

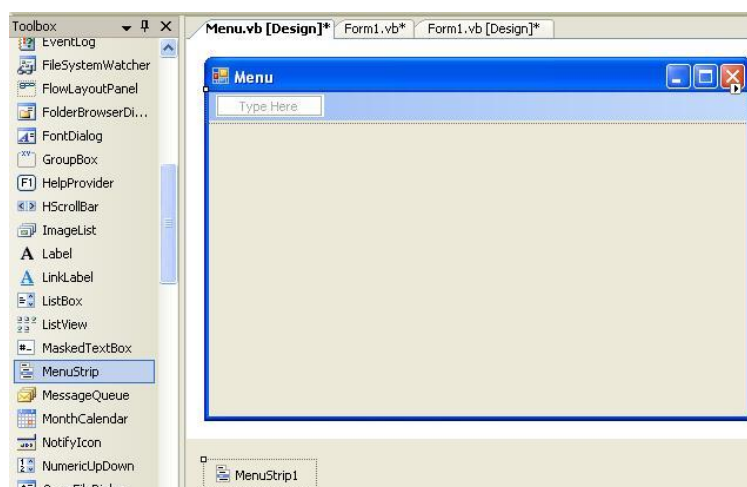
Создание меню

Задача 1

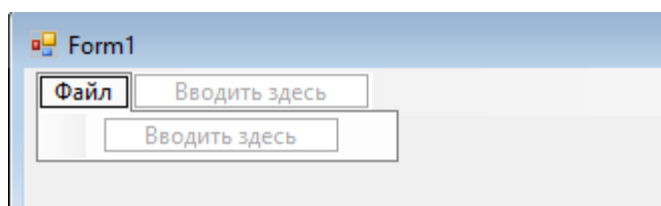
Создать приложение Блокнот. Реализовать функционал пунктов меню Файл и Правка.

Добавление в форму меню осуществляется при помощи элемента управления MenuStrip. Элемент MenuStrip это элемент управления, имеющий видимый интерфейс и тем не менее расположенный внизу формы вместе с другими элементами управления, не имеющими интерфейса (например, элемент управления Timer).

Поместите на форме новый элемент управления MenuStrip, дважды щелкнув на соответствующем элементе набора инструментов. Наверху формы вы увидите текст Type Here.



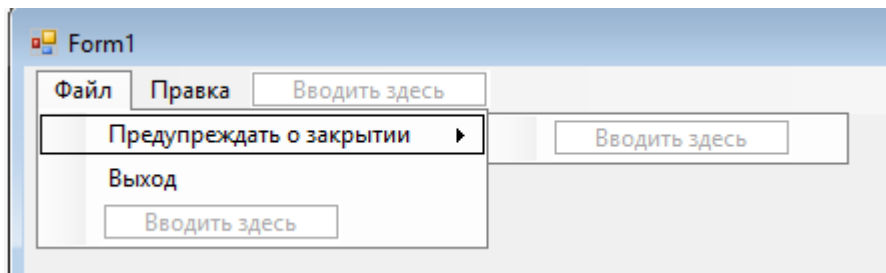
Щелкните на этом тексте и напечатайте Файл. Когда вы начинаете вводить текст, VS отображает два новых поля с надписью Type Here (добавить здесь):



Текст, который был введен, создал новый пункт меню. Каждый пункт меню — это объект, следовательно, у каждого пункта есть свойства, которые отображаются в окне Properties. Введенный текст изменил свойство Text пункта меню.

Щелкните на тексте Type Here, который расположен справа от пункта **Файл**, и введите текст **Правка**. При этом появляются еще два элемента **Type Here**, так же как и при добавлении пункта меню **Файл**.

Щелкните еще раз на пункте **Файл**, чтобы отобразить поле **Type Here** под этим пунктом меню. Щелкните на поле и вводите названия подпунктов меню **Предупреждать о закрытии** и **Выход**.



Чтобы удалить пункт меню, щелкните на нем правой кнопкой мыши и выберите Delete из контекстного меню. Чтобы переместить пункт меню, протяните его с его текущего места на место, где вы хотите его поместить. Чтобы между пунктами меню была разделительная линия, необходимо в списке **Type Here** выбрать Separator.

Создание выделенных пунктов меню

Пункт меню может отображать "птичку" рядом со своим текстом. Значки "птички" в меню используются для создания пунктов меню, по которым отображается состояние – выбран или не выбран пункт меню. Создайте пункт меню с отображением состояния. Для этого щелкните на поле **Type Here** под пунктом меню **Выход** и введите **Предупреждать о закрытии**, затем поменяйте свойство **Checked** этого пункта меню на **True**. Рядом с элементом меню теперь появилась "птичка".

Программирование меню

Для задания команд для пункта меню **Выход** дважды щелкните на нем и в открывшемся шаблоне процедуры **Click** введите оператор закрытия формы:

```
Close()
```

Создайте программный код пункта **Предупреждать о закрытии**.

Дважды щелкните на пункте **Предупреждать о закрытии** и введите следующую строку программы в событии **Click**:

```
предупреждатьОЗакрытииToolStripMenuItem.Checked= !предупреждатьОЗакрытииToolStripMenuItem.Checked;
```

Эта команда программы будет изменять состояние пункта меню с выбранного на невыбранное (изменять значение свойства **Checked**); если **Checked = True**, то оно меняется на **False**, и наоборот.

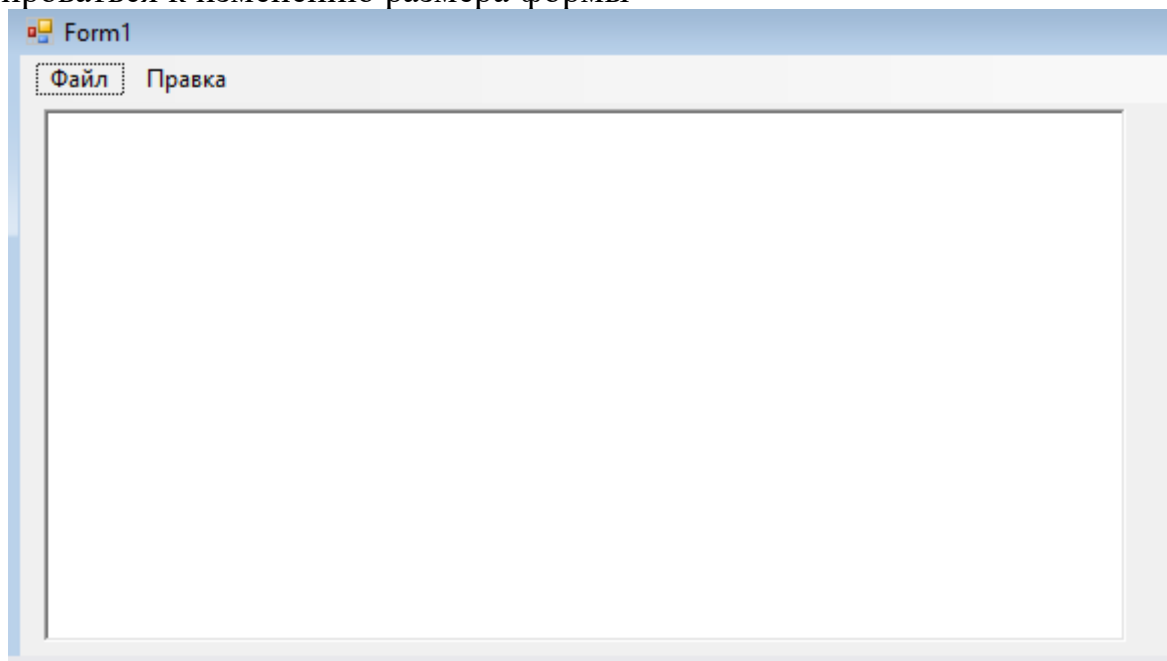
! используется для отрицания логического значения. Следовательно, при каждом щелчке на данном пункте меню его состояние переключается с выбранного на невыбранное и наоборот.

В событии **Click** пункта **Выход** добавьте анализ значения **Checked** пункта меню **Предупреждать о закрытии**:

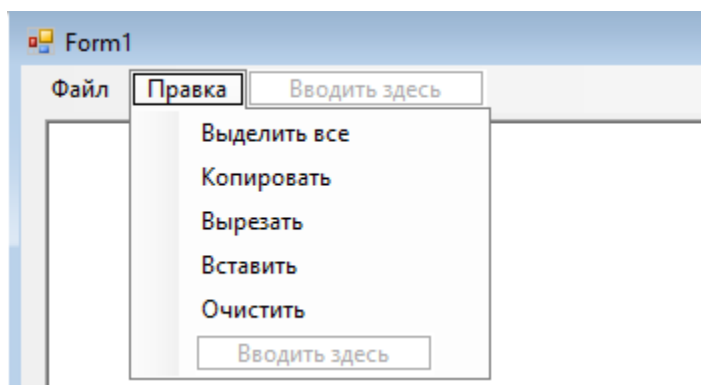
```
DialogResult rz = new DialogResult();
if (предупреждатьОЗакрытииToolStripMenuItem.Checked)
{
    rz = MessageBox.Show("Завершить программу", "Завершение", MessageBoxButtons.YesNo);
    if (rz == DialogResult.Yes)
        Close();
}
else
    Close();
```

Пункт меню Правка

Для работы с текстом в приложении добавьте на форму элемент управления RichTextBox. Чтобы элемент RichTextBox1 изменял свой размер при увеличении окна, то установите для свойства Dock элемента RichTextBox1 значение **Fill**. Это позволит элементу заполнять доступное пространство и автоматически адаптироваться к изменению размера формы



Пункт меню **Правка** должен содержать команды для работы с буфером обмена.



Для этого используются следующие методы текстового поля RichTextBox.

```
RichTextBox1.Copy();
RichTextBox1.Cut();
RichTextBox1.Paste();
```

Очистить и выделить все:

```
RichTextBox1.Clear();
RichTextBox1.SelectAll();
```

RichTextBox

Этот элемент управления напоминает текстовое поле TextBox, но дает более сложные возможности форматирования. Элемент управления RichTextBox происходит непосредственно из класса TextBoxBase. Этот элемент управления предоставляет широкие возможности для создания специальных элементов для конкретных задач. Например, свойства для выделенного текста.

Свойства, методы и события класса RichTextBox, не унаследованные из других классов:

События и методы	О п и с а н и е
SelectionAlignment	Определяет выравнивание текущего выделения или позиции курсора <i>// Устанавливает выравнивание по центру для текущего выделенного текста в RichTextBox</i> <pre>richTextBox1.SelectionAlignment = HorizontalAlignment.Center;</pre>
SelectionCharOffSet	Определяет режим отображения текста: нормальный, верхний индекс или нижний индекс
SelectionColor	Определяет цвет текста для выделенного в данный момент текста или позиции курсора <i>// Обработчик нажатия кнопки "Изменить цвет текста"</i> <i>// Открываем диалоговое окно выбора цвета</i> <i>// ShowDialog() показывает окно и ждет, пока пользователь нажмет ОК или Отмена</i> <pre>if (colorDialog1.ShowDialog() == DialogResult.OK) { // Если пользователь нажал ОК (выбрал цвет и подтвердил) // Применяем выбранный цвет к выделенному тексту в RichTextBox richTextBox1.SelectionColor = colorDialog1.Color; }</pre>
SelectionFont	Определяет шрифт для выделенного в данный момент текста или позиции курсора <i>// Применяем выбранный шрифт к выделенному тексту в RichTextBox</i> <pre>richTextBox1.SelectionFont = fontDialog1.Font;</pre>
SelectionBackColor	Определяет цвет фона выделенного текста <i>/ Устанавливаем цвет фона для выделенного текста</i> <pre>richTextBox1.SelectionBackColor = colorDialog1.Color;</pre>
SelectionLength	Количество выделенных символов
BackColor	Возвращает или задает цвет фона <pre>richTextBox1.BackColor = colorDialog1.Color;</pre>

SelectionTabs	Определяет абсолютные позиции остановок по табуляции в элементе управления
SelectionType	Доступно только для чтения. Возвращает тип выделения для элемента управления. Используется для определения атрибутов выделенного в данный момент текста
ShowSelectionMargin	Определяет, отображается ли в элементе управления граница выделения. Она похожа на заголовок строки в таблице данных и дает возможность пользователю легко выделять полные строки
ZoomFactor	Определяет текущий масштаб элемента управления
CanPaste	Возвращает значение, указывающее на то, можно ли перенести в элемент управления текущее содержимое буфера обмена

Задача 2

Добавить в приложение Панель инструментов. Для всех команд пункта меню Правка создать кнопки на панели инструментов. Добавить для них всплывающие подсказки. Для Команды Очистить создать горячие клавиши. Для элемента RichTextBox создать контекстное меню с командами Копировать, Вырезать, Вставить

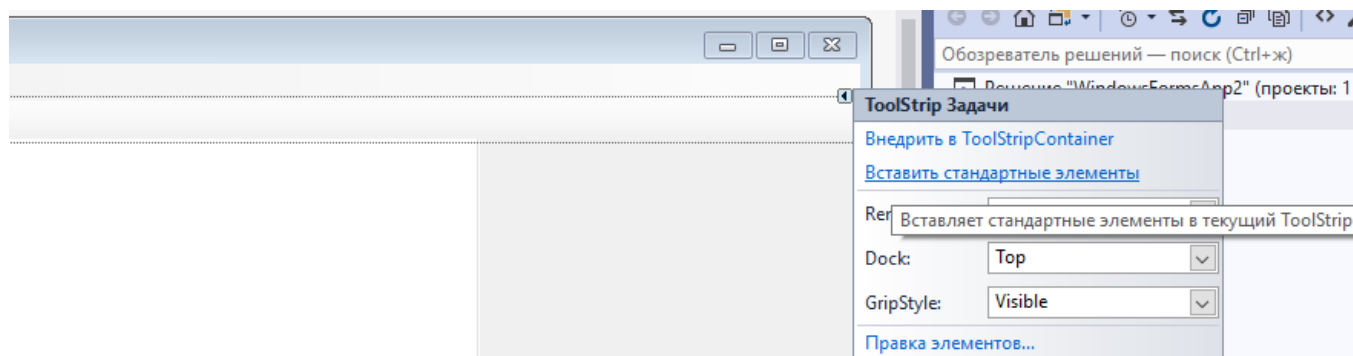
Решение задачи 2:

Панели инструментов

Кнопки на панелях обычно дублируют наиболее часто используемые команды главного меню. Чтобы создавать панель инструментов, поместите на форму элемент ToolStrip.

Установить кнопки можно непосредственно на панели двойным щелчком по пустой кнопке или с помощью свойства Items, которое открывает окно для формирования панели инструментов.

Можно загрузить стандартные кнопки, если открыть дополнительное окно настройки элемента управления, нажав на стрелочку в правом верхнем углу ToolStrip.



Выберите пункт Insert ToolStripContainer (Вставить стандартные элементы) и на панели инструментов появится стандартный набор кнопок:



Из этого набора можно удалить ненужные кнопки (в нашем случае – печать) и добавить недостающие, загрузив в них изображения небольшого размера с помощью свойства Image.

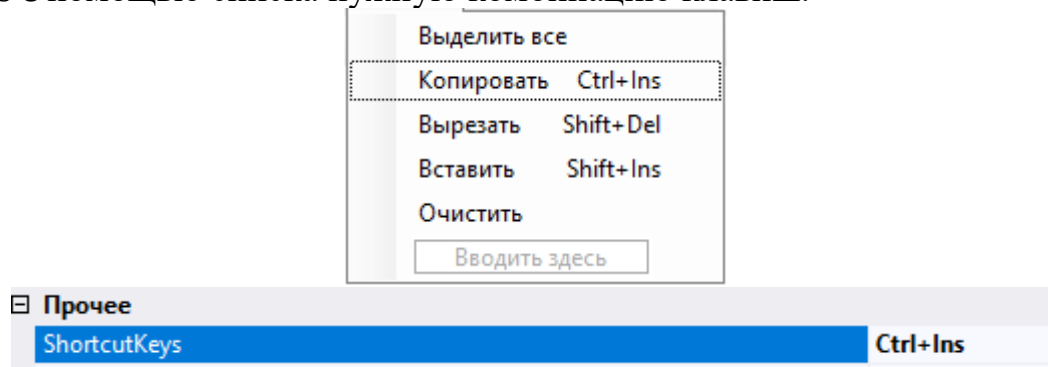
Элемент управления ToolTip (всплывающая подсказка)

Всплывающие подсказки появляются при наведении мышью на элемент управления, находящийся на форме.

С помощью свойства ToolTipText отдельной кнопки измените всплывающую подсказку.

Создание горячих клавиш

В меню можно добавить комбинации клавиш, с помощью которых можно выполнить команды меню. Для этого для пункта меню в окне Properties определите ShortcutKeys с помощью списка нужную комбинацию клавиш.

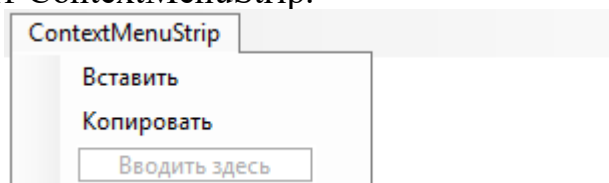


Создание контекстных меню

Контекстное меню – это "всплывающее" меню, которое появляется при щелчке на объекте формы с помощью правой кнопки мыши. Они называются контекстными.

Создание контекстных меню очень напоминает создание обычных меню, хотя контекстные меню используют другой элемент управления – ContextMenuStrip.

Добавьте в форму новое контекстное меню ContextMenuStrip из набора инструментов. Как и элемент управления MenuStrip, элемент ContextMenuStrip появляется в области под конструктором формы. Когда элемент выбран, вверху формы появляется элемент ContextMenuStrip.

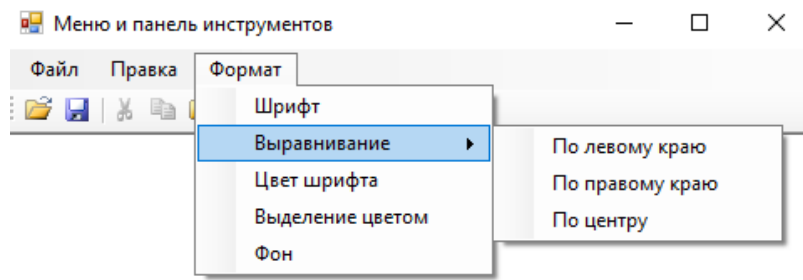


Присоединение элемента управления к контекстному меню RichTextBox осуществляется через задание свойства ContextMenu элемента управления на ContextMenuStrip1.

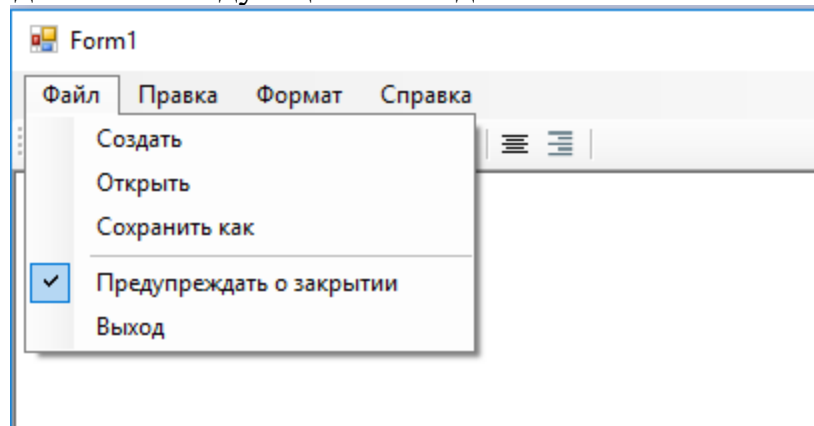
Поведение	
AcceptsTab	False
AutoWordSelection	False
BulletIndent	0
ContextMenuStrip	contextMenuStrip1

Задание для самостоятельной работы

Необходимо добавить в меню команду Формат с командами форматирования текста и оформления документа. Для этого использовать диалоговые окна.



В меню **файл** добавить следующие команды:



Класс CommonDialog

Предназначен для создания стандартных диалоговых окон. CommonDialog имеет дочерние классы: OpenFileDialog; ColorDialog; FontDialog; PageSetupDialog; PrintDialog.

Всего стандартных диалогов, имеющих в WindowsForm – восемь, и все она находятся на панели инструментов Toolbox

- OpenFileDialog – служит для отображения диалогового окна открытия файла;
- SaveFileDialog – используется для отображения диалогового окна сохранения файла;
- ColorDialog – служит для отображения диалогового окна выбора цвета;
- FontDialog – служит для отображения диалогового окна выбора шрифта;
- FolderBrowserDialog – служит для отображения диалогового окна обзора папок
- PrintDialog – предназначен для отображения диалогового окна выбора принтера, количества печатаемых страниц и другой информации;

- `PrintPreviewDialog` – применяется для отображения диалогового окна, в котором будет показан текущий документ таким образом, каким он будет выведен на печать;
- `PageSetupDialog` – служит для отображения диалогового окна выбора параметров страницы, предназначенной для печати.

Класс `FileDialog`

Этот класс предназначен для того, чтобы открывать или сохранять файлы. При этом диалоговое окно, которое выводится с помощью класса `FileDialog`, является модальным. В свою очередь класс `FileDialog` делится на `OpenFileDialog` и `SaveFileDialog`.

Сами по себе диалоговые окна не считывают данные с диска и ничего не записывают; они лишь предлагают пользователю удобный интерфейс для выбора требуемых имен файлов.

FileName – основное свойство, которое используется как результат открытия окон `OpenFileDialog` и `SaveFileDialog`.

Пример использования диалогового окна для выбора файлов.

```
openFileDialog1.ShowDialog();
textBox1.Text = openFileDialog1.FileName;
или
if (openFileDialog1.ShowDialog() == DialogResult.OK)
    textBox1.Text = openFileDialog1.FileName;

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)
    return;
// получаем выбранный файл
string filename = openFileDialog1.FileName;
// читаем файл в строку
string fileText = System.IO.File.ReadAllText(filename);
richTextBox1.Text = fileText;
```

Filter Определяет, файлы какого типа будут открываться или сохраняться при помощи данного диалогового окна. Эти типы файлов будут отображаться в поле "Тип файлов"

Фильтр (значение свойства `Filter`) задается при помощи следующего синтаксиса:

Описание1|Маска1|Описание2|Маска2| ... |ОписаниеМ|МаскаМ

Описания соответствует элементам раскрывающегося списка Тип файлов (Files of type), расположенного на диалоговом окне, а маски — это маски файлов, соответствующих описаниям.

Фильтр диалогового окна для открытия файлов с расширением RAR или ZIP задан при помощи строки:

Архивы в формате RAR|*.rar|Архивы в формате ZIP|*.zip

```
openFileDialog1.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";
saveFileDialog1.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";
```

FilterIndex Определяет, какой тип файлов из списка, заданного свойством **Filter**, будет отображаться при загрузке диалогового окна. Значение, равное 1, соответствует первому элементу списка, значение 2 — второму и т. д.

InitialDirectory Задает начальную папку, отображающуюся при загрузке диалогового окна

Title Строка, отображаемая в заголовке диалогового окна

Multiselect Определяет, разрешено ли пользователю выбирать сразу несколько файлов (только для объектов типа **OpenFileDialog**)

CheckFileExists Управляет проверкой наличия выбранного файла перед возвратом из диалогового окна

CheckPathExists Управляет проверкой наличия выбранного пути перед возвратом из диалогового окна

Для **SaveFileDialog** при сохранении файла, если файл с указанным именем уже существует, то необходимо задать вопрос о том, нужно ли перезаписать файл.

Свойство **OverwritePrompt** отвечает за то, будет ли отображаться вопрос о перезаписи текущего файла взамен уже существующего с таким же именем.

CreatePrompt Определяет, следует ли спрашивать у пользователя разрешение на создание нового файла, если его еще нет. Свойство определено лишь для объектов типа **SaveFileDialog**

ReadOnlyChecked Определяет начальное состояние флажка **Open as read-only** (см. описание свойства **Show/ReadOnly**). Свойство задано только для объектов типа **OpenFileDialog**

RestoreDirectory Определяет, следует ли восстанавливать значение текущей папки, заданное до отображения диалогового окна

ShowHelp Управляет отображением кнопки **Справка** в диалоговом окне

ShowReadOnly Определяет, следует ли отображать в диалоговом окне флажок **Open as read-only**. Свойство задано только для объектов типа **OpenFileDialog**

ValidateNames Определяет, будет ли диалоговое окно автоматически следить за допустимостью вводимой в качестве имени файла последовательности символов

```
saveFileDialog1.ShowDialog();
System.IO.StreamWriter sr = new
System.IO.StreamWriter(saveFileDialog1.FileName);
sr.WriteLine(richTextBox1.Text);
sr.Close();
```

```
openFileDialog1.ShowDialog();
System.IO.StringReader sr = new
System.IO.StringReader(openFileDialog1.FileName);
richTextBox1.Text = sr.ReadToEnd();
sr.Close();
```

Класс **FontDialog**

служит для отображения диалогового окна выбора шрифта

Свойства и методы класса **FontDialog**:

ShowDialog – выводит диалоговое окно

ShowColor – определяет, можно ли в диалоговом окне задавать цвет шрифта;

Color – цвет шрифта, который выбрал пользователь

Font – шрифт, его начертание, размер и эффекты

showApply – определяет, должна ли отображаться кнопка Применить (Apply)

Apply (Событие экземпляра Public) – происходит при щелчке пользователя на кнопке Применить в диалоговом окне

Пример изменения шрифта в текстовом поле:

```
FontDialog1.ShowColor = True
FontDialog1.ShowDialog()
TextBox1.ForeColor = FontDialog1.Color
TextBox1.Font = FontDialog1.Font
```

Класс Font служит для определения характеристик шрифта, используемых для форматирования текста, в том числе набор символов (например, Times New Roman или Courier), размер (размер точки) и стиль (жирный, курсив, нормальный, подчеркивание и др.).

Чтобы создать новый тип шрифта, можно использовать такой фрагмент:

```
Font f1 = new Font("Arial", 30)
```

Текст Arial в этом фрагменте — это название шрифта, установленного на компьютере (он присутствует среди шрифтов, установленных на компьютерах с операционной системой Windows). Если вы зададите название шрифта, который не существует, то используется шрифт по умолчанию. Второй параметр – это размер текста.

Если вы хотите использовать стиль, отличный от нормального, вы можете определить значения стиля в качестве третьего параметра:

```
Font f2 = new Font("Arial", 30, FontStyle.Italic)
```

Значения стиля:

Regular - обычный

Bold – полужирный

Underline – подчеркнутый

Striceout – зачеркнутый

Новый шрифт можно определить через существующий и использовать сразу несколько стилей

```
Font f3 = new Font(f1, FontStyle.Italic or FontStyle.Bold)
```

Можно также использовать шрифт существующего объекта или элемента управления

```
f1 = Me.Font
```

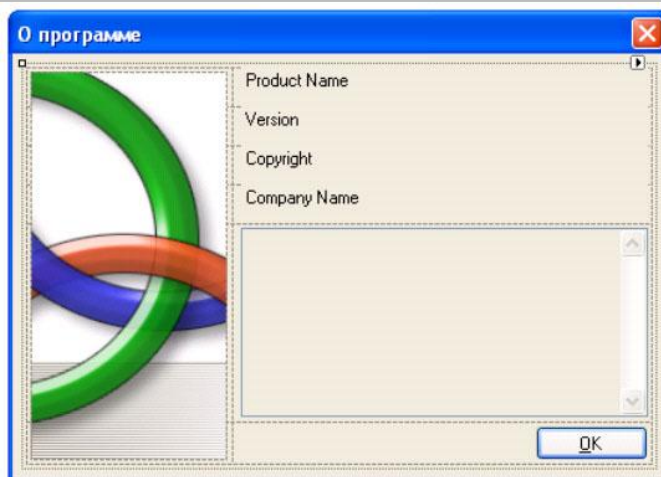
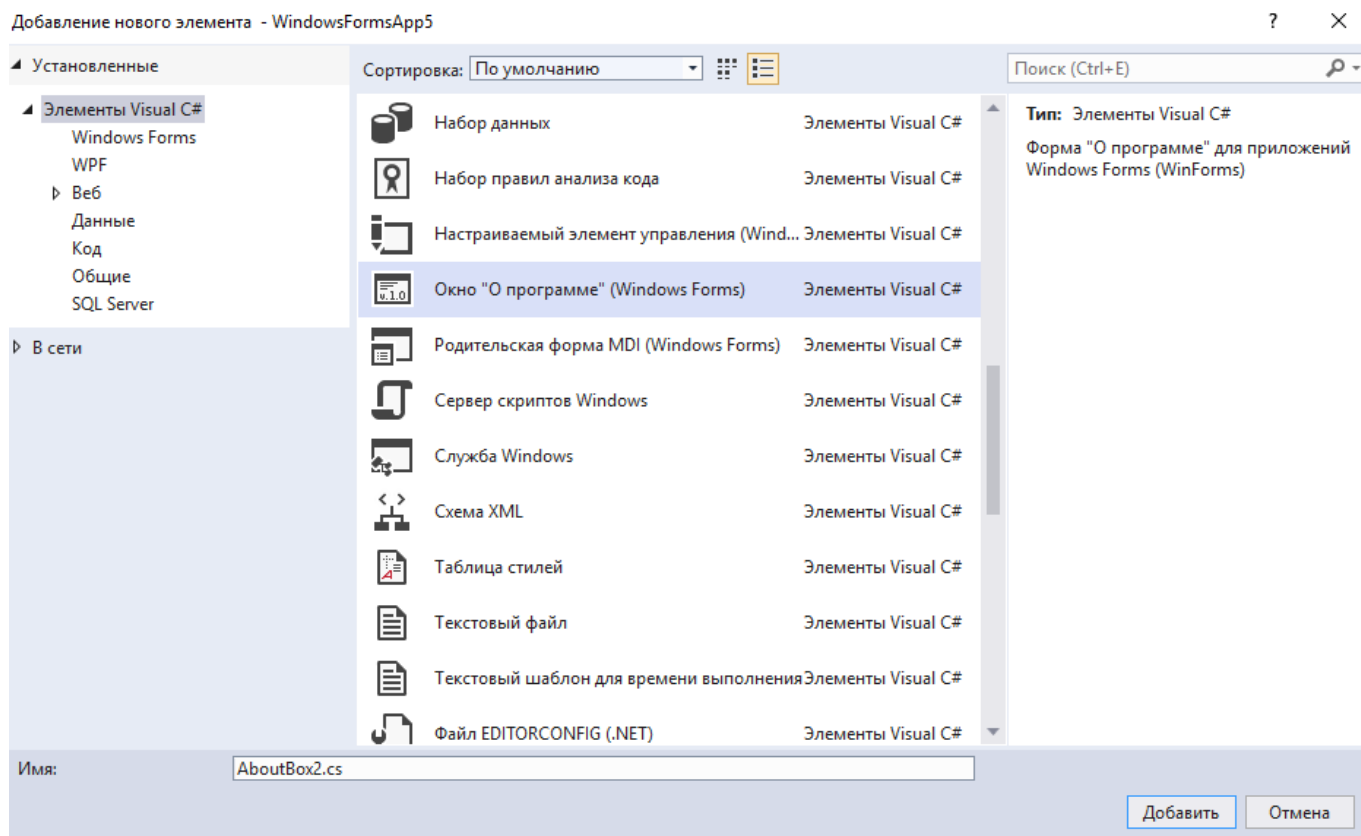
```
Font = textBox1.Font
```

(объект f1 должен существовать)

Нельзя изменить значения свойств объекта класса Font. Нужно создавать новый объект с новыми свойствами.

Диалоговое окно для вывода справочной информации AboutBox

Добавляем новую форму AboutBox (из меню New Item) для вывода справочной информации о разрабатываемом приложении, которое должно иметь вид



По умолчанию класс AboutBox берет дополнительную информацию из метаданных сборки.

Введем собственную информацию. Для этого изменим фрагмент кода конструктора класса AboutBox1 следующим образом.

```
public AboutBox1 ()
{
```

```
InitializeComponent();  
this.Text = String.Format("About {0} {0}", AssemblyTitle);  
this.labelProductName.Text = AssemblyProduct;  
this.labelVersion.Text = String.Format("Version {0} {0}",  
AssemblyVersion);  
this.labelCopyright.Text = "@ПГАУ, 2026"; ;  
this.labelCompanyName.Text = "Иванов И.И."; ;  
this.textBoxDescription.Text = "Студенческий проект";  
}  
  
// Открываем модальное диалоговое окно  
AboutBox1 aboutBox = new AboutBox1();  
aboutBox.ShowDialog();
```