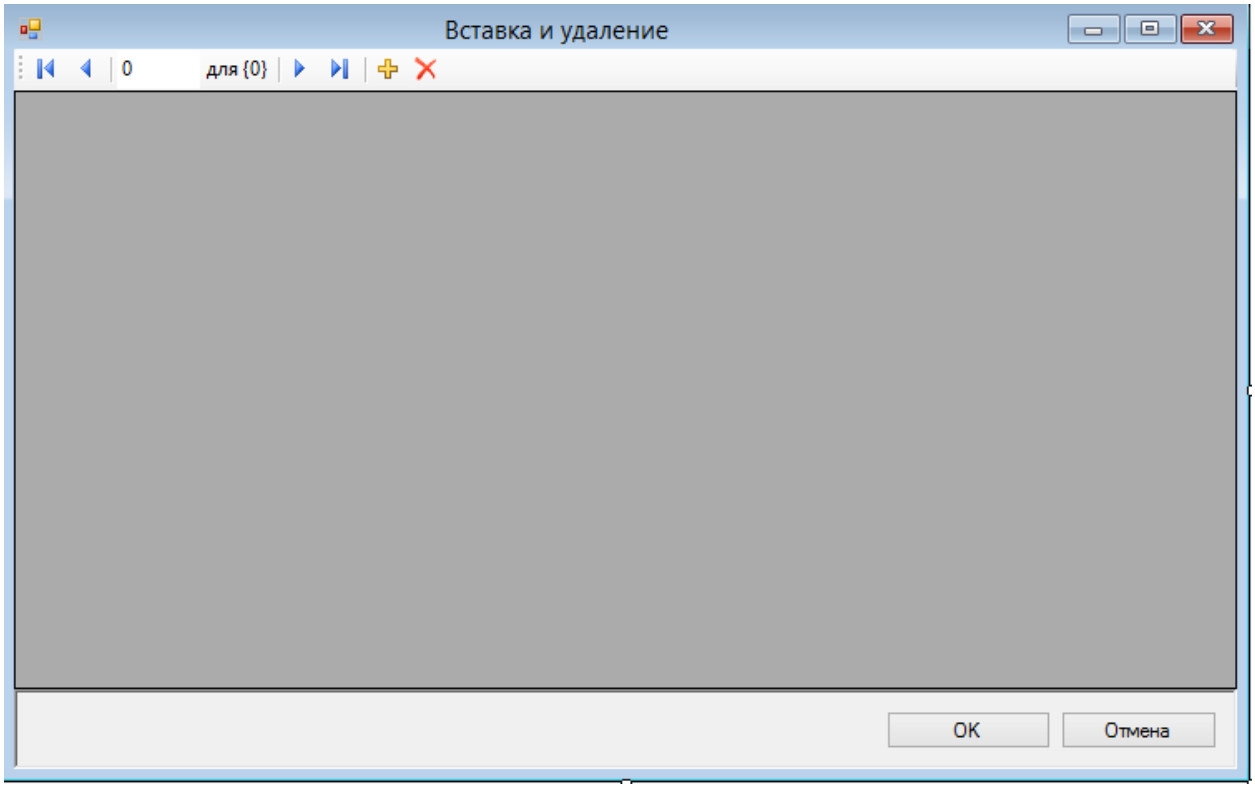


Рис.4.20. Форма «Вставка и удаление»

7. Нам необходимо создать вторую форму для добавления клиентов.

Создайте вторую форму, активизируйте ее и измените имя на Клиент (рис.4.21)**.**

8. Расположите на форме элемент Panel и в свойстве **Dock** заменить стандартное значение, на новое значение Bottom (рис.4.21).

9. Расположите две кнопки Загрузить – Button1 и Очистить – Button2 (рис.4.21).

10. На панели расположите еще две кнопки Добавить – Button3 и Отмена – Button4 (рис.4.21).

11. Добавьте на форму **lebel1…lebel6**, **textBox1…textBox6** (рис.4.21)**.**

Изменить значения **label**, как указано в таб.4.7.

Таблица 4.7.

Свойство **Элемент**	**Text**
lebel1	Фамилия
lebel1	Имя
lebel1	Отчество
lebel1	Паспорт
lebel1	Телефон
lebel1	Адрес

11. Добавьте элемент PictureBox. (рис.4.21).

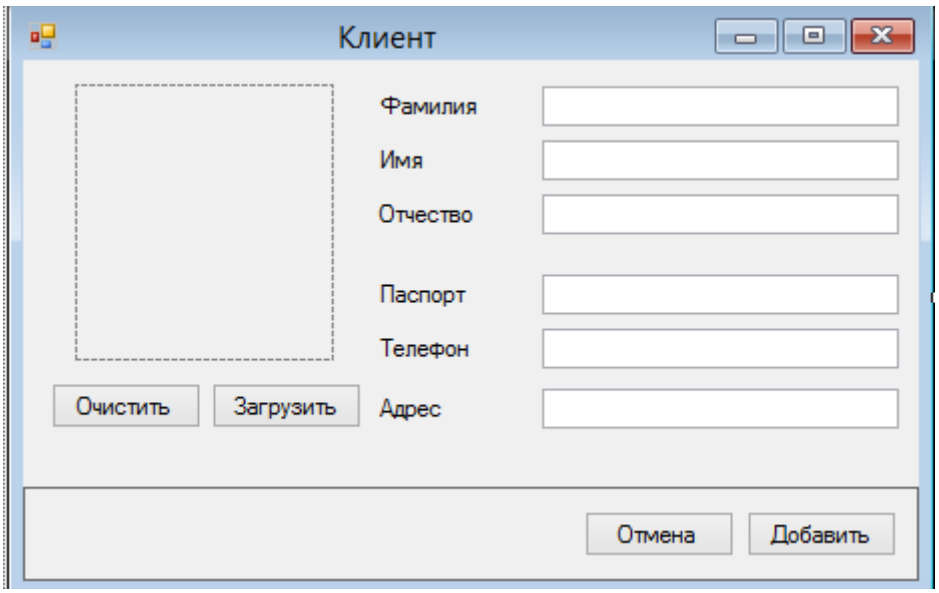


Рис.4.20. Форма «Клиент»

12. Перейдите в редактор кода Form2 и напишите обработчик событий для кнопки Загрузить – Button1, которая будет открывать и загружать фото клиента.

```
private void button1_Click(object sender, EventArgs e)
        {
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                pictureBox1.ImageLocation = openFileDialog1.FileName;
                pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
            }
        }
```

13. Напишите обработчик событий для кнопок Очистить – Button2, Добавить – Button3 и Отмена – Button4.

```
private void button3_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.OK;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.Cancel;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            pictureBox1.Image = null;
             }
```

14. Перейдите в редактор кода Form1 и добавьте подключаемый модуль базы данных MySQL.

```
using MySql.Data.MySqlClient;
```

15. Объявите переменные для работы с базой данных.

```
public partial class Form1 : Form
    {
        MySqlDataAdapter da;
        DataSet ds;
        DataTable dt;
        MySqlConnection cn;
        BindingSource bs;
```

16. Добавьте функцию вычисления максимального числа поля id таблицы «Client».

```
public int max()
     {   int ret = 0;
         string query = "SELECT * FROM Client";
         MySqlCommand com = new MySqlCommand(query, cn);
         MySqlDataReader rd = com.ExecuteReader();
         while (rd.Read())
         {
             ret = rd.GetInt32(0);
         }
         return ret;
     }
```

17. Добавьте обработчик событий **Load** (процедура подключения базы данных) для Form1**.**

Выберете таблицу “Client” базы данных “TravelAgency”, переименуйте поля для удобства использования, а также отключите ненужные поля.

```
private void Form1_Load(object sender, EventArgs e)
        {
            string host = "localhost"; // Имя хоста
            string database = "TravelAgency"; // Имя базы данных
            string user = "root"; // Имя пользователя
            string password = ""; // Пароль пользователя
            string Connect = "Database=" + database + ";Datasource=" + host + ";User="
+ user + ";Password=" + password;
            cn = new MySqlConnection(Connect);
            cn.Open();
            da = new MySqlDataAdapter("SELECT * FROM Client", cn.ConnectionString);
            ds = new DataSet();
            dt = new DataTable("Client");
            bs = new BindingSource();
            da.Fill(dt);
            ds.Tables.Add(dt);

            bs.DataSource = dt;
            dataGridView1.DataSource = bs;
            dataGridView1.Columns["id"].Visible = false;
            dataGridView1.Columns["Surname"].HeaderText = "Фамилия";
            dataGridView1.Columns["Name"].HeaderText = "Имя";
            dataGridView1.Columns["Patronymic"].HeaderText = "Отчество";
            dataGridView1.Columns["Passport"].HeaderText = "Паспорт";
            dataGridView1.Columns["Phone"].HeaderText = "Телефон";
            dataGridView1.Columns["Address"].HeaderText = "Адрес";
            dataGridView1.Columns["Photo"].Visible = false;

            bindingNavigator1.BindingSource = bs;
            cn.Close();

             }
```

18. Добавьте обработчик событий **Click** для кнопок Button1 – Отмена и Button2 – ОК, закрытие приложения без сохранения и сохранение всех изменений в реальной базе данных соответственно.

```
      private void button2_Click(object sender, EventArgs e)
            {
                MySqlCommandBuilder commandBuilder = new MySqlCommandBuilder(da);
                da.Update(ds.Tables[0]);
                Close();
            }

            private void button1_Click(object sender, EventArgs e)
            {
                Close();
                 }
```

19. Добавьте обработчик событий **Click** для bindingNavigatorDeleteItem**.** Сделайте вывод предупреждающего сообщения о удалении записи из базы данных.

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

```
                {
                    if (MessageBox.Show("Вы уверены, что хотите удалить эту запись?",
"", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                    {
                        bs.RemoveCurrent();
                    }
                }
```

20. Добавьте обработчик событий **Click** для bindingNavigatorAddNewItem.

Подключитесь к форме “Клиент” для заполнения полей базы данных.

```
private void bindingNavigatorAddNewItem_Click(object sender, EventArgs e)
        {
            Задача5.Form2 newForm = new Задача5.Form2();
            if (newForm.ShowDialog(this) == DialogResult.OK)
            {
                bs.AddNew();
                DataRowView drw = bs.Current as DataRowView;
                drw["Surname"] = newForm.textBox1.Text;
                drw["Name"] = newForm.textBox2.Text;
                drw["Patronymic"] = newForm.textBox3.Text;
                drw["Passport"] = newForm.textBox4.Text;
                drw["Phone"] = newForm.textBox5.Text;
                drw["Address"] = newForm.textBox6.Text;
                drw["Photo"] = newForm.pictureBox1.ImageLocation;
                bs.EndEdit();
            }

             }
```

21. Сохраните проект и запустите программу для проверки работоспособности, протестируйте добавление и удаление полей. Результат показан на рис.4.21.

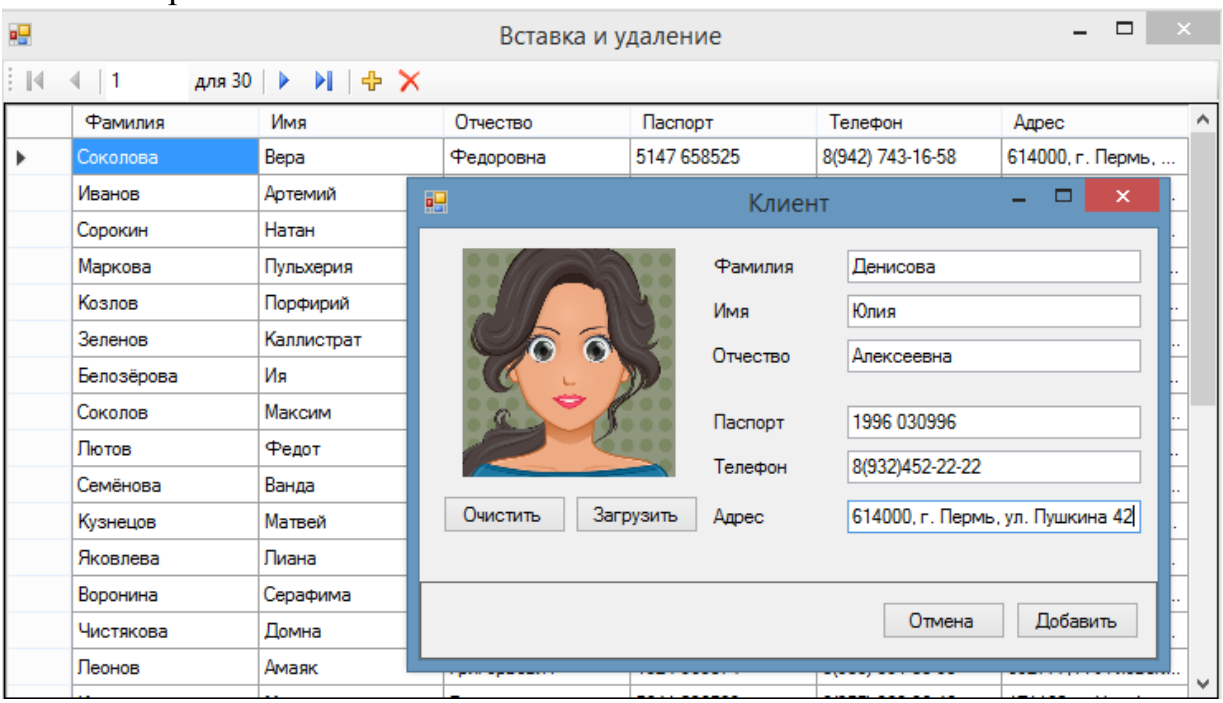


Рис.4.21. Вставка и удаление

4.7. Работа с несколькими связными таблицами

Постановка задачи: Разработать приложение, работающее с несколькими связными таблицами, на примере базы данных “TravelAgency”.

1. Создайте новый проект WindowsForms. Назовите проект RelatedTables.

2. Подключите ссылку MySql.Data.

3. Активизируйте форму и измените ее имя на Связные таблицы.

4. Расположите на форме элемент BindingNavigator (рис.4.22). Удалить ненужные компоненты “Добавить” и “Удалить”.

5. Добавить на форму элементы и установить их свойства, как указано в таб.4.8 и таб.4.9 (рис.4.22).

Таблица 4.8.

Элемент / Свойство	**splitContainer1**	**splitContainer2**
Dock	Fill	Fill
Orientation	Vertical	Horizontal

Таблица 4.9.

Элемент / Свойство	**Button1**	**Panel1**	**DataGriedView1**	**DataGriedView2**	**DataGriedView2**
Text	Выход				
Dock		Bottom	Fill	Fill	Fill

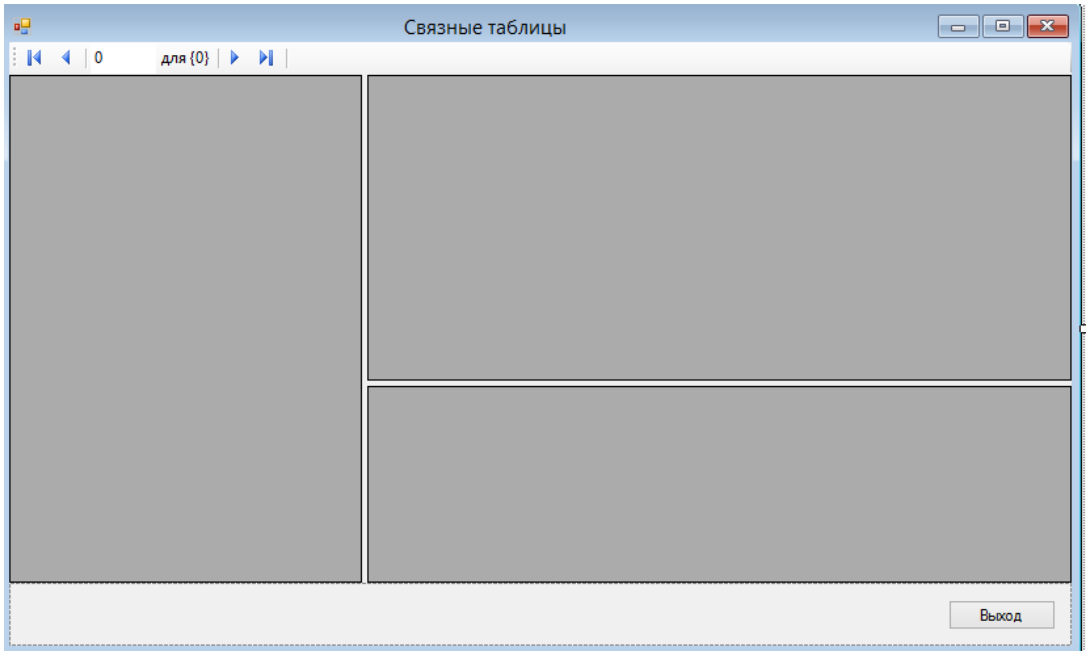


Рис.4.22. Форма «Связные таблицы»

6. Перейдите в редактор кода и добавьте подключаемый модуль базы данных MySQL.

```
using MySql.Data.MySqlClient;
```

7. Объявите переменные для работы с базой данных.

```
public partial class Form1 : Form
    {
        MySqlDataAdapter da1, da2, da3, da4;
        DataSet ds;
        MySqlConnection cn;
        BindingSource bs1, bs2, bs3;
```

8. Добавьте обработчик событий **Click** для кнопок Button1 – Выход, закрытие приложения.

```
private void button1_Click(object sender, EventArgs e)
       {
           Close();
       }
```

9. Добавьте обработчик событий **Load** (процедура подключения базы данных) для **Form1**.

Выбираем таблицы “Client”, “Trip”, “Tour”, базы данных “TravelAgency”, переименовываем поля для удобства использования, а также отключаем ненужные поля.

```
private void Form1_Load(object sender, EventArgs e)
        {
            string host = "localhost"; // Имя хоста
            string database = "TravelAgency"; // Имя базы данных
            string user = "root"; // Имя пользователя
            string password = ""; // Пароль пользователя
            string Connect = "Database=" + database + ";Datasource=" + host + ";User=" +
user + ";Password=" + password;
            cn = new MySqlConnection(Connect);

            ds = new DataSet();
            bs1 = new BindingSource();
            bs2 = new BindingSource();
            bs3 = new BindingSource();

            da1 = new MySqlDataAdapter("SELECT * FROM Trip", cn.ConnectionString);
            da2 = new MySqlDataAdapter("SELECT * FROM Client", cn.ConnectionString);
            da3 = new MySqlDataAdapter("SELECT * FROM Tour", cn.ConnectionString);
            da4 = new MySqlDataAdapter("select a.*, b.* from TourClient a, Client b where
a.Client=b.id", cn.ConnectionString);

            da1.Fill(ds, "Trip");
            da2.Fill(ds, "Client");
            da3.Fill(ds, "Tour");
            da4.Fill(ds, "TourClient");

            DataColumn dcTrip_TourID = ds.Tables["Trip"].Columns["Tour"];
            DataColumn dcTrip_ID = ds.Tables["Trip"].Columns["id"];
            DataColumn dcClientID = ds.Tables["Client"].Columns["id"];
            DataColumn dcTourID = ds.Tables["Tour"].Columns["id"];
            DataColumn dcTourClientTrip = ds.Tables["TourClient"].Columns["Trip"];
```

```
            DataColumn dcTourClientClient = ds.Tables["TourClient"].Columns["Client"];

            DataRelation dataRelation1 = new DataRelation("TripToTourClient", dcTrip_ID,
dcTourClientTrip, false);
            ds.Relations.Add(dataRelation1);
            DataRelation    dataRelation2    =    new    DataRelation("TourClientToClient",
dcTourClientClient, dcClientID, false);
            ds.Relations.Add(dataRelation2);
            DataRelation dataRelation3 = new DataRelation("TripToTour", dcTrip_TourID,
dcTourID, false);
            ds.Relations.Add(dataRelation3);

            bs1.DataSource = ds;
            bs1.DataMember = "Trip";

            bs2.DataSource = bs1;
            bs2.DataMember = "TripToTourClient";

            bs3.DataSource = bs1;
            bs3.DataMember = "TripToTour";

            dataGridView1.DataSource = bs1;
            dataGridView2.DataSource = bs2;
            dataGridView3.DataSource = bs3;

            dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
            dataGridView1.MultiSelect = false;
            dataGridView1.Rows[0].Selected = true;

            dataGridView1.Columns["id"].Visible = false;
            dataGridView1.Columns["Departure"].HeaderText = "Начало";
            dataGridView1.Columns["Arrival"].HeaderText = "Конец";
            dataGridView1.Columns["Transport"].HeaderText = "Транспорт";
            dataGridView1.Columns["Tour"].Visible = false;
            bindingNavigator1.BindingSource = bs1;

            dataGridView2.Columns["id"].Visible = false;
            dataGridView2.Columns["id1"].Visible = false;
            dataGridView2.Columns["Trip"].Visible = false;
            dataGridView2.Columns["Client"].Visible = false;
            dataGridView2.Columns["Surname"].HeaderText = "Фамилия";
            dataGridView2.Columns["Name"].HeaderText = "Имя";
            dataGridView2.Columns["Patronymic"].HeaderText = "Отчество";
            dataGridView2.Columns["Passport"].Visible = false;
            dataGridView2.Columns["Phone"].HeaderText = "Телефон";
            dataGridView2.Columns["Address"].Visible = false;
            dataGridView2.Columns["Photo"].Visible = false;

            dataGridView3.Columns["id"].Visible = false;
            dataGridView3.Columns["Country"].HeaderText = "Страна";
            dataGridView3.Columns["City"].HeaderText = "Город";
            dataGridView3.Columns["Season"].HeaderText = "Сезон";
            dataGridView3.Columns["Description"].HeaderText = "Описание";
            dataGridView3.Columns["Price_of_one_day"].HeaderText = "Цена (день)";
            dataGridView3.Columns["Hotel"].HeaderText = "Отель";
            dataGridView3.Columns["Visa"].HeaderText = "Виза";
            dataGridView3.Columns["Photo"].Visible = false;
             }
```

10. Сохраните проект и запустите программу для проверки работоспособности, протестируйте добавление и удаление полей. Результат показан на рис. 4.23.

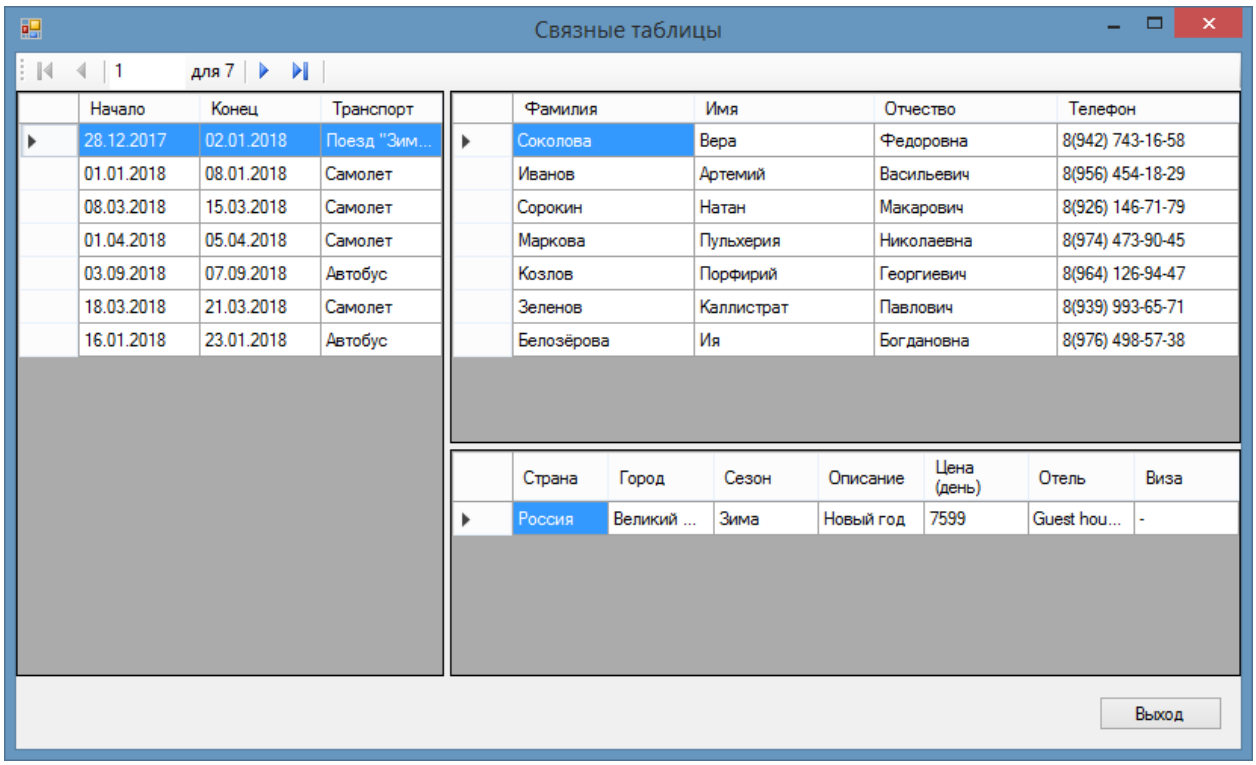


Рис.4.23. Связные таблицы

4.8. Задачи для самостоятельной работы

1. Напишите программу, которая выводит содержимое таблицы, используя вычисляемые поля.

2. Напишите программу, позволяющую выполнить SQL – запрос, введенный пользователем.

3. Напишите программу, при помощи которой можно создать базу данных «Архитектурные памятники Перми». При помощи SQL запроса.

4. Напишите программу для работы с локальной базой данных «Архитектурные памятники Перми».

5. Напишите программу работы с базой данных «Записная книжка». Для создания базы данных используйте визуальные средства Visual Studio.

6. Напишите программу, работающую с несколькими несвязными таблицами.

7. Напишите программу, которая позволяет просматривать и изменять содержимое базы данных, организация компонентов производится на стадии разработки приложения.

8. Напишите программу работы с базой данных «Ежедневник», каждая запись которой содержит информацию о запланированном мероприятии. в начале работы программа должна вывести список дел, запланированных на дату запуска программы и ближайшие дни. Если таблицы данных нет, то программа должна ее создать.

9. Напишите программу, работающую с несколькими связными таблицами, описание связей должно осуществляться программным путем.

10. Напишите программу, которая выбирает для просмотра определенные поля таблицы.

11. Напишите программу, которая самостоятельно формирует дополнительные поля для таблицы (путем SQL – запросов).

12. Напишите программу, позволяющую выполнить SQL-запрос, введенный пользователем. БД Корабли таблица Classes. (БД создается самостоятельно на этапе разработки и имеет 2 таблицы).

13. Напишите программу, работающую с несколькими несвязными таблицами. БД Фирма вторсырья (без связей). (БД создается самостоятельно на этапе разработки и имеет 2 таблицы).

14. Напишите программу, работающую с несколькими связными таблицами, описание связей должно осуществляться программным путем. БД

Аэрофлот. (БД создается самостоятельно на этапе разработки и имеет 2 таблицы).

15. Напишите программу, которая выбирает определенные поля таблицы. БД Аэрофлот таблица Trip. (БД создается самостоятельно на этапе разработки и имеет 2 таблицы).

16. Напишите программу, которая сортирует записи в таблице по выбранному полю, притом должна быть возможность сортировки, как по возрастанию, так и по убыванию. БД «Компьютерная фирма», таблица Printer (БД создается самостоятельно на этапе разработки и имеет 2 таблицы).

17. Напишите программу, которая фильтрует данные по определенным диапазонам (изменяемые пользователем), не используя SQL-запросы. БД «Окраска». (БД создается самостоятельно на этапе разработки и имеет 2 таблицы).

18. Напишите программу, реализующую поиск в БД (не использую SQL-запросы). БД «Компьютерная фирма» таблица Laptop. (БД создается самостоятельно на этапе разработки и имеет 2 таблицы).

19. Напишите программу для работы с локальной базой данных «Архитектурные памятники Перми». (БД создается самостоятельно на этапе разработки и имеет 3 таблицы).

20. Напишите программу работы с базой данных «Записная книжка». Предусмотреть связь таблиц с БД «Ежедневник» (БД создаются самостоятельно на этапе разработки и имеет 4 таблицы).

5. Мультимедиа

Большинство современных программ, работающих в среде Windows, являются мультимедийными. Такие программы обеспечивают просмотр видеороликов и мультипликации, воспроизведение музыки, речи, звуковых эффектов. Типичными примерами мультимедийных программ являются игры и обучающие программы.

Для работы с мультимедиа используется три различные технологии:

- метод ImageAnimator (анимация без звука);
- WMP (Windows Media Player) (подключаемый элемент для просмотра видео);
- Winmm (DLL библиотека для работы с аудиофайлами).

Метод ImageAnimator рассчитан на простые анимационные ролики, которые выводят анимированное изображение на экран. Изображение создается из анимированного GIF-файла.

Подключаемый элемент Windows Media Player. С его помощью проигрываются файлы любых форматов, поддерживаемых данной системой.

Существует единый набор команд для воспроизведения аудиофайлов с помощью любой звуковой карты. В ОС Windows функции, предназначенные для выполнения мультимедийных операций, находятся в библиотеке WinMM.DLL.

5.1. Анимация

Постановка задачи: Разработать приложение, в главном окне которого сразу после появления окна воспроизводится не сопровождаемая звуком анимация, например, рекламный ролик. Добавьте возможность остановки и воспроизведения анимации.

1. Создайте новый проект WindowsForms. Назовите проект Animation.

2. Активизируйте форму и измените ее имя на Анимация. Укажите размер 350; 500. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

3. Добавьте на форму элемент Panel. В свойстве **Dock** установите значение Bottom.

4. Разместите на панели две кнопки Играть и Стоп (рис.5.1).

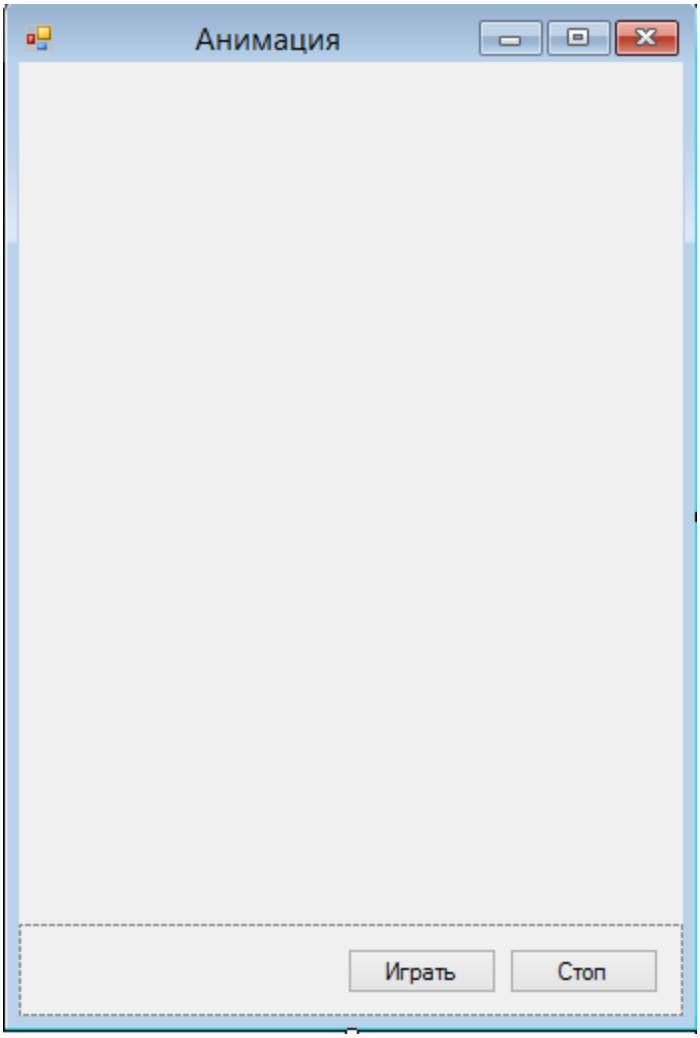


Рис.5.1. Элементы формы «Анимация»

5. В этом проекте нам необходимо использовать ресурсы, где будет хранится наша картинка.

В обозревателе решений щелкните правой кнопкой мыши и выберете **Добавить → Компонент → Общие → Файл ресурсов.**

После создания, открываем файл ресурсов (Resource1.resx) и помещаем необходимую картинку. Это можно сделать посредством Drag and Drop или через меню вставить изображение.

В проекте эта картинка будет доступна через *Resource1.Image1* где Image1 имя наше картинки.

Добавьте любую GIF-анимацию в ресурс.

6. Перейдите в редактор кода и добавьте переменную currentlyAnimating, которая будет отвечать проигрывается ли сейчас анимация.

```
public partial class Form1 : Form
    {
        Image Image = Resource1.Image1;
```

```
bool currentlyAnimating = false;
```

7. Для того, чтобы мы могли проиграть нашу картинку воспользуемся компонентом ImageAnimator, который в свою очередь умеет останавливать проигрывание.

Добавьте два метода AnimateImage и StopAnimation, которые отвечают за проигрывание и остановку анимации.

```
public void AnimateImage()
        {
            if (!currentlyAnimating)
            {

                //  Запускаем анимацию
                ImageAnimator.Animate(Image, new EventHandler(this.OnFrameChanged));
                currentlyAnimating = true;
            }
        }
        public void StopAnimation()
        {
            if (currentlyAnimating)
            {
                //  Останавливаем анимацию
                ImageAnimator.StopAnimate(Image, OnFrameChanged);
                currentlyAnimating = false;
            }
               }
```

8. Опишите метод для изменения размеры формы.

Перерисовываем нашу картинку в зависимости от размера формы.

```
private void OnFrameChanged(object o, EventArgs e)
        {

            this.Invalidate();
               }
```

9. Добавьте метод для отрисовки формы.

```
protected override void OnPaint(PaintEventArgs e)
        {
            if (currentlyAnimating)
            {
                // Переходим на следующий кадр в анимации
                ImageAnimator.UpdateFrames(Image);
                // Покажем этот кадр на экране (будет растягиваться на весь экран)
                e.Graphics.DrawImage(Image, ClientRectangle);
            }
               }
```

10. Добавьте обработчики событий для кнопок Играть и Стоп.

```
public Form1()
        {
            InitializeComponent();
        }

       private void button1_Click(object sender, EventArgs e)
        {
            AnimateImage();
```

```
}

private void button2_Click(object sender, EventArgs e)
{
    StopAnimation();
}
   }
```

11. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис.5.2.

Рис.5.2. Анимация

5.2. Видеофайлы

Постановка задачи: Разработать приложение, используя которое можно просмотреть видеоклип. Клип должен воспроизводится в диалоговом окне программы. Для выбора клипа (AVI-файла) используйте стандартное окно Открытие файла. Программа должна содержать PlayList, а также возможность очистить его и удалить файл.

1. Создайте новый проект WindowsForms. Назовите проект Video.

2. Для работы с видеофайлами необходимо подключить ссылку Windows Media Player.

В обозревателе решений нажмите правой кнопкой мыши в разделе **Ссылки → Добавить ссылку → COM → Windows Media Player.**

3. Необходимо добавить плеер, в котором будет воспроизводится видеофайлы на панель элементов. Для этого на панели инструментов жмем на правую кнопку мыши, выбираем пункт **Выбрать элементы**. На вкладке Компоненты COM выбираем **Windows Media Player.** Элемент появится в стандартных элементах управления.

4. Активизируйте форму и измените ее имя на Просмотр видео. Укажите размер 800; 600. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

5. Добавьте элемент Panel. В свойстве **Dock** укажите Right.

6. На панель добавьте элемент FlowLayoutPanel (Обрабатывает макет компонентов и автоматически компонует их в макет потока). В свойстве **Dock** укажите Bottom.

7. На элемент FlowLayoutPanel добавьте три кнопки Очистить, Добавить, Удалить.

8. Добавьте элемент ListView на панель и в свойстве **Dock** укажите Fill.

Элементы на панели предназначены для работы с playlist (рис.5.3).

9. На форму перенесите ранее добавленный элемент Windows Media Player из стандартных элементов управления и в свойстве **Dock** укажите Fill (рис.5.3).

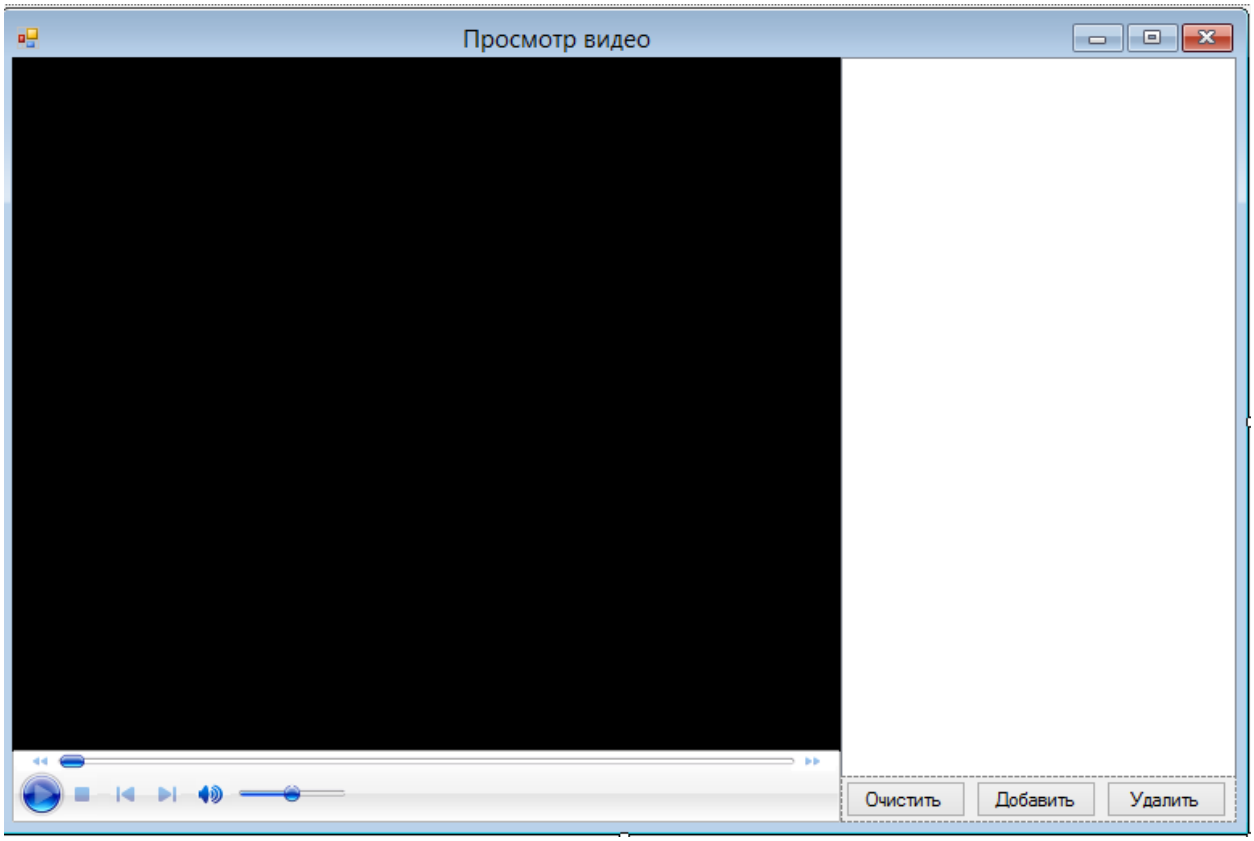


Рис.5.3. Элементы формы «Просмотр видео»

10. Перейдите в редактор кода и создайте поля классов.

Создаем поля классов WMP отвечающий за сам проигрыватель и playlist. Определим позицию текущего элемента в Playlist, а также создадим метод, который определяет есть ли элемент Item в нем.

```
public partial class Form1 : Form
    {
        //  Воспользуемся встроенными возможностями WMP
        WMPLib.IWMPPlaylist songs;
        WMPLib.IWMPMedia media;
        //  Текущая позиция
        int curr = 0;
        bool isContainsPL(string item)
        {
            bool ret = false;
            for (int i = 0; i < songs.count; i++)
            {
                ret = ret || (songs.Item[i].sourceURL == item);
            }
            return ret;
               }
```

11. Добавьте обработчик событий при запуске формы.

Создадим playlist с именем myplaylist.

```
private void Form1_Load(object sender, EventArgs e)
        {
            songs = axWindowsMediaPlayer1.playlistCollection.newPlaylist("MyPlayList");
               }
```

12. Добавьте обработчик событий для кнопки Добавить.

Сначала создаем элемент OpenFileDialog – стандартный диалог открытия файла. Задаем значение, такое как начальный каталог, например, диск D. Указываем фильтр открываемых файлов и если пользователь выбрал файл, то после проверки добавляем его в playlist, при этом запоминаем позицию старого файла, для того чтобы после добавления начать воспроизведение с той же позиции. Обновляем список файлов в ListView, для этого очищаем его и добавляем все элементы из нашего playlist заново.

```
private void button2_Click(object sender, EventArgs e)
        {
            OpenFileDialog dialog = new OpenFileDialog();
            dialog.FileName = String.Empty;
            dialog.InitialDirectory = "D:\\";
            dialog.Filter = "Video Files(*.avi; *.mp4)|*.avi;*.mp4";
            dialog.RestoreDirectory = true;
            if (dialog.ShowDialog() == DialogResult.OK)
            {
                if (!isContainsPL(dialog.FileName))
                {
                    //  Запомним текущую позицию
                    double                        CurrentPosition                        =
axWindowsMediaPlayer1.Ctlcontrols.currentPosition;
                    media = axWindowsMediaPlayer1.newMedia(dialog.FileName);
                    songs.appendItem(media);
                    axWindowsMediaPlayer1.currentPlaylist = songs;
                    //  Продолжаем воспроизведение с запомненной позиции
                    axWindowsMediaPlayer1.Ctlcontrols.currentPosition = CurrentPosition;

                    listView1.Items.Clear();
                    for (int i = 0; i < songs.count; i++)
                    {
                        ListViewItem lvItem1 = new ListViewItem(new string[] {
                            songs.Item[i].name,
                            songs.Item[i].sourceURL,
                            "0"
                        });
                        listView1.Items.Add(lvItem1);
                    }
                    listView1.Items[curr].Selected = true;
                    listView1.Select();
                }
            }

      }
```

13. Добавьте обработчик событий для кнопки Очистить.

Останавливаем воспроизведение, очищаем ListView и очищаем playlist.

```
private void button1_Click(object sender, EventArgs e)
        {
            //  Кнопка очистить
            axWindowsMediaPlayer1.Ctlcontrols.stop();
            listView1.Items.Clear();
            songs.clear();
               }
```

14. Добавьте обработчик событий для кнопки Удалить.

```
private void button3_Click(object sender, EventArgs e)
        {
            if (curr == listView1.SelectedItems[0].Index)
            {
                //  Если удаляем тот файл, который сейчас проигрывается
                //  Останавливаем видео
                axWindowsMediaPlayer1.Ctlcontrols.stop();
                //  Удаляем из playlist
                songs.removeItem(songs.Item[curr]);
                //  Удаляем из ListView
                listView1.SelectedItems[0].Remove();
                //  Уменьшаем текущий индекс и если у нас осталось еще что-то в playlist
                //  то выделяем это видео и запускаем на проигрывание
                curr = (curr==0) ? 0 :--curr;
                if (curr < songs.count)
                {
                    listView1.Items[curr].Selected = true;
                    listView1.Select();
                    axWindowsMediaPlayer1.Ctlcontrols.play();
                }

            }
            else
            {
                //  Если удаляем видео которое сейчас не проигрывается
                songs.removeItem(songs.Item[listView1.SelectedItems[0].Index]);
                listView1.SelectedItems[0].Remove();
                listView1.Items[curr].Selected = true;
                listView1.Select();
            }
           }
```

15. Добавьте обработчик MouseDoubleClick событий для элемента listView.

```
private void listView1_MouseDoubleClick(object sender, MouseEventArgs e)
        {
            if (curr != listView1.SelectedItems[0].Index)
            {
                //  Запомним место на котором было воспроизведение, что бы в дальнейшем,
если мы опять на него перейдем
                //  воспроизведение будет начинаться с той же позиции
                listView1.Items[curr].SubItems[2].Text                                    =
axWindowsMediaPlayer1.Ctlcontrols.currentPosition.ToString();
                //  Изменим текущий индекс
                curr = listView1.SelectedItems[0].Index;
                //  Остановим воспроизведение
                axWindowsMediaPlayer1.Ctlcontrols.stop();
                //  Укажем плееру новый файл и запустим на воспроизведение
                axWindowsMediaPlayer1.Ctlcontrols.playItem(songs.Item[curr]);
                axWindowsMediaPlayer1.Ctlcontrols.currentPosition                         =
Convert.ToDouble(listView1.SelectedItems[0].SubItems[2].Text);
            }
            else
            {
                //  Если плеер не воспроизводит видео и стоит в режиме ожидания
                //  то запускаем видео
                if (axWindowsMediaPlayer1.playState == WMPLib.WMPPlayState.wmppsReady)
                    axWindowsMediaPlayer1.Ctlcontrols.playItem(songs.Item[curr]);

            }
               }
```

16. Добавьте еще один обработчик событий для элемента listView – KeyPress (Возникает, когда этот элемент управления находится в фокусе и пользователь нажимает и отпускает клавишу).

Если пользователь нажал клавишу Enter имитируем MouseDoubleClick по listView.

```
private void listView1_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)Keys.Enter) listView1_MouseDoubleClick(this, null);
        }
```

17. Если в playlist не выбран файл, то кнопку Удалить сделаем недоступной. Для этого добавьте обработчик событий **ItemSelectionChanged** (Событие происходит при изменении выбранного состояния элемента) для элемента listView.

```
private           void           listView1_ItemSelectionChanged(object           sender,
ListViewItemSelectionChangedEventArgs e)
        {
            //  Если в PlayList ничего не выбрано, кнопку удаления делаем
            button3.Enabled = e.IsSelected;
               }
```

18. Добавьте обработчик событий **MediaChange** для элемента Windows Media Player.

```
private          void          axWindowsMediaPlayer1_MediaChange(object          sender,
AxWMPLib._WMPOCXEvents_MediaChangeEvent e)
        {
            //  Видео растянем по размерам формы
            axWindowsMediaPlayer1.stretchToFit = true;
               }
```

19. Добавьте еще один обработчик событий для Windows Media Player – **CurrentItemChange**.

```
private        void        axWindowsMediaPlayer1_CurrentItemChange(object        sender,
AxWMPLib._WMPOCXEvents_CurrentItemChangeEvent e)
        {
            //  Здесь будем обрабатывать событие, если была нажата кнопка
            //  на плеере вперед - назад или если заканчивается видео и следует переход на
следующее
            //  На ListBox также передвинем выделенный объект
            int i =0;
            if (listView1.Items.Count > 0)
            {
                for (i = 0; i < songs.count; i++)
                {
                    //  проверяем на идентичность SourceURL у текущего видео (на который
сейчас перешли)
                    //  и соответствующую ему позицию в PlayList
                    if                    (songs.Item[i].sourceURL                    ==
axWindowsMediaPlayer1.currentMedia.sourceURL)
                        curr = i;
                }
                //  Выделим это видео в ListBox
                listView1.Items[curr].Selected  =  !(axWindowsMediaPlayer1.playState  ==
WMPLib.WMPPlayState.wmppsStopped);
```

```
            listView1.Select();
        }
    }
```

20. Сохраните проект и запустите программу для проверки работоспособности. Загрузите в Playlist видеофайлы и проиграйте их. Результат показан на рис.5.4.

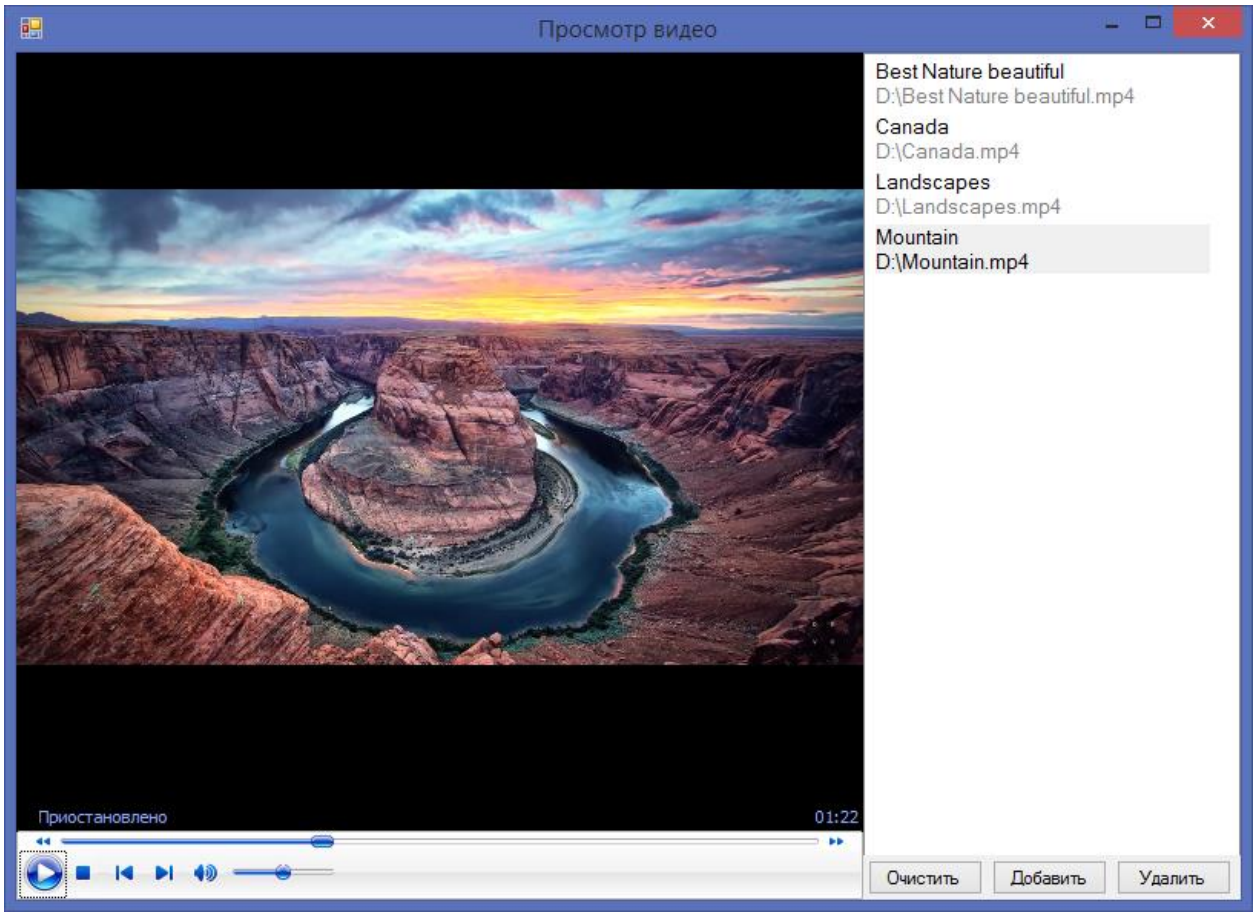


Рис.5.4. Просмотр видео

5.3. Аудиофайлы

Постановка задачи: Написать программу MP3 Player. Программа должна обеспечивать возможность выбора каталога, в котором находится MP3-файлы, а также регулировку громкости звука непосредственно в диалоговом окне программы. Программа должна содержать PlayList.

1. Создайте новый проект WindowsForms. Назовите проект Audio.

2. Активизируйте форму и измените ее имя на MP3 плеер. Укажите размер 350; 500.

3. Добавьте на форму элемент TabelLayotPanel (Обрабатывает макет компонентов и автоматически компонует их в форме таблицы) и в свойстве **Dock** укажите Fill.

Разделим элемент на четыре составляющие (четыре строки), для этого в свойстве элемента ColumnCount (число столбцов в таблице) укажите 1, а в RowCount (Число строк в таблице) 4.

1-я строка будет отвечать за регулировку громкости, дорожку воспроизведения и показ информации о песне. Во 2-й строке разместим кнопки воспроизведение, пауза, переход на следующую и предыдущую песню. В 3-ей строке будет размещен Playlist. В 4-ой строке добавим кнопку для добавления файлов в playlist (рис.5.6).

4. В 1-ю строку перенесите элемент Panel. В свойстве **Dock** укажите Fill.

5. Добавьте на панель два элемента GroupBox (рис.5.6).

Для GroupBox1 задайте свойства согласно таб. 14.

Таблица 14.

Свойство	**Значение**
Text	Громкость
Dock	Right

Разместите в нем элемент TrackBar (Позволяет пользователю выбрать диапазон значений с помощью небольшого ползунка, перемещаемого по полосе) и в свойстве **Orientation** (Ориентация элемента управления) укажите Vertical (Вертикальный) (рис.5.6).

GroupBox2 заполните оставшиеся пространство панели. Разместите в нем Label1 – элемент, который будет отвечать за информацию о песне и TrackBar – полоса прокрутки песни (рис.5.6).

6. На 2-ю строку добавьте элемент FlowLayoutPanel. В свойстве **Dock** укажите Fill.

7. Перенесите элемент ImageList (Управляет коллекцией изображений, которые обычно используются другими элементами управления) на форму.

В данном элементе будут хранится изображения для наших кнопок.

В свойстве Images (Изображения, записанные в ImageList) загрузите картинки в формате .ico для кнопок управления (рис.5.5).

8. Перенесите на панель четыре кнопки и в свойстве ImageList выберете ImageList1. В **ImageKey** (Индекс изображения, которое будет отображаться на данном элементе управления), выберите соответствующее изображение для кнопок согласно рис. 5.5.

Рис. 5.5. Кнопки управления

8. В 3-ю строку перенесите элемент ListBox, в свойстве **Dock** укажите Fill (рис.5.6).

9. На последнюю, 4-ю строку добавьте кнопку Открыть и также в значении свойстве **Dock** укажите Fill (рис.5.6).

10. Перенесите элемент Timer на форму.

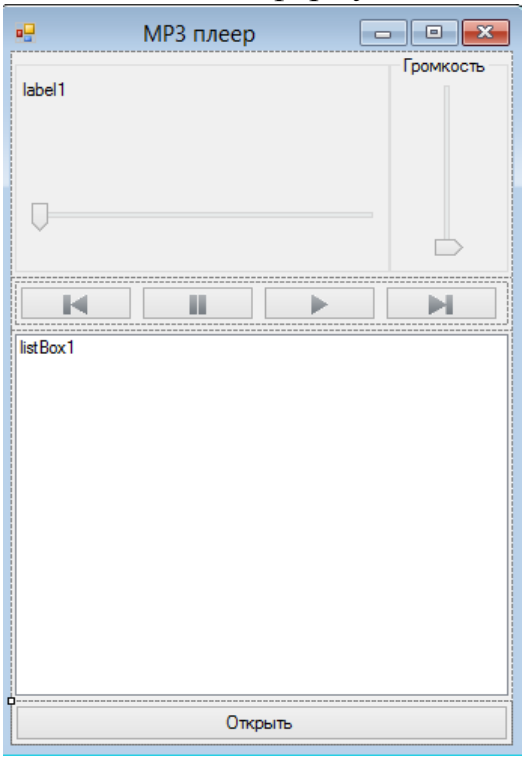


Рис.5.6. Элементы формы «MP3 плеер»

11. Обычно в программу вводят новый класс, когда нужно добавить некоторую функцию, которую можно выделить в отдельный модуль кода. Отдельный модуль и отдельный класс используют для того, чтобы не загромождать кодом основной модуль программы (так проще разбираться в коде и поддерживать его). Рассмотрим добавление в программу класса отдельным файлом.

Щелкните правой кнопкой мыши по проекту и выберете **Добавить → Класс**. Назовите новый класс Mp3Player.

12. Для начала опишем класс Mp3Player. Добавьте код в отдельно созданный класс Mp3Player. Для работы со звуком нам необходимо подключить winmm.dll.

Примечание. DLL (Dynamic Link Library) – динамически подключаемая библиотека функций. Для библиотек DLL предполагается многократное использование различными программами.

```
class Mp3Player: IDisposable
    {
        private int masterVolume;
        public bool isPlayed { get; private set; }
        //  Воспроизводить наши MP3 файлы будем через WinApi
        //  Подключим winmm.dll
        [DllImport("winmm.dll")]
        private static extern long mciSendString(string strCommand, StringBuilder
strReturn, int iReturnLength, IntPtr hwndCallback);
        // Конструктор класса
        public Mp3Player()
        {
            masterVolume = 1000;
        }
        // Открывает файл для будущего его проигрывания
        public void Open(string fileName)
        {
            string Command = @"open ""{0}"" type mpegvideo alias MediaFile";
            Command = string.Format(Command, fileName);
            mciSendString(Command, null, 0, IntPtr.Zero);
            MasterVolume = MasterVolume;
        }
        // Начинает проигрывание файла
        public void Play()
        {
            string Command = "play MediaFile";
            mciSendString(Command, null, 0, IntPtr.Zero);
            MasterVolume = MasterVolume;
            isPlayed = true;
        }
        // Останавливает проигрывание медиа файла
        public void Stop()
        {
            string Command = "stop MediaFile";
            mciSendString(Command, null, 0, IntPtr.Zero);
            isPlayed = false;
        }
        // Закрывает медиа файла
        public void Close()
        {
```

```
    string Command = "close MediaFile";
    mciSendString(Command, null, 0, IntPtr.Zero);
    isPlayed = false;
}
// Приостанавливает проигрывание текущего открытого медиа файла
public void Pause()
{
    string Command = "pause MediaFile";
    mciSendString(Command, null, 0, IntPtr.Zero);
    isPlayed = false;
}
// Возвращает строку текущего статуса
public String Status()
{
    int i = 128;
    System.Text.StringBuilder stringBuilder = new System.Text.StringBuilder(i);
    mciSendString("status MediaFile mode", stringBuilder, i, IntPtr.Zero);
    return stringBuilder.ToString();
}
// Длина трека (в секундах)
public ulong Length
{
    get
    {
        StringBuilder str = new StringBuilder(128);
        ulong ret = 0;
        try
        {
            mciSendString("status MediaFile length", str, 128, IntPtr.Zero);
            ret = Convert.ToUInt64(str.ToString());
        }
        catch { }
        return ret;
    }
}
// Текущая позиция (проигранно с начала трека (сек))
public ulong Position
{
    get
    {
        StringBuilder str = new StringBuilder(128);
        ulong ret = 0;
        try
        {
            mciSendString("status MediaFile position", str, 128, IntPtr.Zero);
            ret = Convert.ToUInt64(str.ToString());
        }
        catch { }
        return ret;
    }
    set
    {
        string Command = "play MediaFile from "+ value;
        mciSendString(Command, null, 0, IntPtr.Zero);
    }
}
// Установка общей громкости воспроизведения
public int MasterVolume
{
    get
    {
        return masterVolume;
    }
    set
```

```
            {
                mciSendString(string.Concat("setaudio MediaFile volume to ", value), null,
0, IntPtr.Zero);
                masterVolume = value;
            }
        }

    }
```

13. Опишите класс MP3Info. Данный класс позволяет получить информацию из тегов аудиофайлов.

```
class MP3Info
    {
        public string Title { get; set; }
        public string Artist { get; set; }
        public string Album { get; set; }
        public string Year { get; set; }
        public string Comment { get; set; }
        public bool HasTag { get; set; }
        public int Track { get; set; }
        public MP3Info(string path)
        {
            HasTag = false;

            // необходимо убедится, что файл существует
            if (File.Exists(path))
            {
                try
                {
                    // открываем файл
                    FileStream fs;
                    fs = new FileStream(path, FileMode.Open);

                    // получаем информацию из тегов
                    byte[] buffer = new byte[128];
                    fs.Seek(-128, SeekOrigin.End);
                    fs.Read(buffer, 0, 128);
                    fs.Close();

                    // преобразование буфера тегов в строку                    UTF8Encoding
                    encoder = new UTF8Encoding();
                    string tag = encoder.GetString(buffer);

                    if (tag.Substring(0, 3) == "TAG")
                    {
                        HasTag = true;

                        Title = RemoveWhiteSpace(tag.Substring(3, 30));
                        Artist = RemoveWhiteSpace(tag.Substring(33, 30));
                        Album = RemoveWhiteSpace(tag.Substring(63, 30));
                        Year = RemoveWhiteSpace(tag.Substring(93, 4));
                        Comment = RemoveWhiteSpace(tag.Substring(97, 28));

                        if (tag[125] == 0)
                            Track = (int)buffer[126];
                        else
                            Track = 0;
                    }
                }
                catch { }
            }
```

```
        }
        private string RemoveWhiteSpace(string s)
        {
            string newstring = "";

            foreach (char c in s)
            {
                if     (char.IsLetterOrDigit(c)     ||     char.IsPunctuation(c)     ||
char.IsSeparator(c))
                {
                    newstring += c;
                }
            }

            return newstring.Trim();
        }
           }
```

14. Создадите еще один класс – MP3PlayList. Класс будет содержать список песен.

```
class MP3PlayList
    {
        public class Song
        {
            // Наименование песни
            public string Name { get; set; }

         // Полный путь до песни
            public string Path { get; set; }

            // Инициализация песни
            public Song(string name, string path)
            {
                Name = name;
                Path = path;
            }
        }
        //  Создадим простейший одномерный индексатор с песнями
        Song[] Songs;
        public Song this[int index]
        {
            set
            {
                if (index >= 0 && index < Length)
                {
                    Songs[index] = value;
                }
            }
            get
            {
                if (index >= 0 && index < Length)
                    return Songs[index];
                else
                    return null;
            }
        }

        // Инициализация плейлиста размером Size
        public MP3PlayList(int Size)
        {
            Songs = new Song[Size];
            Track = 0;
```

```
        }
        // Длина массива
        public int Length
        {
            get { return Songs.Length; }
        }

        // Номер текущего трека
        public int Track { get; set; }

        // Включение/выключение повтора произведения плейлиста
        public bool Repeat { get; set; }

        // Переключение на следующую песню
        public void NextSong()
        {
            if (Track < Length-1) Track++;
            else if (Repeat) Track = 0;
        }

        // Переключение на предыдущую песню
        public void PrevSong()
        {
            if (Track > 0) Track--;
            else if (Repeat) Track = Length-1;
        }
    }
```

15. Перейдите в Form1.cs. Создайте переменные плеера и опишите переменную PlayList.

```
public partial class Form1 : Form
    {

        private Mp3Player player;
        MP3PlayList PlayList;
        public Form1()
        {
            InitializeComponent();
            label1.Text = "";
            player = new Mp3Player();
               }
```

16. Опишем метод PlayFile который воспроизводит песню по заданному индексу. Текущая песня останавливается, вычисляется информация, содержащаяся в теге mp3 файла, устанавливается значение в ползунке равное длине проигрывания песни. А также заполняем label1, который отвечает за вывод информации по файлу и начинается его воспроизведение.

```
private void PlayFile(int index)
        {
            if (index < PlayList.Length)
            {
                player.Stop();
                player.Close();
                MP3Info Info = new MP3Info(PlayList[index].Path);
                label1.Text = "Исполнитель: " + Info.Artist + "\nАльбом: " + Info.Album +
"\nНазвание песни: " + Info.Title;
                player.Open(PlayList[index].Path);

                trackBar1.Maximum = (int)player.Length;
```

```
                TimeSpan ts = TimeSpan.FromMilliseconds(trackBar1.Maximum);
                label1.Text  +=  "\nВремя:  "  +  String.Format("{0:00}:{1:00}:{2:00}",
ts.Hours, ts.Minutes, ts.Seconds);
                listBox1.SelectedIndex = index;
                listBox1.Select();
                player.Play();
            }
               }
```

17. Добавьте обработчик событий кнопки Открыть.

Создается стандартный диалог открытия файлов, устанавливаем фильтры на файлы – .mp3, .wav, а также разрешим выбор нескольких файлов. Если файлы выбраны, мы создаем PlayList и заносим все выбранные файлы в элемент ListView. Включаем повтор воспроизведения листа по кругу.

Т.к. в конструкторе класса mp3 плеера устанавливается максимальная громкость, то ползунок отвечающий за громкость, так же устанавливаем на максимум.

Начинам автоматически проигрывать первый файл из списка.

Кнопку загрузить делаем недоступной.

```
private void btnOpen_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog dialog = new OpenFileDialog())
            {
                dialog.Filter = "Sound Files (*.mp3; *.wav)|*.mp3;*.wav";
                dialog.Multiselect = true;
                if (dialog.ShowDialog() == DialogResult.OK)
                {
                    PlayList = new MP3PlayList(dialog.SafeFileNames.Length);
                    MP3PlayList.Song Track;
                    //  Занесем все выбранные файлы в плейлист
                    for (int i = 0; i < dialog.SafeFileNames.Length; i++)
                    {
                        Track     =     new     MP3PlayList.Song(dialog.SafeFileNames[i],
Path.Combine(Path.GetDirectoryName(dialog.FileName), dialog.SafeFileNames[i]));
                        PlayList[i] = Track;
                        listBox1.Items.Add(PlayList[i].Name);
                    }
                    PlayList.Repeat = true;
                    trackBar2.Maximum = 1000;
                    trackBar2.Value = 1000;
                    PlayFile(0);
                }
                btnOpen.Enabled = false;
            }
               }
```

18. Добавьте обработчик событий для кнопки Пауза.

Когда файл проигрывается ставим его на паузу, если стоит на паузе, начинаем его воспроизведение.

```
private void button2_Click(object sender, EventArgs e)
        {
            if (player.isPlayed)
```

```
            player.Pause();
        else player.Play();
           }
```

19. Добавьте обработчик событий для копки Играть.

Если в PlayList выбранный файл совпадает с тем, который в данный момент проигрывается, и он стоит на паузе, то начинаем его воспроизведение, иначе текущее воспроизведение останавливается, и мы начинаем воспроизведение выделенного файла.

```
private void btnPlay_Click(object sender, EventArgs e)
        {
            if (PlayList.Track == listBox1.SelectedIndex)
            {
                if (!player.isPlayed) player.Play();
            }
            else
            {
                PlayList.Track = listBox1.SelectedIndex;
                PlayFile(PlayList.Track);
            }
           }
```

20. Добавьте обработчик событий Tick для элемента Timer.

Если у нас воспроизводится какой-либо файл, то меняем позицию бекунка в зависимости от текущей позиции.

```
private void timer1_Tick(object sender, EventArgs e)
        {
            if (player != null)
            {
                trackBar1.Value = (int)player.Position;
            }
           }
```

21. Добавьте обработчики событий для элементов TrackBar1 и TrackBar2. Если пользователь передвинул ползунок, меняем позицию воспроизведения и уровень громкости соответственно.

```
private void trackBar1_Scroll(object sender, EventArgs e)
        {
            player.Position = (ulong)trackBar1.Value;
        }

        private void trackBar2_Scroll(object sender, EventArgs e)
        {
            player.MasterVolume = trackBar2.Value;
        }
```

22. Добавьте обработчик событий для кнопки предыдущая песня.

Запомним текущий трек, перейдем на предыдущий и если он не совпадает с тем, что мы запомнили, то начинам его воспроизводить, иначе у в PlayList просто одна песня.

```
private void button1_Click(object sender, EventArgs e)
        {
            int oldTrack = PlayList.Track;
            PlayList.PrevSong();
            if (PlayList.Track != oldTrack) PlayFile(PlayList.Track);
        }
```

23. Добавьте обработчик событий для кнопки следующая песня.

```
private void button3_Click(object sender, EventArgs e)
        {
            int oldTrack = PlayList.Track;
            PlayList.NextSong();
            if (PlayList.Track != oldTrack) PlayFile(PlayList.Track);
               }
```

24. Добавьте событие MouseDoubleClick для элемента listBox.

```
private void listBox1_MouseDoubleClick(object sender, MouseEventArgs e)
        {
            if (PlayList.Track == listBox1.SelectedIndex)
            {
                if (!player.isPlayed) player.Play();
            }
            else
            {
                PlayList.Track = listBox1.SelectedIndex;
                PlayFile(PlayList.Track);
            }
        }
```

25. Сохраните проект и запустите программу для проверки работоспособности. Загрузите в playlist аудиофайлы и проиграйте их. Результат показан на рис.5.7.

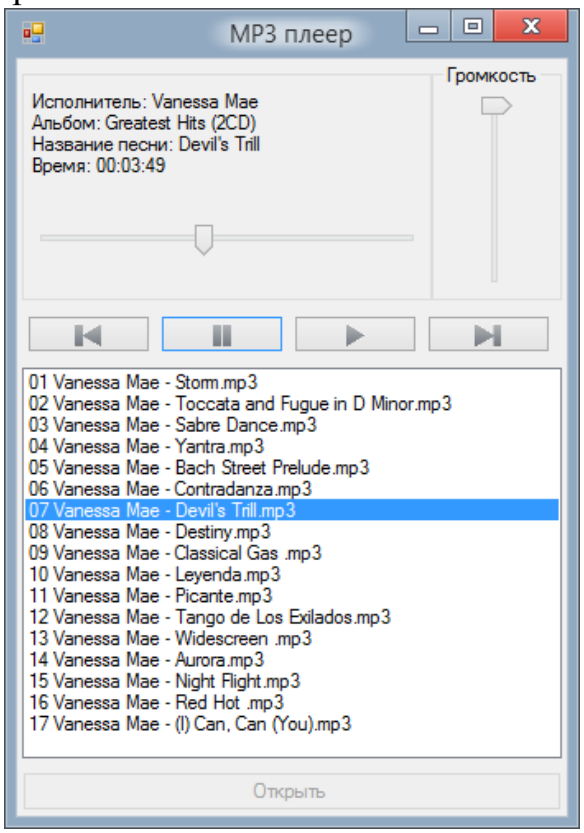


Рис.5.7. MP3 плеер

5.4. Задачи для самостоятельной работы

1. Напишите программу, в окне которой сразу после запуска циклически воспроизводится несопровождаемая звуком анимация.

2. Напишите программу, в окне которой воспроизводилась не сопровождаемая звуком анимация. Пользователь может выбрать каталог, в котором находится файл с анимацией (если в каталоге несколько файлов, то программа выбирает первый).

3. Разработайте приложение, проигрывающее стандартные анимации (поиск папки, поиск файла, поиск компьютера, копирование файлов, удаление файла, удаление минуя корзину, очистка корзины).

4. Напишите программу, которая при наведении мыши на форму начинает проигрывание анимации.

5. Напишите программу, в окне которой отображается галерея анимаций, добавьте кнопки для переключения.

6. Напишите программу, которая перед загрузкой основной формы выводила заставку.

7. Напишите программу «Цифровые часы», где цифры имели бы не статический вид. Для вывода цифр использовать анимированные изображения.

8. Напишите программу, в окне которой сразу после запуска циклически воспроизводится AVI файл.

9. Напишите программу, которая при указании каталога циклически воспроизводит все анимации имеющиеся в нем.

10. Напишите программу, которая воспроизводит AVI файл в новой форме и меняет свой размер и прозрачность до максимального, при начале воспроизведения, до минимального при окончании воспроизведения.

11. Напишите программу, которая воспроизводит AVI файлы, используя компонент WindowsMediaPlayer. Воспроизведение должно осуществляться из PlayList. Не использовать стандартный PlayList wmp.

12. Напишите программу, которая воспроизводит интернет радио, используя компонент WindowsMediaPlayer. Например: http://194.226.152.38:8000/evropa_plus

13. Напишите программу, которая воспроизводит MP3 файлы, используя компонент WindowsMediaPlayer.

14. Доработайте предыдущую программу. Добавьте возможность пользования стандартного PlayList для компонента WindowsMediaPlayer.

15. Напишите программу, которая показывает видео файл и субтитры к нему на одной форме. Воспроизведение осуществлять при помощи компонента WindowsMediaPlayer. Файл с субтитрами содержит время и текст показа подсказки.

16. Напишите программу, которая автоматически определяет, какую анимацию открывает пользователь. Программа самостоятельно выбирает нужный компонент для воспроизведения анимации.

17. Напишите программу, которая отслеживает изменилась ли папка откуда воспроизводятся файлы и корректировала бы PlayList. Использовать в качестве базовой задачу № 14.

18. Напишите программу, главная форма которой разделена на 4 зоны. В каждой зоне свой видео плейер, который играет свой видеофайл по кругу. Пользователь может выбирать, какой плейер (только один) сейчас воспроизводит видео со звуком. Все остальные, воспроизводят видео без звука.

19. Напишите программу, главная форма которой разделена на 4 зоны, в каждой свой плейер и воспроизводит свой файл беззвучно. При наведении мыши на любое видео, появляется большое модальное окно, где данное видео воспроизводится уже со звуком.

20. Доработайте разобранную задачу для воспроизведения MP3-файлов, чтобы по окончании воспроизведения песни, начинала бы воспроизводится следующая песня.

21. Разработайте приложение для прослушивания MP3-файлов. Программа должна содержать полноценный PlayList. Использовать в качестве базового приложения, разобранную задачу на воспроизведения MP3-файлов.

22. Разработайте приложение для прослушивания MP3-файлов. При проигрывании аудио файла, сделать визуальное сопровождение в виде столбиков. Использовать вертикально ориентированные компоненты ProgressBar.

23. Напишите программу, для прослушивания звуковых файлов, находящихся в выбранном пользователем каталоге (приложение обрабатывает только файлы wav).

24. Напишите программу, которая воспроизводит MP3-файлы с нарастающей громкостью в начале проигрывания файла и с убывающей в конце. Длина нарастания и затухания задается пользователем.

25. Напишите программу, в окне которой воспроизводилась сопровождаемая звуком анимация. Пользователь может выбрать каталог, в котором находится файл с анимацией и звуковое сопровождение (если в каталоге несколько файлов, то программа выбирает первый).

6. COM – объекты

Объектная модель программных компонентов (Component Object Model, COM) проектировалась с целью дать возможность создавать компоненты на любом языке/платформе, обладающем поддержкой этой модели, и использовать их в любом языке/платформе (другом), так же обладающем поддержкой COM. Платформа .NET не исключение и позволяет легко использовать сторонние COM-объекты и экспортировать типы .NET в виде COM-объектов.

Вы можете использовать COM-компоненты в проекте C#. Общая процедура следующая.

1. Добавьте в проект ссылку на COM-компонент или библиотеку типов.

2. Создайте экземпляр класса, определенного в вызываемой оболочке времени выполнения. Он, в свою очередь, создает экземпляр COM-объекта.

3. Используйте объект так же, как другие управляемые объекты. Когда объект удаляется сборщиком мусора, экземпляр COM-объекта также удаляется из памяти.

6.1. Работа с файлами PDF

Постановка задачи: Разработать приложение, которое отображает Adobe PDF Reader.

1. Создайте новый проект WindowsForms. Назовите проект pdfReader.

2. Для создания приложения необходимо подключить ссылку Adobe Acrobat Browser.

В обозревателе решений нажмите правой кнопкой мыши в разделе **Ссылки → Добавить ссылку → COM → Adobe Acrobat Browser.**

3. Необходимо добавить элемент, в котором будут отображаться документы в формате .pdf. Для этого на панели инструментов жмем на правую кнопку мыши, выбираем пункт **Выбрать элементы**. На вкладке Компоненты COM выбираем **Adobe PDF Reader.** Элемент появится в стандартных элементах управления.

4. Активизируйте форму и измените ее имя на PDF Reader. Укажите размер 600; 550. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

5. Перетащите на форму элемент MenuStrip (Отображает команды приложения и параметры, сгруппированные по функциональности) (рис.6.1).

В меню добавьте раздел – Файл, а в него два подраздела Открыть и Выход.

6. Из панели элементов перетащите раннее добавленный элемент Adobe PDF Reader на форму. Заполните элементом оставшиеся пространство формы (рис.6.1).

7. Добавьте на форму элемент OpenFileDialog.

Рис.6.1. Элементы формы «PDF Reader»

8. Добавьте обработчик событий для подраздела Файл – Выход.

Удаляем документ и закрываем приложение.

```
private void выходToolStripMenuItem_Click(object sender, EventArgs e)
        {
            axAcroPDF1.Dispose();
            Close();
               }
```

9. Добавьте обработчик событий для подраздела Файл – Открыть.

Пользователь выбирает файл при помощи стандартного диалогового окна – открытие файла. Происходит загрузка файла для показа в элемент Adobe PDF Reader. При этом отображаем панель инструментов.

```
private void открытьToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                axAcroPDF1.setShowToolbar(true);
                axAcroPDF1.LoadFile(openFileDialog1.FileName);
            }
        }
```

10. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис.6.2.

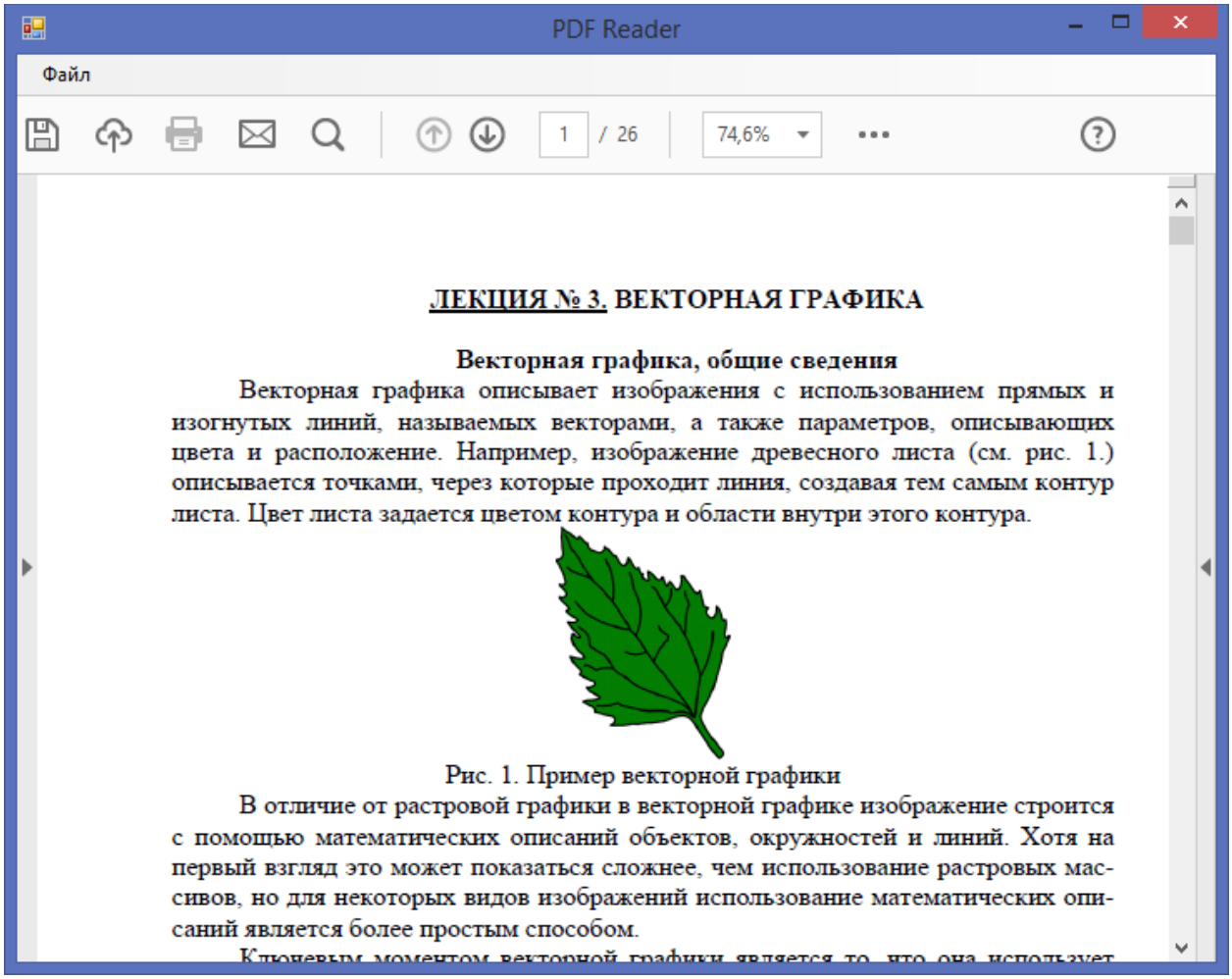


Рис.6.2. PDF Reader

6.2. Создание таблицы MS Excel.

Постановка задачи: Разработать приложение, которое запускает MS Excel, создает новую книгу и формирует таблицу умножения, с использованием значений или формул.

1. Создайте новый проект WindowsForms. Назовите проект Excel.

2. Для создания и работы с таблицами необходимо подключить ссылку Microsoft Excel.

В обозревателе решений нажмите правой кнопкой мыши в разделе **Ссылки → Добавить ссылку → COM → Microsoft Excel.**

3. Активизируйте форму и измените ее имя на Таблица умножения. Укажите размер 300; 190. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

4. Добавьте два элемента RadioButton. В свойстве Text измените значения на Использовать значения и Использовать формулы (рис.6.3).

По умолчанию сделайте кнопку Использовать значения.

5. Добавьте кнопку Создать таблицу (рис.6.3).

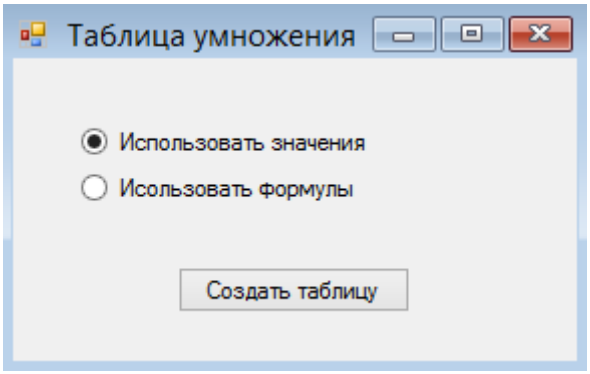


Рис.6.3. Элементы формы «Таблица умножения»

6. Перейдите в редактор кода и добавьте подключаемый модуль Excel.

```
using myExcel = Microsoft.Office.Interop.Excel;
```

7. Создайте переменные.

```
public partial class Form1 : Form
    {
        public myExcel.Application app;
               public myExcel.Workbook Workbook;
```

8. Опишите метод создания таблицы умножения.

```
private void MultiTable()
        {
            myExcel.Worksheet Sheet;
            // Новая книга
            Workbook = app.Workbooks.Add(Type.Missing);
```

```
            // Новый лист
            Sheet = (myExcel.Worksheet)Workbook.Worksheets.get_Item(1);
            //  Зададим имя
            Sheet.Name = "Таблица умножения";
            string[]                    letter                    =                    {
"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T" };
            for (int i = 1; i <= 10; i++)
            {

                for (int j = 1; j <= 10; j++)
                {
                    //  Закрасим ячейки по диагонали таблицы
                    if (i == j)
                        Sheet.Cells[i, j].Interior.Color = Color.Coral;
                    //  Боковую колонку и верхнюю строку покрасим в свой цвет
                    if (i == 1 || j == 1)
                    {
                        Sheet.Cells[i, j].Interior.Color = Color.BurlyWood;
                        Sheet.Cells[i, j].Font.Bold = true;
                        Sheet.Cells[i, j].Value = (i == 1) ? j : i;
                    }
                    else
                    //  В зависимости от выбранного типа данных
                    //  в ячейки прописываем соответствующие значения
                    if (rbFormula.Checked)
                        Sheet.Cells[i, j].Formula = "=A" + i.ToString() + "*" + letter[j-
1] + "1";
                    else
                        Sheet.Cells[i, j].Value = i * j;
                }
            }
            myExcel.Shapes Sel = Sheet.Shapes;

Sel.AddLabel(Microsoft.Office.Core.MsoTextOrientation.msoTextOrientationHorizontal,   567,
35, 72, 72);
            Sel.AddTextEffect(
                Microsoft.Office.Core.MsoPresetTextEffect.msoTextEffect12,
                "Таблица умножения",
                "+mn-lt",
                48,
                Microsoft.Office.Core.MsoTriState.msoFalse,
                Microsoft.Office.Core.MsoTriState.msoFalse,
                11,
                189);

            // отображение окна Excel
            app.Visible = true;
        }
```

9. Создаем Excel-приложение.

```
public Form1()
        {
            InitializeComponent();
            //  Создание нового Excel
            app = new myExcel.Application();
           }
```

10. Добавьте обработчик событий **FormClosed** (Возникает при каждом завершении работы с формой после того, как форма была закрыта, и определяет причины этого закрытия) для формы.

```
        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
```

```
{
    try
    {
        Workbook.Close();
        Workbook = null;
        app.Quit();
    }
    catch { app.Quit(); }
       }
```

11. Добавьте обработчик событий для кнопки Создать таблицу. Вызываем метод MultiTable.

```
private void button1_Click(object sender, EventArgs e)
        {
            MultiTable();
               }
```

12. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис.6.4.

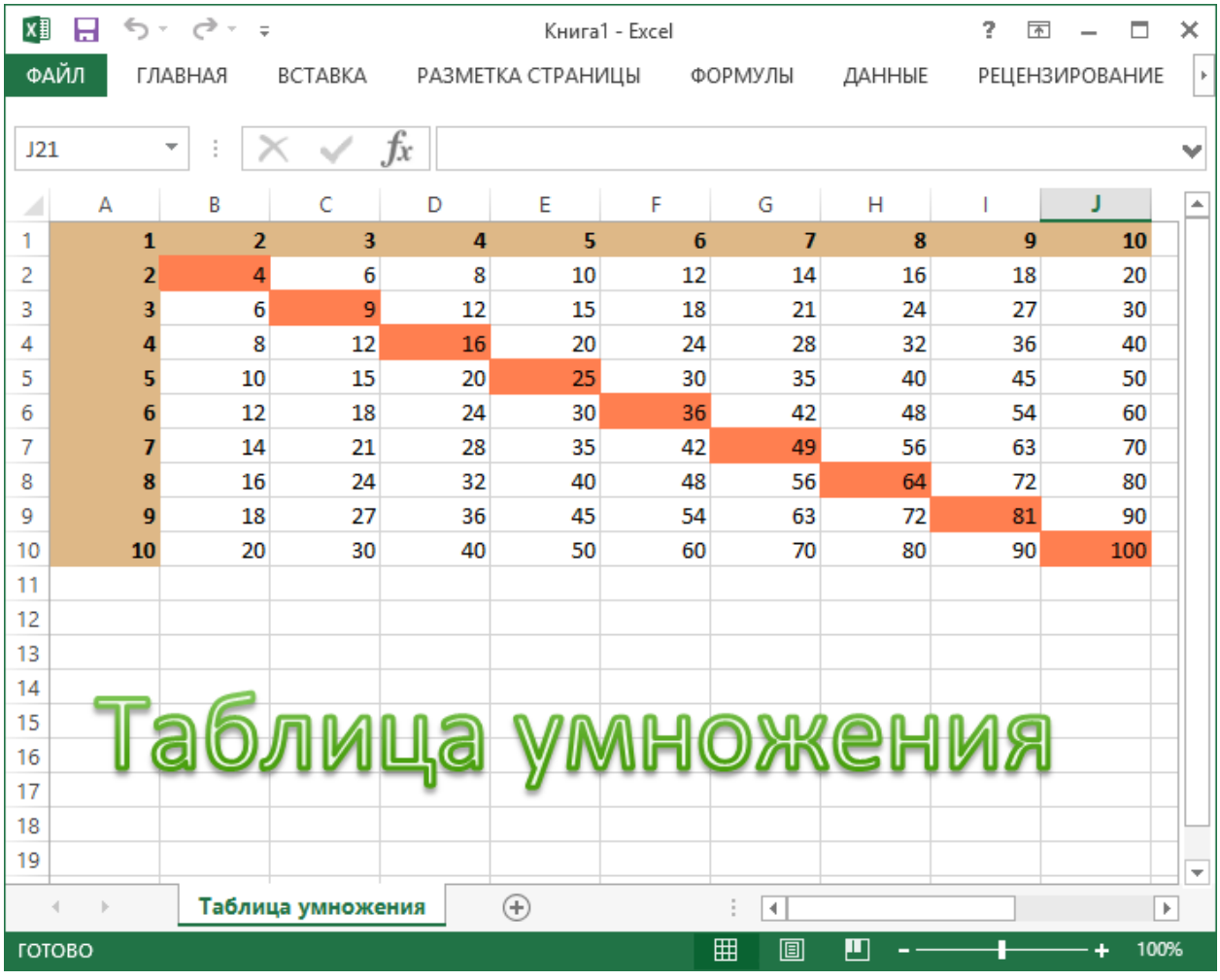


Рис.6.4. Создание таблицы умножения в MS Excel

6.3. Заполнение шаблона MS Word

Постановка задачи: Разработать приложение, которое запускает MS Word и заполняет шаблон сообщения. Данные вводятся программным путем.

1. Создайте новый проект WindowsForms. Назовите проект Word.

2. Для создания приложения необходимо подключить ссылку Microsoft Word.

В обозревателе решений нажмите правой кнопкой мыши в разделе **Ссылки → Добавить ссылку → COM → Microsoft Word.**

3. Активизируйте форму и измените ее имя на Создание письма. Укажите размер 560; 550. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

4. Для заполнения письма нам необходимы такие параметры как: Наименование компании, Слоган компании, Дата, Входящий номер, Заголовок документа, Адрес, Телефон, Факс и Текст письма. Для этого разместите элемент Panel, в свойстве BorderStyle укажите Fixed3D. На панель добавьте элементы label и в свойстве Text задайте соответствующие значения (рис.6.5).

5. Для каждого элемента label добавьте свое текстовое поле (рис.6.5).

6. Над панелью добавьте указатель пути шаблона (рис.6.5).

7. Добавьте кнопки – Обзор, Сгенерировать, Сформировать и Выход, согласно рис.6.5.

8. Перенесите на форму OpenFileDialog и ErrorProvider.

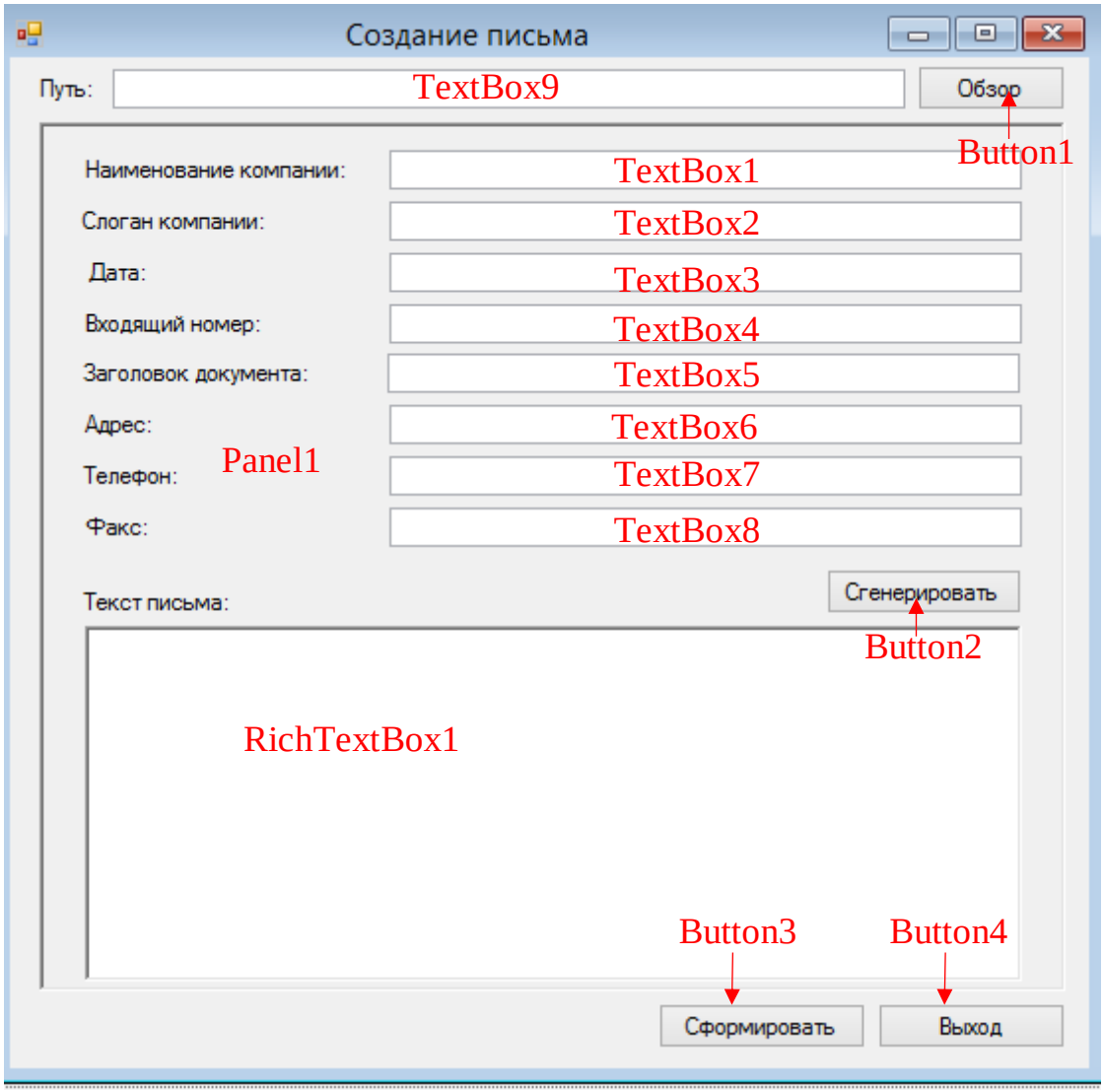


Рис.6.5.Элементы формы «Создание письма»

9. Измените свойство **Name** элементов, согласно таб. 15. Для того, чтобы не запутаться в элементах из-за большого количества данных.

Таблица 15.

Элемент	Name
TextBox1	tbName
TextBox2	tbSlogan
TextBox3	tbDate
TextBox4	tbNumber
TextBox5	tbTitle
TextBox6	tbAddress
TextBox7	tbTelephon
TextBox8	tbFax
TextBox9	tbPath
RichTextBox1	Content
Button1	btnBrowse

Button2	btnFish
Button3	btnMail
Button4	btnClose

10. Что бы из формы можно было перенести данные, в шаблоне определенным образом указываем закладки, в которые будем вставлять текст из формы. *Закладки типа {наименование поля}.*

Выберете любой шаблон MS Word и заполните объекты надписи по образцу, согласно рис.6.6. Сам шаблон поместите в папку проекта → **bin** → **Debug**.

Рис.6.6.Пример шаблона MS Word

11. Перейдите в редактор кода и добавьте подключаемый модуль Word.

```
using Word = Microsoft.Office.Interop.Word;
```

12. Создайте переменные для приложения и документа Word.

```
public partial class Form1 : Form
    {
        Word.Application app;
               Word.Document doc;
```

13. Опишите метод, который заполняет наш шаблон вводимыми данными. Создадим приложение и документ, заменим наши закладки на значения методом ReplaceWordStub.

```
private void documentWriter()
        {
            app = new Word.Application();
            doc = app.Documents.Open(tbPath.Text, Visible:true);
            try
            {
                //  Заменим наши закладки на нужный текст
                ReplaceWordStub("{NAMECOMPANY}", tbName.Text, doc, true);
                ReplaceWordStub("{slogan}", tbSlogan.Text, doc, true);
                ReplaceWordStub("{date}", tbDate.Text, doc, true);
                ReplaceWordStub("{number}", tbNumber.Text, doc, true);
                ReplaceWordStub("{TITLE}", tbTitle.Text, doc, true);
                ReplaceWordStub("{content}", Content.Text, doc, false);
                ReplaceWordStub("{address}", tbAddress.Text, doc, true);
                ReplaceWordStub("{telephon}", tbTelephon.Text, doc, true);
                ReplaceWordStub("{fax}", tbFax.Text, doc, true);
                app.Visible = true;
            }
            catch{}
               }
```

14. Опишите метод замены. Мы пробегаемся по всем объектам типа надпись и если мы нашли нужную закладку, то производим замену.

```
private void ReplaceWordStub(string stubToReplace, string text, Word.Document wordDocument,
bool isSmallReplace)
        {
            object replaceTypeObj = Word.WdReplace.wdReplaceAll;
            object missingObj = System.Reflection.Missing.Value;
            object toReplace = stubToReplace;
            object ReplText = text;
            Word.Shape wordShape;
            object matchCase = false;
            object replace = 2;
            for (int i = 1; i <= wordDocument.Shapes.Count; i++)
            {
                wordShape = wordDocument.Shapes[i];

                try
                {
                    if (isSmallReplace)

wordDocument.Shapes[i].TextFrame.TextRange.Find.Execute(toReplace, missingObj, missingObj,
                        missingObj,   missingObj,   missingObj,   missingObj,   missingObj,   missingObj,
missingObj, ReplText,
                        replaceTypeObj, missingObj, missingObj, missingObj, missingObj);
                    else
                    {
                        if
(wordDocument.Shapes[i].TextFrame.TextRange.Find.Execute(toReplace,           missingObj,
missingObj,
                            missingObj,  missingObj,  missingObj,  missingObj,  missingObj,
missingObj, missingObj,
                            missingObj, missingObj, missingObj, missingObj, missingObj))
                        {
                            wordShape.TextFrame.TextRange.Text = text;
                        };
                    }
```

```
                }
                catch (Exception e)
                {
                    continue;
                }
            }
        }
```

15. Добавьте обработчик событий для кнопки Закрыть. Закрываем документ, приложение и саму программу.

```
private void btnClose_Click(object sender, EventArgs e)
        {
            try
            {
                doc.Close();
                doc = null;
                app.Quit();
            }
            catch { }
            Close();
               }
```

16. Добавьте обработчик событий для кнопки Обзор. Вызываем стандартный диалог открытия файла с установленным фильтром .docx. Если пользователь выбрал шаблон заносим его имя в элемент tbPath.

```
private void btnBrowse_Click(object sender, EventArgs e)
        {
            dialog.Filter = "Файлы документа Word (*.docx)|*docx";
            if (dialog.ShowDialog() == DialogResult.OK)
            {
                tbPath.Text = dialog.FileName;
            }
               }
```

17. Добавьте обработчик событий для кнопки Сформировать. Для каждого TextBox определим значение, пустое или нет. И в зависимости от этого, элементу присвоим текст ошибки и передадим фокус обратно на данный элемент, чтобы пользователь заполнил данный элемент, при этом поле будет подсвечиваться. Если все поля заполнены правильно вызываем метод заполнения документа.

```
private void btnMail_Click(object sender, EventArgs e)
        {
            bool valid = true;
            error.Clear();
            if (tbName.Text == string.Empty)
            {
                error.SetError(tbName,  "Все  поля  должны  быть  заполнены\nЗаполните
наименование компании");
                tbName.Focus();
                valid = false;
            }
            if (tbSlogan.Text == string.Empty)
            {
                error.SetError(tbSlogan, "Все поля должны быть заполнены\nЗаполните слоган
компании");
                tbSlogan.Focus();
```

```
                valid = false;
            }
            if (tbDate.Text == string.Empty)
            {
                error.SetError(tbDate, "Все поля должны быть заполнены\nЗаполните дату
документа");
                tbDate.Focus();
                valid = false;
            }
            if (tbNumber.Text == string.Empty)
            {
                error.SetError(tbNumber, "Все поля должны быть заполнены\nЗаполните
входящий номер документа");
                tbNumber.Focus();
                valid = false;
            }
            if (tbAddress.Text == string.Empty)
            {
                error.SetError(tbAddress, "Все поля должны быть заполнены\nЗаполните
почтовый адрес компании");
                tbAddress.Focus();
                valid = false;
            }
            if (tbTelephon.Text == string.Empty)
            {
                error.SetError(tbTelephon, "Все поля должны быть заполнены\nЗаполните
телефон компании");
                tbTelephon.Focus();
                valid = false;
            }
            if (tbFax.Text == string.Empty)
            {
                error.SetError(tbFax, "Все поля должны быть заполнены\nЗаполните факс
компании");
                tbFax.Focus();
                valid = false;
            }
            if (Content.Text == string.Empty)
            {
                error.SetError(Content, "Все поля должны быть заполнены\nЗаполните текст
письма");
                Content.Focus();
                valid = false;
            }
            if (valid)
                documentWriter();
              }
```

18. Добавьте обработчик событий TextChanged для элемента tbPath. Если не выбран шаблон, то панель с элементами делаем недоступной.

```
private void tbPath_TextChanged(object sender, EventArgs e)
        {
            panel1.Enabled = btnMail.Enabled = (tbPath.Text != string.Empty);
              }
```

19. Добавьте обработчик событий для кнопки Сгенерировать.

```
private void btnFish_Click(object sender, EventArgs e)
        {
            string str;
            using (StreamReader strr = new StreamReader(
```

```
                HttpWebRequest.Create(@"https://fish-
text.ru/get?type=paragraph&number=2&format=html").GetResponse().GetResponseStream()))
                str = strr.ReadToEnd();
            //  Ответ с сайта придет в формате html
            //  Абзацы заключены в теги <p></p>
            //  Преобразуем наш текст перед выводом
            Content.Text = str.Replace("<p>", "\n       ").Replace("</p>", "\n").Remove(0,
1);
              }
```

20. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис.6.7.

Рис.6.7. Создание письма Word

6.4. Задачи для самостоятельной работы

1. Напишите программу, которая отображает документы Word через COM-объекты.

2. Напишите программу, которая отображает документы Excel через COM-объекты.

3. Напишите программу, которая отображает документы PowerPoint через COM-объекты.

4. Напишите программу, которая открывает документы Word, Excel. Программа определяет тип файла и запускает соответствующую программу.

5. Напишите программу для формирования в MS Word приглашения, на какое-либо торжество.

6. Напишите программу, которая выделяет несколько первых символов в документе Word, и выводит их в окно сообщений MessageBox.

7. Напишите программу, которая создает документ Word и в этом документе создает таблицу умножения.

8. Доработать предыдущую программу. После создания документа, автоматически сохранять документ. Имя файла задается пользователем.

9. Напишите программу, открывает документ Word и заполняет его случайным текстом, содержащим буквы, цифры, пробелы, точки, запятые. Объем заполняемого текста задается пользователем.

10. Доработайте предыдущую программу, необходимо отформатировать случайный текст следующим образом: все цифры выделить жирным, после каждой запятой вставить пробел, после каждой точки вставить символ “конец строки”. В полученных абзацах применить красною строку.

11. Напишите программу, которая открывает документ Word и вставляет на каждую страницу нижний колонтитул, содержащий номера страниц, отформатированные по центру и обрамленные знаками "-" (например: - 1 -)

12. Напишите программу, которая создает документ Word, содержащий логотип компании. Рисунок, пользователь указывает самостоятельно. Использовать вставку объекта "Рисунок". Подогнать размер рисунка по размерам страницы, применить художественный эффект "Кисть", задать сквозное обтекание.

13. Напишите программу, которая формирует служебную записку компании. Шаблон фирменного бланка, а также поля (название фирмы,

заголовок, текст письма, и т.д.) указывает пользователь. Для вставки текста в шаблон, использовать закладки (Bookmark).

14. Напишите программу, которая запускает MS Excel, создает новую книгу, сохраняет документ под именем Example.xls, завершает работу MS Excel.

15. Напишите программу, которая запускает MS Excel, создает новую книгу и в этой книге создает таблицу умножения в шестнадцатеричной системе счисления используя формулы.

16. Напишите программу, которая автоматически запускает MS Excel, и вводит данные в таблицу из поля, находящегося на форме.

17. Напишите программу, которая запускает MS Excel и рисует график функции $y = \sin x$. Данные расположить на одном листе, график на другом.

18. Напишите программу, которая производит решения треугольника средствами Excel, ввод и вывод данных должен происходить через разрабатываемое приложение.

19. Напишите программу, которая запускает MS Excel и строит гистограмму по данным, указанным в таблице на форме.

20. Напишите программу, которая запускает MS Excel и рисует на листе алгоритм подсчета суммы чисел от 1до 100. Использовать вставку фигур и надписей.

21. Напишите программу калькулятор, производящую вычисления в MS Excel

22. Напишите программу, которая запускает MS Excel и заполняет ячейки следующим образом: 1) колонка - число, 2- месяц, 3-год, 4 - день недели. Диапазон дат с 01.01.2017 по 30.08.2018. Задать "фильтр и сортировку", на диапазон, так, чтобы показывались только четверги, числа которого начинаются на цифру 2) Отсортировать от макимального года к минимальному.

23. Напишите программу, которая запускает MS Excel и заполняет 5 столбцов произвольными значениями с плавающей точкой в диапазоне от -1000 до 2000. Подсчитать среднее значение, сумму, минимум и максимум по строкам и столбцам. По каждой строке сформировать среднее арифметическое из округленных до целого значений, сравнить с предыдущим значением и, если они совпадают, выделить строку красным цветом.

24. Напишите программу, которая запускает MS Acsess и создает БД BDShop. Создает 2 таблицы Product, содержащий поля id, Name, Price и Customer, содержащий поля id, idProduct, Name, Count.

25. Доработайте предыдущую программу, которая создает первичные ключи к таблицам по полю id и вторичные по полю Name.

26. Доработайте предыдущую программу, которая заполняет таблицы произвольными данными. В форме предусмотреть поле ввода для произвольного запроса SQL, после ввода которого показывается результат в DataGridView.

27. Напишите программу, которая воспроизводит презентацию PowerPoint. Пользователь сам выбирает файл с презентацией.

28. Напишите программу, которая запускает OneNote и создает произвольную запись.

29. Напишите программу, которая запускает MS Outlook и отправляет Email указанному адресату.

30. Напишите программу, которая конвертирует документ MS Word в PDF и показывает этот документ на форме.

7. Сетевые компоненты

Сегодня миллионы компьютеров и устройств связаны в глобальную сеть интернет, либо в отдельные локальные подсети. В связи с этим возникает необходимость создания приложений, которые бы использовали все преимущества передачи данных по сети. Например, одним из распространенных приложений, которое использует передачу по сети, является веб-браузер. И платформа .NET и язык программирования C# предоставляют все необходимые возможности для создания приложений, которые могут взаимодействовать по сети и использовать различные сетевые протоколы.

Прежде чем отправить запрос к какому-нибудь ресурсу, компьютер обращается к DNS-северу, чтобы по имени ресурса получить его ip-адрес.

7.1. Протокол SMTP

Постановка задачи: Разработать приложение, которое представляет из себя почтового клиента и отправляет сообщения указанному адресату.

Для отправки почты в среде интернет используется протокол SMTP. Данный протокол указывает, как почтовые сервера взаимодействуют при передаче электронной почты.

Для работы с протоколом SMTP и отправки электронной почты в .NET предназначен класс SmtpClient из пространства имен System.Net.Mail.

1. Создайте новый проект WindowsForms. Назовите проект Email.

2. Активизируйте форму и измените ее имя на Отправка E-mail. Укажите размер 500; 400. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

3. Через форму будем указывать почту отправителя, тему и сам текст сообщения. На форму добавьте элементы label и в свойстве Text задайте соответствующие значения (рис.7.1).

5. Для каждого элемента label добавьте свое текстовое поле (рис.7.1).

6. Во второй части приложения добавьте элемент GroupBox. Здесь будут содержаться данные для подключения к почте отправителя.

В свойстве Text элемента GroupBox укажите Настройка почтового клиента.

7. На GroupBox перенесите три элемента label – Имя пользователя, Пароль и Почтовый сайт. Для каждого элемента добавьте свое текстовое поле (рис.7.1).

8. Добавьте кнопку Отправить (рис.7.1).

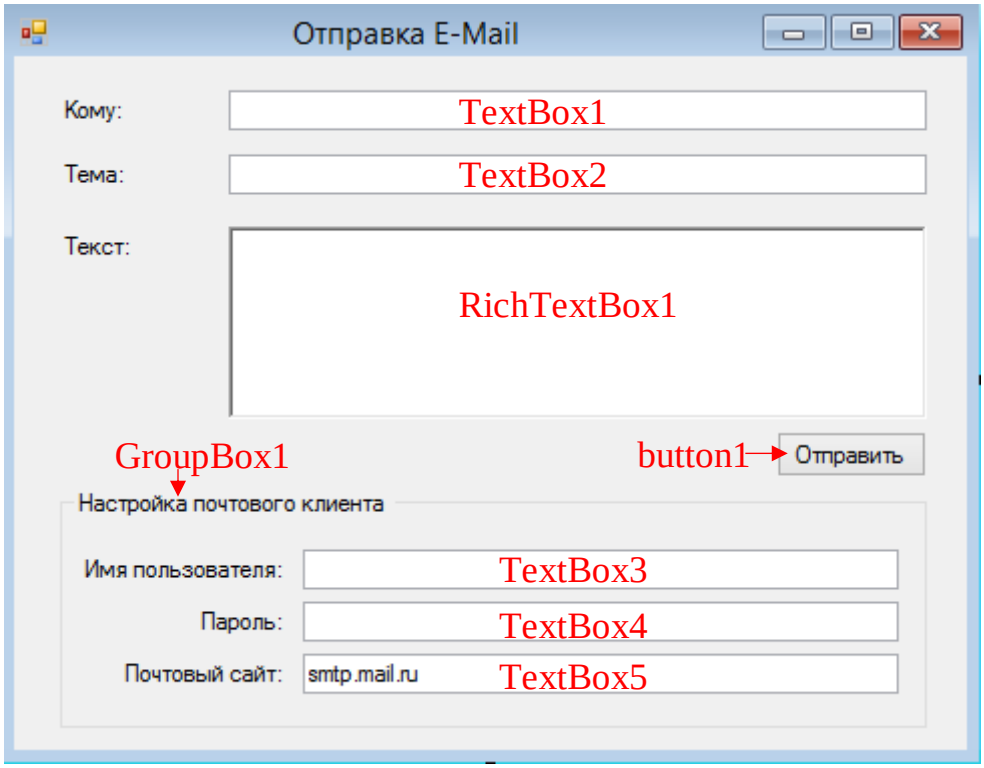


Рис.7.1. Элементы формы «Отправка E-mail»

9. Каждый пользователь, работающий с почтовыми приложениями, так или иначе сталкивается с настройкой параметров входящей и исходящей корреспонденции. Одним из основных элементов является конфигурация SMTP-сервера.

Сокращение SMTP происходит от английского словосочетания Simple Mail Transfer Protocol, что в переводе означает «простой протокол отправки почты». В основном область его применения ограничивается сетями на основе TCP/IP и пользовательским уровнем.

Для этого в свойстве Text элемента TextBox5 укажите stmp.mail.ru. Большинство популярных почтовых клиентов используют SMTP-сервера.

10. В свойстве **PasswordChar** (Указывает символ для отображения пароля при вводе в однострочном поле редактирования) элемента TextBox4 укажите символ *.

11. Добавьте на форму элемент ErorProvider.

12. Перейдите в редактор кода и добавьте подключаемый модуль Net.

```
using System.Net.Mail;
```

13. Добавьте обработчик событий для кнопки Отправить.

Создаем объект типа NetworkCredential, который принимает логин и пароль. Задаем имя сервера по порту 587 (для того чтобы отправить письмо используется протокол SMTP – по порту 25 (устаревший), либо по порту 587.

Для разных почтовых клиентов адрес должен быть свой. В настройках можно выяснить имя почтового сайта, который отвечает за доставку писем, а также можно узнать номер порта.

Создаем объект MailMessage где заполняем все поля: от кого письмо, кому письмо, тему и сам текст сообщений. Для наглядности покажем имя отправителя, который будет показан при получении письма.

Далее отправляем письмо в безопасном режиме, при появлении исключительной ситуации работа нашей программы не остановится.

```
private void btnSend_Click(object sender, EventArgs e)
        {
            System.Net.NetworkCredential crdSupport = new
System.Net.NetworkCredential(tbLogin.Text, tbPassword.Text);
             SmtpClient smtpConfirmation = new SmtpClient(tbSMTPClient.Text, 587);//тут
можно указать адрес сервер, логин и пароль доступа к которому казан выше
            smtpConfirmation.Credentials = crdSupport;
            smtpConfirmation.EnableSsl = true;

            MailMessage mmConfirmation = new MailMessage();
            mmConfirmation.From = new MailAddress(tbLogin.Text, "John Dow");
            mmConfirmation.To.Add(tbFrom.Text);
            mmConfirmation.Subject = tbSubject.Text;
            mmConfirmation.Body = tbBody.Text;
            try
            {
                smtpConfirmation.Send(mmConfirmation);
                MessageBox.Show("Письмо успешно отправлено ", "Отправка");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, ex.Source);
            }

        }
```

14. Добавьте событие Validating для элементов TextBox1 и TextBox3, для введения корректного e-mail адреса.

```
private void tbFrom_Validating(object sender, CancelEventArgs e)

            {
                errorProvider1.Clear();
                if (tbFrom.Text == string.Empty) return;
                try
                {
                    var mail = new System.Net.Mail.MailAddress(tbFrom.Text);
                    e.Cancel = false;

                }
                catch
                {
                    e.Cancel = true;
                    errorProvider1.SetError(tbFrom, "E-Mail адресс введен некорректно");
                }

        }

        private void tbLogin_Validating(object sender, CancelEventArgs e)
```

```
{
    errorProvider1.Clear();
    if (tbLogin.Text == string.Empty) return;
    try
    {
        var mail = new System.Net.Mail.MailAddress(tbLogin.Text);
        e.Cancel = false;

    }
    catch
    {
        e.Cancel = true;
        errorProvider1.SetError(tbLogin, "E-Mail адрес введен некорректно");
    }
}
```

15. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис.7.2.

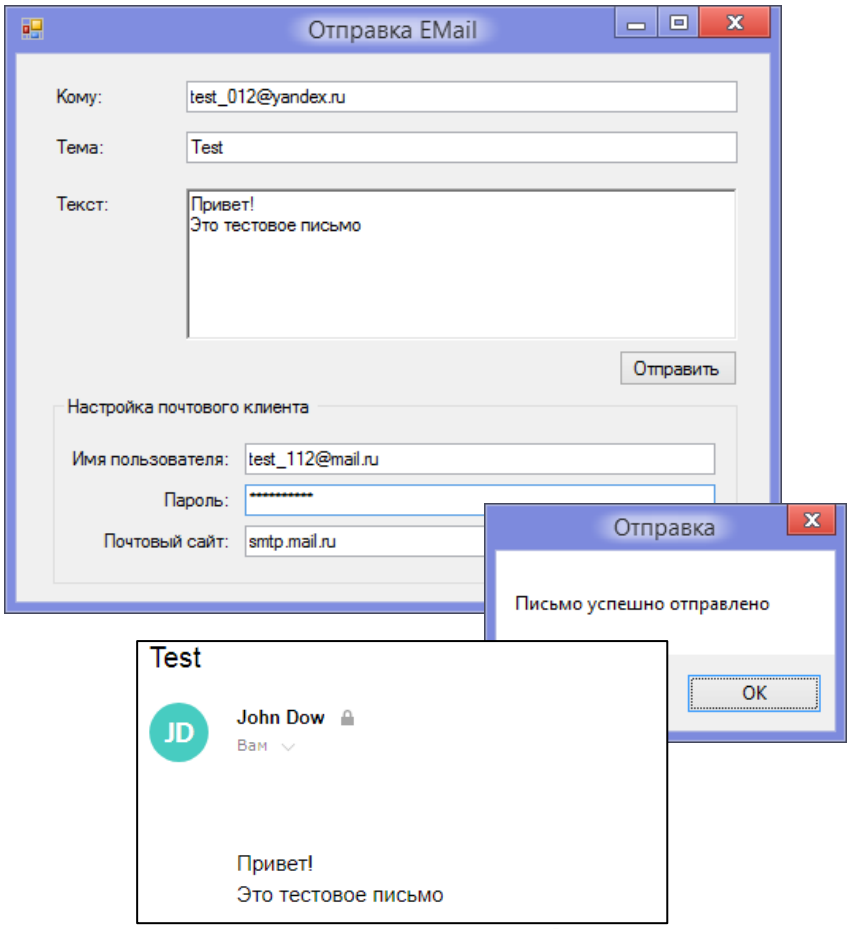


Рис.7.2. Отправка E-mail

7.2. Очередь сообщений

В Microsoft есть очередь сообщений, она устанавливается, как компоненты Windows. Администратор домена создает очередь сообщений, и пользователи могут прислать и получать сообщения используя эту очередь. Особенность заключается в том, что пользователю не нужно быть в сети в тот момент, когда ему отправляется сообщение, эти сообщения будут хранится на сервере и когда пользователь подключится к домену он сможет их прочитать.

В пространстве имен **System.Messaging** определены классы, которые позволяют выполнять чтение и запись сообщений с использованием такого предлагаемого в составе операционной системы Windows средства для организации сообщений, как **Message Queuing**. В **Message Queuing** сообщения записываются и читаются из специальной очереди сообщений.

Сообщение включает тело, содержащее пересылаемые данные, и метку — заголовок сообщения. В тело сообщения может быть помещена любая информация. В .NET имеется набор форматировщиков, преобразующих данные, которые помещаются в тело. Помимо метки и тела сообщение включает дополнительную информацию об отправителе, конфигурацию таймаута, идентификатор транзакции или приоритет.

Очередь сообщений представляет собой своего рода "накопительный бункер" для сообщений. Сообщения, сохраняемые на диске, размещаются в каталоге <windows>\system32\msmq\storage.

Очереди сообщений могут быть созданы программно, вызовом метода **Create()** класса **MessageQueue**. Методу **Create()** должен быть передан путь к новой очереди. Этот путь состоит из имени хоста, где расположена очередь, и имени очереди.

Постановка задачи: Написать программу, создающую и отправляющую сообщения в очередь сообщений. Предусмотреть возможность прочтения и последующего удаления сообщений.

1. Создайте новый проект WindowsForms. Назовите проект MessageQueue.

2. Активизируйте форму и измените ее имя на Очередь сообщений. Укажите размер 450; 350. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

3. Добавьте элементы на форму согласно рисунку 7.3.

4. Для элемента RichTextBox в свойстве BorderStyle укажите Fixed3D.

5. Существует несколько типов очереди сообщений. Общедоступные или частные очереди обычно применяются для отправки сообщений.

Общедоступная (public) очередь – Публикуется в Active Directory. Информация о таких очередях реплицируется в доменах Active Directory. Для получения информации о таких очередях можно воспользоваться средствами просмотра и поиска. К общедоступной очереди можно обращаться, не зная имени компьютера, на котором она расположена. Такую очередь можно переместить с одной системы на другую, и клиент этого не заметит.

Частные (private) очереди доступны, только когда известны их полные путевые имена. Частные очереди могут использоваться в среде Workgroup.

Добавьте коллекцию для элемента NumericUpDown – Общедоступная (public) очередь; Частные (private) очереди.

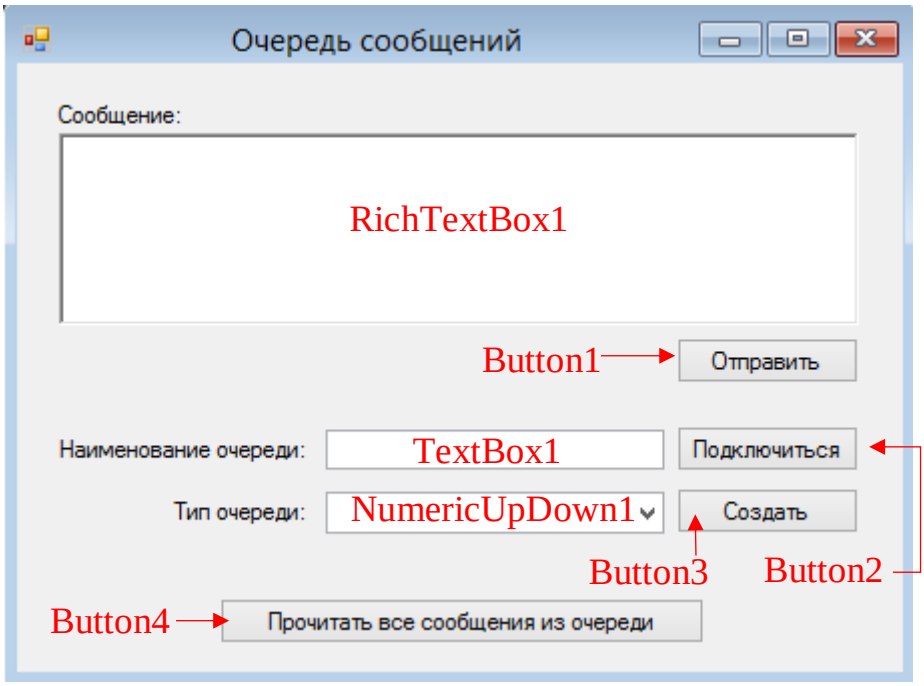


Рис.7.3. Элементы формы «Очередь сообщений»

Принцип работы заключается в том, что сначала пишем наименование очереди, если такой нет, то создаем, затем подключаем ее, при прочтении всех сообщений они удаляются из очереди.

Примечание. Через очередь сообщений можно передавать не только текст, но и классы, картинки.

6. Перейдите в редактор кода и добавьте подключаемый модуль Messaging.

```
using System.Messaging;
```

7. Создайте поле, отвечающее за то, открыта ли очередь.

```
public partial class Form1 : Form
    {
        private bool _isConnectQueuing;
        public bool isConnectQueuing
```

```
        {
            get
            {
                return _isConnectQueuing;
            }
            set
            {
                _isConnectQueuing = value;
                btRecieve.Enabled = value;
                btSend.Enabled = value;
            }
        }
```

8. Опишите переменную отвечающую за очередь сообщений.

```
MessageQueue queue;
```

9. Добавьте функцию формирующую полный путь к очереди. Частная очередь – private$, общая – public$.

```
private string GetPathQueuing()
        {
            return ((cbTypeQueuing.SelectedIndex == 1) ? ".\\private$\\" : ".\\public$\\")
+ tbNameQueuing.Text;
              }
```

10. Необходимо определить существует ли такая очередь. Добавьте функцию определяющую существует ли данная очередь: PathQueuing в системе.

Функция принимает одно значение – имя очереди. В цикле перебираем все очереди, которые зарегистрированы в системе и возвращаем признак содержится ли имя очереди в просматриваемых очередях в системе.

```
private bool FindQueuing(string PathQueuing)
        {
            bool isFindPathQueuing = false;
            foreach                    (var                    queue                    in
MessageQueue.GetPrivateQueuesByMachine(Environment.MachineName))
            {
                isFindPathQueuing = PathQueuing.Contains(queue.QueueName);
            }
            return isFindPathQueuing;
        }
```

11. При инициализации добавьте начальное значение объектов.

```
public Form1()
        {
            InitializeComponent();
            cbTypeQueuing.Text = cbTypeQueuing.Items[1].ToString();
            cbTypeQueuing.SelectedIndex = 1;
            isConnectQueuing = false;
               }
```

12. Добавьте обработчик событий для изменения выбора очереди. Если выбрано, хоть какое-то значение, кнопки создать и подключить становятся активными.

```
private void tbNameQueuing_TextChanged(object sender, EventArgs e)
        {
            btConectQueuing.Enabled = btCreateQueuing.Enabled = tbNameQueuing.Text != 
string.Empty;
        }
```

13. Добавьте обработчик событий для кнопки подключить.

```
private void btConectQueuing_Click(object sender, EventArgs e)
        {
            //  Подключение к очереди
            string PathQueuing = GetPathQueuing();      //  Формируем строку подключения
            if (FindQueuing(PathQueuing))               //  Если такая очередь найдена
            {
                queue = new MessageQueue(@PathQueuing); //  Подключаем очередь
                MessageBox.Show("Очередь " + PathQueuing + " подключена.", "Информация", 
MessageBoxButtons.OK, MessageBoxIcon.Information);
                isConnectQueuing = true;
            }

            else MessageBox.Show("Очередь " + PathQueuing + " не найдена.", "Ошибка", 
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
```

14. Аналогично добавьте обработчик событий для кнопки создания очереди.

```
private void btCreateQueuing_Click(object sender, EventArgs e)
        {
            //  Создадим очередь
            string PathQueuing = GetPathQueuing();      //  Формируем строку подключения
            if (FindQueuing(PathQueuing))               //  Если такая очередь найдена
                MessageBox.Show("Очередь " + PathQueuing + " найдена. Создание 
невозможно.\nДля подключения к данной очереди нажмите Подключиться", "Ошибка", 
MessageBoxButtons.OK, MessageBoxIcon.Error);
            else
                try
                {
                    using (var queue = MessageQueue.Create(@PathQueuing))    //  Создаем 
очередь
                    {
                        queue.Label = tbNameQueuing.Text;
                        MessageBox.Show("Очередь " + PathQueuing + " создана.\n" + "Путь: 
" + queue.Path, "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        isConnectQueuing = true;
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Ошибка создания очереди " + PathQueuing + 
"\nНеобходимо обладать правами администратора домена", "Ошибка", MessageBoxButtons.OK, 
MessageBoxIcon.Error);
                }
        }
```

15. Добавьте обработчик событий для кнопки Отправить. Посылаем в очередь текст сообщения, при этом его удалям из RichTextBox1 (MessageText).

```
        private void btSend_Click(object sender, EventArgs e)
        {
            try
            {
                queue.Send(MessageText.Text, "Сообщение");
```

```
                MessageText.Text = "";
                MessageBox.Show("Сообщение отправлено в очередь!", "Информация",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                MessageText.Focus();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString(), "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
```

16. Добавьте обработчик событий для кнопки Прочитать все сообщения из очереди.

```
  private void btRecieve_Click(object sender, EventArgs e)
        {
            //  Просмотрим все сообщения в очереди и удалим их
            TimeSpan timeout = TimeSpan.FromSeconds(1);     //  Считывать сообщения будем
с интервалом в 1 секунду
            MessageEnumerator enumerator = queue.GetMessageEnumerator2();    //  Считаем
всю очередь сообщений
            while (enumerator.MoveNext(timeout))
            {
                System.Messaging.Message Message = enumerator.Current;       //  Текущее
сообщение
                Message.Formatter = new XmlMessageFormatter(new String[] {
"System.String,mscorlib" }); //  укажем формат сообщения
                //  Выведем сообщение на экран
                MessageBox.Show(
                    Message.Body.ToString(),        //  Тело сообщения
                    enumerator.Current.Label,       //  Заголовок сообщения
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information
                    );
                enumerator.RemoveCurrent();         //  Удалим, прочитанное сообщение
                enumerator.Reset();                 //  Вернемся в начало очереди
            }
        }
```

17. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис.7.4.

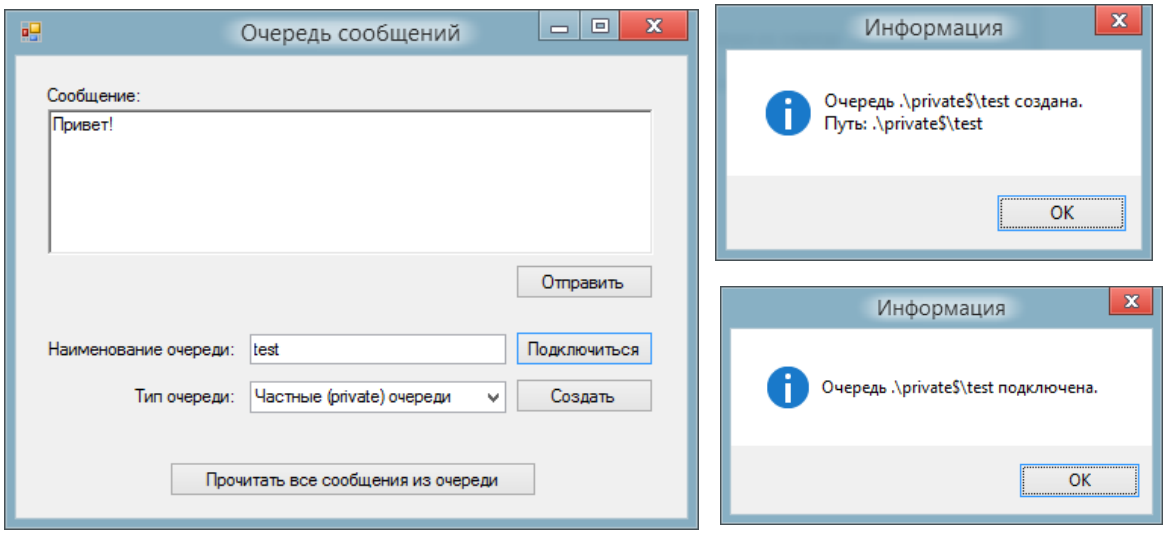


Рис.7.4. Очередь сообщений

7.3. Клиент-серверное приложение на сокетах TCP

В основе межсетевых взаимодействий по протоколам TCP и UDP лежат сокеты. В .NET сокеты представлены классом System.NET.Sockets.Socket, который предоставляет низкоуровневый интерфейс для приема и отправки сообщений по сети.

Постановка задачи: Создать клиент-серверное сетевое приложение для отправки сообщения серверу. Для реализации приложения необходимо использовать потоковые сокеты протокола TCP

КЛИЕНТ

1. Клиент – отправляет введённое сообщение. Для этого подключаем сокет по протоколу TCP 904. Перед отправкой сообщение шифруем DES.

Примечание. DES – блочный алгоритм, то есть при шифровании исходное сообщение переводится в двоичный код, а затем разбивается на блоки и каждый блок отдельно зашифровывается (расшифровывается).

Создайте новый проект WindowsForms. Назовите проект SocketSecurityClient.

2. Активизируйте форму и измените ее имя на Клиент. Укажите размер 550; 200. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

3. Добавьте элементы label, RichTextBox, Button1, Button 2 на форму согласно рисунку 7.5.

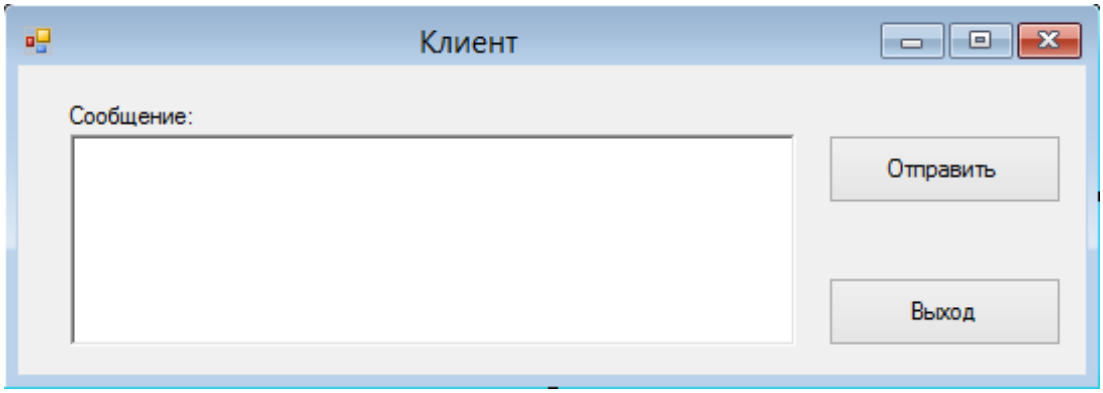


Рис.7.5. Форма «Клиент»

4. Перейдите в редактор кода и опишите поля сокет, буфер для передачи данных (данные передаются не текстом, а массивом байтов) и задайте секретный ключ для шифрования.

```
public partial class Form1 : Form
    {
        static Socket socket;
        private byte[] data;
```

```
        // Ключ безопасности
              private const string MyKey = "„B_itмґЃ©_[7ГёMzKЪ_¬H:©_yGP_–#ћI";
```

5. Добавьте обработчик событий для кнопки Закрыть.

```
private void button2_Click(object sender, EventArgs e)
        {
            Close();
               }
```

6. Добавьте обработчик событий для RichTextBox. Если текст не введен, кнопка отправить становятся не активной.

```
private void MessageText_TextChanged(object sender, EventArgs e)
        {
            btnSend.Enabled = MessageText.Text != string.Empty;
               }
```

7. Опишите функцию шифрования строки. Генерация случайного вектора инициализации, допустим передается очень много однотипной информации, тогда зашифрованные строки тоже будут одинаковые и злоумышленникам проще получить информацию, для того чтобы этого избежать, в начало строки добавляем, случайную последовательность символов, это приводит к тому, что даже одинаковые строки после шифрования всегда будут выглядеть по-разному.

```
public static byte[] EncryptData(string data)
        {
            // Преобразовать строку data в байтовый массив
            byte[] ClearData = Encoding.UTF8.GetBytes(data);

            // Создадим алгоритм шифрования
            SymmetricAlgorithm Algorithm = SymmetricAlgorithm.Create();
            Algorithm.Key = Encoding.ASCII.GetBytes(MyKey);

            // Зашифровать информацию
            MemoryStream Target = new MemoryStream();

            // Сгенерировать случайный вектор инициализации (IV)
            // для использования с алгоритмом
            Algorithm.GenerateIV();
            Target.Write(Algorithm.IV, 0, Algorithm.IV.Length);

            // Зашифровать реальные данные
            CryptoStream cs = new CryptoStream(Target, Algorithm.CreateEncryptor(),
CryptoStreamMode.Write);
            cs.Write(ClearData, 0, ClearData.Length);
            cs.FlushFinalBlock();

            // Вернуть зашифрованный поток данных в виде байтового массива
            return Target.ToArray();
               }
```

8. Добавьте обработчик событий для кнопки отправить.

```
private void btnSend_Click(object sender, EventArgs e)
        {
            try
            {
```

```
                //  Передача данных будет вестись по протоколу TCP
                socket   =   new   Socket(AddressFamily.InterNetwork,   SocketType.Stream,
ProtocolType.Tcp);
                //  Сокет цепляем на локальный хост по порту 904
                //  Если меняем адрес, то на удаленном компьютере, где запускаем сервер
                //  необходимо открыть порт в межсетевой экран
                socket.Connect("127.0.0.1", 904);
                if (socket.Connected)
                {
                    //  Просто так сообщение не отправить, его нужно перевести в байты
                    //  При этом мы его зашифруем
                    data = EncryptData(MessageText.Text);
                    socket.Send(data);
                    socket.Close();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }
```

СЕРВЕР

Сервер получает зашифрованное сообщение от клиента, расшифровывает его и выводит на экран.

9. Добавьте в решение еще один проект WindowsForms. Назовите проект SocketSecurityServer.

10. Активизируйте форму и измените ее имя на Сервер. Укажите размер 400; 250. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

11.. Добавьте элементы label, RichTextBox, Button1, на форму согласно рисунку 7.6.

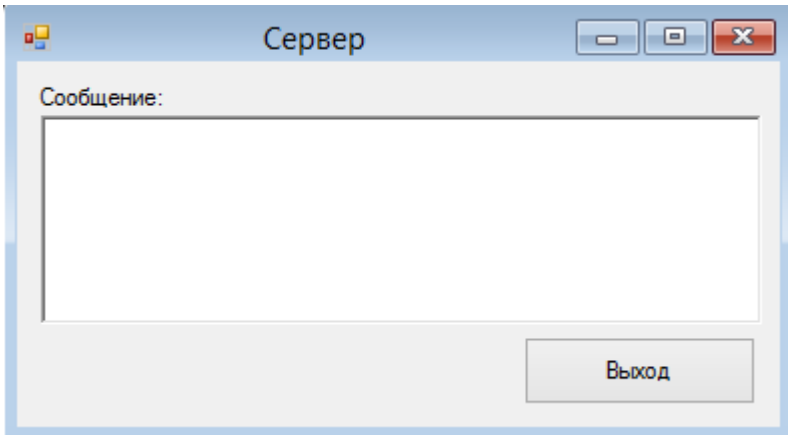


Рис.7.6. Форма «Сервер»

12. Опишите переменные для сокета, буфера, потока и приватного ключа, который должен в точности совпадать с ключом на клиенте.

```
public partial class Form1 : Form
    {
        Thread thReceive;
```

```
        static Socket socket;
        byte[] data;
        int byteRead;
        // Ключ безопасности
               private const string MyKey = "„B_itmґЃ©_[7ГёMzKЪ_¬H:©_yGP_–#ћI";
```

13. Опишите функцию получения сообщения.

```
private void ReceiveMessage()
        {
            //  Передача данных будет вестись по протоколу TCP
            socket    =    new    Socket(AddressFamily.InterNetwork,    SocketType.Stream,
ProtocolType.Tcp);
            socket.Bind(new IPEndPoint(IPAddress.Any, 904));     //  Слушать будем все IP
адреса по порту 904
            while (true)
            {
                socket.Listen(10);                                //  Запустим прослушивание
порта
                //  Сюда мы попадаем, если нам пришло сообщение
                byteRead = socket.Accept().Receive(data);           //  Получим сообщение
                //  Расшифруем и добавим наше сообщение в RichTextBox
                string dd= DecryptData(data, byteRead);
                MessageText.Text = DecryptData(data, byteRead);
            }
               }
```

14. Опишите функцию дешифрования сообщения.

На вход подается массив зашифрованных байтов на выходе получаем расшифрованную строку.

```
public static string DecryptData(byte[] data, int Len)
        {
            // Создать алгоритм
            SymmetricAlgorithm Algorithm = SymmetricAlgorithm.Create();
            Algorithm.Key = Encoding.ASCII.GetBytes(MyKey);

            // Расшифровать информацию
            MemoryStream Target = new MemoryStream();

            // Прочитать вектор инициализации (IV)
            // и инициализировать им алгоритм
            int ReadPos = 0;
            byte[] IV = new byte[Algorithm.IV.Length];
            Array.Copy(data, IV, IV.Length);
            Algorithm.IV = IV;
            ReadPos += Algorithm.IV.Length;

            CryptoStream cs = new CryptoStream(Target, Algorithm.CreateDecryptor(),
                CryptoStreamMode.Write);
            cs.Write(data, ReadPos, Len - ReadPos);
            cs.FlushFinalBlock();

            // Получить байты из потока в памяти и преобразовать их в текст
            return Encoding.UTF8.GetString(Target.ToArray());
               }
```

15. При инициализации формы задайте первоначальное значение и запустите в отдельном потоке прослушку порта.

```
public Form1()
```

```
        {
            InitializeComponent();
            data = new byte[1024]; //  Создадим буфер, который будем использовать при
получении собщений
            //  Т.к. прослушивание порта является операцией, которая тормозит приложение
            //  запустим его в отдельном потоке
            thReceive = new Thread(ReceiveMessage);     //  Создадим поток
            thReceive.IsBackground = true;              //  Наш поток будет фоновым
            thReceive.Start();                          //  Запустим поток
        }
```

16. Добавьте обработчик событий для кнопки Закрыть.

```
private void button2_Click(object sender, EventArgs e)
        {
            Close();
               }
```

17. Сохраните проект и скомпилируйте exe-файлы, запустите программы для проверки работоспособности. Результат показан на рис.7.7.

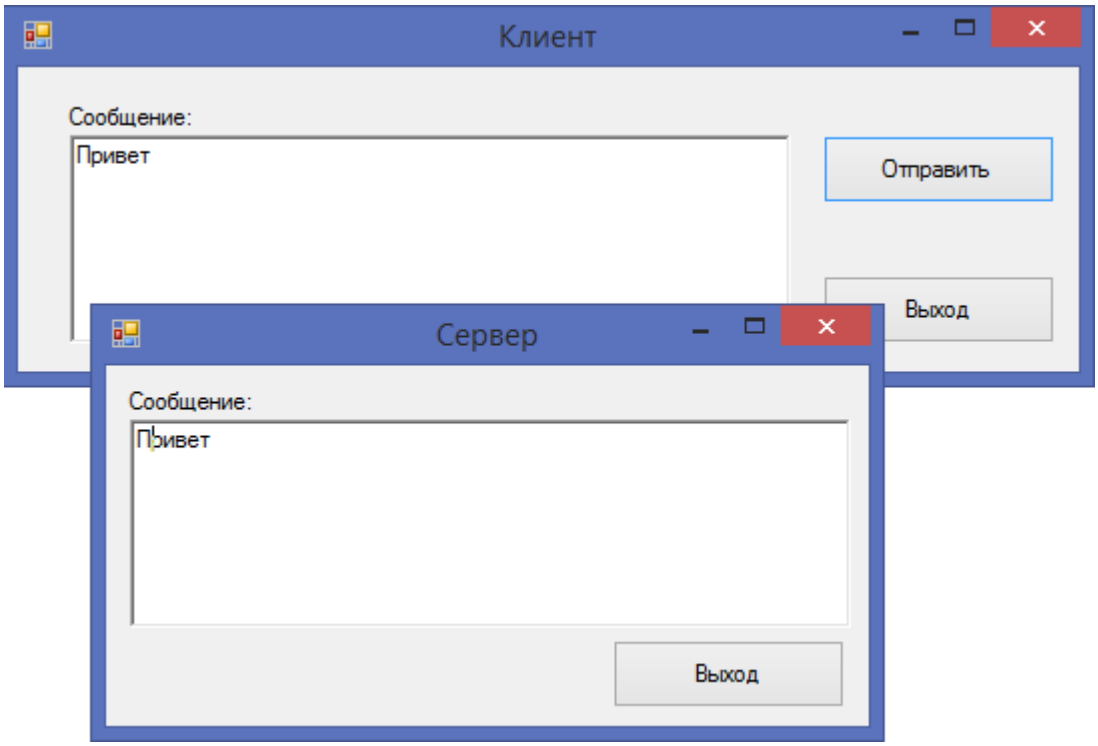


Рис.7.7. Клиент-серверное приложение

7.4. Задачи для самостоятельной работы

1. Напишите программу, которая зашифровывает, введенный пользователем текст в файл и расшифровывает зашифрованный файл.

2. Доработайте программу № 1, которая применяет разные алгоритмы шифрования. Методы шифрования и дешифрования пользователь выбирает самостоятельно.

3. Напишите программу, которая делает скрин экрана и пересылает картинку через сокет по протоколу TCP на сервер.

4. Напишите программу, которая переслать файл через сокет по протоколу UDP на сервер. Файл пользователь выбирает самостоятельно.

5. Напишите программу, представляющую из себя почтового клиента, который принимает и отправляет Email. Использовать протоколы SMTP и IMAP. Пользователь указывает свои данные для подключения к почтовому серверу.

6. Доработайте предыдущую программу, которая отправляет Email нескольким пользователям. Реализовать возможность прикреплять различные файлы к письму.

7. Напишите программу, которая умеет отправлять сообщения и принимать их по протоколу TCP. Обе операции должны быть разделены на потоки.

8. Доработайте программу № 7, которая умеет отправлять сообщения и принимать их. При этом протокол, IP-адрес, шлюз пользователь задает самостоятельно.

9. Доработайте предыдущую программу, которая шифрует и расшифровывает сообщения. Пользователь может самостоятельно выбрать тип шифрования (DES, TripleDES, ...).

10. Напишите программу, которая передает файл через очередь сообщений. Пользователь выбирает файл самостоятельно.

11. Разработайте приложение, позволяющее считывать каталоги и файлы с ftp сервера.

12. Разработайте приложение, которое синхронизирует системное время, с каким-либо сервером точного времени.

13. Напишите программу, которая показывает курс валют. Курсы брать с сервера Центробанка.

8. Собственные компоненты

Составные элементы управления предоставляют средства для создания и повторного использования настраиваемых графических интерфейсов. Составной элемент управления — это компонент, имеющий визуальное (или не визуальное) представление. Таким образом, он может состоять из одного или нескольких элементов управления, компонентов или блоков кода Windows Forms, позволяющих расширить функциональные возможности за счет проверки введенных пользователем данных, изменения свойств отображения или выполнения других предусмотренных разработчиком действий. Составные элементы управления можно вставлять в Windows Forms точно так же, как другие элементы управления.

8.1. Визуальный компонент

Постановка задачи: 1) Разработать компонент «Бегущая строка» на основе элемента бегущая строка. 2) Написать программу для использования собственного компонента «Бегущая строка», предусмотреть прокрутку как в лево, так и вправо.

1. Создайте новый проект Пользовательский элемент. Назовите проект LabelRunText.

2. В конструктор перетащите два элемента Label и Timer.

Label будет отвечать за сам текст, а Timer за скорость прокрутки.

3. Перейдите в редактор кода и определите свойство TextRun.

Данный текст и будет прокручиваться на экране. Browsable означает, что TextRun будет показан в свойстве элемента LabelTextRun. Мы присваем Label1 текст с введеным значением TextRun. Вызываем метод Invalidate (делает недействительной всю поверхность элемента управления и вызывает его перерисовку).

```
public partial class LabelRunText: UserControl
    {

        // Текст, показанный на экране
        private string _Text="label1";
        [Browsable(true)]
        public string TextRun
        {
            get { return _Text; }
            set
            {
                _Text = value;
                label1.Text = _Text;
                this.Invalidate();
            }
        }
```

4. Задайте свойство Speed, которое отвечает за скорость прокрутки текста.

Description – описание данного свойства, DefaultValue – значение по умолчанию.

При изменении свойства Speed меняем значение свойства Interval у Timer1, вызываем при этом метод Invalidate, для того чтобы скорость сразу поменялась.

```
public LabelRunText()
        {
            InitializeComponent();
        }

        private int _Speed = 150;
        // Изменение скорости прокрутки текста
        [Description("Изменение скорости прокрутки текста")]
        [DefaultValue(150)]
        [Browsable(true)]
        public int Speed
        {
            get { return _Speed; }
            set
            {
                _Speed = value;

                timer1.Interval = value;
                Invalidate();
            }
        }
```

5. Загадайте свойство, которое включает или отключает прокрутку текста.

При задании данного свойства меняем значение свойство Enabled элемента Timer1, затем вызываем метод Invalidate.

```
private bool _isEnableRun = false;
        // Вкл/Выкл. прокрутку текста
        [Description("Вкл/Выкл. прокрутку текста")]
        [DefaultValue(false)]
        [Browsable(true)]
        public bool isEnableRun
        {
            set
            {
                _isEnableRun = value;
                timer1.Enabled = _isEnableRun;
                Invalidate();
            }
            get { return _isEnableRun; }
        }
```

6. Задайте перечисление направления бегущего текста.

```
public enum RunOrientation { left, right}
```

7. Задайте свойство RunOrientation, которое будет отвечать за направление.

```
    // Устанавливает направление ориентации текста
        [Description("Устанавливает направление ориентации текста")]
        [DefaultValue(RunOrientation.left)]
        [Browsable(true)]
        public RunOrientation OrientationRun { get; set; }
```

8. Добавьте обработчик событий Tick для Timer1.

Если в данный момент мы не находимся в режиме создания приложения, то в зависимости от направления сдвигаем текст.

```
private void timer1_Tick(object sender, EventArgs e)
        {
            if (!DesignMode)
            {
                if (OrientationRun == RunOrientation.left)
                    TextRun = TextRun.Remove(0, 1) + TextRun[0];
                else
                    TextRun = TextRun[TextRun.Length - 1] + TextRun.Remove(TextRun.Length
- 1, 1);
                Invalidate();
            }

               }
```

9. Переопределите метод OnPaint (перерисовка элемента).

Зададим свойству Text элемента Label1 значение нашего поля TextRun, затем вызовем базовый метод OnPaint.

```
protected override void OnPaint(PaintEventArgs e)
        {
            label1.Text = TextRun;
            base.OnPaint(e);
               }
```

10. Скомпелируйте проект, в папке Debug сформируется файл с разрешением .dll. Это и будет наш компонент.

ИСПОЛЬЗОВАНИЕ ВИЗУАЛЬНОГО КОМПОНЕНТА

11. Создайте новый проект WindowsForms. Назовите проект Ticker.

12. Подключите ссылку на созданный файл dll, нажмите правой кнопкой мыши на проекте, выберете **Добавить → Добавить ссылку → Обзор**.

Добавьте элемент на панель, нажмите правой кнопкой мыши на панели элементов, выберете **Добавить → Выбрать элементы → Обзор**. Элемент отоброзится в вкладке компоненты.

13. Активизируйте форму и измените ее имя на Демонстрация пользовательского элемента. Укажите размер 600; 200. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

14. Добавьте на форму элемент GropBox (рис.7.8). В свойстве текст укажите Бегущая строка.

15. На элемент GroupBox перетащите созданный элемент labelRunText (рис.7.8). Укажите свойства согласно таб. 15.

Таблица 15.

Свойство	Значение
Font	Menuet script; 20pt
AutoSize	True
AutoSizeMode	GrowAndShrink
isEnebleRun	True
TextRun	Этот текст, показывающий возможность прокрутки!

16. Добавьте на форму TrackBar и две кнопки Влево и Вправо согласно рис.8.1.

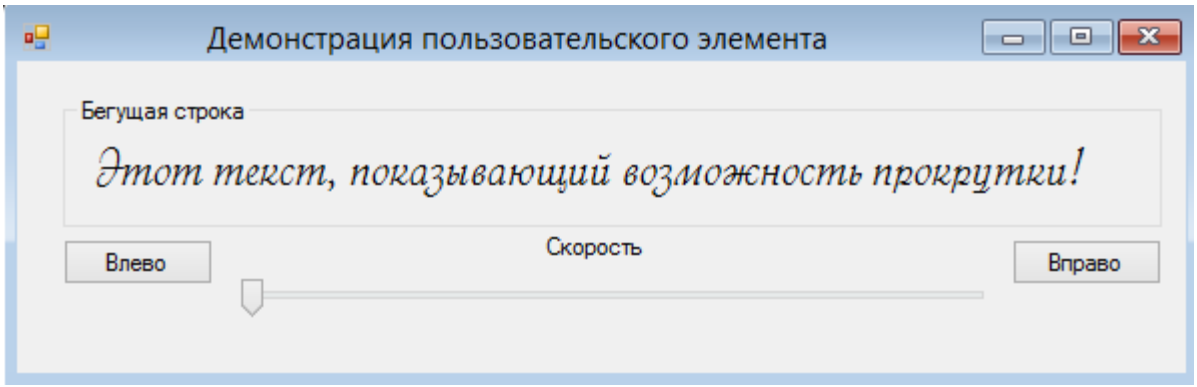


Рис.8.1. Элементы формы «Демонстрация пользовательского элемента»

17. Перейдите в редактор кода и добавьте обработчик событий Load для формы. Зададим начальную скорость.

```
private void Form1_Load(object sender, EventArgs e)
        {
            labelRunText1.Speed = 200;
               }
```

18. Добавьте обработчик событий для кнопки Влево.

```
private void button1_Click(object sender, EventArgs e)
        {
            labelRunText1.OrientationRun = LabelRunText.LabelRunText.RunOrientation.left;
               }
```

19. Добавьте обработчик событий для кнопки Вправо.

```
  private void button2_Click(object sender, EventArgs e)
        {
            labelRunText1.OrientationRun = LabelRunText.LabelRunText.RunOrientation.right;
               }
```

20. Добавьте обработчик событий для ползунка, в зависимости от его значения мы присеваем свойство Speed. Определим параметр, который вычисляется как обратное значение ползунка. При этом исключим присваиванию параметру Speed значение 0.

```
private void trackBar1_Scroll(object sender, EventArgs e)
        {
            if (trackBar1.Maximum != trackBar1.Value)
                labelRunText1.Speed = trackBar1.Maximum - trackBar1.Value;
              }
```

21. Сохраните проект и запустите программу для проверки работоспособности.

8.2. Не визуальный компонент

Постановка задачи: 1) Создать не визуальный компонент, который определяет треугольник по трем сторонам, по двум сторонам и углу между ними, либо по одной стороне и двум прилежащим углам. 2) Написать программу для использования собственного не визуального компонента «Треугольник», предусмотреть графическое отображение.

Данный не визуальный компонент способен вычислять все стороны и все углы треугольника, также здесь описан метод определяющий вырожденный (треугольник, все вершины которого лежат на одной прямой) треугольник.

1. Создайте проект Библиотека классов. Назовите проект Triangle.

2. Задайте поля стороны треугольника и углы между сторонами треугольника.

```
public class Triangle
    {
        //  Треугольник -это фигура, которая состоит из трёх точек, не лежащих на одной прямой, и трёх отрезков, попарно соединяющих эти точки.
        //  Точки называются вершинами треугольника, а отрезки - его сторонами.
        //  Сумма всех углов в треугольнике равна 180°.

        //  Стороны треугольника
        public double SideA { get; set; }
        public double SideB { get; set; }
        public double SideC { get; set; }
        //  Углы межде сторонами треугольника
        public double AngleA { get; set; }
        public double AngleB { get; set; }
        public double AngleC { get; set; }
```

3. Добавьте метод, который определяет, может ли существовать треугольник, имеющий такие длины сторон.

```
public bool isRegularTriangle(double SideA, double SideB, double SideC)
        {
            //  Воспользуемся свойством треугольника
            //   Любая сторона треугольника меньше суммы двух других сторон и больше их разности
            bool _isRegularTriangle =
                (SideA < (SideB + SideC)) &
                (SideA > (SideB - SideC)) &
                (SideB < (SideA + SideC)) &
                (SideB > (SideA - SideC)) &
                (SideC < (SideA + SideB)) &
                (SideC > (SideA - SideB));
            return _isRegularTriangle;
        }
```

4. Определите метод который создает треугольник по трем сторонам.

Примечание. При известных сторонах углы проще всего определить, пользуясь теоремой косинусов, частным случаем которой является теорема Пифагора. $c^2 = a^2 + b^2 - 2ab\cos\gamma$, откуда $\gamma = arccos\left(\frac{a^2+b^2-c^2}{2ab}\right)$

Методы создают треугольник, при этом определяются стороны и углы.

```
//  Создания треугольника по трем сторонам
        public void Create1(double SideA, double SideB, double SideC)
        {
            this.SideA = SideA;
            this.SideB = SideB;
            this.SideC = SideC;

            this.AngleC = Math.Acos((Math.Pow(this.SideB, 2) + Math.Pow(this.SideC, 2) -
Math.Pow(this.SideA, 2)) / (2 * this.SideB * this.SideC)) / (Math.PI / 180);
            this.AngleA = Math.Acos((Math.Pow(this.SideA, 2) + Math.Pow(this.SideC, 2) -
Math.Pow(this.SideB, 2)) / (2 * this.SideA * this.SideC)) / (Math.PI / 180);
            this.AngleB = Math.Acos((Math.Pow(this.SideA, 2) + Math.Pow(this.SideB, 2) -
Math.Pow(this.SideC, 2)) / (2 * this.SideA * this.SideB)) / (Math.PI / 180);
        }
        //  Создание треугольника по двум сторонам и углу между ними
        public void Create2(double SideA, double SideC, double AngleA)
        {
            this.SideA = SideA;
            this.SideC = SideC;
            this.AngleA = AngleA;

            this.SideB = Math.Sqrt(Math.Pow(this.SideA, 2) + Math.Pow(this.SideC, 2) - 2 *
this.SideA * this.SideC * Math.Cos(this.AngleA * (Math.PI / 180)));
            this.AngleC = Math.Acos((Math.Pow(this.SideB, 2) + Math.Pow(this.SideC, 2) -
Math.Pow(this.SideA, 2)) / (2 * this.SideB * this.SideC)) / (Math.PI / 180);
            this.AngleB = Math.Acos((Math.Pow(this.SideA, 2) + Math.Pow(this.SideB, 2) -
Math.Pow(this.SideC, 2)) / (2 * this.SideA * this.SideB)) / (Math.PI / 180);
              }

  //  Создание треугольника по стороне и прилегающим углам
        public void Create3(double SideA, double AngleA, double AngleB)
        {
            this.SideA = SideA;
            this.AngleA = AngleA;
            this.AngleB = AngleB;

            this.AngleC = 180 - this.AngleA - this.AngleB;
            this.SideC  =  this.SideA  *  Math.Sin(this.AngleC  *  (Math.PI  /  180))  /
Math.Sin(this.AngleA * (Math.PI / 180));
            this.SideB  =  this.SideA  *  Math.Sin(this.AngleB  *  (Math.PI  /  180))  /
Math.Sin(this.AngleA * (Math.PI / 180));
              }
```

5. Скомпелируйте проект, в папке Debug сформируется файл с разрешением .dll. Это и будет наш компонент.

ИСПОЛЬЗОВАНИЕ НЕВИЗУАЛЬНОГО КОМПОНЕНТА

Создадим проект для тестирования нашего компонента, который бы строил треугольник по трем сторонам, треугольник для наглядности будем рисовать.

6. Создайте новый проект WindowsForms. Назовите проект TriangleTest.

7. Подключите ссылку на созданный файл dll, нажмите правой кнопкой мыши на проекте, выберете **Добавить → Добавить ссылку → Обзор**.

8. Активизируйте форму и измените ее имя на Треугольник. Укажите размер 650; 350. Измените стандартное значение свойства формы **AutoScaleMode** – None на Font.

9. Добавьте элементы на форму согласно рис. 8.2.

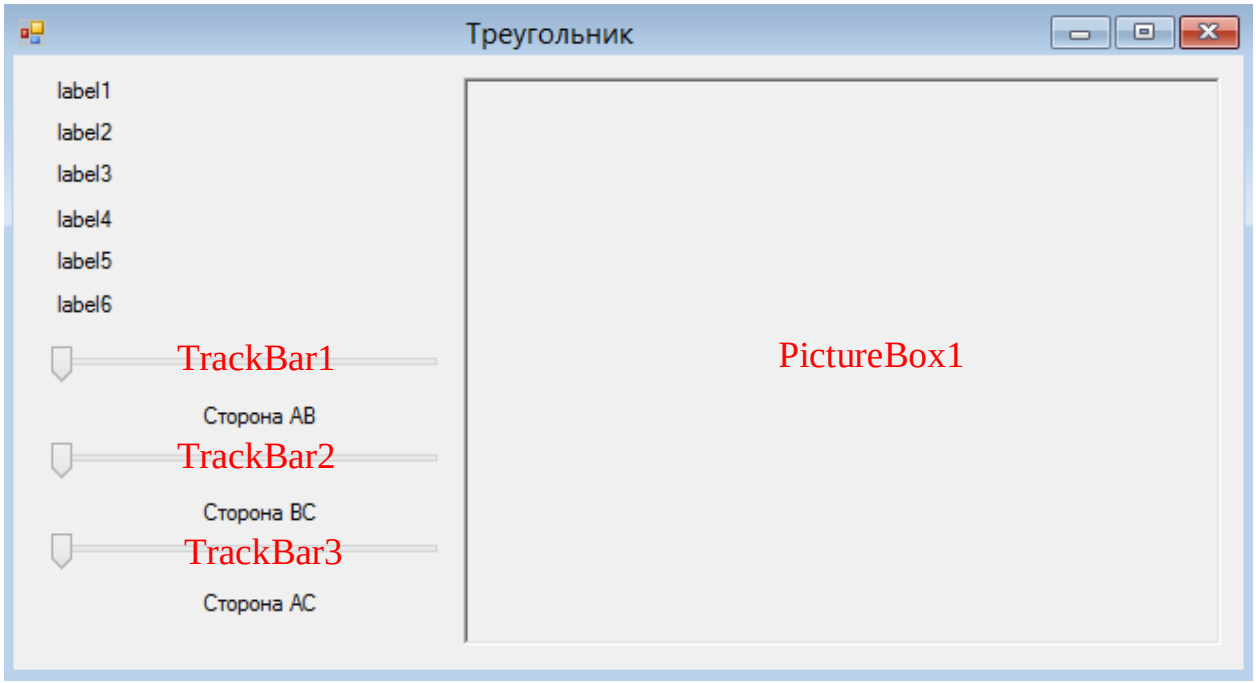


Рис.8.2. Элементы формы «Треугольник»

10. Перейдите в редактор кода и определите объект класса Triangle.

```
public partial class Form1 : Form
    {
        //  Определим наш треугольник
        Triangle.Triangle tr = new Triangle.Triangle();
               public Form1()
```

11. При инициализации формы, задайте первоначальное значение треугольника, который однозначно существует. Вызовем метод PrintText.

```
public Form1()
        {
            InitializeComponent();
            //  Зададим первоначальные значения треугольника
            trackBar1.Value = 30;
            trackBar2.Value = 40;
            trackBar3.Value = 50;
            //  Зададим первоначальные значения наших бегунков
            tr.Create1(trackBar1.Value, trackBar2.Value, trackBar3.Value);
            PrintText();
               }
```

12. Определите метод PrintText, который выводит на экран значение сторон углов треугольника и рисует его в PictureBox.

```
private void PrintText()
        {
            //  Выведем информацию по сторонам и углам
            lbSideA.Text = "Сторона AB: " + Math.Round(tr.SideA, 2).ToString();
            lbSideB.Text = "Сторона BC: " + Math.Round(tr.SideB, 2).ToString();
            lbSideC.Text = "Сторона AC (основание): " + Math.Round(tr.SideC, 2).ToString();
            lbAngleA.Text = "Угол альфа: " + Math.Round(tr.AngleA, 2).ToString();
            lbAngleB.Text = "Угол бетта: " + Math.Round(tr.AngleB, 2).ToString();
            lbAngleC.Text = "Угол гамма: " + Math.Round(tr.AngleC, 2).ToString();
            //   Создадим карандаш
            Pen blackPen = new Pen(Color.Black, 3);
            Pen redPen = new Pen(Color.Red, 2);
            // Массив точек треугольника.
            Point[] points = new Point[3];
            //  Первые две точки известны: (0;0) и (длина стороны основания;0)
            //  Сдвинем наш треугольник вниз, чтобы он был в центре
            //  Прибавим ко всем координатам по X - 100, по Y - 250
            points[0].X = 100; points[0].Y = 250;
            points[1].X = (int)(100 + tr.SideC); points[1].Y = 250;
            // вычисление третьей точки
            points[2].Y =(int)((tr.SideA * tr.SideA + tr.SideC * tr.SideC - tr.SideB *
tr.SideB) / (2 * tr.SideA));
            points[2].X = (int)Math.Sqrt(tr.SideC * tr.SideC - points[2].Y * points[2].Y);
            //  Вычислим полупериметр треугольника
            double p = (tr.SideA + tr.SideB + tr.SideC) / 2;
            //  Вычислим площадь треугольника
            double S = Math.Sqrt(p * (p - tr.SideA) * (p - tr.SideB) * (p - tr.SideC));
            //  Вычислим длину высоты, опущенной к основанию -это и будет координата по Y
            double y = 2*S / (tr.SideC);
            //  Вычислим координату по X через катеты и гиппотенузу
            double x = Math.Sqrt(tr.SideA * tr.SideA - y * y);
            //  Зададим вычисленные координаты в точку
            points[2].X = 100+(int)x;
            points[2].Y = 250-(int)y;

            //  Нарисуем наш треугольник по трем точкам
            Bitmap bmp = new Bitmap(pictureBox1.Width, pictureBox1.Height);
            using (Graphics grfx = Graphics.FromImage(bmp))
            {
                grfx.Clear(Color.White);
                //  Рисуем по точкам треугольник
                grfx.DrawPolygon(blackPen, points);
                //  Закрасим его
                grfx.FillPolygon(Brushes.Tan, points);
                // Нарисуем для наглядности высоту
                grfx.DrawLine(redPen, points[2], new Point(points[2].X, 250));
                //  Подпишем наши точки
                Font fontAxis = new Font("Segoe UI", 8);
                SolidBrush sbAxis = new SolidBrush(Color.DodgerBlue);
                grfx.DrawString("A", fontAxis, sbAxis, points[0].X - 10, points[0].Y + 5);
                grfx.DrawString("C", fontAxis, sbAxis, points[1].X, points[1].Y + 5);
                grfx.DrawString("B", fontAxis, sbAxis, points[2].X - 5, points[2].Y - 15);
            }

            // Устанавливаем изображение.
            pictureBox1.Image = bmp;
               }
```

13. Добавьте обработчик событий для TrackBar1.

Один обработчик событий для всех трех элементов. Если треугольник не вырожденный пересоздадим его и нарисуем.

```
private void trackBar1_Scroll(object sender, EventArgs e)
        {
            //  Проверим,существет ли треугольник с такими сторонами
            if (tr.isRegularTriangle(trackBar1.Value, trackBar2.Value, trackBar3.Value))
            {
                //  Если существует,зададим треугольник и перерисуем его
                tr.Create1(trackBar1.Value, trackBar2.Value, trackBar3.Value);
                PrintText();
            }
               }
```

14. Сохраните проект и запустите программу для проверки работоспособности. Результат показан на рис.8.3.

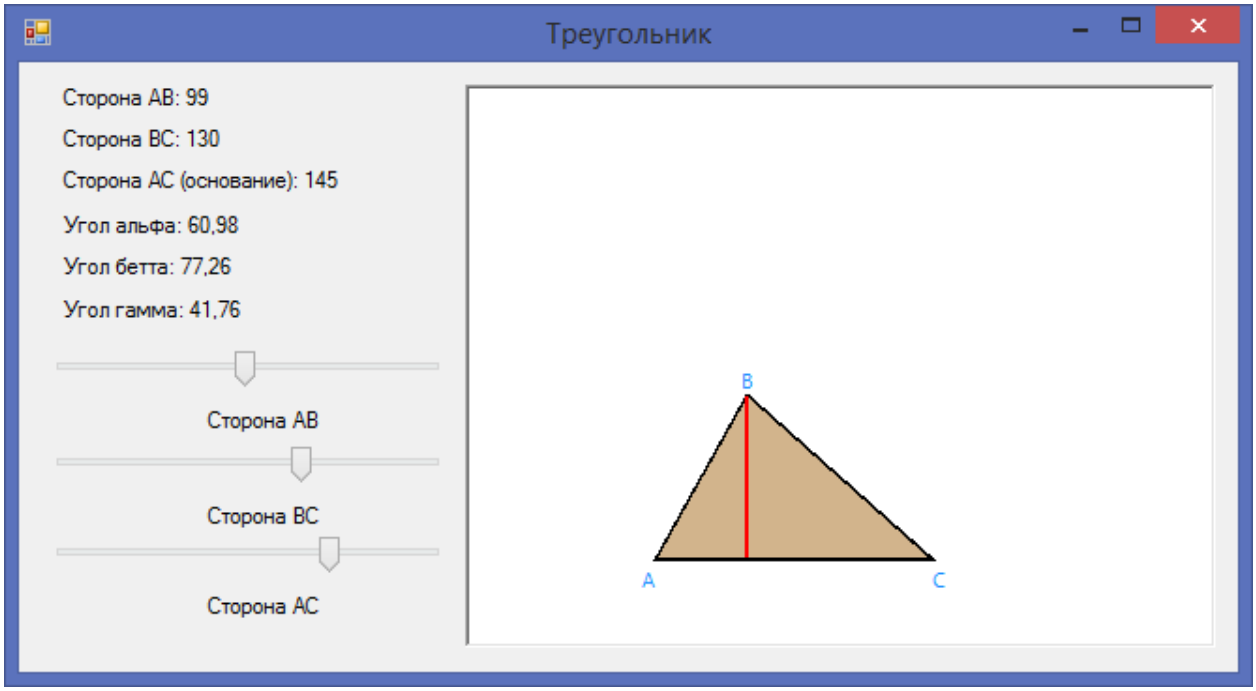


Рис.8.3. Треугольник

,

8.3. Задачи для самостоятельной работы

1. Разработайте компонент, основанный на элементе TextBox, который будет конвертировать текущее содержимое поля ввода в заглавные буквы и возвращать количество вхождений заданного символа в поле ввода.

2. Разработайте компонент, основанный на элементе ProgressBar, у которого на полосе отображался бы процент заливки.

3. Разработайте компонент, основанный на элементе Label, который бы с определенной скоростью изменял цвет с одного на другой. Цвета и скорость, настраиваемые параметры.

4. Разработайте компонент, основанный на элементе Label, который умеет, при наведении на него мышью, прокручивать текст.

5. Разработайте компонент, основанный на элементе Button, представляющий из себя круглую кнопку

6. Разработайте компонент, основанный на элементе Button: Очень смешная кнопка - которая ускользает от мыши.

7. Разработайте компонент, основанный на элементе Button, который изменяет свойства шрифта, когда мышь перемещается над ним.

8. Разработайте компонент, основанный на элементе TextBox, который гарантирует, что свойство text содержит действительное число. Определяемые пользователем свойства: длина целой части, количество десятичных знаков, символ, который представляет десятичную точку.

9. Разработайте компонент, основанный на элементе TextBox, для ввода и редактирования IP-адреса.

10. Разработайте компонент, основанный на элементе TextBox, который проверяет правильность ввода Email, если введенная строка не соответствует адресу электронной почты, то выдается сообщение пользователю, иначе ничего не происходит.

11. Разработайте компонент, основанный на элементе TextBox, который расширяет возможности стандартного компонента добавлением поддержки выбора файла.

12. Разработайте компонент: Графический секундомер, который будет показывать часы, минуты, секунды.

13. Разработайте компонент: Простой калькулятор, работающий только с целыми числами. Кнопки с цифрами и операциями (+-*/). При делении дробную часть не учитывать.

14. Разработайте компонент, показывающий модальное окно "О программе". Можно задавать заголовок, текст и ссылку на произвольный сайт.

15. Дополните не визульную библиотеку по работе с треугольниками методами вычисления периметра, площади и вычисления всех точек вершин треугольника.

16. Разработайте не визульную библиотеку по работе с четырёхугольниками. Библиотека должна содержать функции задания фигуры, вычисления площади, периметра, длин диагоналей и вычисления точек вершин.

17. Разработайте не визульную библиотеку, которая умеет вычислять периметр, площадь и объем основных геометрических фигур. Предусмотреть, что для одной и той же фигуры могут быть реализованы различные методы вычисления периметра, площади или объема, в зависимости от входных значений.

18. Разработайте не визульную библиотеку по работе с комплексными числами. Библиотека должна содержать функции задания комплексного числа и действий над числами (+-*/).

19. Разработайте не визульную библиотеку по работе с шестнадцатеричными числами. Библиотека должна содержать функции задания шестнадцатеричного числа и действий над числами (+-*/).

20. Реализовать родительский класс «операции над двумя числами» и дочерний класс «операции над 3 числами» в C# без создания компонента.

21. Разработайте не визульную библиотеку для поиска файлов и папок в директориях и поддиректориях, используя маску (*.*, *.exe, ...). Можно определить критерии поиска: поиск в поддиректориях, поиск readonly-файлов, поиск скрытых файлов, поиск системных файлов и т.д.

22. Разработайте не визульную библиотеку для работы с сетью, позволяющую узнать имя машины, IP адрес, MAC адрес;

23. Разработайте не визульную библиотеку, которая генерирует случайный текст путем запроса к сайту fish-text.ru. Можно настраивать тип текста (предложения или параграф), количество, необходимость конвертации из html в string.

24. Разработайте не визульную библиотеку, которая может генерировать строки символов, подходящие для использования в качестве паролей. Пользователь может установить свойства, чтобы определить длину паролей и какие символы они должны содержать.

25. Разработайте не визульную библиотеку для работы с логированием. Умеет создавать, удалять лог файлы. Имеет метод добавления записи в лог-файл. Пользователь устанавливает свойства, чтобы добавлять в строку логирования текущую дату, время, тип записи (info, error, alert) и дополнительное поле. Предусмотреть наличие методов статистики - количество записей, период, количество записей с определенным типом.

26. Разработайте не визульную библиотеку для работы с массивами данных. Данные представляют целые числа. Пользователь задет размер массива, интервал, в котором лежат данные. Библиотека должна уметь заполнять массив случайными данными, сортировать (3 способа), поиск данных, замена. Вычислять максимальное, минимальное и среднее значение.

ПРИЛОЖЕНИЕ № 2

Разработка контента сайта в поддержку учебного курса

В поддержку курса разработан сайт «Объектно-ориентированное и визуальное программирование на C#» для обучающихся среднего профессионального обучения на платформе WordPress. На сайте размещены материалы для практической работы обучающихся (рис.4.1).

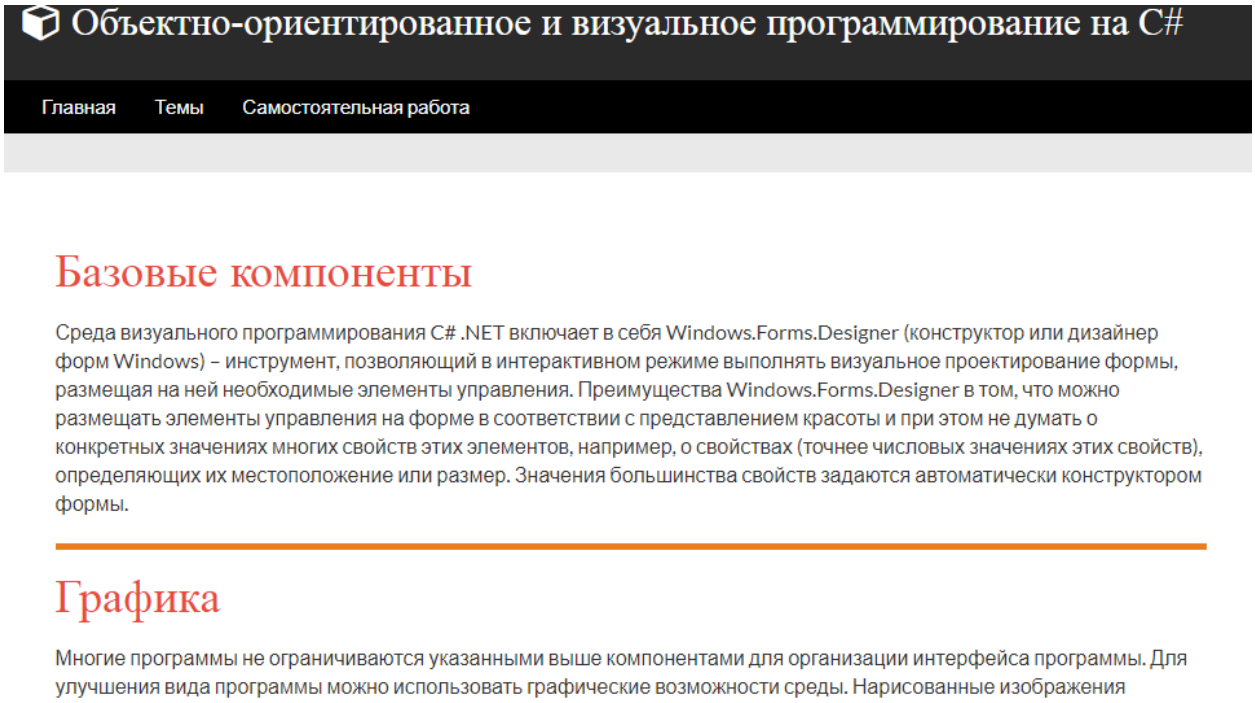


Рис.4.1. Сайт «Объектно-ориентированное и визуальное программирование на C#»

Сайт для поддержки дисциплины «Объектно-ориентированное и визуальное программирование» состоит из страниц:

- Главная
- Темы
    - Базовые компоненты
    - Графика
    - Файловая система и исключения
    - Базы данных
    - Мультимедиа
    - COM-объекты
    - Сетевые компоненты
    - Собственные компоненты

- Самостоятельная работа
    - Базовые компоненты
    - Графика
    - Файловая система и исключения
    - Базы данных
    - Мультимедиа
    - COM-объекты
    - Сетевые компоненты
    - Собственные компоненты

Важная стадия работы над сайтом – разработка интерфейса. Организация интерфейса – очень важное потребительское свойство продукта учебного назначения. Так как с курсом работает обучающийся, а число сеансов работы обычно относительно невелико, так что особое значение приобретает быстрота и легкость освоения управления сайта. Сайт, созданный с помощью системы WordPress, отличается простотой в управлении, обучающиеся с легкостью разберутся в расположение объектов на сайте.

ПРИЛОЖЕНИЕ № 3

Диск с материалами работы