Лабораторная работа №8

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И МЕХАТРОНИКИ

Кафедра информационных технологий и мехатроники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по проведению лабораторных работ по дисциплине
«Объектно-ориентированное программирование»
для студентов специальности 6.050101 “Компьютерные науки”

Разработчик - доцент кафедры информационных технологий и мехатроники
кандидат технических наук, старший научный сотрудник
Тимонин Владимир Алексеевич

Харків 2015

Лабораторная работа №8
Исследование возможностей интегрированной среды разработки Visual C# для создания простых приложений ОС Windows.

Цель работы – исследовать возможности интегрированной среды разработки Visual C# и получить практические навыки по созданию простых приложений ОС Windows.

1. Теоретические сведения

Интегрированная Среда Разработки Visual C# (Integrated Development Environment - IDE, в дальнейшем будем использовать для нее аббревиатуру ИСР) — это среда, позволяющая облегчение процесса создания программ и содержащая все необходимое для проектирования, запуска и тестирования при ложений (редактор кодов, отладчик, инструментальные панели, редактор изображений, инструментарий баз данных и др.). ИСР предоставляет возможность расширять меню, включая в него необходимые дополни тельные программы, в том числе и собственные.

1.1. Структура главного окна ИСР Visual Studio C#

Чтобы начать работу в Microsoft Visual C#, надо запустить Microsoft Studio - сделать щелчок на кнопке **Пуск** и в меню **Все програм мы→Microsoft Visual Studio 2010** выбрать **Microsoft Visual Studio 2010.** На экране появится ос новное окно ИРС, представляющее собой совокупность нескольких окон (рис.1).

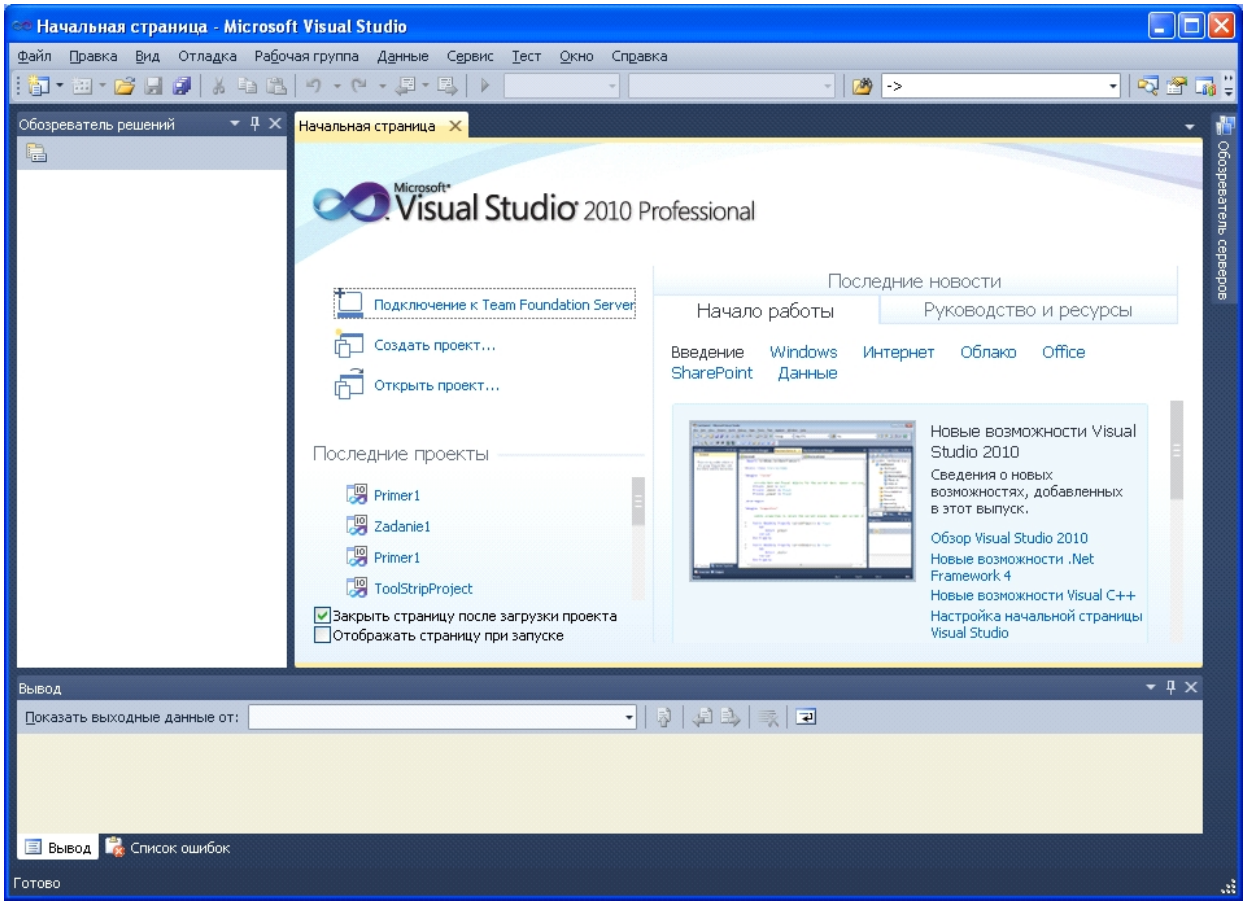

Рис. 1. Вид главного окна ИРС после загрузки Visual C#

Самое верхнее окно имеет заголовок **Microsoft Visual Studio**, которое отражает название среды разработки и имя нового решения, из которого будет получена работающая программа. На строке заголовка проекта находятся кнопки свертывания, восстановления и закрытия окна.

Под заголовком размещается строка главного меню, которая предоставляет доступ ко всем функциям и командам среды разработки. Ее состав частич но зависит от варианта Visual C#. При вызове этих команд (**Файл, Правка, …**) открываются так называемые «выпадающие меню», представляющие собой набор команд.

Справа от названия элемента меню может находить ся комбинация клавиш быстрого доступа для вывода элемента меню, используя клавиатуру. В зависимости от ситуации в среде некоторые пункты меню недоступны для использования (отображаются светло-серым цветом). Напри мер, нельзя воспользоваться командой сохранения файла, если нет открытых файлов.

Главное меню состоит из следующих подменю:

Файл - позволяет создать новое решение, от крыть ранее созданное решение, сохранить решения проекты или формы в фай лах с заданными именами;

Правка - позволяет выполнять обычные для приложений операции обмена с буфером Clipboard, осуществлять поиск и контекстные замены в коде приложения, которые свойственны большинству известных тексто вых редакторов;

Вид - позволяет вызывать на экран различные окна, необходимые для проектирования;

Проект - позволяет добавлять и убирать из проекта элементы, задавать опции проекта;

Построение - позволяет работать с решениями и проектами (создание, модификация, удаление и др.);

Отладка – дает возможность выполнять проект в нормальном или отладочном режимах, продвигаясь по шагам, останавливаясь в указанных точках кода, просматривая значения переменных и т.д.;

Рабочая группа – позволяет работать с группами проектов;

Данные – позволяет использовать инструментарий для работы с базами данных;

Формат - позволяет выравнивать группы размещенных на форме компонентов по размерам и местоположению;

Сервис – включает ряд подразделов, позволяющих настраи вать ИСР и выполнять различные вспомогательные программы, работать с программами, конфигури рующими базы данных и т.д. Кроме того, в это меню можно сами включить любые разделы, вызывающие те или иные приложения, и таким образом расши рить возможности главного меню Visual C#, приспособив его для своих задач;

Тест – позволяет работать с тестовыми проектами;

Окно - по зволяют ориентироваться среди массы окон, обычно одновременно открытых в процессе проектирования и переключаться в нужное окно;

Справка - содержит разделы, помогающие работать со встроенной в Visual C# справочной системой.

Ниже строки главного меню размещаются инструментальные панели, содержащие кнопки быстрого вызова, дублирующие некоторые наиболее часто используемые ко манды меню. Все эти кнопки имеют всплывающие подсказки (при наведении курсора мыши на кнопку появляется подсказка о том, для чего предназначена кнопка). Рядом с такими кнопками могут быть дополнительные кнопки для раскрытия списка значений основной кнопки. Так как все кнопки не помещаются в отведенное им место на рабочем столе, то они свернуты в небольшие полосы с кнопками их развертывания.

Для того чтобы приступить к созданию программы в Visual C# или, как принято говорить, начать работу над проектом, надо:

1. В меню **Файл** окна **Microsoft Visual Studio** выбрать команду **Файл→Создать→Проект** или на начальной странице выбрать пункт «**Создать проект**» или с помощью кнопки быстрого вызова «**Создать проект**» (первая слева в строке, расположенной ниже строки главного меню).

2. В открывшемся окне **Создать проект** (рис. 2) из списка предопределенных шаблонов выбрать раздел «Другие языки», раскрыть список **Visual C#** и выбрать тип приложения — **Windows**.

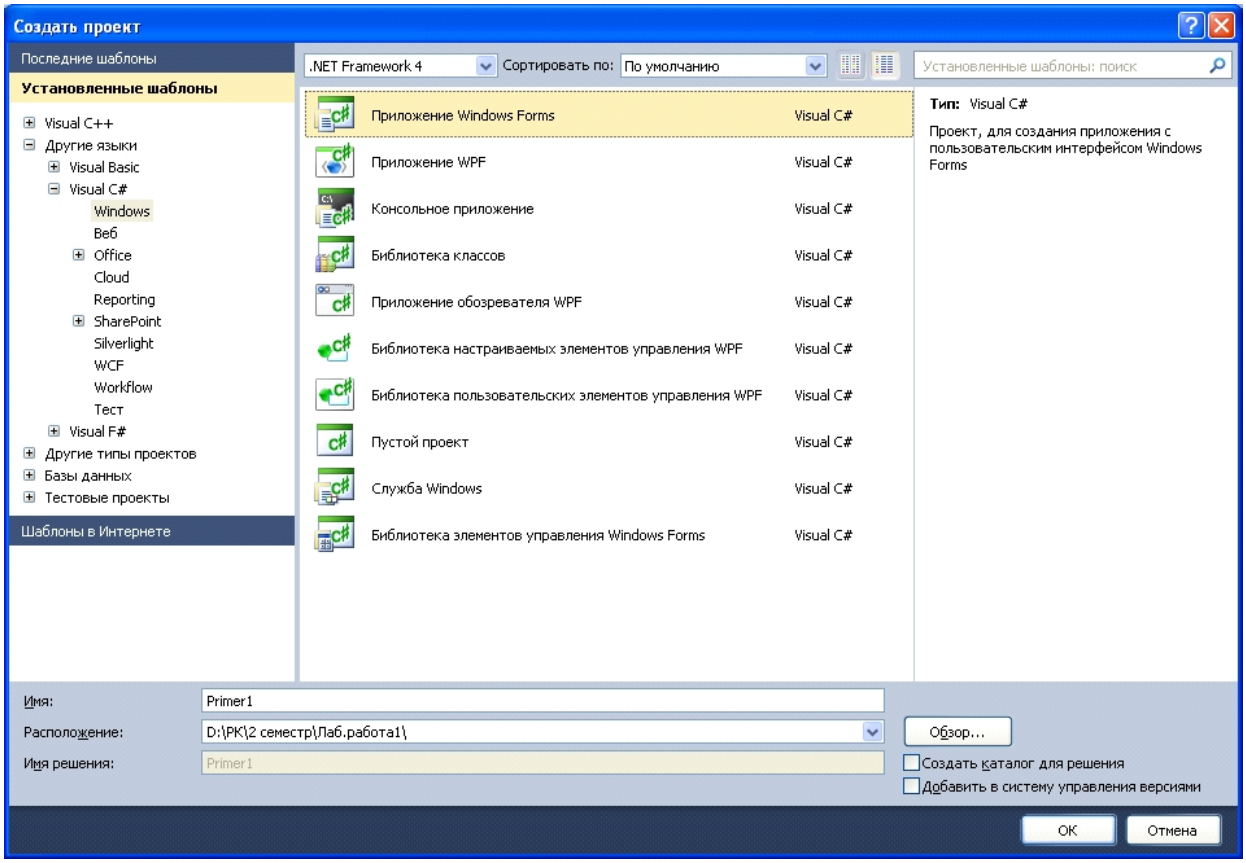

Рис. 2. Вид окна «Создать проект»

3. Выбрать вид приложения — **Приложение Windows Forms Visual C#.**

4. В поле **Имя** ввести имя проекта, в поле **Расположение** ввести папку размещения проекта (по умолчанию это C:\documents and settings\User\Мои документы\Visual Studio 2010\Projects\, гдеUser — имя пользователя в систе ме) и нажать кнопку **ОК.**

В результате описанных действий будет создан проект Visual Studio — сово купность файлов, необходимых для создания программы (установленный флажок **Создать каталог для решения** сбросить). Имя папки проекта определяет содержимое поля **Имя** (имя проекта).

Окно среды Microsoft Visual Studio 2010 в начале работы над новым проектом приведено на рис. 3. В заголовке окна отображается имя проекта, над кото рым в данный момент идет работа.

В верхней части окна находится строка меню и область отображения панелей инструментов. По умолчанию в области отображения панелей инструментов выводится панель **Стандартная**. Чтобы сделать доступными другие па нели инструментов, надо выбрать команду **Сервис→Настройка** и в раскрывшем ся списке сделать щелчок на имени нужной панели (например, на рис. 3 показана дополнительно панель «**Макет**»).

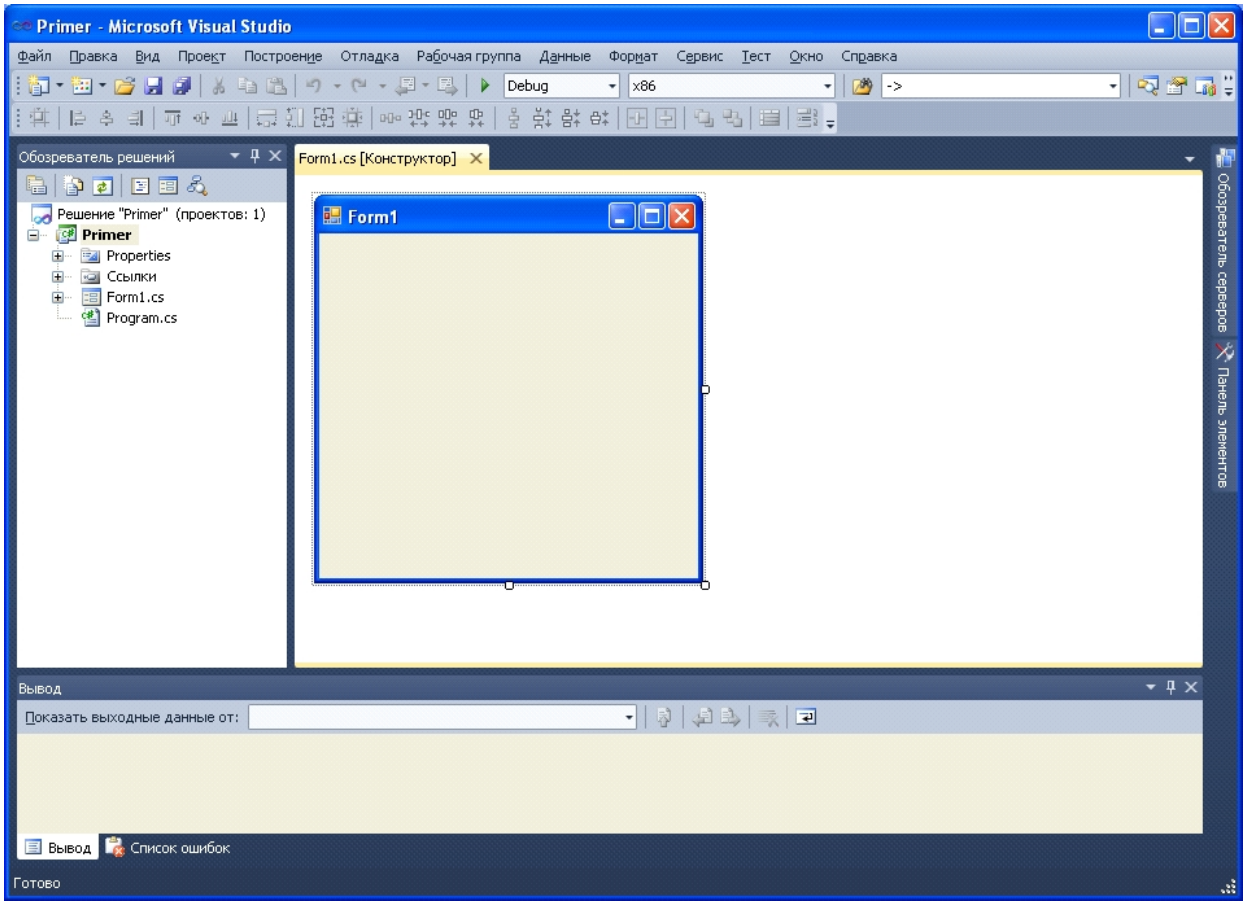

Рис. 3. Вид окна «Primer - Microsoft Visual Studio»

Visual Studio предоставляет множество окон, которые отображают информацию, необходимую для создания приложений. Некоторые окна, такие как **Обозреватель решений** отображаются по умолча нию, другие, например отладочные появляются при запуске приложения в режиме отладки Полный список доступных окон расположен в главном меню под опцией **Вид** (рис. 4).

В целом, интегрированная среда разработки содержит два типа окон: окна инструментов и окна документов. Окна инструментов и документов можно упорядочить перетаскиванием, коман дами меню **Окно** или командами контекстного меню (щелкнув правой кнопкой мыши заголовок перемещаемого окна). Панели инструментов можно перетаскивать или упорядочивать с помощью диалогового окна **Настроить**, доступ к которому можно получить через команду меню **Сервис→Настройка.**

Каждое окно — это обычное Windows-окно, имеющее стандартную заголовочную полосу в своей верхней час ти. За эту полосу можно окно перемещать, протягивая мышью. У окон име ются свойства, которые открываются, если на заголовочной части щелкнуть правой кнопкой мыши (перечень свойств окна **Панель элементов** показан на рис. 5):

- **Плавающая область** - такое окно можно перетягивать в любую часть рабо чего стола;
- **Закрепить –** окно может перемещаться, но будет захватываться другим окном;
- **Закрепить как вкладку** – отображается в качестве вкладки в основное окно рабочего стола. Это окно, в котором первоначально в качестве вкладки располагается **Начальная страница**;
- **Автоматически скрывать –** окно автоматически "прячется" в качестве вкладки к ближайшей боковой сто роне основного окна рабочего стола, а при наведении курсора мыши на имя окна оно автоматически всплывает;
- **Скрыть -** окно исчезает с экрана. Чтобы оно снова появилось, надо воспользоваться опцией **Вид** главного меню.

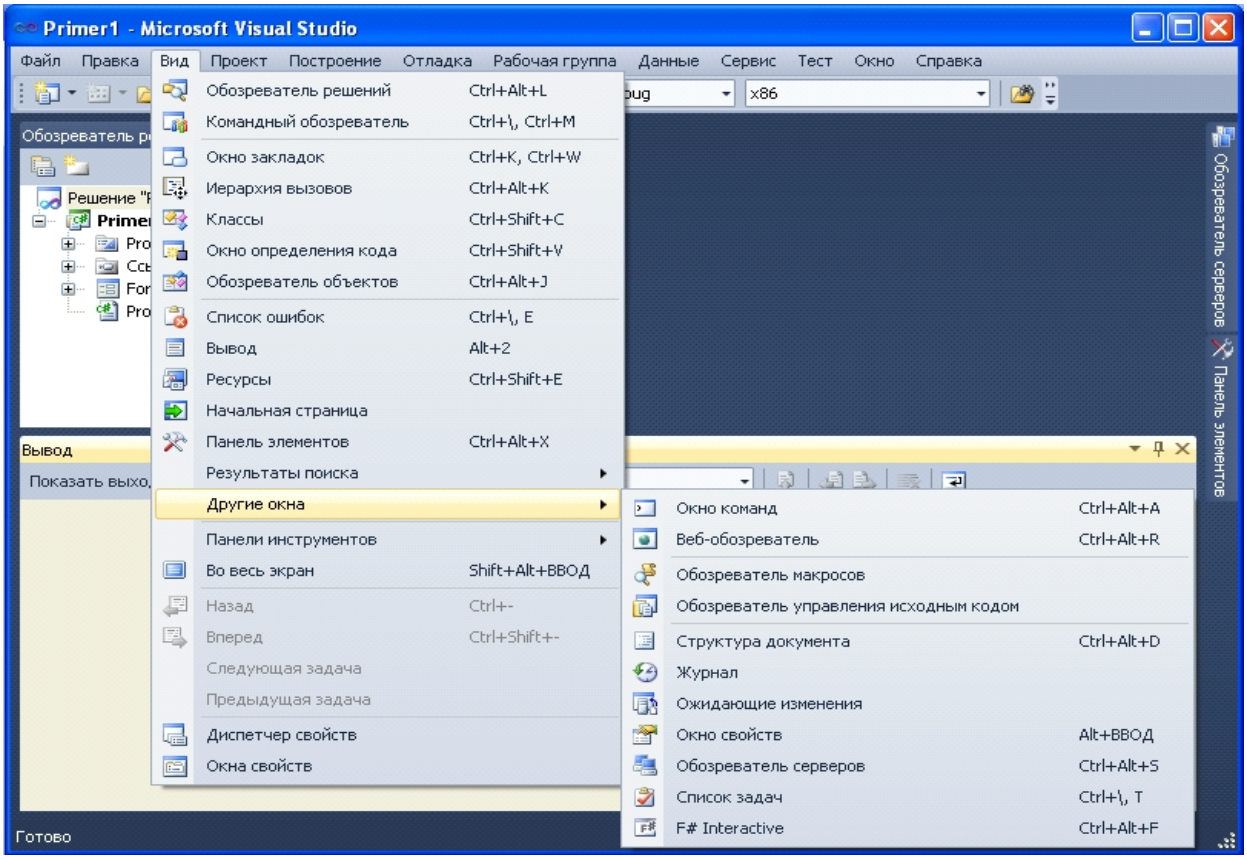

Рис. 4. Список окон ИСР Visual Studio

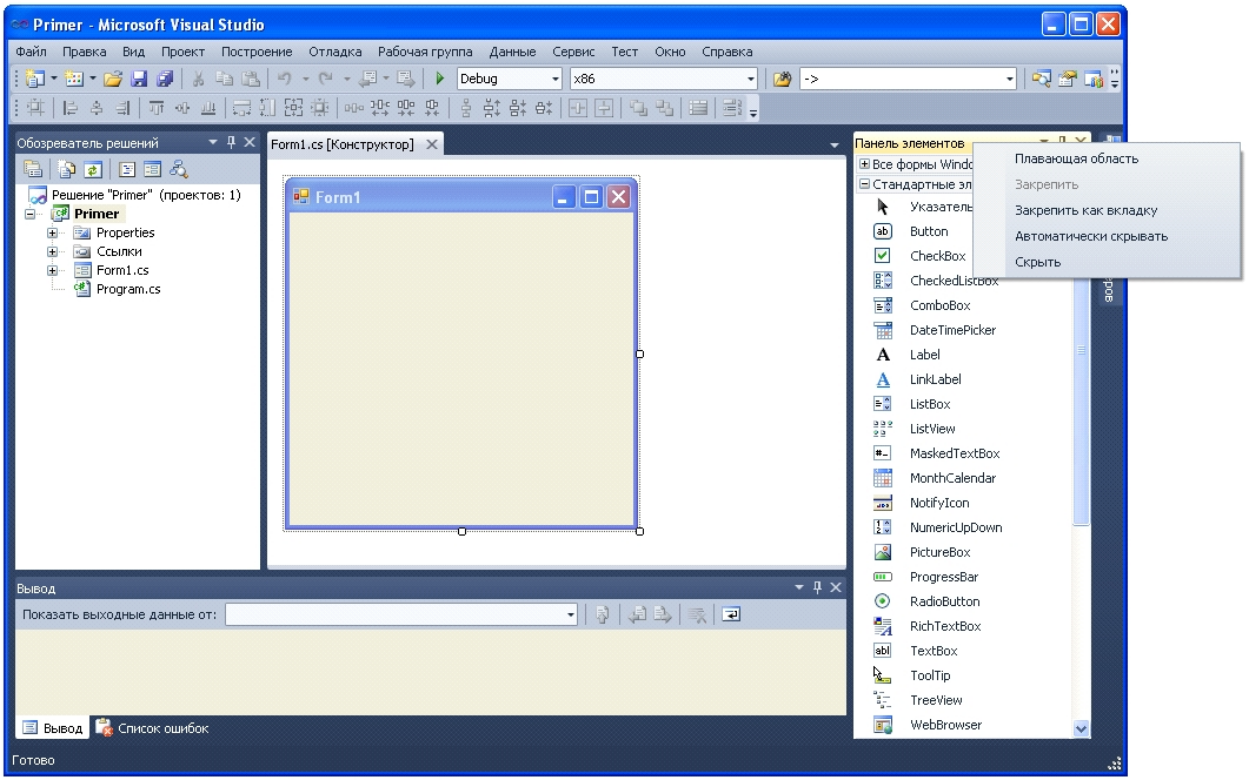

Рис. 5. Контекстное меню свойств окна

Чтобы манипулировать положением окон на рабочем столе, надо хорошо по нимать все свойства, иначе может сложиться ситуация, когда на рабочем сто ле соберется множество окон, которые просто будут мешать работать. В та ком случае некоторые окна надо будет "разогнать" по

боковым сторонам основного окна, а другие просто спрятать.

Например, для установки окна **Панель элементов** или **Обозреватель серверов** справа-сбоку основного окна рабочего стола необходимо установить у этих окон свойство (через правую кнопку мыши) **Закрепить**. Тогда в заглавной строке окна появится значок **Автоматически скрывать**. Значок имеет вид вертикальной кнопки. Если на этом значке щелкнуть, то окно причалит к правой стороне главного окна среды и спрячется (останется видимым только его имя). Если теперь навести курсор мыши на это имя, то окно автоматически всплывет, и в его заголовочной части мы увидим значок **Автоматически скрывать** в виде горизонтальной кнопки.

Любое окно инструментов или окно документов может быть отстыковано от интегрированной среды разработки и помещено в любое место на рабочем столе. Ес ли два связанных окна документов отображаются одновременно, при изменении содержимого в любом из них обновляются оба этих окна.

Окна инструментов можно прикрепить к одной из сторон фрейма интерфейса ИРС. При перетаскивании окна инструментов в новое расположение в интегриро ванной среде разработки появляется маркер в виде ромба. Четыре стрелки ром ба указывают на четыре стороны панели редактирования. Если окно является ок ном инструментов, то дополнительные четыре стрелки указывают на углы окна ИРС. Маркер помогает закре пить окно инструментов на одной из четырех сторон интегрированной среды раз работки или во фрейме редактирования.

При пере таскивании окна инструментов или документов к центру интегрированной среды разработки появляются специальные маркеры (рис. 6). Когда перетаскиваемое окно достигнет нужного расположения, наведите указа тель на соответствующую часть ромба-маркера (указанная область будет отобра жена затемненной). Чтобы закрепить окно, отпустите кнопку мыши, и оно останется прикрепленным в выбранном месте среды разработки.

Например, если обозреватель решений закреплен на правой стороне среды раз работки, а вы хотите закрепить его на левой стороне, перетащите обозреватель ре шений в центр среды разработки, наведите указатель на самую левую стрелку ром ба и отпустите кнопку мыши.

Кроме того, окно инструментов можно прикрепить к боковой, верхней или ниж ней части окна среды разработки, перетащив его в сторону до появления второго маркера в форме ромба. Щелкните одну из четырех стрелок, чтобы закрепить окно рядом с данной частью боковой стороны окна.

Чтобы переместить закрепляемое окно без его привязки к какому-либо месту, нужно при его перетаскивании держать нажатой клавишу **<Ctrl>**.

Все окна инструментов, названия которых встречаются в меню **Вид**, поддержи вают возможность автоматического скрытия (значок канцелярской кнопки в заго ловке окна). Автоматическое скрытие сдвигает окно в сторону, когда активно дру гое окно. Когда окно скрыто, его имя и значок отображаются на вкладке на краю интегрированной среды разработки. Чтобы снова сделать окно активным, наведите указатель на вкладку, и оно вернется в поле зрения.

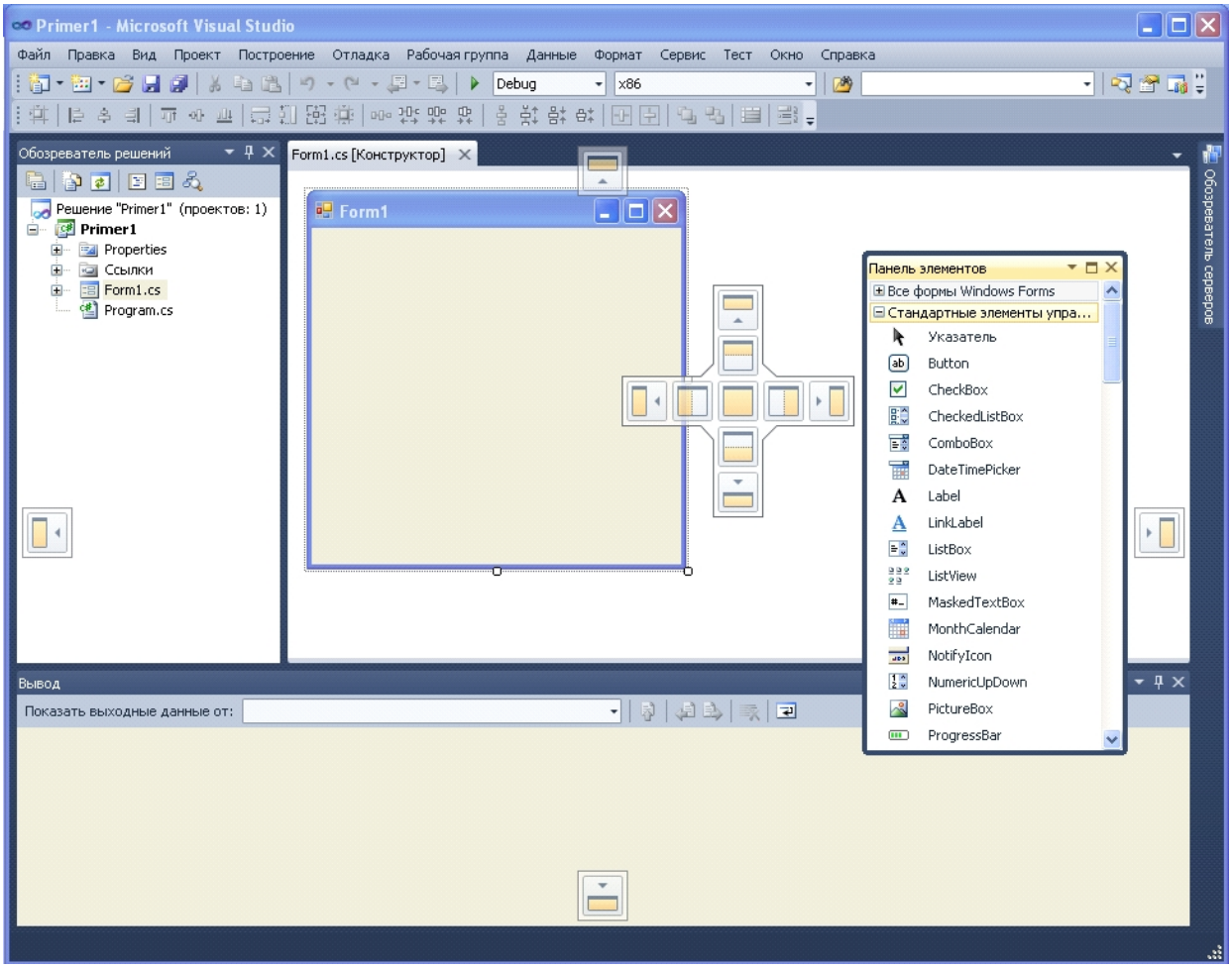

Рис. 6. Отображение специальных маркеров при перетаскивании окон

Центральную часть окна Visual C# занимает окно проектирования формы (отображена закладка **Form1.cs[Конструктор]**) (рис. 3). В нем находится форма — заготовка окна приложения. Форма — это главный контейнер, в котором размещаются компоненты самой среды. С помощью этих компонентов и реализуется конкретный алгоритм определенной задачи.

Когда форма появится на экране, в нее в соответствии с имеющимся алгоритмом задачи помещают необходимые компоненты из палитры (т. е. из набора компонентов среды), придают свойствам компонентов необходимые значения и определяют реакции на события компонентов. Реакции задаются в программах, которые называются обработчиками событий. Все программы-обработчики событий компонентов, расположенных в данной форме, помещаются в тот же программный модуль, который создается вместе с появлением формы на экране.

Поскольку по правилам проектирования в форму надо будет помещать компоненты, которые расположены в окне **Панель элементов**, то может потребоваться изменение размеров формы. Это делается протяжкой формы за анкерные точки — небольшие квадратики, выделенные по углам и сторонам формы.

Чтобы увидеть программный модуль, надо открыть контекстное меню формы и выполнить в нем команду **Перейти к коду**. При этом в верхней части окна проектирования, где находится форма, и которое уже содержит вкладку **Form1.cs[Конструктор],** появится новая вкладка **Form1.cs.** Если открыть эту вкладку, то можно увидеть, что экземпляр класса **Form**, который виден на рабочем столе, действительно является наследником **Form** и получил имя **Form1** (рис. 7).

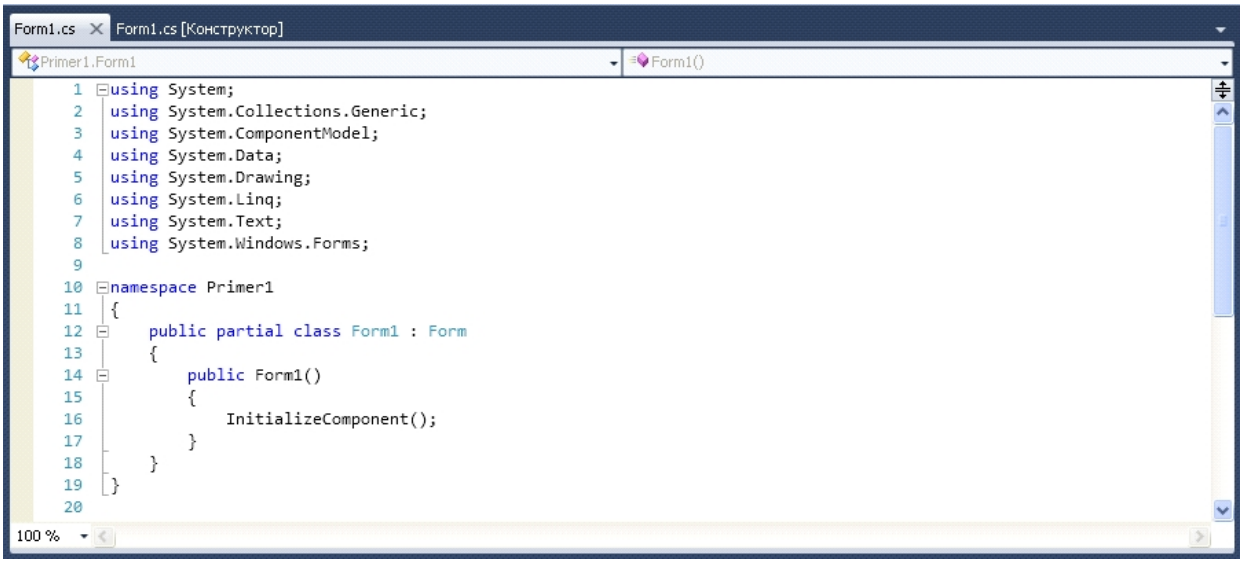

Рис. 7. Заготовка программного кода

Если по каким-либо причинам не видна (например, случайно была закрыта вкладка **Form1.cs[Конструктор]**), то чтобы форма появилась на рабочем столе, следует выполнить опцию контекстного меню элемента **Form1.cs** «**Открыть в конструкторе**».

Размер области для просмотра и редактирования кода устанавливается в зависимости от размещения окон в интегрированной среде разработки.

В левой части главного окна размещается окно **Обозреватель решения**, в котором отображается структура файлов проекта и среди них - программный модуль заготовка для размещения в нем программ-обработчиков событий компонентов, которые будут размещены в форме(рис. 8). Окно **Обозреватель решения** удобно использовать для быстрого доступа к нужно му элементу проекта.

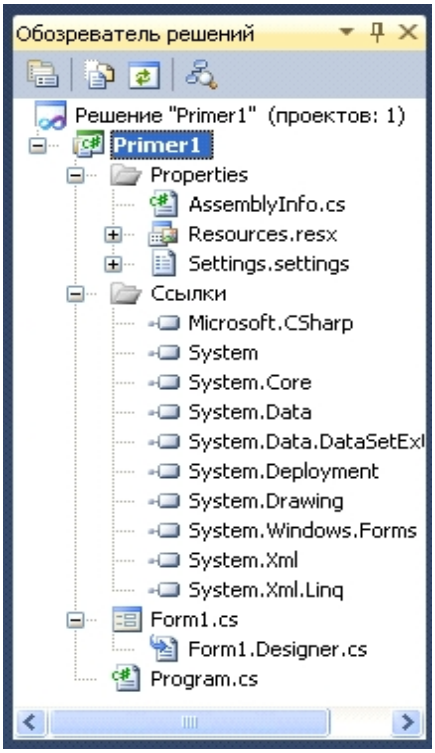

Рис. 8. Окно «Обозреватель решений»

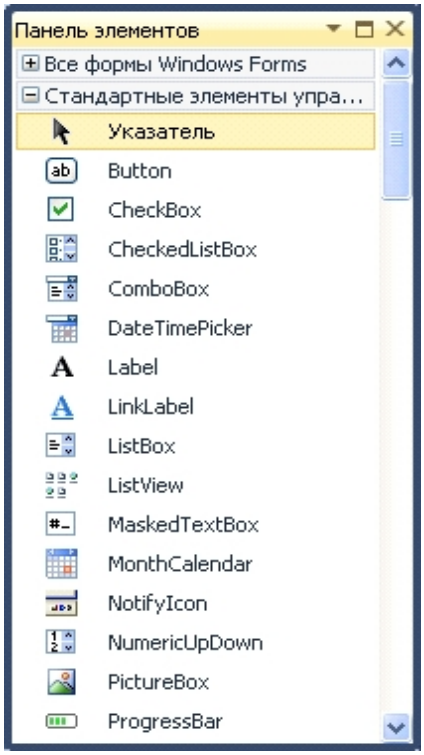

Рис. 9. Окно «Панель элементов»

В правой части главного окна Visual C# отображены окна **Обозреватель серверов** и **Панель**

элементов (при первоначальном создание проекта отображаются в виде закладок) (рис. 3). Для отображения этих окон необходимо навести курсор мыши на соответствующее окно (автоматически отобразится окно в главном окне) и щелкнуть на заголовке окна правой кнопкой мыши (отобразится всплывающее меню) и выбрать пункт меню «**Плавающая область**» (на рис. 9 отображено окно **Панель элементов**).

Окно **Обозреватель серверов** используется для получения сведений о компьютерах в сети в процессе написания кода (рис. 10). Оно позволяет получать информацию о соединениях с базами данных, службах, журналах регистрации событий, а также другую подобную информацию.

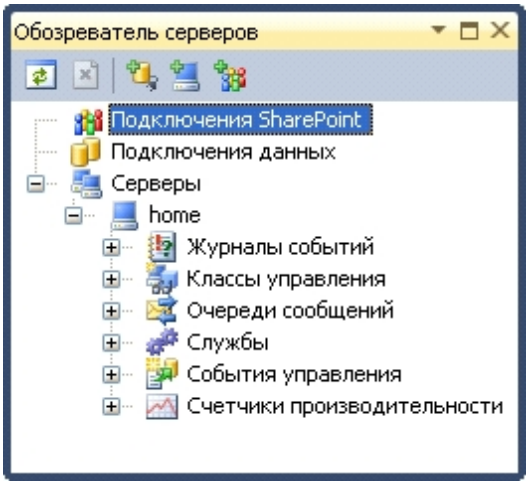

Рис. 10. Окно «Обозреватель серверов»

В окне **Панель элементов** находятся компоненты, которые можно поместить (перетащить) на форму. Компонент — это объект, реализующий некоторую функциональность. Например, в группе **Стандартные элементы управления** находятся компо ненты, реализующие пользовательский интерфейс (**Label** — область отобра жения текста; **TextBox** — поле ввода/редактирования текста; **Button** — командная кнопка), а в группе **Данные** — компоненты доступа к базам данных.

Кроме того, в правой части размещается окно **Свойства** (рис. 11), которое можно отобразить с помощью команды главного меню **Вид->Окно свойств**.

В окне **Свойства** отображаются свойства выбранного (выделенно го) в данный момент объекта — компонента или, если ни один из элементов не выделен, самой формы (имя и тип объекта отобража ются в верхней части окна **Свойства**) (рис. 13). Вызывается это окно несколькими способами:

- в меню **Вид** выбираем пункт **Другие окна** или используем клавишу **<F4>**;
- на выбранном объекте щелкаем правой кнопкой мыши и в контекстном меню находим пункт **Свойства**;
- выбираем объект и нажимаем клавишу **<F4>**;
- просто выбираем объект и переходим в окно **Свойства**.

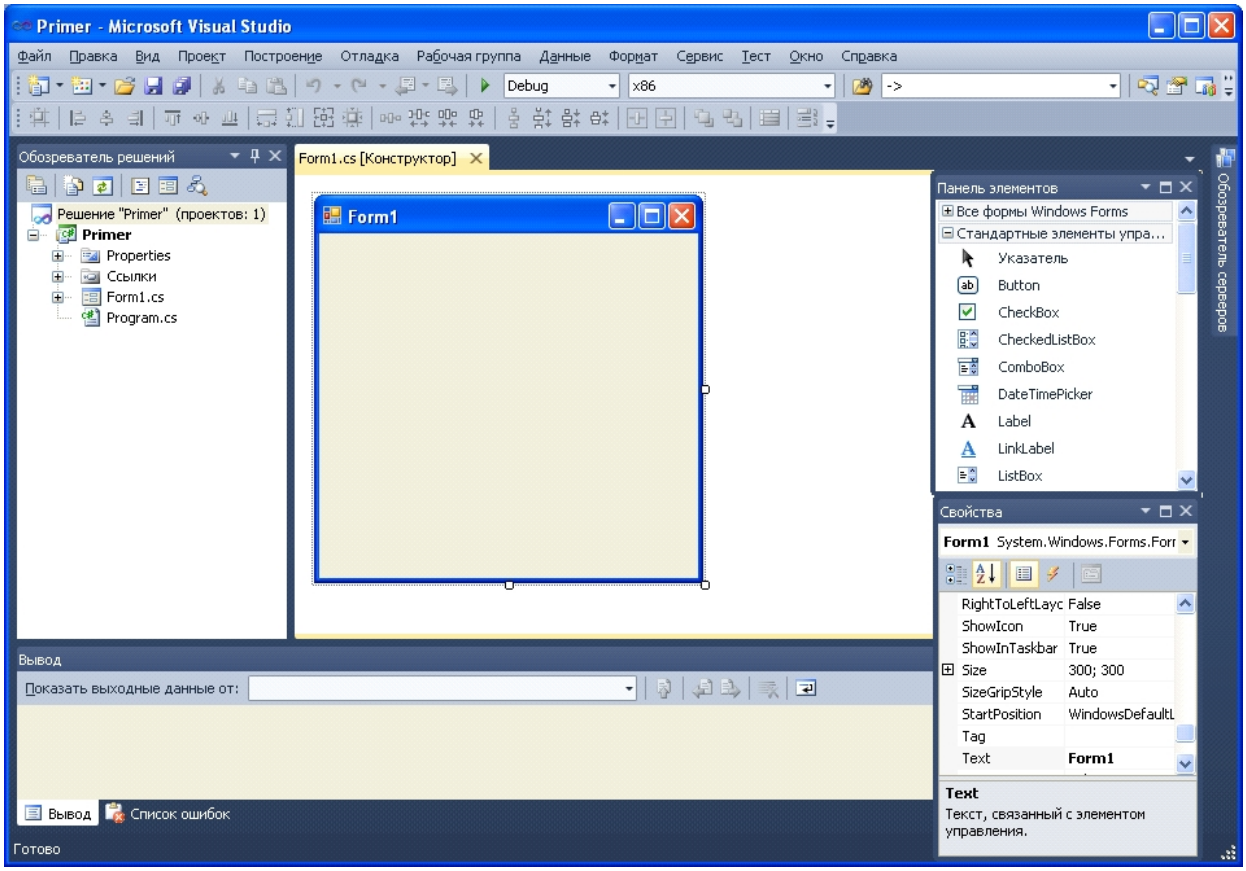

Рис. 11. Внешний вид главного окна при создании проекта

При создании проекта в окне **Свойства** отображаются свойства проекта. Окно **Свойства** используется для редактирования значений свойств объек тов и, как следствие, изменения их вида и поведения. Например, чтобы из менить текст в заголовке формы, надо изменить значение свойства **Text**. Окно **Свойства** содержит панель быстрых кнопок (рис.12).

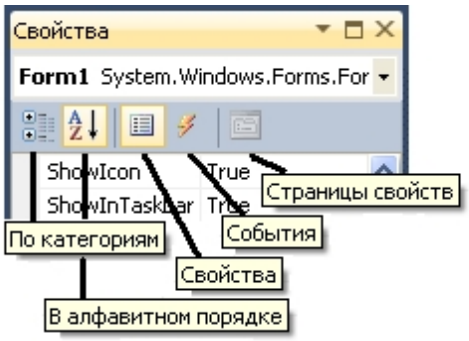

Рис. 12. Кнопки управления

При отображении свойства могут быть объединены в группы по функ циональному признаку (кнопка «**По категориям**») или упорядочены по алфавиту (кнопка «**В алфавитном порядке**»). Чтобы изменить способ отображения, надо сделать щелчок на соответствующей кнопке.

Сделав в окне **Свойства** щелчок на кнопке **События** (рис. 12), можно увидеть перечень событий (рис. 14), которые способен воспринимать выбранный объект (компонент или форма). Событие – это то, что происходит во время работы программы. Например, командная кнопка **Button** может реагировать на щелчок кнопкой мыши – событие **Click**.

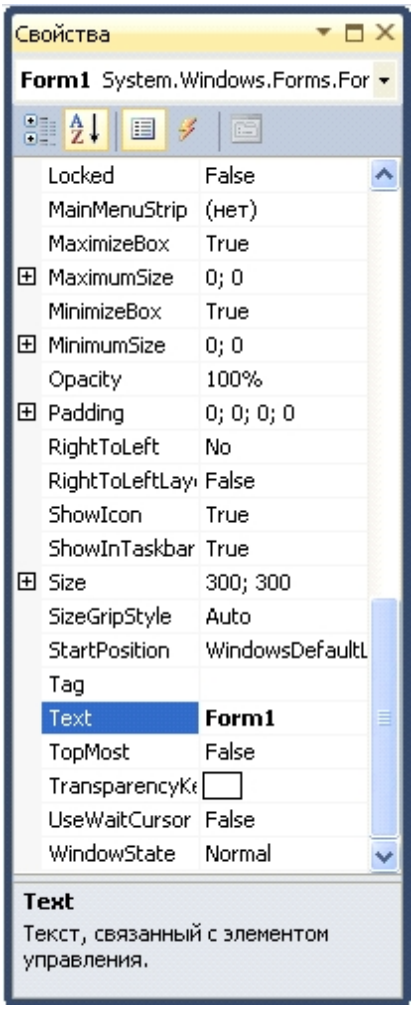

Рис. 13. Окно «Свойства» при щелчке на кнопке «**Свойства**»

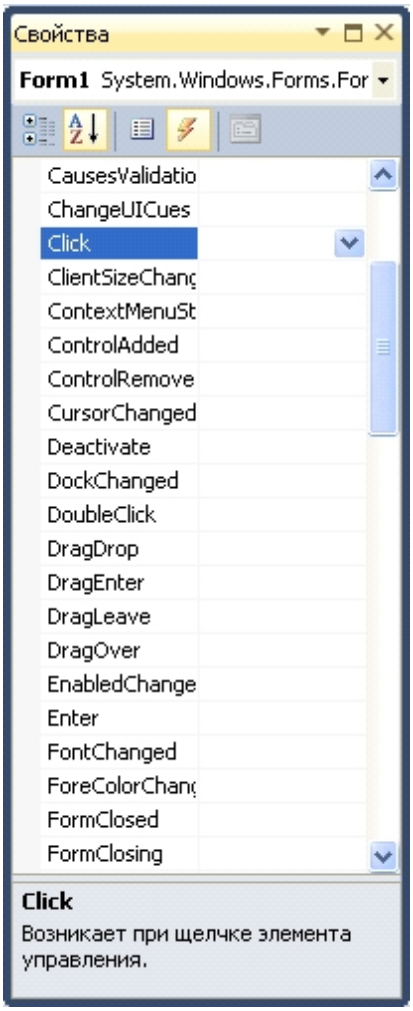

Рис. 14. Окно «Свойства» при щелчке на кнопке «**События**»

Окно **Обозреватель объектов** предназначено для получения информации о том, какие методы и другие элементы программного кода доступны в базовых классах и других библио теках, на которые в сборках имеются ссылки (рис. 15). Окно **Обозреватель объектов** отображает древовидное представление структуры классов приложения, позволяя просматривать члены каждого класса. С помощью этого окна можно просмотреть не только пространства имен и классы, входящие в состав проекта, но и все сборки, на которые в проекте имеются ссылки. При работе с окном **Обозреватель объектов** необходимо учитывать, что классы в нем группируются сначала по сборкам, в которых они находятся, и лишь затем по про странствам имен.

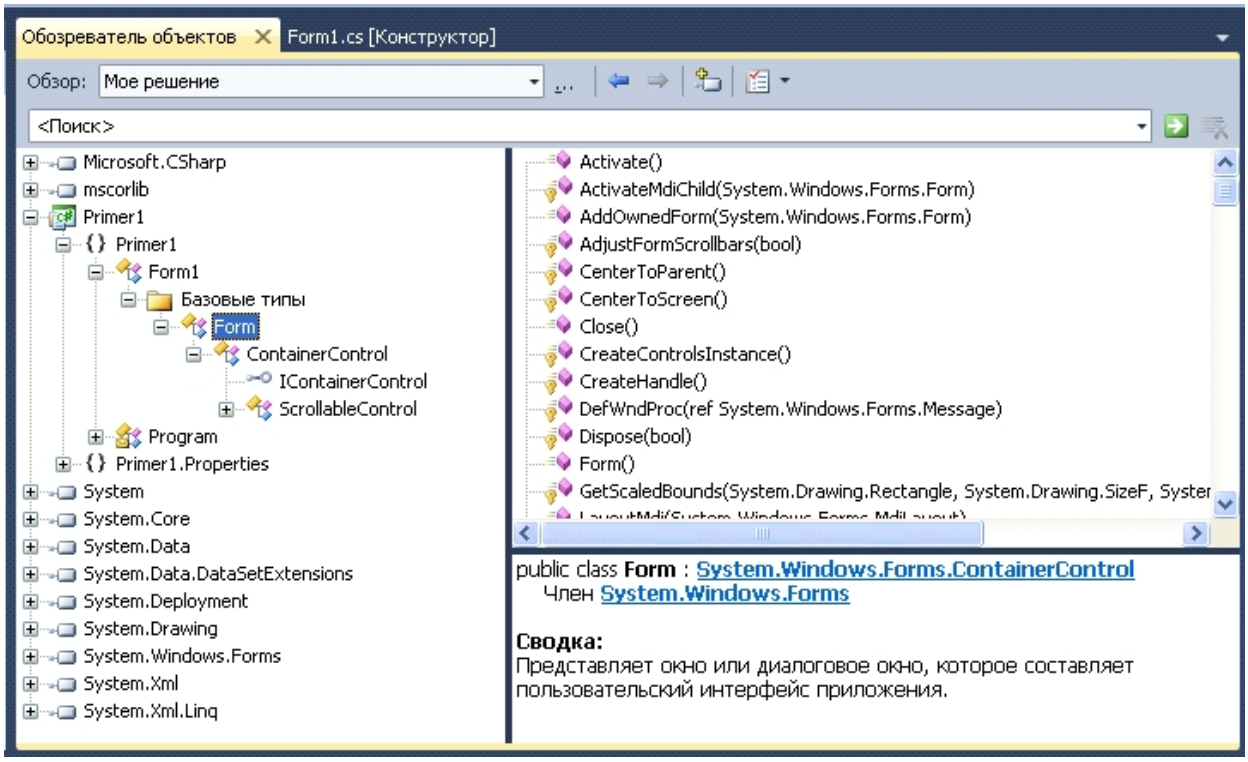

Рис. 15. Внешний вид окна **Обозреватель объектов**

1.2. Методика создания приложения в ИСР Visual C#

В настоящее время для разработки программ (приложений) используется среда быстрой разработки (RAD – среда, Rapid Application Development). В основе такой среды лежит технология визуального проектирования интерфейса пользователя и событийного программирования.

Создание приложения в Visual C# подразумевает создание формы, размещение на ней необходимых компонентов, определение свойств (характеристики компонентов), описание методов (функ ции, выполняющие необходимые действия над компонентами) этих компонентов.

Процесс создания приложения в Visual C# разбивается на две части:

- разработка формы приложения, включающая в себя создание интерфейса приложения (“внешнего вида”) и определение свойств у всех элементов;
- написание программного кода.

Разработка формы осуществляется с помощью выбора ком понентов из **Панели элементов** с последующим размещением их на форме. Для размещения элемента на форму с помощью палитры ком понентов необходимо произвести следующие действия:

1. выполнить щелчок левой кнопкой мыши на изображении нужного компонента на **Панели элементов**;
2. перевести указатель мыши на форму (указатель соответству ет местоположению левого верхнего угла компонента);
3. щелкнуть левой кнопкой мыши для создания объекта.

Еще один способ размещения элемента на форме состоит в вы полнении двойного щелчка на изображении требуемого компо нента в палитре компонентов. В этом случае новый элемент будет добавлен в верхний левый угол формы. Используя мышь, можно перемещать компонент на форме и изменять его размеры.

После того как необходимые компоненты помещены на форму, следует установить их свойства, которые определяют внешний вид и особенности работы приложения. У становка свойств выполняет ся в окне **Свойства** (рис. 13). Свойства компонентов могут изменяться динамически (в процессе работы приложения).

Так как реакцией на событие должно быть какое-либо действие (в Visual C# реакция на событие реализуется как метод обработки события), программист должен создать обработчик события, чтобы

придать компоненту необходимое поведение. Для создания заготовки обработчика события можно использовать события окна **События** (рис. 14), выполнив двойной щел чок в поле справа от имени необходимого события.

Обработчик события оформляется в виде метода (например, щелчок на кнопке **Button1**):

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

    }
```

При автоматическом формировании заголовка обработчика события, выполняемом средой ИСР, имя метода представляет собой объединение названия ком понента и имени события, соединенных символом подчеркивания.

Тело метода ограничено символами {} и состоит из от дельных операторов языка программирования C#. После каждой строки ставится точка с запятой.

Очень часто возникают казусы со случайным созданием событий. Например, если дважды, щелкнуть по компоненту в визуальном дизайнере, то будет создан обработчик события по умолчанию для этого компонента, для компонента Button со бытием по умолчанию является Click.
Случайно созданные обработчики событий необходимо убрать, чтобы он не мешался. Для этого существует несколько вариантов:

1. Выделить компонент в визуальном дизайнере и перейти в режим «События» панели «Свойства». Напротив события удалить в поле название метода, созданного для обработчика события. Визуальный редактор удалит регистрацию события, которую он автоматически добавил в свой метод InitializeComponent(). Если в обработчике события не было кода, то заготовка для метода исчезнет и из кода.
2. Если обработчик события содержит код, но он уже не нужен, то можно сначала удалить код из обработчика события, а потом выполнить действия из пункта 1.
3. Если обработчик события содержит код, то можно сначала удалить имя обработчика события в режиме «События» панели «Свойства», а потом безболезненно удалить код метода
4. Если был создан обработчик события и тут же был удален метод в редакторе кода, то в этом случае среда разработки не удалит регистрацию события в методе InitializeComponent(). Это придется делать вручную.

Рассмотрим эту ситуацию на примере. Создадим новое приложение, поместим на его форму кнопку, дважды щелк нем по кнопке, и среда разработки переключится в редактор кода с создан ным для обработки события методом:

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

У далив эту заготовку кода из редактора и запустив проект на компиляцию, получим сообщение об ошибке. В этой ошибке компилятор сообщает, что обработчик события не найден. При двойном щелчке по ошибке откроется закладка с файлом Form1.Designer.cs, где найде на ошибка, и будет выделена строка кода, которую добавил визуальный редактор для регистрации события. Для того, чтобы избавиться от ошибки, нужно удалить строку, которую выделил редактор. Таким образом, вручную удаляет регистрация, и проект откомпилируется без проблем.

1.2.1. Создание простого приложения

Пример 1. Создадим приложение, в кото ром при щелчке пользователя на кнопке появлялась бы какая-нибудь надпись. Для этого последовательно выполним следующие шаги:

1. Запустите Visual C# с помощью меню **Пуск→Программы→Microsoft Visual Studio 2010.**

В поле **Имя** ввести имя проекта **Primer**, в поле **Расположение** ввести папку размещения проекта **D:\PK11\Semestr2\Лаб.работа1**.

Работа над новым проектом начинается с создания стартовой формы – главного окна приложения. Стартовая форма создается путем изменения значений свойств формы **Form1** и добавления к форме необходимых компонентов: полей ввода, полей вывода текстовой информации, командных кнопок и др.

Для изменения свойств объектов, в том числе и формы, используется окно **Свойства** (рис. 13). В левой колонке этой вкладки перечислены свойства выбранного объекта, а в правой – указаны значения свойств. На рис. 16 приведен перечень свойств формы, отображенных в окне **Свойства**.

При создании формы можно изменить ряд свойств. Например, для того чтобы изменить текст в заголовке формы **Form1** на заголовок приложения необходимо в окне **Свойства** щелкнуть левой копкой мыши на строке **Text** (в результате будет выделено значение свойства и в правом окошке появится курсор) и ввести соответствующий текст заголовка приложения.

Некоторые свойства являются сложными, т.е. значение определяется совокупностью значений уточняющих свойств. Перед именами сложных свойств стоит значок «+», в результате щелчка на котором раскрывается список уточняющих свойств.

В результате выбора некоторых свойств, рядом со значением свойства появляется командная кнопка с тремя точками (например, свойство **Font**). Это значит, что задать значение свойства можно в дополнительном диалоговом окне, которое появляется в результате нажатия на эту кнопку.

2. После создания формы необходимо ее сохранить, выполнив команду **Файл→Сохранить все**.

3. Теперь можно начинать собственно разработку проекта: помещать в форму (в ту, которая сохранилась на экране после записи) различные компоненты из Панели элементов и реализовывать алгоритм задачи.

Перенесите на пустую форму кнопку типа **Button** из раздела «**Стандартные элементы управления**» панели элементов. Для этого выделите пиктограмму кноп ки и затем щелкните курсором мыши в нужном вам месте формы. На форме поя вится кнопка, которой ИСР Visual C# по умолчанию присвоит имя — **button1** (рис. 17).

4. Аналогичным образом перенесите на форму из раздела «**Стандартные элементы управления**» метку **Label.** Visual C# присвоит метке имя **label1.** В этой метке в процессе выполнения приложе ния будет появляться текст при нажатии пользователем кнопки.

5. Разместите компоненты на форме так, как показано на рис. 17. Окно формы можно изменить. Это можно сделать двумя спо собами. Надо подвести указатель мыши к краю формы (появится стрелка, указывающая, в каком направлении можно менять раз мер) и, нажав левую кнопку мыши, изменить размер. Также мож но указать значения высоты и ширины формы в ее свойствах **ClientHeight** и **ClientWidth** соответственно.

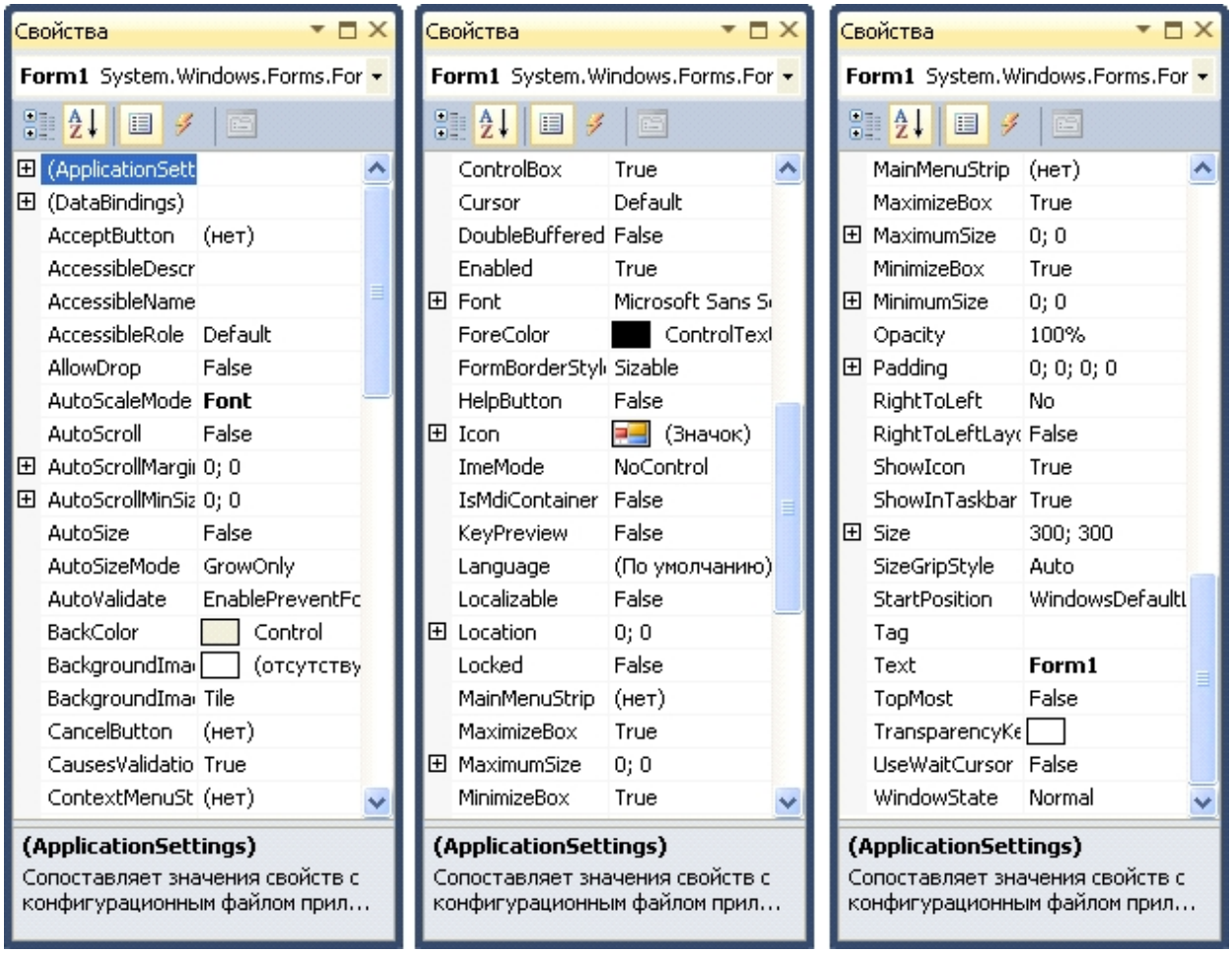

Рис. 16. Перечень свойств формы

6. Выделите на форме компонент **button1**. Перейдите в окно **Свойства** и измените свойство **Text** (надпись, которая по умолчанию равна **button1**) на «**Пуск**».

7. Укажите метке **label1,** что надписи на ней надо делать жирным шрифтом. Для этого выделите метку, в окне **Свойства** выберите свойство **Font** (шрифт), затем щелчком на кнопке с двоеточием откройте диалоговое окно «**Шрифт**» и установите в начертание шрифта «**жирный**», размер – **12**.

8. Сотрите текст в свойстве **Text** метки **label1,** чтобы он не отображался, пока пользователь не нажмет кнопку приложения.

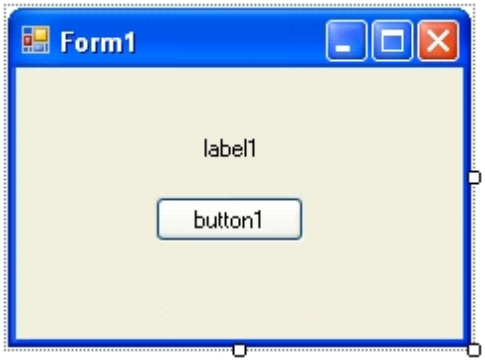

Рис. 17. Форма приложения

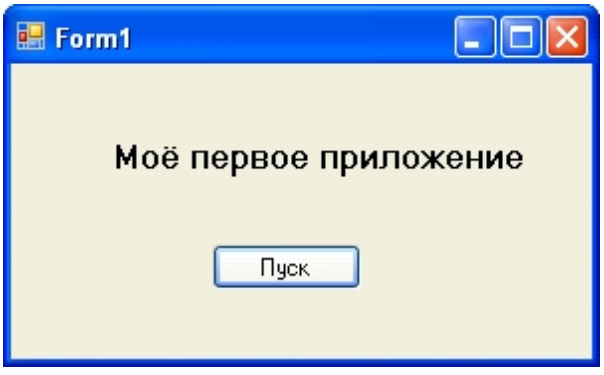

Рис. 18. Окно приложения в процессе выполнения

9. Теперь вам осталось только написать оператор, который заносил бы в свойство **Text** метки

label1 нужный вам текст в нужный момент. Этот момент определя ется щелчком пользователя на кнопке **Пуск**. При щелчке в кнопке генерируется собы тие **Click**. Следовательно, обработчик этого события вы и должны написать.

Выделите кнопку **button1** на форме, перейдите в окно **Свойства**, от кройте в нем окно **События**, найдите событие кнопки **Click** и сделайте двойной щелчок в окне справа от имени этого собы тия. Это стандартный способ задания обработчиков любых событий. Но перей ти в обработчик события **Click** (только этого события) можно и иначе: достаточно сделать двойной щелчок на компоненте **button1** на форме. В обоих случаях оказываемся в окне Редактора кода и видим текст:

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

    }
```

Заголовок этого метода складывается из имени компонента **button1** и имени события **Click**, соединенных символом подчеркивания.

10. Напишите в обработчике оператор задания надписи метки **label1.** Этот опера тор может иметь вид:

```
label1.Text = "Мое первое приложение";
```

Оператор, который написан, означает следующее. Символ «= » обозначает операцию присваивания, в которой тому, что написано перед этим символом, при сваивается значение того, что написано после символа присваивания. Слева на писано ***label1.Text***, что значит присваивание значения свойству **Text** компонента **label1.** Все указания свойств и методов производятся анало гичным образом: пишется имя компонента, затем ставятся сим волы операции доступа **«.»** (точка). После этого символа пишется имя свойства или метода. В данном слу чае свойству **Text** присваивается строка текста ***«Мое первое приложение».***

Таким образом, полностью обработчик события имеет вид (рис. 19).

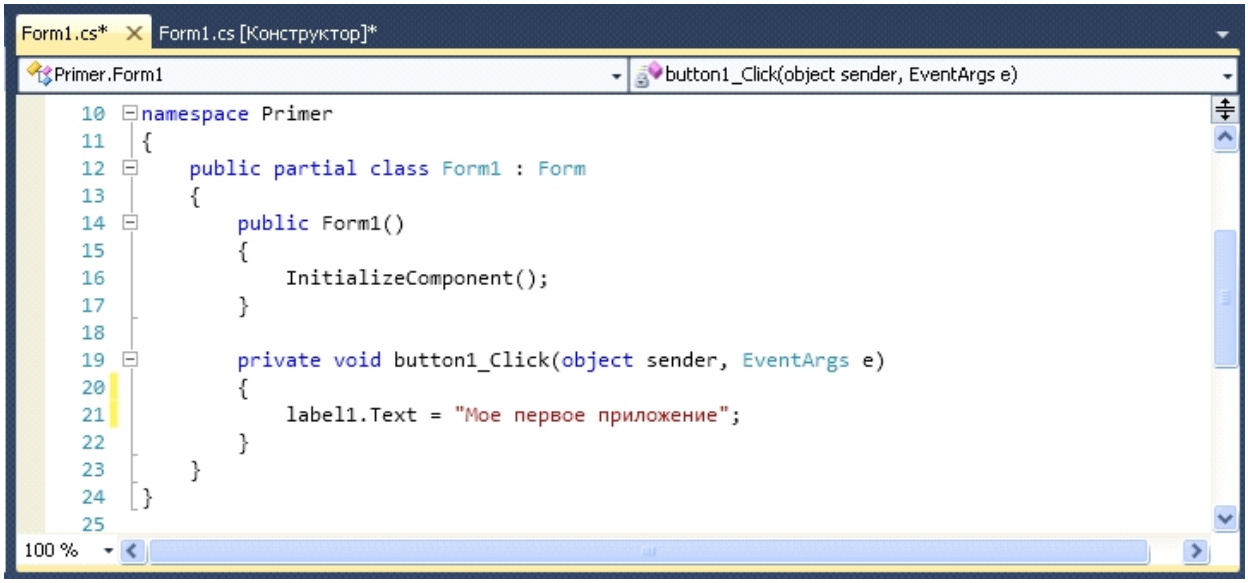

Рис. 19. Окно Редактора кода

В процессе разработки программы часто возникает необходимость переключения между окном редактора кода и окном конструктора формы. Это можно сделать при помощи закладок **Form1.cs** и **Form1.cs[Конструктор]**.

12. Итак, приложение готово. Можно откомпилировать и выполнить его. Процесс построения программы активизируется в результате выбора в меню **Построение** команды **Построить *<имя проекта>***, а также в результате запуска программы из среды разработки (меню **Отладка** команда «**Начать отладку**» или «**Запуск без отладки**»), если с момента последней компи ляции в программу были внесены изменения.

Процесс и результат компиляции отражаются в окне **Вывод.** Если в про грамме нет ошибок, то по завершении процесса компиляции в окне **Вывод** отображается сообщение о результате построения (рис. 20).

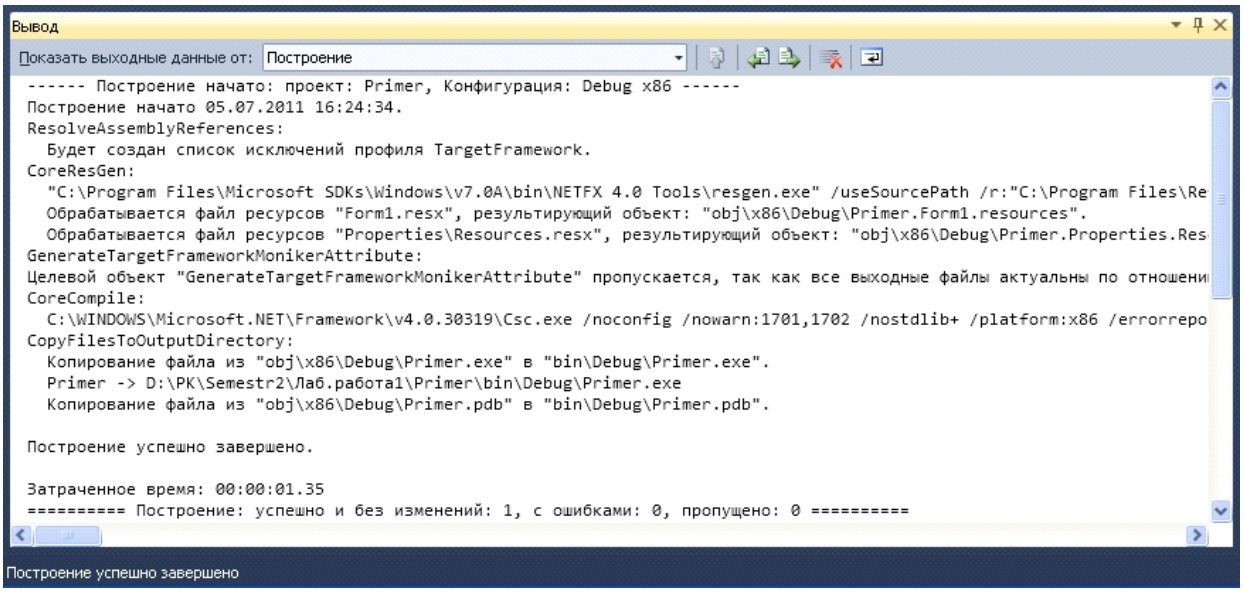

Рис. 20. Результат построения (в программе нет ошибок)

Если в процессе построения в программе обнаруживаются ошибки, то в окно **Вывод** выводятся сообщения о них (рис. 21).

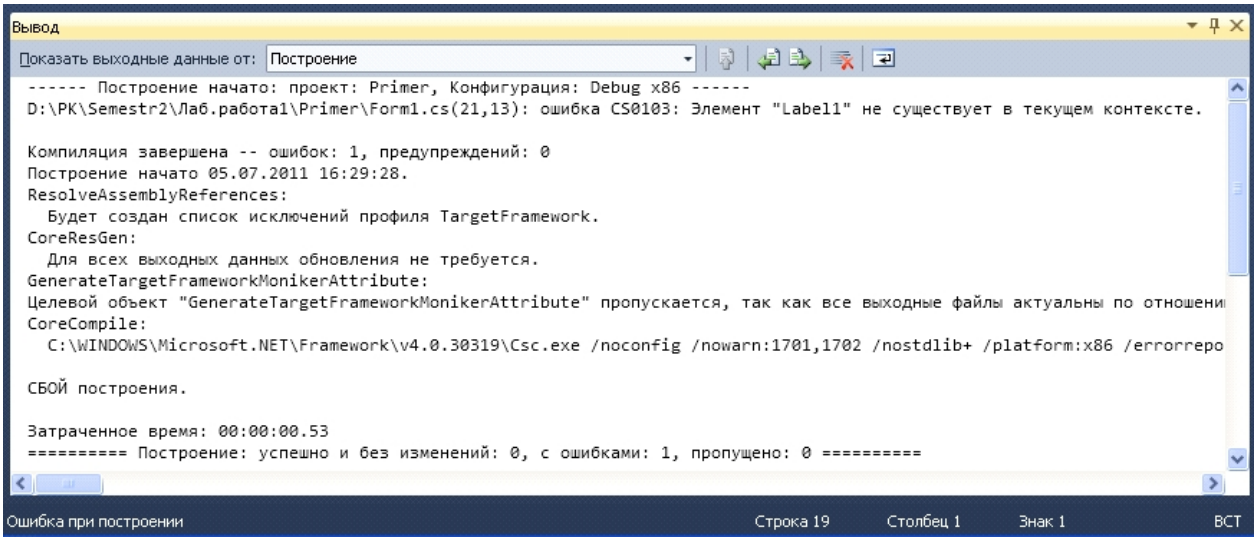

Рис. 21. Результат построения (в программе есть ошибки)

Чтобы перейти к фрагменту кода, который содержит ошибку, надо выбрать сообщение об ошибке и дважды щелкнуть на нем мышью. В редакторе кода будет отмечена строка(стрелка синего цвета), в которой обнаружена ошибка, а в строке состояния будет указано содержание ошибки и её местоположение (рис. 22).

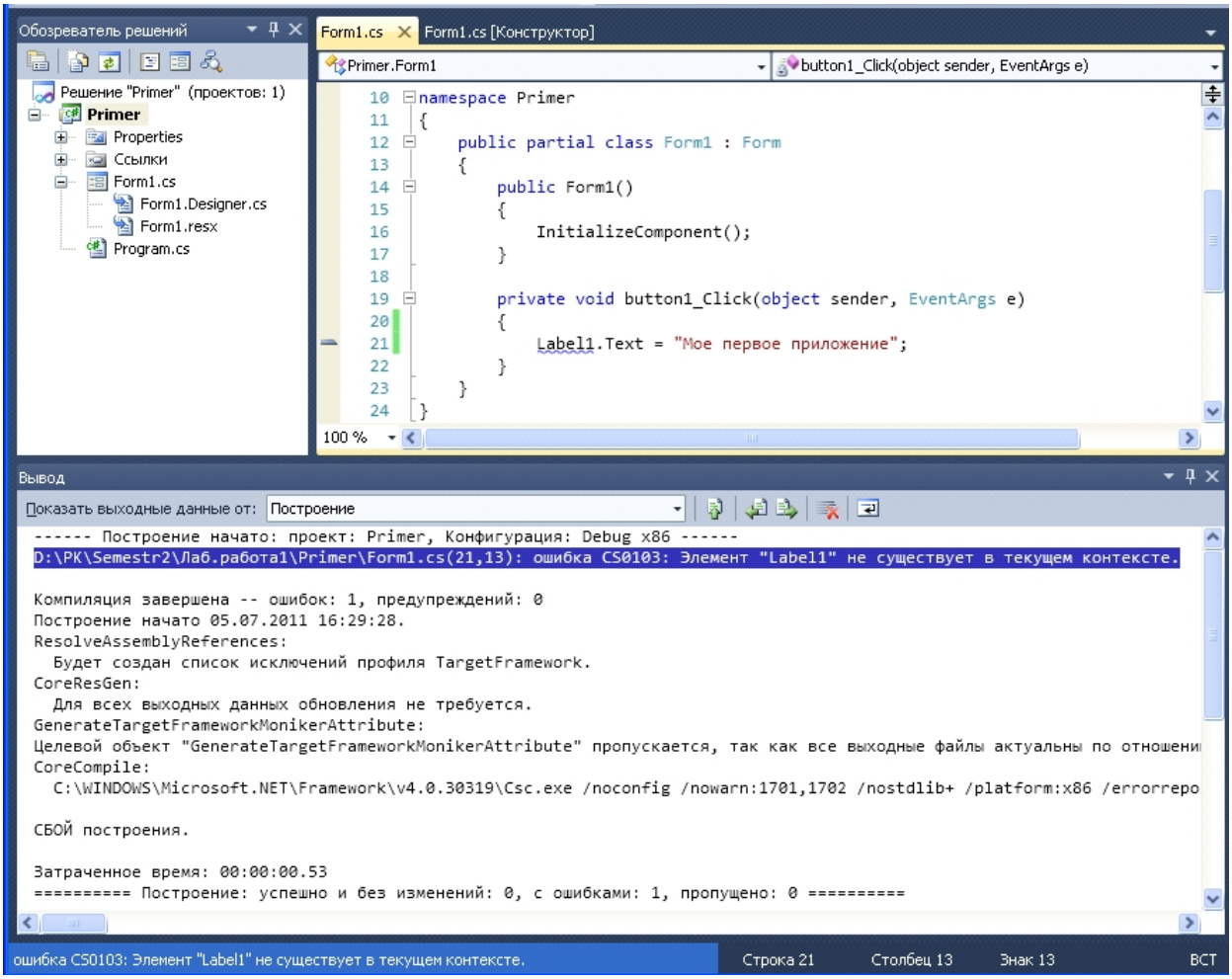

Рис. 22. Внешний вид редактора кода при наличии ошибок

13. Пробный запуск программы можно выполнить непосредственно из Visual Studio, не завершая работу со средой разработки. Для этого в меню **Отладка** надо выбрать команду «**Начать отладку**» или «**Запуск без отладки**». Можно также сделать щелчок на кнопке «**Начать отладку**» или нажать кла вишу <**F5** >.

Команда «**Начать отладку**» запускает программу в режиме отладки. Команда «**Запуск без отладки**» запускает программу в обычном режиме, даже в том случае, если в ней есть отладочная информация (заданы точки останова, указаны переменные, значения которых надо контролировать).

Во время работы программы могут возникать ошибки, которые называются ошибками времени выполнения или исключениями. В большинстве случаев причинами исключений являются неверные исходные данные.

14. После завершения создания проекта его надо сохранить командой **Файл→Сохранить все**.

1.2.2. Создание более сложного приложения

Пример 2. Создадим приложение, которое позволяет вычислять силу тока в электрической цепи. Сила тока вычисляется по известной формуле: **I=U/R**, где **U** – напряжение источника (Вольт), а **R** – величина сопротивления (Ом). Вид диалогового окна программы показан на рис.23.

1. Перед началом разработки приложения желательно открыть новый проект, используя команду главного меню **Файл→Создать→Проект…**

В поле **Имя** ввести имя проекта **Amper**, в поле **Расположение** ввести папку размещения проекта **D:\PK11\Semestr2\Лаб.работа1**.

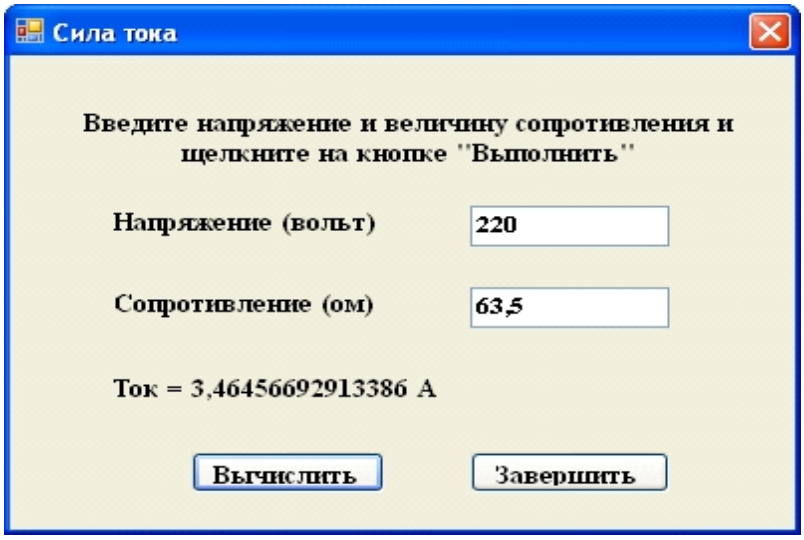

Рис. 23. Внешний вид приложения «Сила тока»

2. Установим свойства формы разрабатываемого приложения в соответствии со значениями, приведенными в табл. 3.

Таблица 3. Перечень свойств формы

Свойство	**Значение**	**Назначение**
Text	Сила тока	Заголовок формы
Size.Height	300	Высота формы
Size.Width	400	Ширина формы
FormBorderStyle	FixedSingle	Тонкая граница формы. Во время работы программы пользователь не может изменить размер окна путем захвата и перемещения его границы
MinimizeBox	False	В заголовке программы не отображать кнопку «Свернуть»
MaximizeBox	False	В заголовке программы не отображать кнопку «Развернуть»
StartPosition	CenterScreen	Окно программы появляется в центре экрана
Font.Name	Times New Roman	Название шрифта
Font.Bold	True	Начертание шрифта
Font.Size	10	Размер шрифта

После создания формы необходимо ее сохранить, выполнив команду **Файл→Сохранить все**.

3. Программа вычисления тока в электрической цепи должна получить от пользователя исходные данные – напряжение и величину сопротивления, которые могут быть введены с клавиатуры в поля редактирования. Поэтому в форму надо добавить поля редактирования.

Чтобы добавить в форму разрабатываемого приложения поле редактирования, надо из раздела «**Стандартные элементы управления**» в окне «**Панель элементов**» выбрать компонент **TextBox**. Затем установить курсор в ту точку, в которой должен быть левый верхний угол компонента и еще раз щелкнуть левой кнопкой мыши. В результате на форме появится компонент **textBox1** – поле редактирования.

Каждому добавленному компоненту автоматически присваивается имя, которое состоит из названия компонента и его порядкового номера. Нам необходимо добавить в форму два компонента **TextBox:** компонент **TextBox1** предназначен для ввода величины напряжения, **TextBox2** – для ввода величины сопротивления (их имена соответственно будут **textBox1** и **textBox2**).

Программист путем изменения свойства **Name** может изменить имя компонента. Однако в простых программах имена компонентов, как правило, не изменяют.

4. Помимо полей редактирования в окне программы находится текст – краткая информация о программе, назначение полей ввода и поле вывода результата

Для вывода текста на поверхность формы используют поля вывода текста. Поле вывода текста – это компонент **Label**. Свойства компонента **Label** примерно такие же, как и компонента **TextBox**. Одними из отличий является свойства:

- **AutoSize** – признак того, что размер поля определяется его содержимым;
- **TextAlign** – способ выравнивания (расположения) текста в поле компонента (наиболее используемые выравнивание по левой верхней границе (**TopLeft**), посередине (**TopCentre**) и по центру (**MiddleCenter**).

В форму разрабатываемого приложения надо добавить четыре компонента **Label**:

label1 - для ввода информационного сообщения о программе;

label2 и **label3** – для определения назначений полей **textBox1** и **textBox2**.

label4 – для вывода результата расчета (величины тока в цепи).

Размеры и местоположение компонентов **Label** установите мышью. Свойства выбранных компонентов перечислены в таблице 4.

Таблица 4. Свойства компонентов

Свойство	**Компонент**			
	label1	**label 2**	**label3**	**label 4**
AutoSize	false	true	true	false
TextAling	MiddleCenter	TopLeft	TopLeft	TopLeft
Text	Введите напряжение и величину сопротивления и щелкните на кнопке «Выполнить»	Напряжение (Вольт)	Сопротивление (Ом)	

5. Последнее, что надо сделать на этапе создания формы – это добавить в форму две командные кнопки (компонент **Button**): **«Вычислить»** и **«Завершить»**. Добавляется в форму так же, как и другие компоненты. После добавления к форме двух командных кнопок нужно установить значения их свойств. Для **button1** в свойстве **Text** записать **Вычислить**, а для **button2** – **Завершить,** т.е. таким образом подписать эти кнопки. Размер и местоположение установить мышью.

Завершив работу над формой надо ее запомнить (команда **Файл->Сохранить все**), и после этого приступать к созданию программы.

6. Теперь вам осталось только написать оператор, который заносил бы в свойство **Text** метки **label4** значение силы тока в нужный момент. Этот момент определя ется щелчком пользователя на кнопке **«Вычислить»**, после ввода в поля редактирования исходных данных. При щелчке в кнопке генерируется собы тие **Click**. Следовательно, необходимо написать обработчик этого события.

Для нашей задачи необходимо выделить на форме кнопку **«Вычислить»** и в окне **События** выбрать событие **Click**. Правое окошко будет пустым – метод обработки события не определен. Двойным щелчком мыши в поле метода обработки открываем окно редактора кода и в шаблоне между фигурными скобками набираем операторы, реализующие метод обработки события.

Ниже приведены программа метода обработки события **Click** при щелчке на кнопке **«Вычислить»**:

```
private void button1_Click(object sender, EventArgs e)
    {
        double u; // напряжение
        double r; // сопротивление
        double i; // ток
        // получить данные из полей ввода
        u = System.Convert.ToDouble (textBox1.Text);
        r = System.Convert.ToDouble (textBox2.Text);
        // вычислить ток
        i = u/r;
        // вывести результат в поле метки label4
        label4.Text = "Ток = " + i.ToString("g") + " A";
```

}

Метод **button1_Click** выполняет расчет силы тока и выводит результат расчета в поле **label4**. Исходные данные вводятся из полей редактирования **textBox1** и **textBox2** путем обращения к свойству **Text**. Чтобы программа работала правильно, пользователь должен ввести в каждое поле редактирования целое или дробное число. При вводе дробного числа для разделения целой и дробной частей надо использовать **запятую**. Так как поле редактирования содержит текст, то необходимо выполнить преобразование строки в число. Эту задачу решает метод **ToDouble()**, в которой в качестве параметра передается содержимое поля редактирования **textbox.Text**. Другие методы преобразования строк приведены в таблице 5.

Таблица 5. Методы преобразования строк

Метод	Значение
ToSingle(s)	Дробное типа Single, Double
ToDouble(s)	
ToByte(s)	Целое типа Byte, Int16, Int32, Int64
ToInt16(s)	
ToInt32(s)	
ToInt64(s)	
ToUInt16(s)	Целое типа Unt16, Unt32, UInt64
ToUInt32(s)	
ToUInt64(s)	

После того, как исходные данные будут помещены в переменные **u** и **r**, выполняется расчет силы тока **i**. Вычисленная величина силы тока выводится в поле **label4** путем присвоения значения свойству **Text**. Для преобразования числа в строку символов используется метод **ToString().** Параметр метода **ToString()** задает формат строки результата: "c" — финансовый; "n" — числовой. В таблице 6 приведе ны возможные форматы представления числовой информации.

Таблица 6. Форматы представления чисел

Параметр метода ToString()	Формат	Пример
c, C	финансовый (денежный). Используется для представления денежных величин. Обозначение денежной единицы, разделитель групп разрядов, способ отображения отрицательных чисел определяют соответствующие на стройки операционной системы	123.456 ("C", en-US) => $123.46 123.456 ("C", fr-FR) => 123,46 € 123.456 ("C", ja-JP) => ?123 -123.456 ("C3", en-US) => ($123.456) -123.456 ("C3", fr-FR) => -123,456 € -123.456 ("C3", ja-JP) => -?123.456
d, D	Числовой. Используется для представления целочисленных цифр с необязательным отрицательным знаком. Поддерживается только целочисленными типами данных.	1234 ("D") => 1234 -1234 ("D6") => -001234
e, E	научный. Используется для представления очень маленьких или очень больших чисел. Разделитель целой и дробной частей числа задается в на стройках операционной системы	1052.0329112756 ("E", en-US) => 1.052033E+003 1052.0329112756 ("e", fr-FR) => 1,052033e+003 -1052.0329112756 ("e2", en-US) => -1.05e+003 -1052.0329112756 ("E2", fr_FR) => -1,05E+003
f, F	число с фиксированной точкой. Используется для представления дробных чисел. Количество цифр дробной части, способ отображения отрицательных чисел опреде ляют соответствующие настройки операционной системы	1234.567 ("F", en-US) => 1234.57 1234.567 ("F", de-DE) => 1234,57 1234 ("F1", en-US) => 1234.0 1234 ("F1", de-DE) => 1234,0 -1234.56 ("F4", en-US) => -1234.5600 -1234.56 ("F4", de-DE) => -1234,5600
n, N	числовой. Используется для представления дробных	1234.567 ("N", en-US) => 1,234.57

	чисел. Количество цифр дробной части, символ-разделитель групп разрядов, способ отображения отрица тельных чисел определяют соответствующие настройки операционной системы	1234.567 ("N", ru-RU) => 1 234,57 1234 ("N", en-US) => 1,234.0 1234 ("N", ru-RU) => 1 234,0 -1234.56 ("N", en-US) => -1,234.560 -1234.56 ("N", ru-RU) => -1 234,560
g, G	универсальный формат. Наиболее компактная запись из двух вариантов: экспоненциального и с фиксированной запятой. Поддерживается всеми числовыми типами данных.	-123.456 ("G", en-US) => -123.456 123.456 ("G", sv-SE) => -123,456 123.4546 ("G4", en-US) => 123.5 123.4546 ("G4", sv-SE) => 123,5 -1.234567890e-25 ("G", en-US) => -1.23456789E-25 -1.234567890e-25 ("G", sv-SE) => -1,23456789E-25
r, R	без округления. В отличие от формата N, этот формат не выполняет округления (количество цифр дроб ной части зависит от значения числа)	123456789.12345678 ("R") => 123456789.12345678 -1234567890.12345678 ("R") => -1234567890.1234567
x, X	Числовой. Используется для представления чисел в шестнадцатеричном виде. Поддерживается только целочисленными типами данных.	255 ("X") => FF -1 ("x") => ff 255 ("x4") => 00ff -1 ("X4") => 00FF

7. Метод обработки события **Click** на командной кнопке **«Завершить»** создается так же, как и метод обработки события **Click** для кнопки **«Выполнить».** В результате щелчка на кнопке **«Завершить»** программа должна завершить работу. Чтобы это произошло надо закрыть окно программы. Это делает метод **Close().**

Программа обработки **Click** события на кнопке **«Завершить»** имеет следующий вид:

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

8. Откомпилируйте и выполните разработанное приложение.

2. Рабочее задание

Задание 1. Разработать приложение «Приветствие», с помощью которого осуществляется диалог пользователя с приложением. Имя пользователя определить в коде приложения.

При разработке интерфейса приложения использовать компоненты **Label** и **Button** (один из вариантов интерфейса после запуска приложения (до нажатия кнопки «Моё имя») представлен на рис. 24, после нажатия кнопки «Моё имя» представлен на рис. 25).

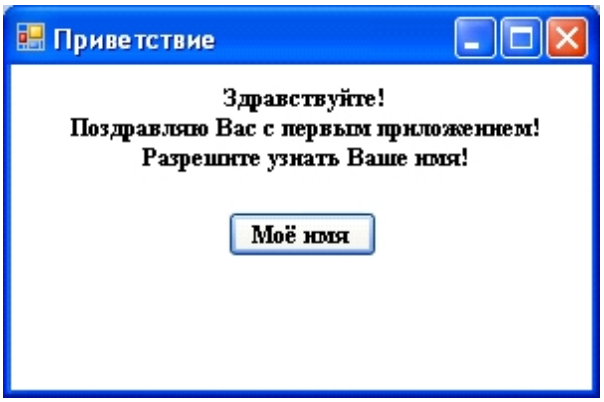

Рис. 24. Внешний вид приложения после запуска

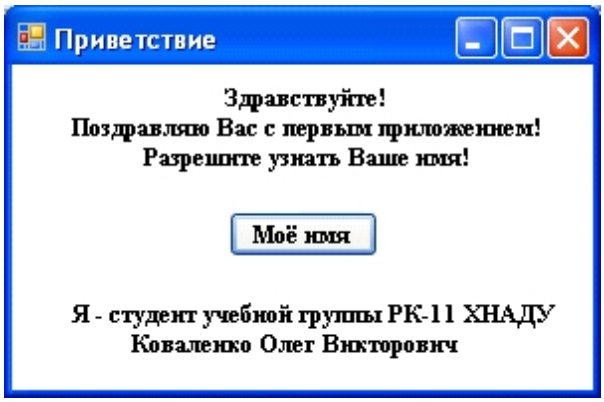

Рис. 25. Внешний вид приложения после нажатия кнопки «Моё имя»

Задание 2. Разработать приложение «Изучение форматов чисел», с помощью которого можно изучить форматы представления чисел. В качестве результата, выводимого на экран, использовать операцию деления двух чисел, значения которых определяются как константы.

При разработке интерфейса приложения использовать компоненты **Label** и **Button** (один из вариантов интерфейса представлен на рис. 26).

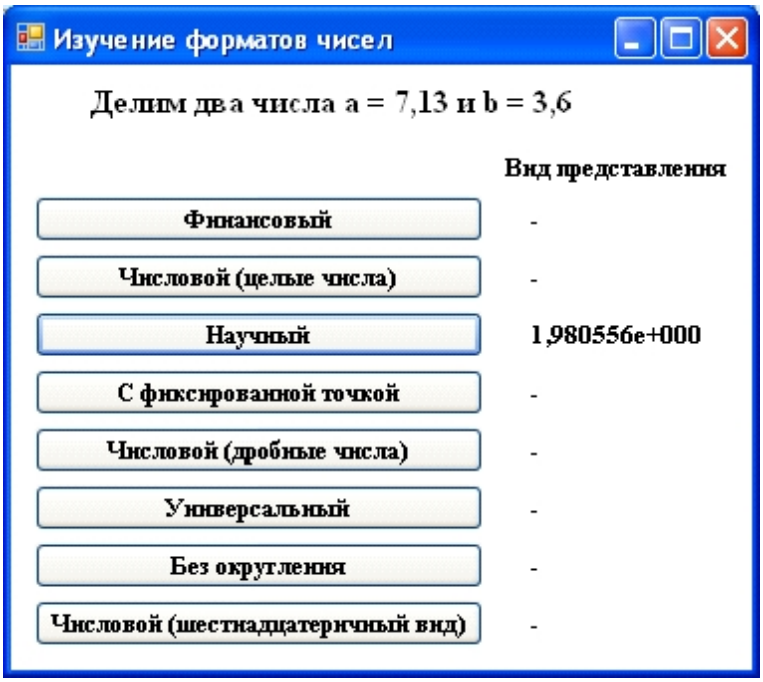

Рис. 26. Внешний вид приложения «Изучение форматов чисел»

Задание 3. Модифицировать приложение «Изучение форматов чисел» таким образом, чтобы исходные числа **a** и **b** вводились с клавиатуры.

Задание 4. Разработать приложение «Определение среднего арифметического значения» 10 чисел. Ввод значения очередного числа осуществляется при каждом нажатии кнопки «Ввод» с выдачей сообщения о вводе n-го значения, результат после ввода последнего значения числа выдается автоматически.

При разработке интерфейса приложения использовать компоненты **Label** и **Button** (один из вариантов интерфейса представлен на рис. 27).

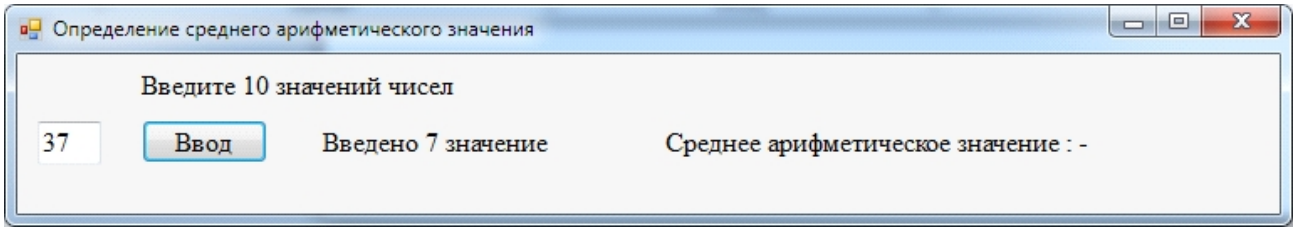

Рис. 27. Внешний вид приложения «Определение среднего арифметического значения»

3. Контрольные вопросы

Литература

1. Голощапов А.Л. Microsoft Visual Studio 2010. – СПб.:БХВ-Петербург, 2011. – 544 с.: ил.
2. Культин Н.Б. Microsoft Visual C# в задачах и примерах. – СПб.: БХВ-Петербург, 2009. – 320 с.: ил.
3. Лабор В.В. Си Шарп: Создание приложений для Windows. – Мн.: Харвест, 2003. – 384 с.
4. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. Пер. с англ. - М.: «Русская Редакция», 2002.- 576 с.: ил.
5. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2. Пер. с англ. - М.: «Русская Редакция», 2002.- 624 с.: ил.
6. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0. Пер. с англ. - М.: Издательский дом "Вильямс", 2011. — 1392 с.: ил.
7. Фаронов В.В. Программирование на языке C#. – СПб.: Питер, 2007. – 240 с.: ил.
8. Фленов М.Е. Библия C#. - СПб.: БХВ-Петербург, 2011.– 560с.: ил.