



Основы
разработки приложений
с использованием
Windows Forms

Урок №6

Меню

Содержание

1. Создание и использование меню	3
2. Акселераторы	10
3. Класс MenuStrip	12
4. Создание меню на основе шаблона	14
5. Динамическое создание меню	17
6. Контекстное меню. Класс ContextMenuStrip	19
7. Тулбар. Класс ToolStrip	22
8. Домашнее задание	27

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

1. Создание и использование меню

Большинство современных приложений имеют как меню, так и панели инструментов, которые содержат более или менее идентичный набор команд. Меню дают пользователю возможность заняться "исследованием" приложения, изучать доступные команды, а панели инструментов служат для быстрого доступа к командам, используемым наиболее часто.

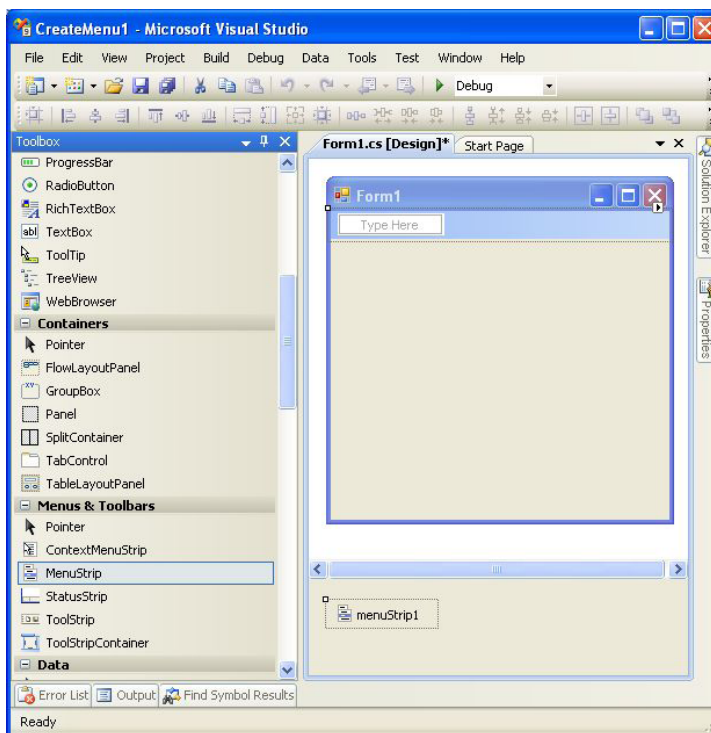


Рисунок 1

Для создания меню, находим в окне **ToolBox** элемент управления **MenuStrip** и перетаскиваем его на форму (рис. 1).

Внизу, под формой вы увидите свое меню. Теперь можно его заполнить. Как только вы введете текст в первый элемент, автоматически меню расширится на один элемент вниз и вправо (рис. 2).

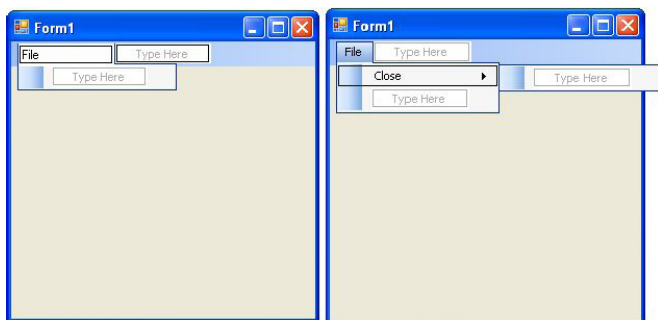


Рисунок 2

Чтобы добавить обработчик для пункта меню, достаточно выполнить на нем двойной щелчок левой кнопкой мыши.

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

    private void vcwToolStripMenuItem_Click(object
        sender, EventArgs e)
    {
        this.Close();
    }
}
```

Пункт меню может представлять из себя обычный текст, выпадающий список, окошко для ввода значений или разделитель, так называемый сепаратор (separator). По умолчанию, когда вы вводите тест, создается обычный текст. Чтобы сделать пункт меню разделителем, достаточно при вводе текста набрать символ «-» (рис. 3).

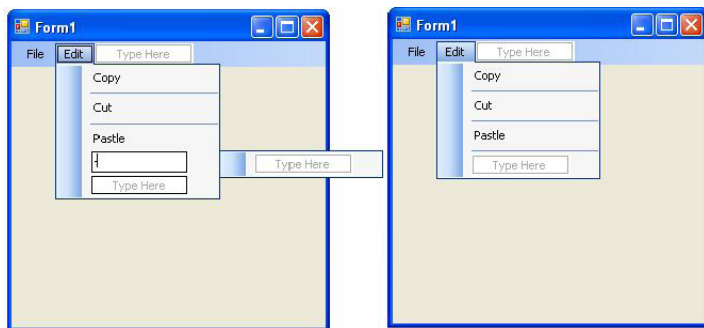


Рисунок 3

Выбрать один из четырех типов меню можно следующим образом:

- Подведите курсор мыши к будущему пункту меню (рис. 4)

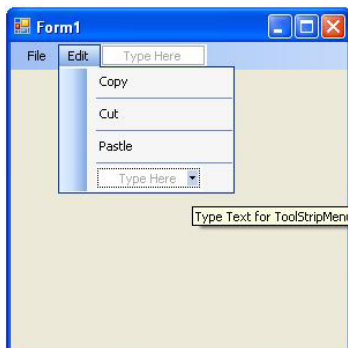


Рисунок 4

Появиться кнопка выпадающего списка

- Нажмите на эту кнопку и выберите желаемый элемент из списка (рис. 5)

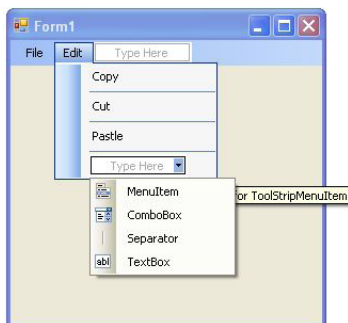


Рисунок 5

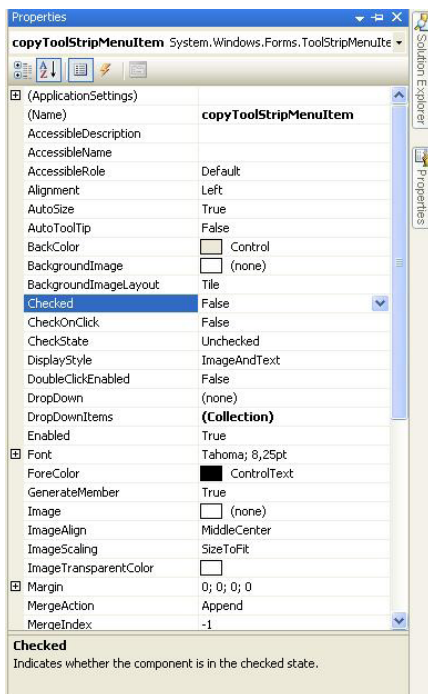


Рисунок 6

Каждый пункт меню может быть помечен галочкой, для этого необходимо зайти в окно **Properties** и установить свойство **Checked** в **true** (рис. 6).

Если вы хотите, чтобы галочка ставилась и снималась при нажатии, то необходимо установить также свойство **CheckOnClick** в **true**.

Давайте рассмотрим пример. Приложение содержит следующее меню (рис. 7):



Рисунок 7

Для пункта меню **Red** установим свойство **CheckOnClick** в **true**. При выборе этого пункта меню, мы будем менять цвет фона формы на красный.

```
public partial class Form1 : Form
{
    Color c;
    public Form1()
    {
        InitializeComponent();
        c = this.BackColor; // запомнили текущий цвет
                           // формы
    }

    private void closeToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
```

```

private void redToolStripMenuItem_Click(
    object sender, EventArgs e)
{
    ToolStripMenuItem it =(ToolStripMenuItem) sender;
    // если галочка стоит - меняем фон на красный

    if (it.Checked == true)
    {
        this.BackColor = Color.Red;
    }
    else // меняем обратно фон на серый
    {
        this.BackColor = c;
    }
}
}

```

Вот что получилось (рис. 8):

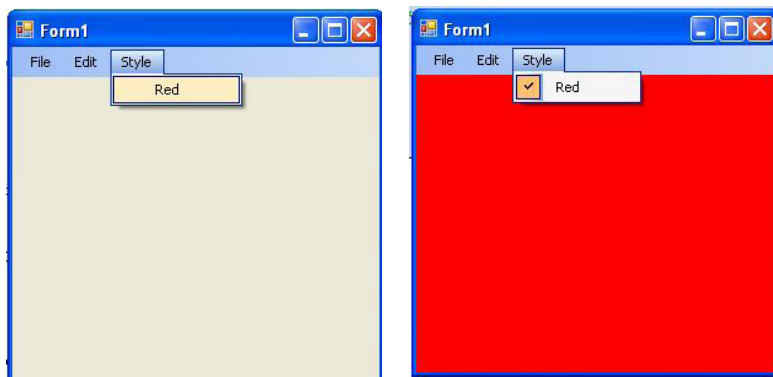


Рисунок 8

Создавая меню, вы также можете указать ориентацию текста, для этого нужно задать свойство **TextDirection**.

И можете задать картинку, воспользовавшись свойством **Image** (рис. 9).

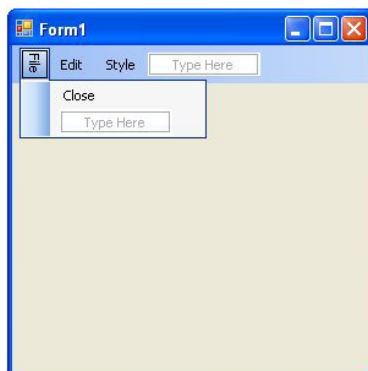


Рисунок 9

2. Акселераторы

Акселератор — это некоторая комбинация клавиш, которая дублирует команду меню. Например, для некоторого меню вы можете создать акселератор **Ctrl+D** — это означает, что и при выборе этого пункта меню мышкой и при нажатии комбинации клавиш **Ctrl+D** будет производиться одно и то же действие.

Для привязки акселератора к пункту меню, необходимо вызвать для пункта меню окно свойств и задать свойство **ShortcutKeys**. Вы можете выбрать любую клавишу, а также комбинацию любой клавиши с **Ctrl**, **Shift** или **Alt** (рис. 10).

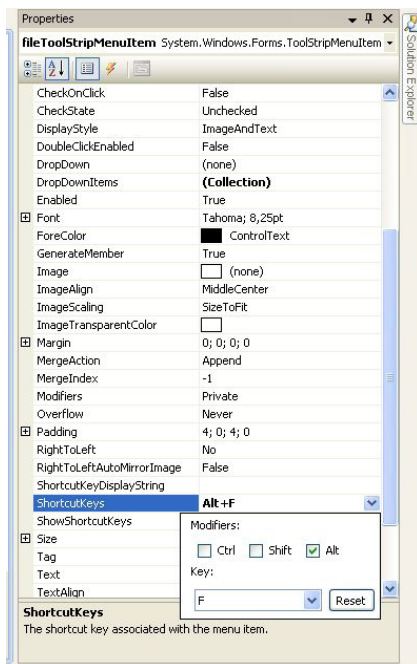


Рисунок 10

Для того, чтобы подсказать пользователю, какой акселератор связан с конкретным пунктом меню, можно установить свойство `ShowShortcutKeys` в `true`. Но для верхнего уровня меню важна краткость (чтобы можно было вместить больше пунктов), поэтому для него не будет отображаться текст акселераторов. Для верхнего уровня можно сделать подсказку, подчеркнув нужную букву, это можно сделать, набрав символ «&» перед желаемой буквой. Такой метод подскажет пользователю, что вызвать данный пункт меню можно также с помощью комбинации `Alt` и подчеркнутой клавиши (рис. 11).

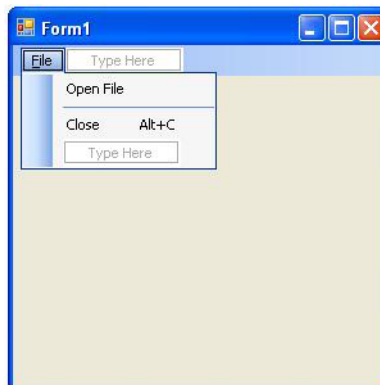


Рисунок 11

3. Класс MenuStrip

За создание меню отвечает класс **MenuStrip**. Давайте познакомимся с ним поближе.

Элемент управления **MenuStrip** представляет контейнер для структуры меню формы. **MenuStrip** может содержать следующие объекты:

- ToolStripMenuItem;
- ToolStripTextBox;
- ToolStripComboBox.

Можно добавить объекты **ToolStripMenuItem** в объект **MenuStrip**, который представляет отдельные команды в структуре меню. Каждый объект **ToolStripMenuItem** может быть командой для приложения или родительским меню для других элементов вложенного меню.

Основные методы класса **MenuStrip**:

Метод	Описание
Contains	Возвращает истину, если меню содержит указанный элемент управления.
GetItemAt	Возвращает элемент, расположенный в заданной точке клиентской области.
Hide	Скрывает элемент управления.
Refresh	Вызывает перерисовку себя и всех дочерних элементов

Основные свойства класса **MenuStrip**:

Свойства	Описание
Items	Получает все элементы, принадлежащие объекту ToolStrip .

Свойства	Описание
MdiWindowsListItem	Возвращает или задает объект <code>ToolStripMenuItem</code> , который используется для отображения списка дочерних форм интерфейса MDI.
Name	Возвращает или задает имя объекта.
Parent	Возвращает или задает родительский контейнер элемента управления.
Visible	Возвращает или задает значение, указывающее отображаются ли элемент управления и все его родительские элементы управления.

Основные свойства класса `ToolStripMenuItem`:

Свойства	Описание
Checked	Получает все элементы, принадлежащие объекту <code>ToolStrip</code> .
CheckOnClick	Получает или задает значение, указывающее, должен ли объект <code>ToolStripMenuItem</code> при щелчке автоматически появляться как установленный или не помеченный.
CheckState	Получает или задает значение, указывающее состояние элемента <code>ToolStripMenuItem</code> — установлен, не отмечен или неопределенное состояние.
DisplayStyle	Получает или задает значение, указывающее, отображаются ли текст и изображения на <code>ToolStripItem</code> .
DropDownItems	Возвращает или задает значение, указывающее отображаются ли элемент управления и все его родительские элементы управления.
Name	Возвращает или задает имя объекта.
Parent	Возвращает или задает родительский контейнер элемента управления.
Visible	Возвращает или задает значение, указывающее отображаются ли элемент управления и все его родительские элементы управления.

4. Создание меню на основе шаблона

Довольно часто в программе возникает необходимость реализовать меню на разных языках или необходимость изменять меню в зависимости от прав пользователя или входных данных. Один из способов реализовать данную функциональность — создать несколько видов меню и организовать «подменю» меню в случае необходимости.

Рассмотрим пример. Создадим форму и два вида меню — русское и английское (рис. 12).



Рисунок 12

Довольно часто в программе возникает необходимость реализовать меню на разных языках или необходимость изменять меню в зависимости от прав пользователя или входных данных. Один из способов реализовать данную функциональность — создать несколько видов меню и организовать «подменю» меню в случае необходимости.

Рассмотрим пример. Создадим форму и два вида меню — русское и английское (рис. 13).

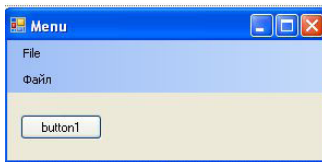


Рисунок 13

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
        menuStripEnglish.Visible = false;
        button1.Text = "English";
    }

    private void closeToolStripMenuItem_Click(
        object sender, EventArgs e)
    {
        this.Close();
    }

    private void button1_Click(
        object sender, EventArgs e)
    {
        if (button1.Text.CompareTo("English") == 0)
        {
            button1.Text = "Русский";
            menuStripEnglish.Visible = true;
            menuStripRussian.Visible = false;
            this.MainMenuStrip = menuStripEnglish;
        }
        else
        {
            button1.Text = "English";
            menuStripEnglish.Visible = false;
            menuStripRussian.Visible = true;
            this.MainMenuStrip = menuStripRussian;
        }
    }
}
```

Вот что получилось:

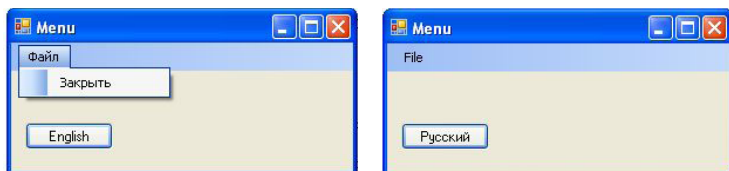


Рисунок 14

5. Динамическое создание меню

Любое меню можно создать динамически. Давайте рассмотрим пример создания простого меню динамически.

```
public partial class Form1 : Form
{
    MenuStrip m;
    public Form1()
    {
        InitializeComponent();
        m = new MenuStrip();

        // добавляем меню верхнего уровня
        ToolStripMenuItem file=
            (ToolStripMenuItem)m.Items.Add("File");
        ToolStripMenuItem edit = (
            (ToolStripMenuItem)m.Items.Add("Edit");
        this.MainMenuStrip = m;
        this.Controls.Add(m); // добавляем меню на форму

        // добавляем выпадающее меню для пункта
        Edit edit.DropDownItems.Add("Cut");

        // добавляем сепаратор
        edit.DropDownItems.Add(new ToolStripSeparator());
        edit.DropDownItems.Add("Copy");

        // добавляем сепаратор
        edit.DropDownItems.Add(new ToolStripSeparator());
        edit.DropDownItems.Add("Paste");
    }
}
```

```

// Добавляем выпадающее меню для пункта File
ToolStripMenuItem close =(ToolStripMenuItem)
    file.DropDownItems.Add("Close");

// связываем меню с акселератором Alt+C
close.ShortcutKeys = Keys.Alt | Keys.C;
close.ShowShortcutKeys = true; // отображать
                               // акселераторы

// добавляем обработчик для пункта меню Close
close.Click += new EventHandler(close_Click);
}

void close_Click(object sender, EventArgs e)
{
    this.Close();
}
}

```

Вот что у нас получилось:

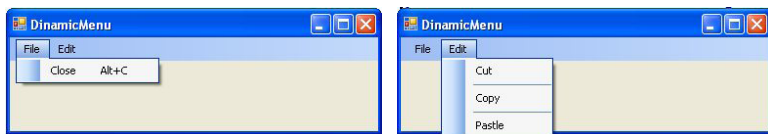


Рисунок 15

6. Контекстное меню. Класс ContextMenuStrip

Контекстное меню — это меню, которое появляется при нажатии правой кнопки мыши. Оно привязывается не к форме, а к любому элементу управления и может быть разным у разных элементов одной формы.

Чтобы добавить контекстное меню, необходимо перетащить на форму из окна **ToolBox** элемент управления **ContextMenuStrip**. Внизу, под формой появиться контекстное меню (рис. 16).

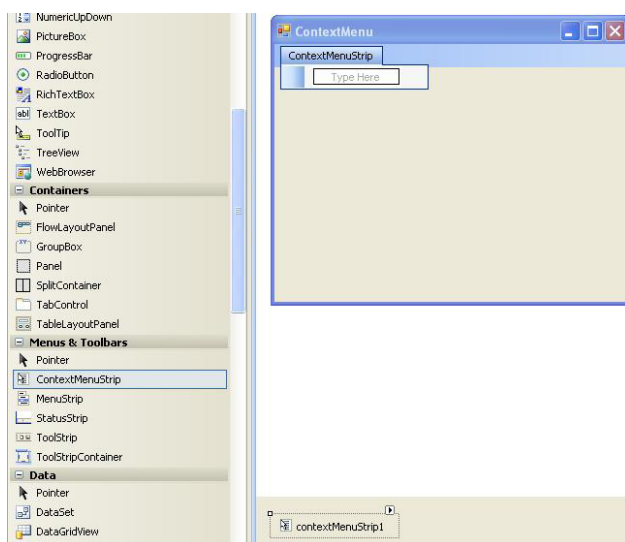


Рисунок 16

Контекстное меню не может содержать элементов верхнего уровня, только выпадающий список. Контекстное меню автоматически не привязывается к форме, для этого

необходимо задать свойство формы `ContextMenuStrip`. Это же свойство можно задать у любого элемента управления (рис. 17).

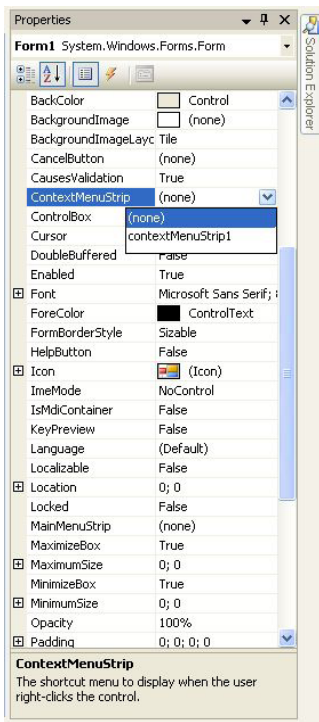


Рисунок 17

Основные методы класса `ContextMenuStrip`:

Метод	Описание
<code>Contains</code>	Возвращает истину, если меню содержит указанный элемент управления.
<code>GetItemAt</code>	Возвращает элемент, расположенный в заданной точке клиентской области.
<code>Hide</code>	Скрывает элемент управления.
<code>Refresh</code>	Вызывает перерисовку себя и всех дочерних элементов

Основные свойства класса ContextMenuStrip:

Свойства	Описание
Items	Получает все элементы, принадлежащие объекту ToolStrip.
Name	Возвращает или задает имя объекта.
Parent	Возвращает или задает родительский контейнер элемента управления.
Visible	Возвращает или задает значение, указывающее отображаются ли элемент управления и все его родительские элементы управления.

Контекстное меню также можно создавать динамически. Рассмотрим пример, в котором контекстное меню добавляется к элементу управления `TextBox`.

```
{
ContextMenuStrip m;
public Form1 ()
{
    InitializeComponent();
    m = new ContextMenuStrip();
    m.Items.Add("Open");
    m.Items.Add("Close");
    textBox1.ContextMenuStrip = m;
}
}
```

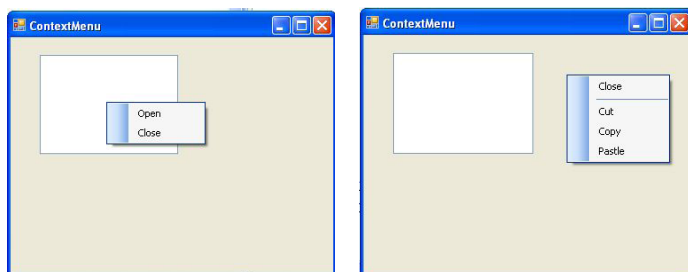


Рисунок 18

7. Тулбар. Класс ToolStrip

ToolBar представляет панель инструментов **Windows**. Хотя элемент управления **ToolStrip** заменяет элемент управления **ToolBar** предыдущих версий и расширяет его функциональные возможности, при необходимости элемент управления **ToolBar** можно сохранить для обратной совместимости и использования в будущем.

Элементы управления **ToolBar** используются для отображения элементов управления **ToolStripButton**, которые могут представлять собой обычные кнопки, кнопки-выключатели или кнопки с раскрывающимся списком. Для кнопок можно назначить изображения. Для этого следует создать объект **ImageList**, назначить его свойству **ImageList** панели инструментов, а затем присвоить значение индекса изображения свойству **ImageIndex** каждой кнопки **ToolStripButton**. Затем можно задать текст, который будет отображаться под изображением или справа от него, установив свойство **Text** объекта **ToolStripButton**. Кнопки панели инструментов можно разделить на логические группы с помощью разделителей. Разделителем является кнопка панели инструментов, у которой свойство **Style** имеет значение **ToolStripButtonStyle.Separator**. Для всех элементов **ToolBar** можно задать только один обработчик.

Рассмотрим пример, который реализует простейший текстовый редактор. Для этого разместим на форме **TextBox** установим ему свойство **MultiLine** в **true** и свойство **Dock** в **Fill**, чтобы растянуть его на всю форму. **ToolBox** будем

создавать динамически, состоящий из трех кнопок: Open, Save, Exit.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
namespace CreateToolBar
{
    public partial class Form1 : Form
    {
        ToolStrip tBar;
        ImageList list;
        public Form1()
        {
            InitializeComponent();
            list = new ImageList();
            list.ImageSize = new Size(50, 50);
            list.Images.Add(new Bitmap("open.bmp"));
            list.Images.Add(new Bitmap("save.bmp"));
            list.Images.Add(new Bitmap("exit.bmp"));
            tBar = new ToolStrip();
            tBar.ImageList = list; // привяжем список
                                // картинок к тулбару
            ToolStripButton toolBarButton1 =
                new ToolStripButton();
            ToolStripButton toolBarButton2 =
                new ToolStripButton();
            ToolStripButton toolBarButton3 =
                new ToolStripButton();
            ToolStripButton separator =
                new ToolStripButton();
        }
    }
}
```

```

separator.Style = ToolBarButtonStyle.
    Separator;
toolBarButton1.ImageIndex = 0; //Open
toolBarButton2.ImageIndex = 1; // save
toolBarButton3.ImageIndex = 2; //exit

tBar.Buttons.Add(toolBarButton1);
tBar.Buttons.Add(separator);
tBar.Buttons.Add(toolBarButton2);
tBar.Buttons.Add(separator);
tBar.Buttons.Add(toolBarButton3);

tBar.Appearance = ToolBarAppearance.Flat;
tBar.BorderStyle = BorderStyle.Fixed3D;
tBar.ButtonClick +=
    new ToolBarButtonClickEventHandler(
        tBar_ButtonClick);
this.Controls.Add(tBar);
}

void tBar_ButtonClick(object sender,
    ToolBarButtonClickEventArgs e)
{
    OpenFileDialog f1;
    SaveFileDialog f2;
    switch (e.Button.ImageIndex)
    {
        case 0:
            f1 = new OpenFileDialog();
            if (f1.ShowDialog() ==
                DialogResult.OK)
            {
                StreamReader r= File.
                    OpenText(f1.FileName);
                textBox1.Text = r.ReadToEnd()
            }
            break;
        case 1:

```



```

        f2 = new SaveFileDialog();
        if (f2.ShowDialog() ==
            DialogResult.OK)
        {
            StreamWriter w =
                new StreamWriter(
                    f2.FileName);
            w.WriteLine(textBox1.Text);
            w.Close();
        }
        break;
    case 2: Close();
        break;
    }
}
}
;

```

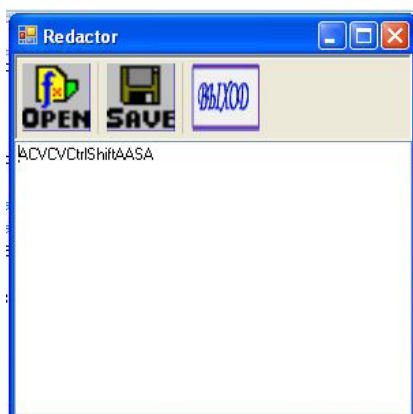


Рисунок 19

ToolStrip является базовым классом для классов MenuStrip, StatusStrip и ContextMenuStrip. Класс ToolStrip class предоставляет множество элементов, обеспечивающих

управление рисованием, вводом с помощью мыши и клавиатуры, а также функции перетаскивания. **ToolStrip** может содержать один из следующих элементов управления:

- **ToolStripButton**;
- **ToolStripSeparator**;
- **ToolStripLabel**;
- **ToolStripDropDownButton**;
- **ToolStripSplitButton**;
- **ToolStripTextBox**;
- **ToolStripComboBox**.

8. Домашнее задание

1. Разработать текстовый редактор, организовать открытие \ сохранение текстовых файлов.
 - В панели инструментов расположить кнопки (Открыть, сохранить, новый документ, копировать, вырезать, вставить, отменить, кнопка настройки редактора (цвет шрифта, цвет фона, ШРИФТ)).
 - Меню должно дублировать панель инструментов (+ выделить всё, +сохранить как).
 - В Заголовке окна находится полный путь к файлу.
 - Организовать контекстное меню для окна редактора (Копировать, Вырезать, Вставить, Отменить).
2. Написать программу «Проводник»
 - При первом запуске программа отображает список доступных дисков.
 - Программа должна содержать дерево дисков, строку адреса, меню, панель инструментов и окно для отображения содержимого папки.
 - Дерево дисков отображает только диски и папки (можно реализовать с помощью [ListBox](#)).
 - При двойном щелчке по папке — в окне содержимого отображаются файлы и подпапки.
 - Программа должна иметь развернутое меню, контекстное меню и возможность работы с горячими клавишами.



Урок №6

Меню

© Компьютерная Академия «Шаг», www.itstep.org

Все права на охраняемые авторским правом фото-, аудио- и видеопроизведения, фрагменты которых использованы в материале, принадлежат их законным владельцам. Фрагменты произведений используются в иллюстративных целях в объеме, оправданном поставленной задачей, в рамках учебного процесса и в учебных целях, в соответствии со ст. 1274 ч. 4 ГК РФ и ст. 21 и 23 Закона Украины «Про авторське право і суміжні права». Объем и способ цитируемых произведений соответствует принятым нормам, не наносит ущерба нормальному использованию объектов авторского права и не ущемляет законные интересы автора и правообладателей. Цитируемые фрагменты произведений на момент использования не могут быть заменены альтернативными, не охраняемыми авторским правом аналогами, и как таковые соответствуют критериям добросовестного использования и честного использования.

Все права защищены. Полное или частичное копирование материалов запрещено. Согласование использования произведений или их фрагментов производится с авторами и правообладателями. Согласованное использование материалов возможно только при указании источника.

Ответственность за несанкционированное копирование и коммерческое использование материалов определяется действующим законодательством Украины.