Лабораторная работа №12

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 №12
Исследование возможностей ИСР Visual C# для создания приложений с использованием меню.

Цель работы – исследовать возможностей интегрированной среды разработки Visual C# и получить практические навыки по созданию приложений с использованием меню.

1. Теоретические сведения

В Visual C# имеется два компонента, представляющие меню: **MenuStrip** - главное меню и **ContextMenuStrip** – контекстное (всплывающее) меню.

1.1. Компонент MenuStrip

Компонент **MenuStrip** представляет собой главное меню приложения, с помощью которого управляют всей работой приложения и его частей. Разные части приложения запускаются на выполнение отдельными командами, собранными в эту структуру. Выход из приложения тоже происходят через меню. Структуру меню определяет заказчик приложения и его исполнитель. Меню формируется в форме после того, как его значок перенесен из палитры компонентов в форму. С этой формой меню будет связано через свойство формы **MainMenuStrip**, в окне которого и появляется имя компонента.

Когда меню сформировано, то после запуска приложения на выполнение в левой верхней части формы будет расположена строка, содержащая главные опции этого меню. Главные опции могут распадаться на более детальные команды (если таковые заданы), располагающиеся на этот раз уже в столбик (сверху вниз).

После того как в форму будет добавлен компонент **MenuStrip**, в верхней части формы появляется строка меню (синяя полоса), в начале которой находится область ввода текста (прямоугольник, внутри которого находится текст "Вводить здесь"). Значок компонента располагается в нижней части рабочего стола (на специальной полосе под формой), а не в самой форме (рис. 1).

Рис. 1. Внешний вид компонента MenuStrip в конструкторе

Чтобы создать первый пункт меню, надо сделать щелчок в об ласти ввода текста и ввести название пункта меню, например, **Пункт1**. Как только будет введена первая буква, справа и снизу от области ввода появятся еще две области ввода текста. В нижнюю область надо ввести название команды меню, например, **Подпункт11,** а в правую — название следующего пунк та меню, например, **Пункт2**. Если при этом пункт (раздел) или подпункт ввелся не в нужном месте, то можно отбуксировать его мышью туда, куда надо.

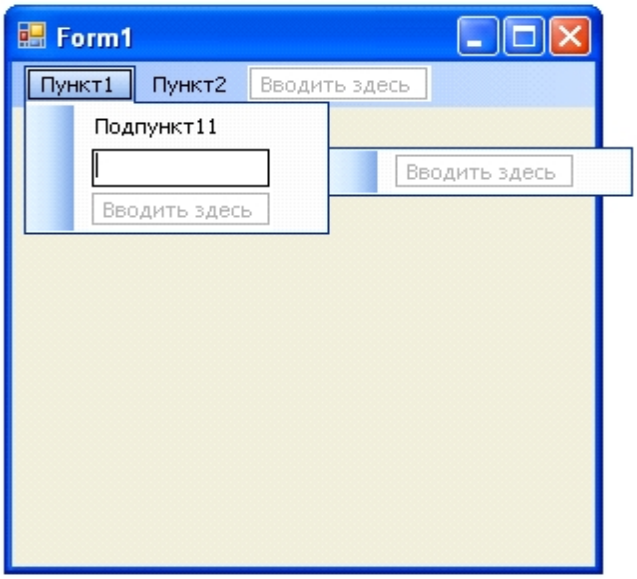

Рис. 2. Процесс формирования структуры меню

Формирование структуры меню может происходить с помощью свойства **Items** компонента **MenuStrip** путем нажатия кнопки с многоточием рядом со свойством **Items** в окне «**Свойства**». В результате откроется окно «Редактор коллекции элементов» (рис. 3).

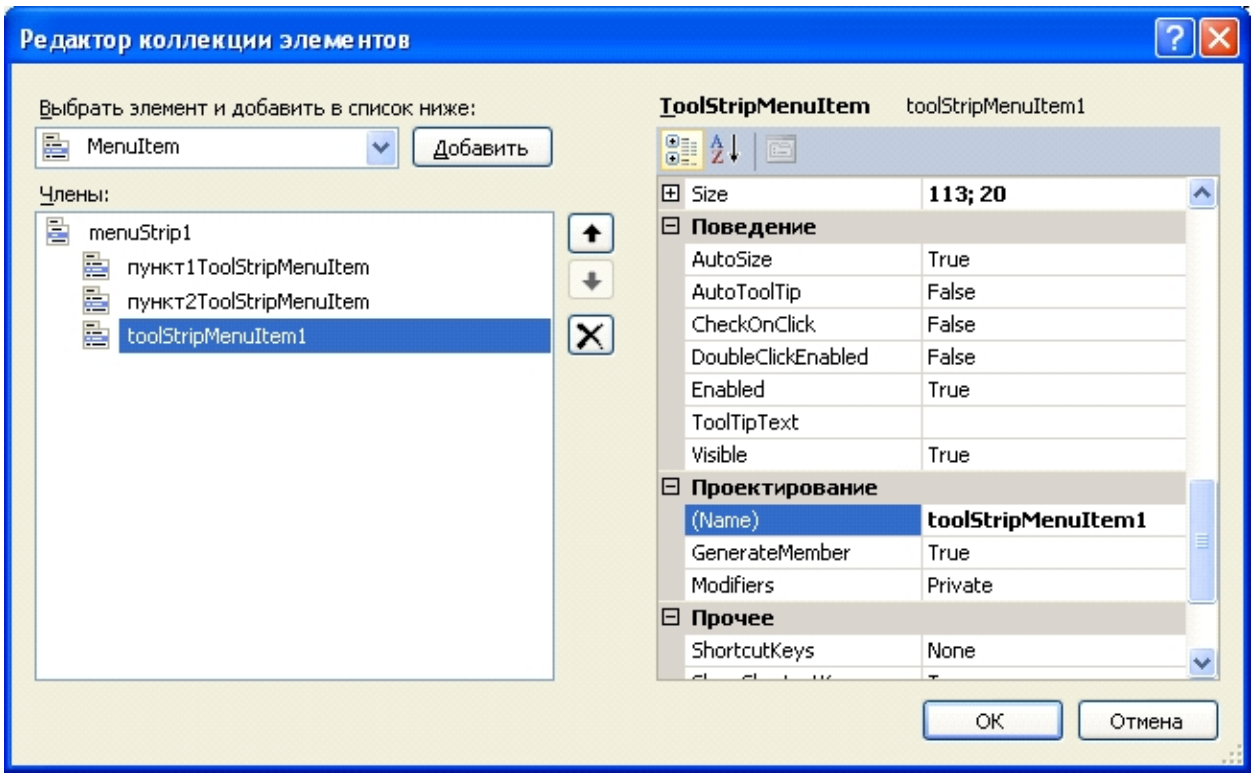

Рис. 3. Вид диалогового окна «Редактор коллекции элементов»

Диалоговое окно позволяет выбирать, добавлять в меню и удалять из меню элементы. При добавлении нового пункта меню необходимо задать название пункта меню (свойство **Text**) и имя (**Name**) элемента **ToolStripMenuItem** (см. табл. 2). При необходимости можно задавать и другие свойства элемента **ToolStripMenuItem** с помощью окна «**Свойства**» (раздел **ToolStripMenuItem**).

При выборе нового элемента в «Редакторе …» можно видеть множество свойств данного объекта, так как каждый раздел (пункт) меню, т.е. каждый элемент свойства **Items**, является объектом типа **MenuItem**, обладающим своими свойствами, методами, событиями. Кроме того, оно является массивом, хранящим все опции меню. Например, количество строк меню с именем **menuStrip1** можно подсчитать с помощью оператора:

menuStrip1.Items.Count;

К опции, которая является **i** –м элементом меню, можно обратиться так:

menuStrip1.Items[i];

Для формирования подпунктов (подопций) меню необходимо воспользоваться свойством **DropDownItems** элемента **ToolStripMenuItem**. Путем нажатия кнопки с многоточием свойства **DropDownItems** откроется диалоговое окно «Редактор коллекции элементов (…)»

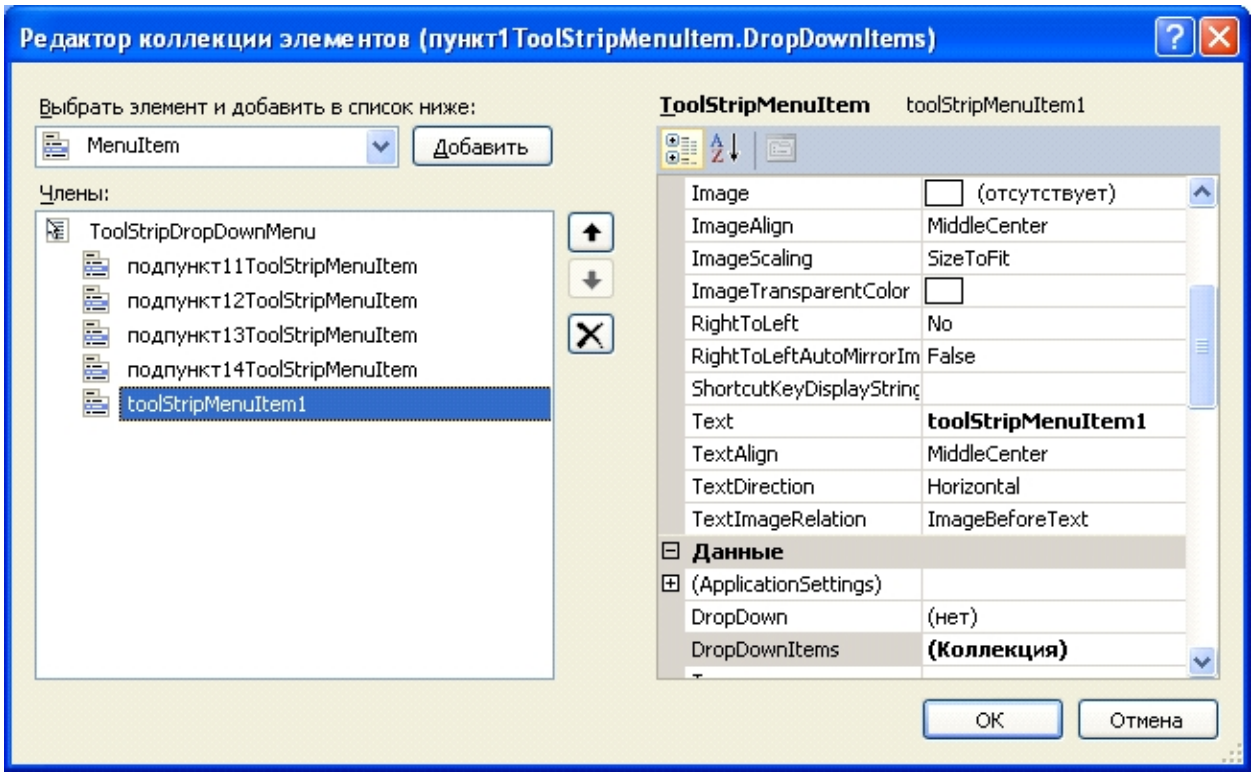

Рис. 4. Вид диалогового окна «Редактор коллекции элементов (…)»

Количество строк (подпунктов) пункта меню с именем **toolStripMenuItem1** можно подсчитать с помощью оператора:

toolStripMenuItem1.Items.Count;

К опции, которая является **i** –м элементом раздела (пункта) меню, можно обратиться так:

toolStripMenuItem1. DropDownItems[i];

Кроме команды в меню можно добавить и другие элементы: **ComboBox, TextBox** и **Separator** (разделитель). Разделитель (горизонтальная ли ния) обычно служит для объединения в группы однотипных ко манд. Чтобы добавить его в список команд, следует выделить команду, перед которой надо поместить разделитель, и из контекстного меню выбрать команду **Вставка→Separator** (рис. 5).

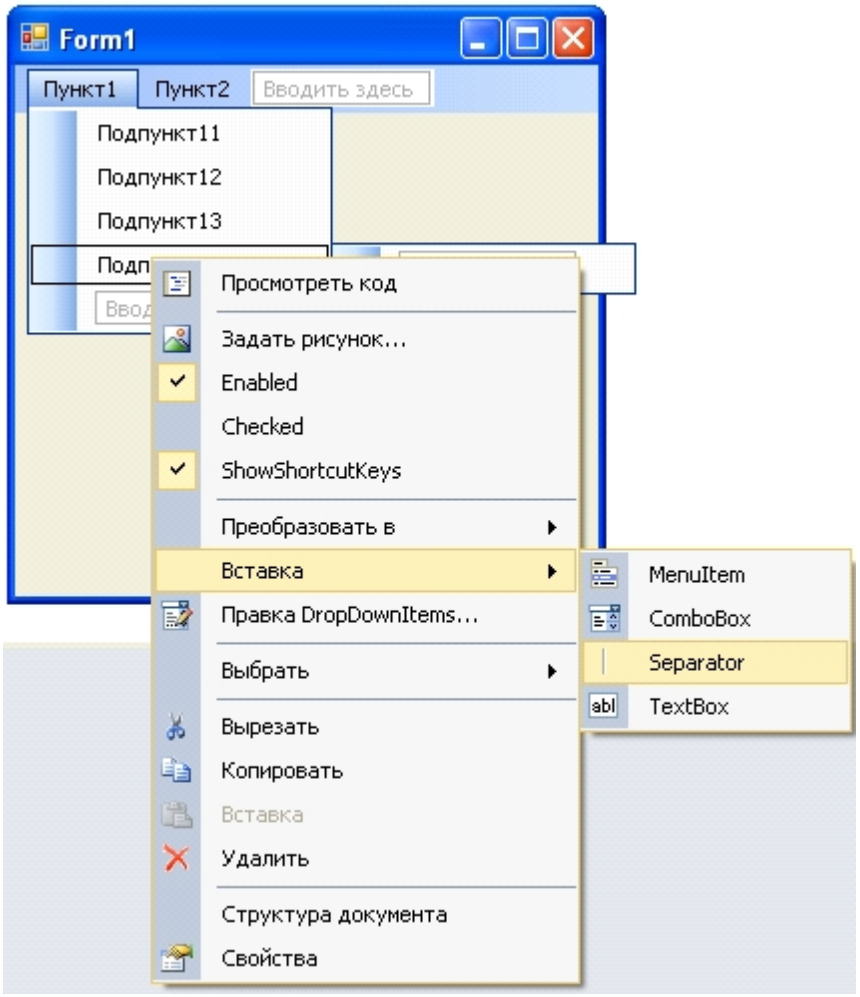

Рис. 5. Вставка в список команд разделителя

После того как структура меню будет сформирована, можно выполнить настройку меню. Настройка выполняется путем изменения значений свойств компонента **MenuStrip** (свойства компонента **MenuStrip** приведены в табл. 1.) и свойств пунктов меню (объектов типа **ToolStripMenuItem**). Свойства объекта **ToolStripMenuItem** приведены в табл. 2.

Таблица 1. Свойства компонента MenuStrip

Свойство	Описание
Name	Имя элемента.
BackgroundImage	Задает (с помощью выбора через диалоговое окно) фоновое изображение, которое помещается в полосу меню и на фоне которого будут видны его опции С помощью свойства **BackgroundImageLayout** изображение можно "подогнать" под соответствующий формат
Dock	Определяет схему причаливания меню к той или иной стороне формы
Enabled	Определяет доступность меню
Items	Через диалоговое окно этого свойства формируются опции меню.
Items.Count	Количество элементов (разделов, пунктов) меню. Расчет количества ведется от 1 (если в списке 10 строк, то **Count** будет равен 10).
LayoutStyle	Стиль размещения меню. Выбирается из выпадающего списка.
Checked	С помощью этого свойства можно контролировать, была ли выбрана данная команда меню. Это очень важно при работе приложения: если в меню множество опций, а некоторые из них уже выполнились, то если не пометить выполненные, возможно по ошибке начать выполнять какую-нибудь опцию снова. Существует свойство **CheckOnClick**, которое (если установить его значение в **True**) обеспечит

	необходимую пометку выполненной опции (но при условии, что её свойство **Checked** тоже имеет значение **True**).
Visible	Свойство позволяет скрыть (**False**) элемент

Таблица 2. Свойства объекта ToolStripMenuItem

Свойство	Описание
Name	Имя элемента, которое образуется из названия элемента и названия объекта **ToolStripMenuItem**, например, **подпункт11ToolStripMenuItem**
Text	Название элемента меню
Image	Картинка, которая отображается рядом с командой
DropDownItems	Через диалоговое окно этого свойства формируются подопции меню.
Enabled	Признак доступности элемента меню. Если значение свойства равно **False**, то элемент меню недоступен (в результате щелчка на элементе меню событие **Click** не происходит, название элемента меню отобра жается инверсным, по отношению к доступному пункту меню, цветом)
Checked	Признак того, что элемент меню выбран. Если значение свойства рав но **True**, то элемент помечается галочкой. Свойство **Checked** обычно применяется для тех элементов меню, которые используются для отображения параметров.
CheckState	Устанавливает трех видовое состояние: флажок включен – **Checked,** флажок выключен – **Unchecked,** состояние неопределенности (ромбик черного цвета) – **Indeterminate.**
ShortcutKeys	Свойство определяет комбинацию клавиш, нажатие которой активизирует выполнение команды
ShowShortcutKeys	Определяет, будут ли отображаться клавиши быстрого вызова для данного элемента меню

Объект **ToolStripMenuItem** способен воспринимать событие **Click**, которое происходит в результате щелчка на элементе меню, нажатия клавиши <**Enter**>, если элемент меню выбран, или в результате нажатия функциональ ной клавиши, указанной в свойстве **ShortcutKeys**.

Для формирования структуры меню можно воспользоваться диалоговым окном **MenuStrip Задачи**, которое раскрывается с помощью щелчка на кнопке в виде квадратика с черных треугольников в верхнем правом углу компонента **MenuStrip** (см. рис. 1). Это диалоговое окно обеспечивает доступ к типичным командам и свойствам (рис. 6):

- Внедрить в ToolStripContainer — позволяет (щелчком мыши) поместить меню в специальный контейнер (вместо расположения его в форме). Контейнер — это объект со своим набором свойств, установка которых позволяет создавать меню, более удобное для пользователя.

- Вставить стандартные элементы — добавляет общепринятые опции меню.

- RenderMode — опция, которая дает возможность выбора из выпадающего списка способа изображения меню: системного (System), профессионального (Professional) или управляемого (ManagerRenderMode).

- Dock — выводит (по щелчку на кнопке) в поле этой опции схему причаливания меню к той или иной стороне формы.

- GripStyle — в этой опции существует выпадающий список, задающий элемент стиля полосы меню: невидима или видима будет специальная пунктирная канавка в верхней части полосы.

- Правка элементов — с помощью этой опции и задаются опции самого меню. Если щелкнуть на этой опции, то откроется диалоговое окно для задания опций меню, причем в левой его части существует окно для добавления новых опций, а в правой части открывается окно для настройки свойств добавляемых опций. С помощью этого окна можно не только добавлять новые опции, но и удалять и реорганизовывать их.

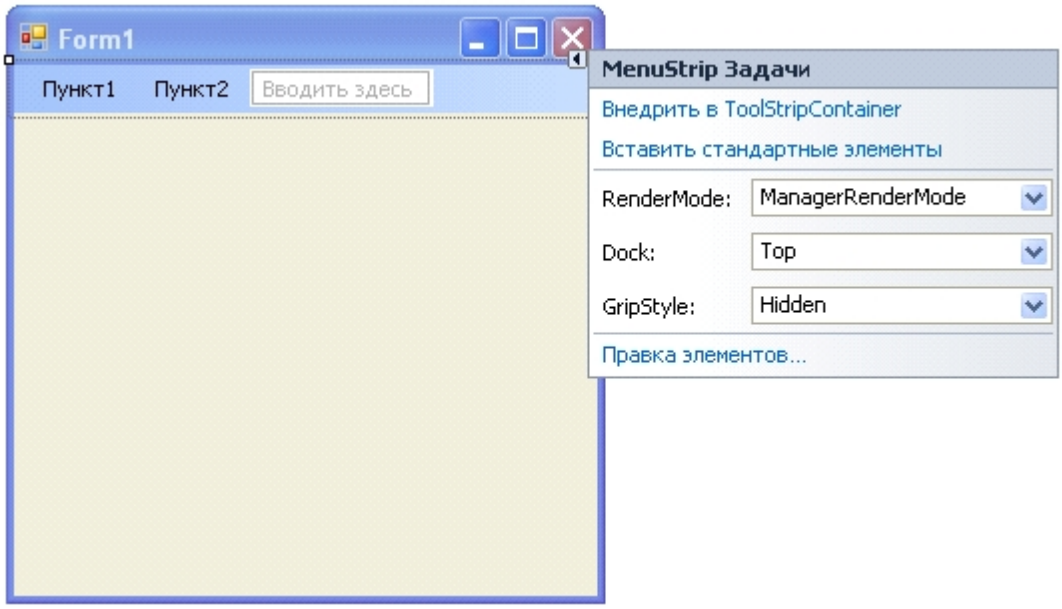

Рис. 6. Диалоговое окно «MenuStrip Задачи»

1.2. Компонент ContextMenuStrip

Компонент **ContextMenuStrip** представляет собой контекстное меню — список команд, который отображается в результате щелчка правой кнопкой мыши на форме или в поле компонента Этот компонент придает другому компоненту, с которым он связывается, дополнительные функциональные возможности, он может быть связан с любым другим компонентом (формой, кнопкой и т. д.), имеющим свойство **ContextMenuStrip**. Когда компонент помещается в форму, его имя будет видно в любом из компонентов формы, у которого есть свойство **ContextMenuStrip**. Это обычное меню, где пользователь определяет порядок действий при активизации компонента, с которым данное меню связано. Если меню связано с формой, то оно появляется, когда пользователь нажимает в активной форме правую кнопку мыши. Меню появляется в том месте, где находится мыши. Задание опций контекстного меню аналогично заданию опций в главном меню.

Команды меню определяют значение свойства **Items**, представляющего собой коллекцию объектов **MenuItem**. Свойства объ екта **MenuItem** приведены в табл. 3. Чтобы задать контекстное меню компо нента или формы, нужно указать имя контекстного меню (компонента **ContextMenuStrip**) в качестве значения свойства **ContextMenuStrip**.

Таблица 3. Свойства объекта MenuItem

Свойство	Описание
Text	Команда
Enabled	Признак доступности команды. Если значение свойства равно **False**, то команда недоступна (название команды отображается серым цветом)
Image	Картинка, которая отображается в строке команды
Checked	Признак того, что элемент меню выбран. Если значение свойства равно **True**, то элемент считается выбранным и помечается галочкой. Свой ство **Checked** обычно используется для тех элементов меню, которые предназначены для отображения параметров

2. Рабочее задание

Задание 1. Разработать приложение «Пример ”Меню”», которое реализует выбор пользователем различных текстов и позволяет изменять свойства текста. Приложение должно решать следующие задачи:

- регистрировать пользователя (администратора) – пункт меню «**Регистрация**». Должен содержать подпункты «Администратор», «Гость» и «Выход»;
- изменять содержания текста – пункт меню «**Текст**». Должен содержать подпункты «Текст1», «Текст2», «Текст3» и «Текст4» и соответствующие комбинации горячих клавиш (см. рис. 8);
- изменять свойств текста – пункт меню «**Свойства**». Должен содержать разветвлённое меню (на рис. 9 представлены разновидности цветов шрифта). Разновидности шрифтов задать: **«Arial», «Courier New»** и **«Times New Roman»**. Размер шрифта задать: **10, 12** и **14**;
- справочные сведения – пункт меню «**Помощь**». Должен содержать подпункт «О программе».

Алгоритм работы приложения должен быть следующим:

- при первоначальном запуске доступен только пункт «Регистрация». Остальные пункты меню недоступны. При выборе пункта «Регистрация» появляется приглашение зарегистрироваться (вариант приглашения представлен на рис. 7);

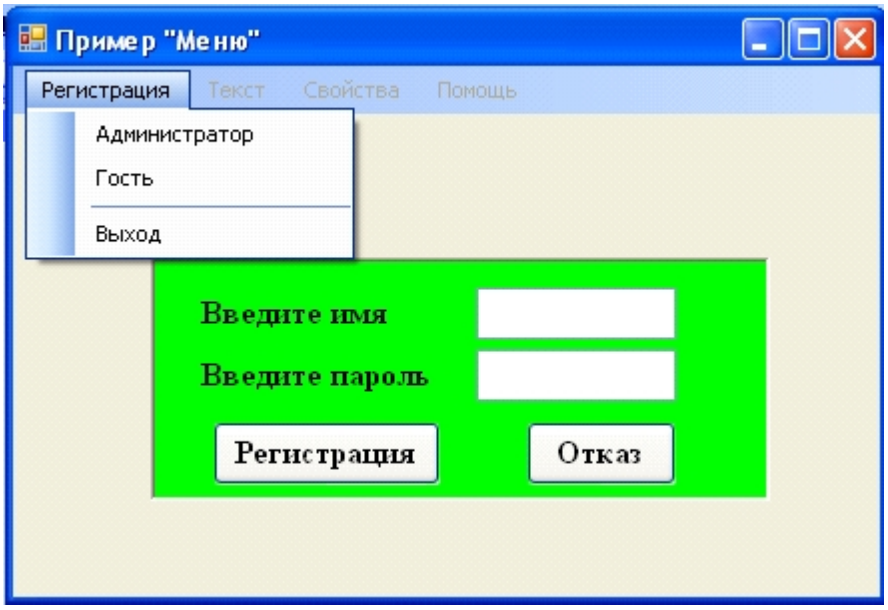

Рис. 7. Вид приглашения к регистрации

- при правильном вводе имени и пароля пункты меню «Текст», «Свойства» и «Помощь» становятся доступны, а пункт «Регистрация» - недоступен. При неправильном вводе должно выдаваться сообщение об ошибке и повторном вводе имени и пароля;
- при выборе подпунктов «Текст» должно меняться содержание текста. В качестве элемента отображения необходимо выбрать компонент **TextBox** (вариант выбора подпункта «Текст1» представлен на рис. 6). Тексты должны содержаться в свойствах компонентов **textBox1, textBox2, textBox3, textBox4**, в которые они записываются в процессе проектирования или формируются во время работы приложения;

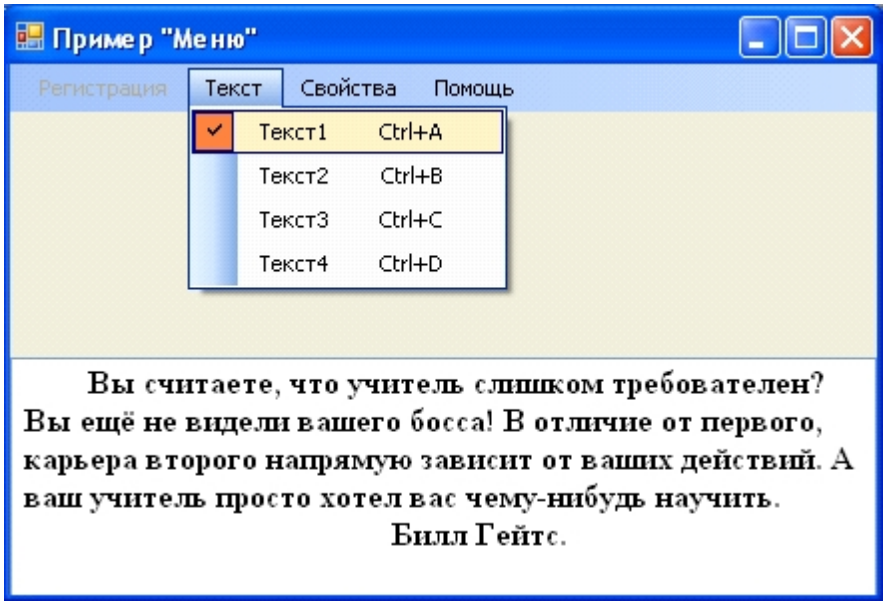

Рис. 8. Вариант вида приложения при работе с пунктом меню «Текст»

- при выборе подпунктов «Свойства» должны меняться свойства текста: цвет, шрифт, размер (вариант выбора подпункта «Цвет» представлен на рис. 7).

Для изменения цвета, например, красного цвета, используется оператор:

textBox1.ForeColor = Color.Red;

Для изменения шрифта и размера, например, **Times New Roman**, размер **12**, используется оператор:

textBox1.Font = new System.Drawing.Font("Times New Roman", 12);

где **Font("Times New Roman", 12)** – конструктор, который инициализирует новый шрифт, используя указанный размер (первый параметр – имя шрифта, второй – размер шрифта).

Для изменения шрифта, размера и начертания, например, **Tahoma**, размер**16**, начертание – **полужирный**, используется оператор:

textBox1.Font = new System.Drawing.Font("Tahoma", 16, FontStyle.Bold);

Класс **Font** определен в пространстве имен **System.Drawing**.

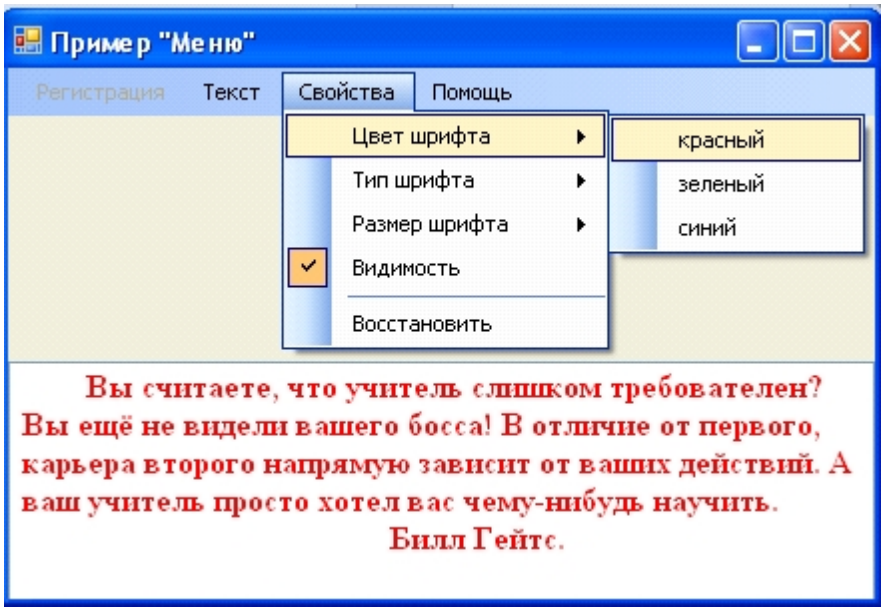

Рис. 9. Вариант вида приложения при работе с пунктом меню «Свойства»

При выборе подпункта «Видимость» содержание текста должно становиться невидимым или видимым в зависимости от значения свойства **Checked**.

При выборе подпункта «Восстановить» свойства текста должны соответствовать первоначальным значениям, т. е. значениям выбранных при проектировании;

- при выборе подпункта «О программе» пункта «Помощь» приложение должно выдавать информацию о разработчике приложения.

 **Задание 2.** Модифицировать приложение путем добавления контекстного меню для компонента **TextBox**. В контекстном меню поместить команды пункта главного меню «Свойства» и в файл реализации проекта «Пример ”Меню”» дописать коды обработки пунктов всплывающего меню.

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.