

Тема 5. Меню и диалоговые окна.

Цель занятия:

**Изучить работу с меню и
диалоговыми окнами.**

Учебные вопросы:

- 1. Меню в Windows Forms.**
- 2. Контекстное меню в Windows Forms.**
- 3. Диалоговые окна в Windows Forms.**
- 4. Компоненты для создания панелей инструментов.**
- 5. Windows Forms обработчик событий**

1. Меню в Windows Forms.

Меню — это элемент пользовательского интерфейса, который предоставляет доступ к различным функциям и командам приложения.

Оно обычно располагается в верхней части окна (главное меню) или может быть контекстным (всплывающим при нажатии правой кнопки мыши).

Меню помогает организовать функционал приложения, делая его более структурированным и удобным для пользователя.

Основные компоненты меню

В Windows Forms для создания меню используются следующие элементы управления:

1. ToolStrip – основной элемент для создания меню

ToolStrip — это контейнер, который представляет собой область для размещения пунктов меню. Он располагается в верхней части формы и может содержать несколько вложенных пунктов и подпунктов.

Основные свойства ToolStrip:

Items — коллекция элементов меню (пунктов, подпунктов, разделителей).

Dock — определяет, как ToolStrip будет закреплен на форме (обычно Top – в верхней части).

RenderMode — позволяет настроить внешний вид меню (например, использовать системный стиль или кастомный).

2. ToolStripMenuItem – элементы меню (пункты, подпункты)

ToolStripMenuItem — это элемент, который представляет собой пункт меню. Он может быть как самостоятельным элементом (например, "Файл"), так и вложенным (например, "Открыть" внутри "Файл").

Основные свойства ToolStripMenuItem:

Text – текст, который отображается в пункте меню.

ShortcutKeys – сочетание клавиш для быстрого вызова команды (например, Ctrl+O для "Открыть").

Click – событие, которое срабатывает при выборе пункта меню.

DropDownItems – коллекция вложенных элементов (подпунктов).

3. Разделители (Separator)

Разделитель — это горизонтальная линия, которая используется для визуального разделения пунктов меню. Это помогает группировать связанные команды и улучшает читаемость меню.

Пример использования:

```
ToolStripMenuItem saveMenuItem = new ToolStripMenuItem("Сохранить");
ToolStripMenuItem exitMenuItem = new ToolStripMenuItem("Выход");

// Добавляем разделитель между "Сохранить" и "Выход"
fileMenuItem.DropDownItems.Add(saveMenuItem);
fileMenuItem.DropDownItems.Add(new ToolStripSeparator());
fileMenuItem.DropDownItems.Add(exitMenuItem);
```

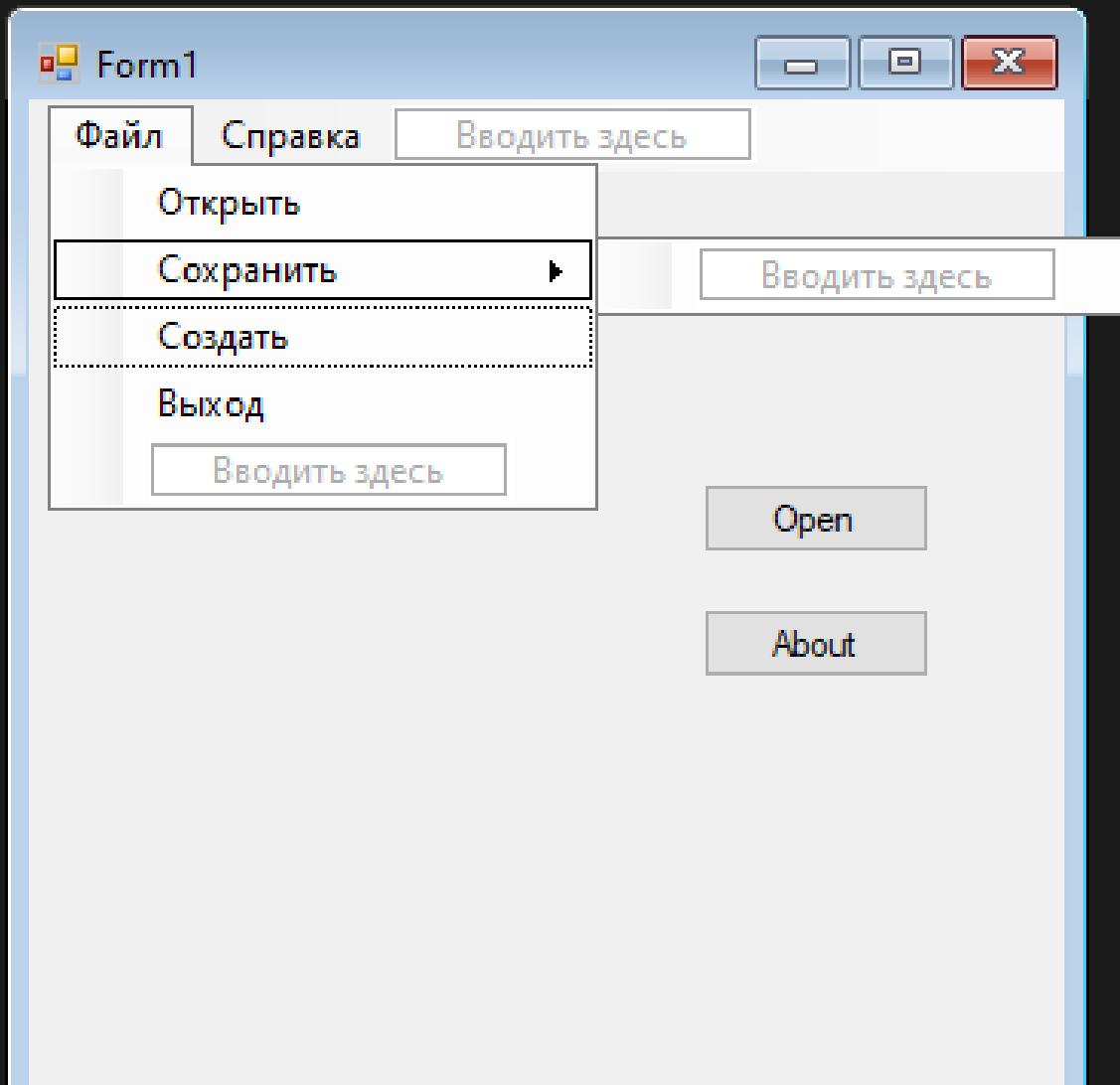
Панель элементов

Поиск по панели элеме

- HelpProvider
- HScrollBar
- ImageList
- Label
- LinkLabel
- ListBox
- ListView
- MaskedTextBox
- MenuStrip**
- MessageQueue
- MonthCalendar
- NotifyIcon
- NumericUpDown
- OpenFileDialog
- PageSetupDialog
- Panel
- PerformanceCoun...
- PictureBox

Form1.cs*

Form1.cs [Конструктор]*



menuStrip1

```
private void openToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Open file");
}

private void helpToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Version 1.0");
}
```

2. Контекстное меню в Windows Forms.

Контекстное меню (Context Menu) — это всплывающее меню, которое появляется при нажатии правой кнопки мыши на элементе управления или области формы.

Оно содержит команды, которые относятся к текущему контексту (например, выделенному тексту, изображению или элементу интерфейса).

Контекстное меню обеспечивает быстрый доступ к часто используемым функциям, делая взаимодействие с приложением более удобным.

Основные компоненты контекстного меню

В Windows Forms для создания контекстного меню используется элемент управления ContextMenuStrip. Он аналогичен ToolStrip, но предназначен для всплывающих меню.

1. ContextMenuStrip – основной элемент для создания контекстного меню

ContextMenuStrip — это контейнер для пунктов контекстного меню. Он может быть привязан к любому элементу управления или форме.

Основные свойства ContextMenuStrip:

Items – коллекция элементов меню (пунктов, подпунктов, разделителей).

ShowImageMargin – отображать ли поле для иконок в пунктах меню.

Opening и **Closing** – события, которые срабатывают при открытии и закрытии меню.

Панель элементов

Поиск по панели элементов

▼ Все формы Windows F...

Указатель

BackgroundWorker

BindingNavigator

BindingSource

Button

CheckBox

CheckedListBox

ColorDialog

ComboBox

ContextMenuStrip

DataGridView

DataSet

DateTimePicker

DirectoryEntry

DirectorySearcher

DomainUpDown

ErrorProvider

EventLog

FileSystemWatcher

Form1.cs*

Form1.cs [Конструктор]*

Form1

Файл Справка

ContextMenuStrip

Копировать

Вставить

Вводить здесь

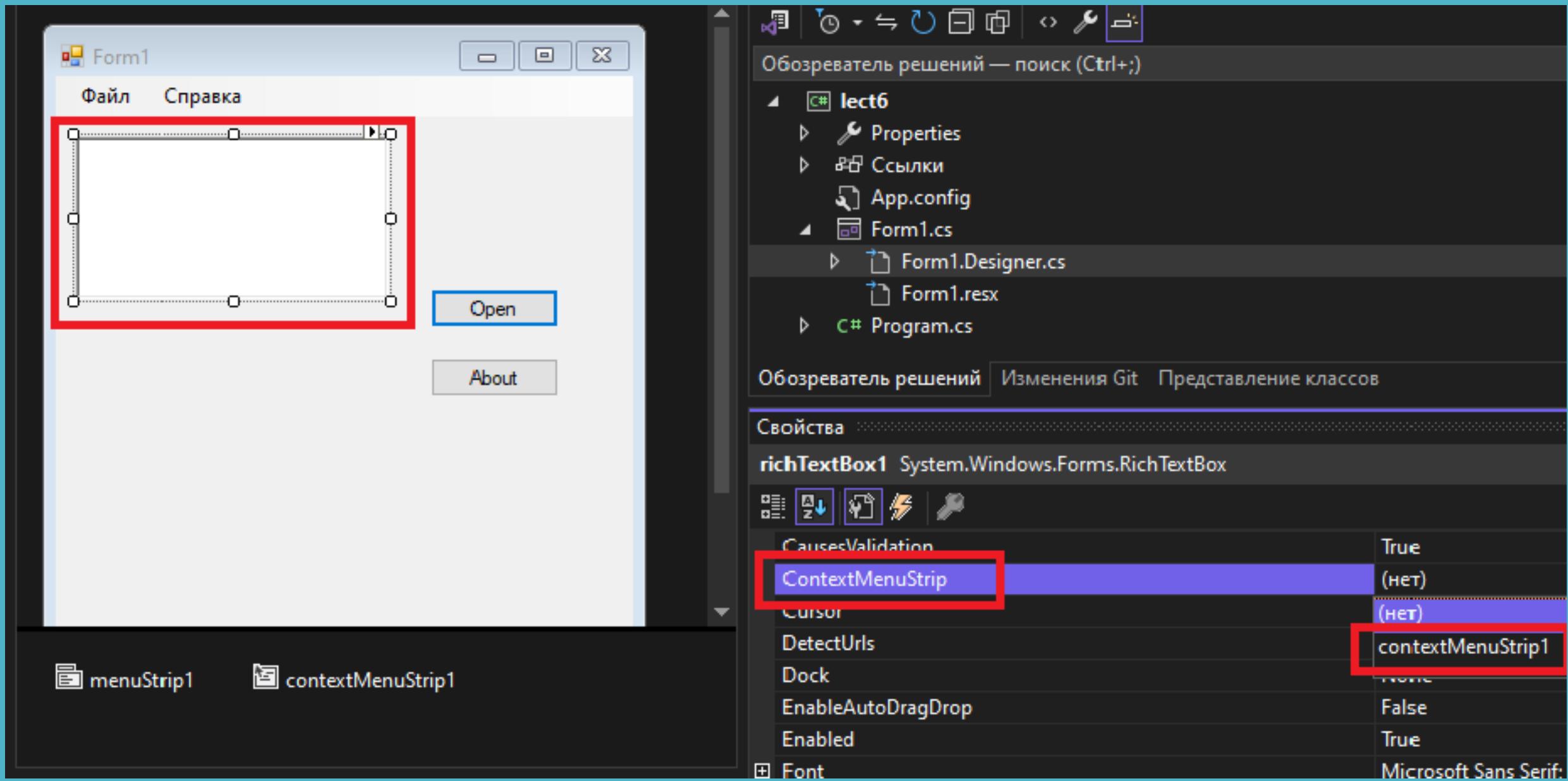


Open

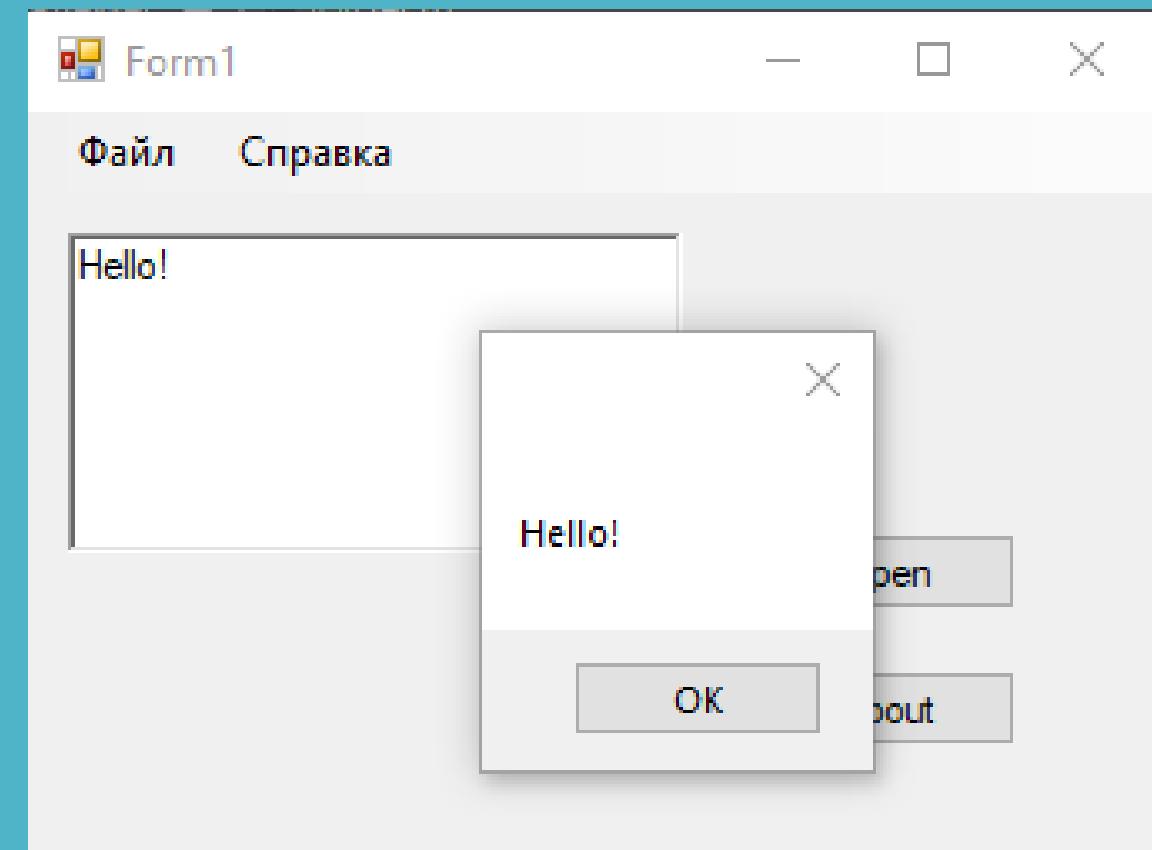
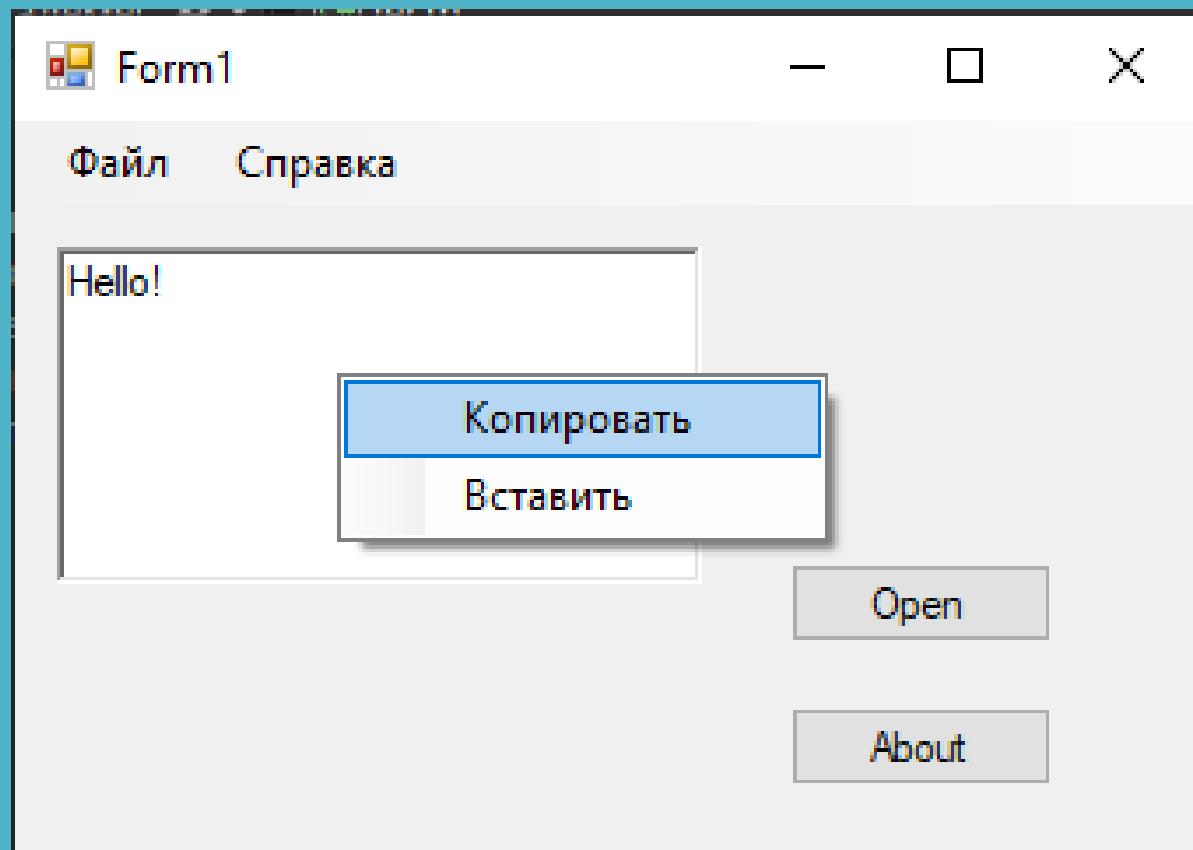
About

menuStrip1

contextMenuStrip1



```
private void копироватьToolStripMenuItem_Click(object sender, EventArgs e)
{
    string text = richTextBox1.Text;
    MessageBox.Show(text);
}
```



2. ToolStripMenuItem – элементы контекстного меню

Как и в главном меню, пункты контекстного меню создаются с помощью ToolStripMenuItem. Они могут содержать текст, иконки, сочетания клавиш и обработчики событий.

Пример использования:

```
ToolStripMenuItem copyMenuItem = new ToolStripMenuItem("Копировать");
copyMenuItem.Click += CopyMenuItem_Click;
contextMenu.Items.Add(copyMenuItem);
```

3. Разделители (ToolStripSeparator)

Разделители используются для визуального разделения пунктов меню. Они добавляются с помощью ToolStripSeparator.

Пример использования:

```
contextMenu.Items.Add(new ToolStripSeparator());
```



Form1

Файл Справка

ContextMenuStrip

Копировать

Вставить

Ввод...

Редактор коллекции элементов

- Выбрать элемент и добавить в список ниже:
- MenuItem
 - MenuItem
 - ComboBox
 - Separator
 - TextBox

Добавить

ContextMenuStrip



> MaximumSize

> MinimumSize

> Padding

> Size

▼ Поведение

AllowDrop

AllowMerge

AutoClose

Enabled

ImeMode

ShowItemToolTips

TabStop

▼ Прочее

DropShadowEnabled

Как привязать контекстное меню к элементу управления

1. Чтобы контекстное меню появлялось при нажатии правой кнопки мыши на элементе управления, нужно установить свойство ContextMenuStrip этого элемента.

Пример:

```
TextBox textBox = new TextBox();
textBox.ContextMenuStrip = contextMenu;
```

Как привязать контекстное меню к элементу управления

2. В дизайнере:

- Перетащите ContextMenuStrip на форму.
- Выберите элемент управления (например, TextBox, Button, Label или саму форму), для которого нужно добавить контекстное меню.
- В свойствах элемента управления найдите свойство **ContextMenuStrip**.
- Выберите из выпадающего списка созданный вами ContextMenuStrip.

Пример создания контекстного меню

Рассмотрим пример создания контекстного меню для текстового поля (TextBox), которое будет содержать команды "Копировать", "Вставить" и "Вырезать".

Шаги:

- Создать ContextMenuStrip.
- Добавить пункты меню (ToolStripMenuItem).
- Написать обработчики событий для пунктов меню.
- Привязать контекстное меню к элементу управления.

Пример кода:

```
InitializeComponent();

// Создаем текстовое поле
TextBox textBox = new TextBox();
textBox.Dock = DockStyle.Top;
textBox.Multiline = true;
textBox.Height = 100;
this.Controls.Add(textBox);

// Создаем контекстное меню
ContextMenuStrip contextMenu = new ContextMenuStrip();

// Добавляем пункты меню
ToolStripMenuItem copyMenuItem = new ToolStripMenuItem("Копировать");
copyMenuItem.Click += (sender, e) =>
{
    if (!string.IsNullOrEmpty(textBox.SelectedText))
    {
        Clipboard.SetText(textBox.SelectedText);
    }
};
```

```
ToolStripMenuItem pasteMenuItem = new ToolStripMenuItem("Вставить");
pasteMenuItem.Click += (sender, e) =>
{
    if (Clipboard.ContainsText())
    {
        textBox.Paste();
    }
};

ToolStripMenuItem cutMenuItem = new ToolStripMenuItem("Вырезать");
cutMenuItem.Click += (sender, e) =>
{
    if (!string.IsNullOrEmpty(textBox.SelectedText))
    {
        Clipboard.SetText(textBox.SelectedText);
        textBox.SelectedText = "";
    }
};
```

```
// Добавляем разделитель
contextMenu.Items.Add(copyMenuItem);
contextMenu.Items.Add(pasteMenuItem);
contextMenu.Items.Add(new ToolStripSeparator());
contextMenu.Items.Add(cutMenuItem);

// Привязываем контекстное меню к текстовому полю
textBox.ContextMenuStrip = contextMenu;
}
```

Особенности контекстного меню

Динамическое изменение меню:

Вы можете изменять пункты меню в зависимости от контекста. Например, отображать или скрывать пункты в зависимости от состояния приложения.

Для этого используйте событие Opening контекстного меню.

```
contextMenu.Opening += (sender, e) =>
{
    pasteMenuItem.Enabled = Clipboard.ContainsText();
};
```

Иконки в контекстном меню:

Пункты меню могут содержать иконки для лучшей визуализации команд.

Для этого используйте свойство `Image` у `ToolStripMenuItem`.

Контекстное меню для нескольких элементов:

Один и тот же `ContextMenuStrip` можно привязать к нескольким элементам управления.

```
TextBox textBox1 = new TextBox();
TextBox textBox2 = new TextBox();
textBox1.ContextMenuStrip = contextMenu;
textBox2.ContextMenuStrip = contextMenu;
```

Выбрать элемент и добавить в список ниже:

MenuItem

Добавить

Члены:

- contextMenuStrip1
- копироватьToolStripMenuItem
- вставитьToolStripMenuItem

↑
↓
X

ToolStripMenuItem

копироватьToolStripMenuItem

| | | |
|---|----------------------------|-----------------|
| | DisplayStyle | ImageAndText |
| > | Font | Segoe UI; 9pt |
| | ForeColor | ControlText |
| | Image | (отсутствует) |
| | ImageAlign | ImageCenter |
| | ImageScaling | SizeToFit |
| | ImageTransparentColor | |
| | RightToLeft | No |
| | RightToLeftAutoMirrorImage | False |
| | ShortcutKeyDisplayString | |
| | Text | Копировать |
| | TextAlign | MiddleCenter |
| | TextDirection | Horizontal |
| | TextImageRelation | ImageBeforeText |
| ▼ | Данные | |
| > | (ApplicationSettings) | |

OK

Отмена

3. Диалоговые окна в Windows Forms.

Диалоговые окна — это специальные окна, которые используются для взаимодействия с пользователем. Они могут запрашивать данные, выводить сообщения, предлагать выбор файлов, цветов, шрифтов и многое другое. Диалоговые окна делятся на два типа:

Модальные окна – блокируют взаимодействие с главным окном приложения до тех пор, пока пользователь не закроет диалоговое окно.

Немодальные окна – позволяют пользователю взаимодействовать как с диалоговым окном, так и с главным окном приложения.

В Windows Forms предусмотрены стандартные диалоговые окна, которые можно легко интегрировать в приложение.

Модальные формы (ShowDialog())

Модальная форма — это форма, которая блокирует взаимодействие с другими окнами приложения до тех пор, пока она не будет закрыта.

Для отображения модальной формы используется метод ShowDialog().

Пример:

```
private void button1_Click(object sender, EventArgs e)
{
    // Создание экземпляра формы
    Form modalForm = new Form();
    modalForm.Text = "Модальная форма";

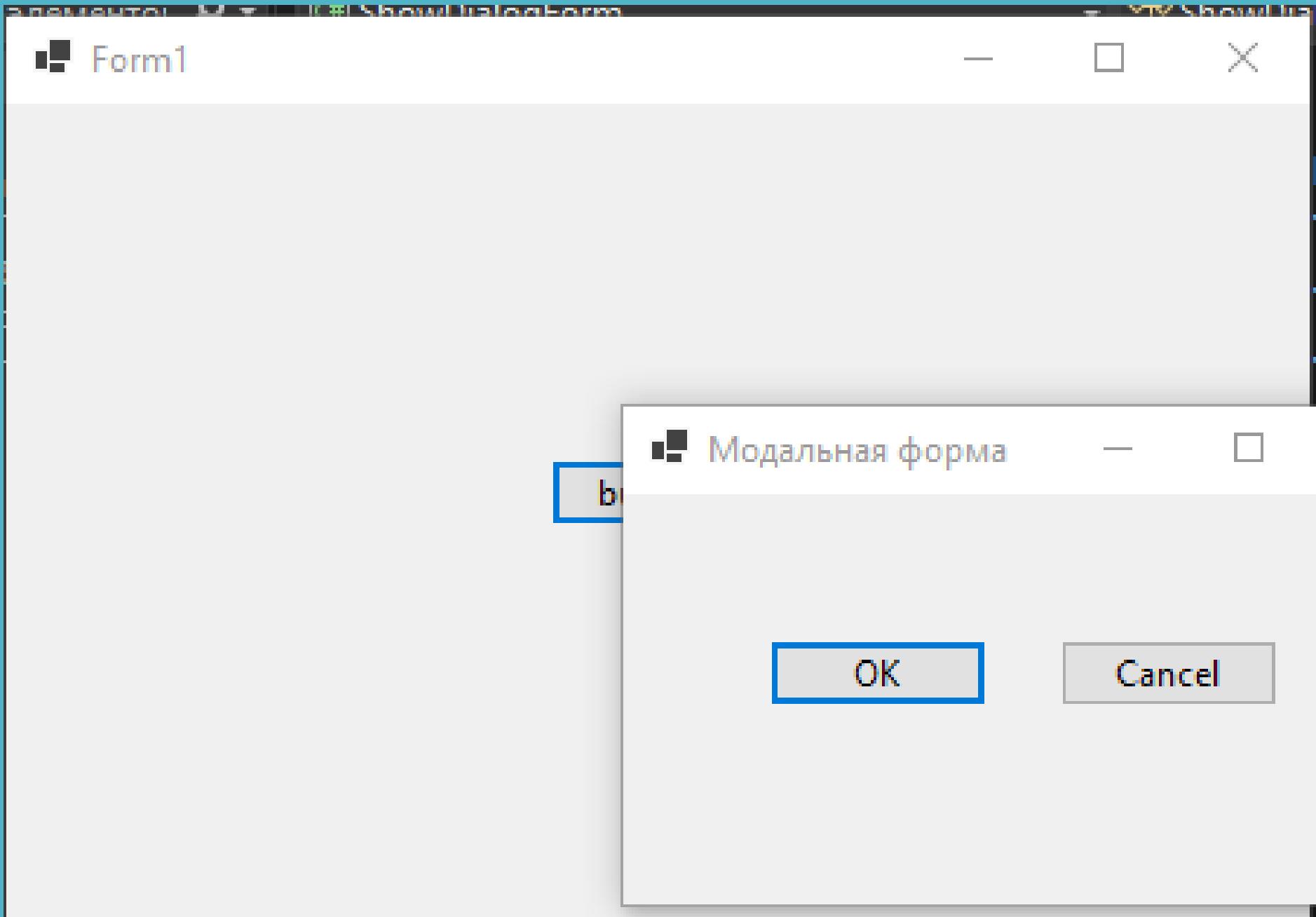
    // Добавление кнопок OK и Cancel на форму
    Button okButton = new Button();
    okButton.Text = "OK";
    okButton.DialogResult = DialogResult.OK;
    okButton.Location = new System.Drawing.Point(50, 50);
    modalForm.Controls.Add(okButton);

    Button cancelButton = new Button();
    cancelButton.Text = "Cancel";
    cancelButton.DialogResult = DialogResult.Cancel;
    cancelButton.Location = new System.Drawing.Point(150, 50);
    modalForm.Controls.Add(cancelButton);
}
```

Windows

```
// Отображение формы как модальной
DialogResult result = modalForm.ShowDialog();

// Обработка результата диалога
if (result == DialogResult.OK)
{
    MessageBox.Show("Пользователь нажал OK");
}
else if (result == DialogResult.Cancel)
{
    MessageBox.Show("Пользователь нажал Отмена");
}
```



Системные диалоги

Windows Forms предоставляет несколько стандартных диалоговых окон, таких как OpenFileDialog, SaveFileDialog, FolderBrowserDialog, ColorDialog, FontDialog и другие.

Эти диалоги позволяют пользователю выбирать файлы, папки, цвета, шрифты и т.д.

1. OpenFileDialog – диалог выбора файла

Используется для выбора одного или нескольких файлов.

Основные свойства:

Title – заголовок окна.

Filter – фильтр файлов (например, "Текстовые файлы (.txt)|.txt|Все файлы (.)|.").

Multiselect – разрешает выбор нескольких файлов.

FileName – путь к выбранному файлу.

Filenames – массив путей к выбранным файлам (если Multiselect = true).

Пример использования:

```
 OpenFileDialog openFileDialog = new OpenFileDialog();
openFileDialog.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
openFileDialog.Multiselect = false;

if (openFileDialog.ShowDialog() == DialogResult.OK)
{
    string selectedFile = openFileDialog.FileName;
    MessageBox.Show("Выбран файл: " + selectedFile);
}
```

2. SaveFileDialog – диалог сохранения файла

Используется для выбора места сохранения файла и указания его имени.

Основные свойства:

Title – заголовок окна.

Filter – фильтр файлов.

FileName – имя файла по умолчанию.

OverwritePrompt – запрашивать подтверждение, если файл уже существует.

Пример использования:

```
SaveFileDialog saveFileDialog = new SaveFileDialog();
saveFileDialog.Filter = "Текстовые файлы (*.txt)|*.txt";
saveFileDialog.FileName = "document.txt";

if (saveFileDialog.ShowDialog() == DialogResult.OK)
{
    string savePath = saveFileDialog.FileName;
    MessageBox.Show("Файл будет сохранен по пути: " + savePath);
}
```

3. FolderBrowserDialog – диалог выбора папки

Используется для выбора папки.

Основные свойства:

Description – описание, которое отображается в диалоге.

SelectedPath – путь к выбранной папке.

Пример использования:

```
FolderBrowserDialog folderBrowserDialog = new FolderBrowserDialog();
folderBrowserDialog.Description = "Выберите папку для сохранения";

if (folderBrowserDialog.ShowDialog() == DialogResult.OK)
{
    string selectedFolder = folderBrowserDialog.SelectedPath;
    MessageBox.Show("Выбрана папка: " + selectedFolder);
}
```

4. ColorDialog – диалог выбора цвета

Используется для выбора цвета.

Основные свойства:

Color – выбранный цвет.

AllowFullOpen – разрешает доступ к расширенным настройкам цвета.

AnyColor – разрешает выбор любого цвета.

FullOpen – открывает расширенную палитру цветов сразу.

Пример использования:

```
ColorDialog colorDialog = new ColorDialog();
colorDialog.AllowFullOpen = true;
colorDialog.AnyColor = true;

if (colorDialog.ShowDialog() == DialogResult.OK)
{
    Color selectedColor = colorDialog.Color;
    this.BackColor = selectedColor; // Изменяем цвет фона формы
}
```

5. FontDialog – диалог выбора шрифта

Используется для выбора шрифта, его размера и стиля.

Основные свойства:

Font – выбранный шрифт.

ShowColor – отображать ли выбор цвета шрифта.

ShowEffects – отображать ли настройки эффектов (зачеркивание, подчеркивание).

Пример использования:

```
FontDialog fontDialog = new FontDialog();
fontDialog.ShowColor = true;

if (fontDialog.ShowDialog() == DialogResult.OK)
{
    Font selectedFont = fontDialog.Font;
    Color selectedColor = fontDialog.Color;
    textBox1.Font = selectedFont; // Применяем шрифт к текстовому полю
    textBox1.ForeColor = selectedColor; // Применяем цвет текста
}
```

6. MessageBox – диалог вывода сообщений

Используется для вывода сообщений пользователю.
Это статический класс, который не требует создания
экземпляра.

Основные методы:

Show – отображает сообщение с кнопками и иконкой.

Пример использования:

```
DialogResult result = MessageBox.Show(  
    "Вы уверены, что хотите выйти?", // Текст сообщения  
    "Подтверждение", // Заголовок окна  
    MessageBoxButtons.YesNo, // Кнопки  
    MessageBoxIcon.Question // Иконка  
);  
  
if (result == DialogResult.Yes)  
{  
    Application.Exit();  
}
```

4. Компоненты для создания панелей инструментов.

В Windows Forms для создания панелей инструментов (toolbars) и строк состояния (status bars) используются компоненты ToolStrip и StatusStrip.

Эти компоненты предоставляют гибкие и настраиваемые интерфейсы для добавления кнопок, меню, меток и других элементов управления.

ToolStrip — это панель инструментов, на которой можно размещать кнопки, выпадающие меню, текстовые поля и другие элементы. Она обычно располагается в верхней части формы и предоставляет быстрый доступ к часто используемым командам.

Основные элементы ToolStrip:

- **ToolStripButton** — кнопка с иконкой и/или текстом.
- **ToolStripLabel** — метка для отображения текста или изображения.
- **ToolStripDropDownButton** — кнопка с выпадающим меню.
- **ToolStripComboBox** — выпадающий список.
- **ToolStripTextBox** — текстовое поле.
- **ToolStripSeparator** — разделитель между элементами.

Пример создания ToolStrip:

```
private void InitializeToolStrip()
{
    // Создание панели инструментов
    ToolStrip toolStrip = new ToolStrip();
    toolStrip.Dock = DockStyle.Top; // Закрепление в верхней части формы

    // Добавление кнопки
    ToolStripButton newButton = new ToolStripButton();
    newButton.Text = "Новый";
    newButton.Image = Properties.Resources.NewIcon; // Иконка из ресурсов
    newButton.Click += NewButton_Click; // Обработчик события
    toolStrip.Items.Add(newButton);

    // Добавление разделителя
    toolStrip.Items.Add(new ToolStripSeparator());
}
```

```
// Добавление выпадающей кнопки
ToolStripDropDownButton dropDownButton = new ToolStripDropDownButton("Файл");
dropDownButton.DropDownItems.Add("Открыть");
dropDownButton.DropDownItems.Add("Сохранить");
toolStrip.Items.Add(dropDownButton);

// Добавление текстового поля
ToolStripTextBox searchBox = new ToolStripTextBox();
searchBox.Text = "Поиск...";
toolStrip.Items.Add(searchBox);

// Добавление панели инструментов на форму
this.Controls.Add(toolStrip);
}
```

```
// Обработчик события для кнопки "Новый"  
private void NewButton_Click(object sender, EventArgs e)  
{  
    MessageBox.Show("Создан новый документ.");  
}
```

Компонент **StatusStrip**

StatusStrip — это строка состояния, которая обычно располагается в нижней части формы. Она используется для отображения информации о состоянии приложения, например, статуса выполнения операции, текущего времени или подсказок.

Основные элементы **StatusStrip**:

- **ToolStripStatusLabel** — метка для отображения текста или изображения.
- **ToolStripProgressBar** — индикатор выполнения.
- **ToolStripDropDownButton** — выпадающая кнопка.
- **ToolStripSplitButton** — кнопка с разделителем.

Пример создания StatusStrip:

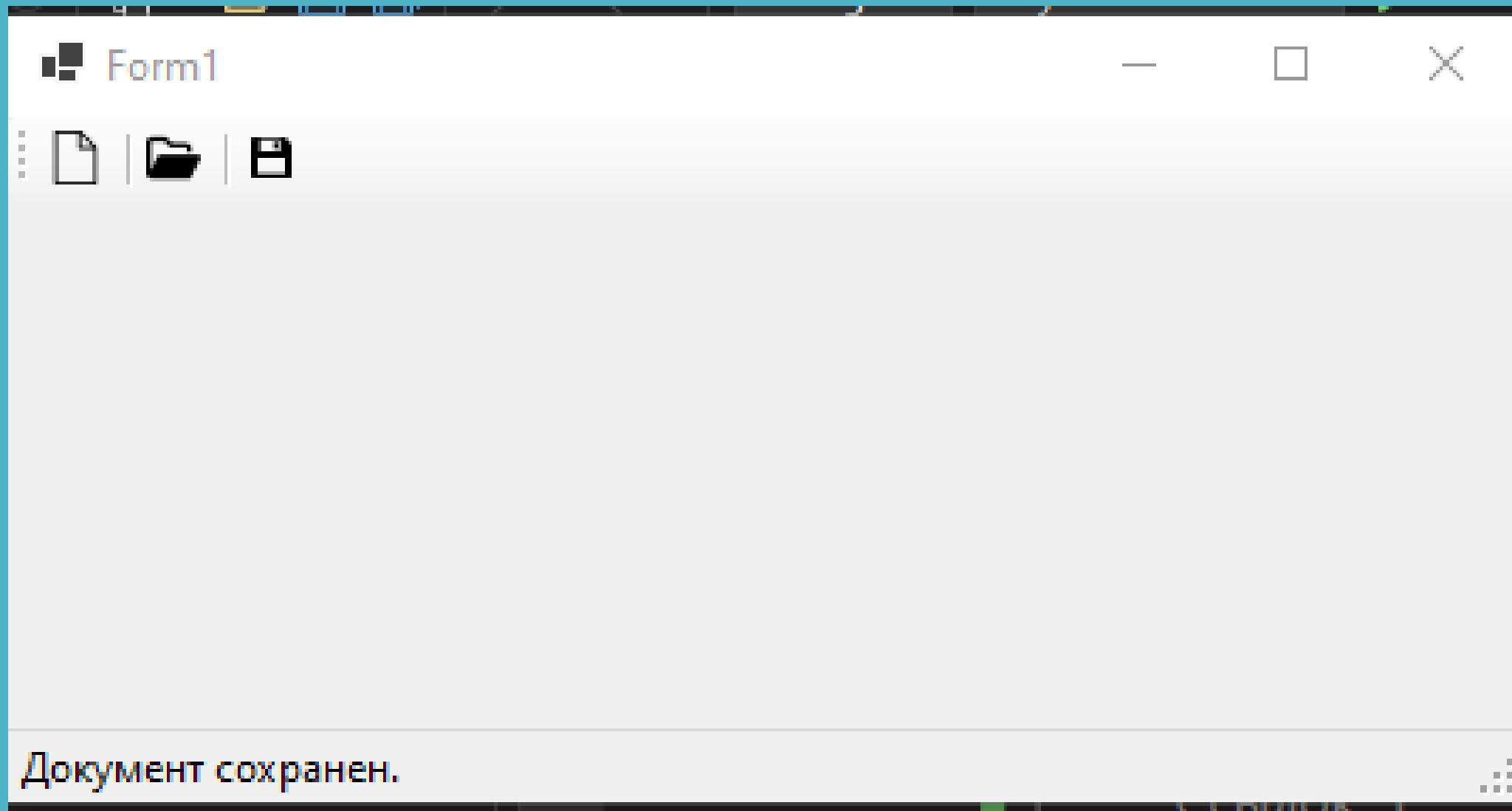
```
private void InitializeStatusStrip()
{
    // Создание строки состояния
    StatusStrip statusStrip = new StatusStrip();
    statusStrip.Dock = DockStyle.Bottom; // Закрепление в нижней части формы

    // Добавление метки
    ToolStripStatusLabel statusLabel = new ToolStripStatusLabel();
    statusLabel.Text = "Готово";
    statusStrip.Items.Add(statusLabel);

    // Добавление индикатора выполнения
    ToolStripProgressBar progressBar = new ToolStripProgressBar();
    progressBar.Value = 50; // Установка значения
    statusStrip.Items.Add(progressBar);

    // Добавление строки состояния на форму
    this.Controls.Add(statusStrip);
}
```

Пример



5. Windows Forms обработчик событий

В Windows Forms обработчики событий используются для реагирования на действия пользователя или системные события, такие как нажатие кнопки, изменение текста в текстовом поле или закрытие формы.

Обработчик события — это метод, который вызывается при возникновении определенного события.

Пример:

```
// Обработчик события Click  
private void Button_Click(object sender, EventArgs e)  
{  
    MessageBox.Show("Кнопка была нажата!");  
}
```

Параметры (object sender, EventArgs e) — это стандартная сигнатура метода-обработчика событий в .NET, включая Windows Forms.

Они передаются в метод, который вызывается при возникновении события. Давайте разберем, что означает каждый из этих параметров:

1. object sender

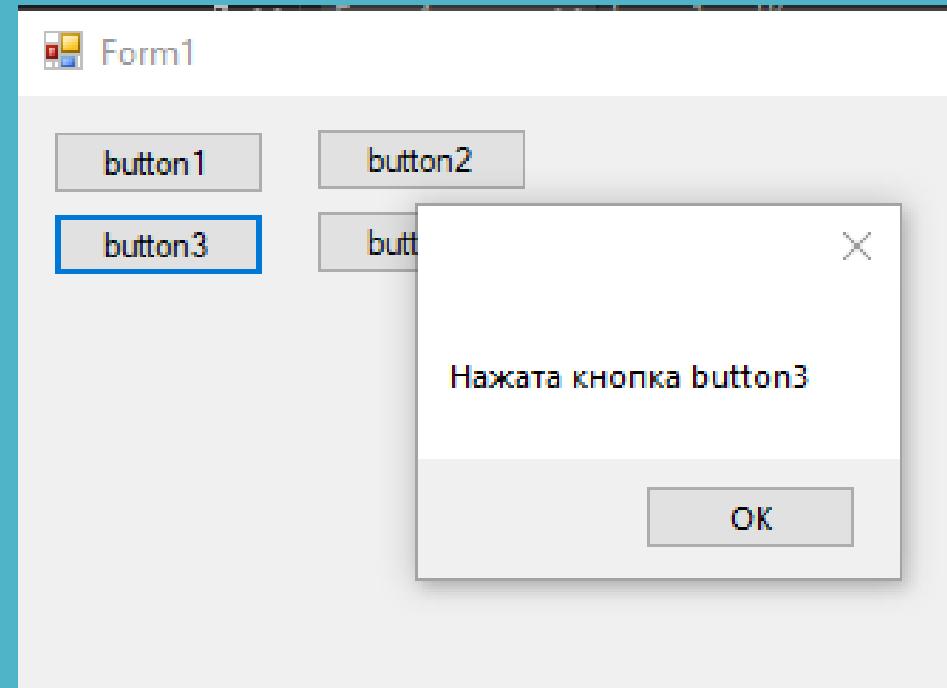
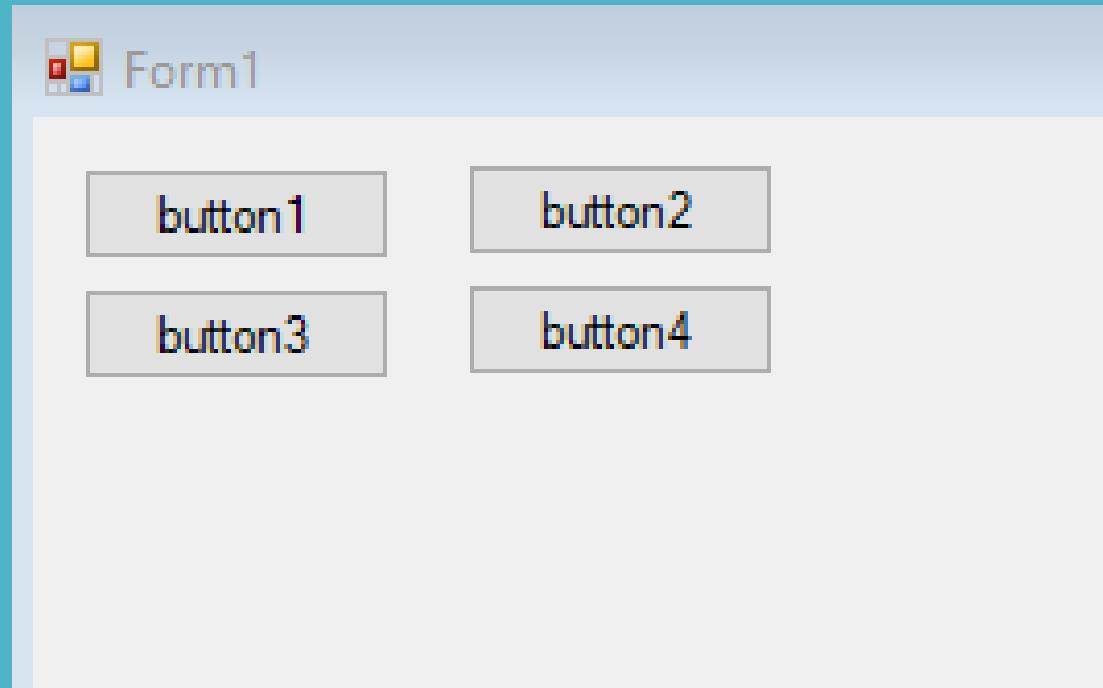
Тип: object

Назначение: Это объект, который вызвал событие (**источник события**).

Пример: Если событие вызвано нажатием кнопки, то sender будет ссылкой на эту кнопку (экземпляр класса Button).

Использование: Вы можете использовать sender, чтобы определить, **какой именно элемент управления вызвал событие**, особенно если **один обработчик используется для нескольких элементов**.

Примеры:



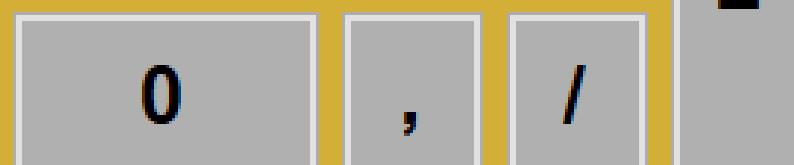
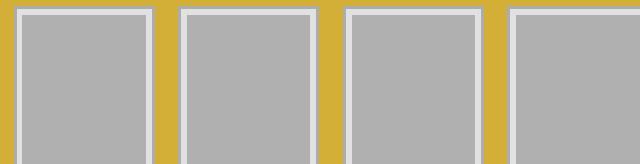
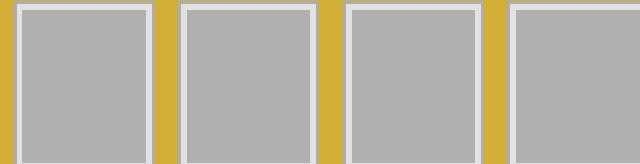
```
private void button1_Click(object sender, EventArgs e)
{
    Button btn = (Button)sender; // Приводим sender к типу Button
    MessageBox.Show($"Нажата кнопка {btn.Text}");
}
```

Form1.Designer.cs

Form1.cs

Form1.cs [Конструирование] ✎ X

Калькулятор



```
private void button_Click(object sender, EventArgs e)
{
    Button button = (Button)sender;

    if (char.IsDigit(button.Text[0])) // Если нажата цифра
    {
        currentNumber += button.Text;
        textBox1.Text = currentNumber; // Отображаем на дисплее
    }
    else if (button.Text == "=") // Если нажато равно
    {
        double result = Calculate();
        textBox1.Text = result.ToString();
        firstNumber = result.ToString();
        operation = "";
    }
    else // Если нажат оператор
    {
        if (firstNumber != "")
        {
            double result = Calculate();
            firstNumber = result.ToString();
        }
        firstNumber = currentNumber;
        operation = button.Text;
        currentNumber = "";
    }
}
```

2. EventArgs e

Тип: EventArgs (или его производные классы)

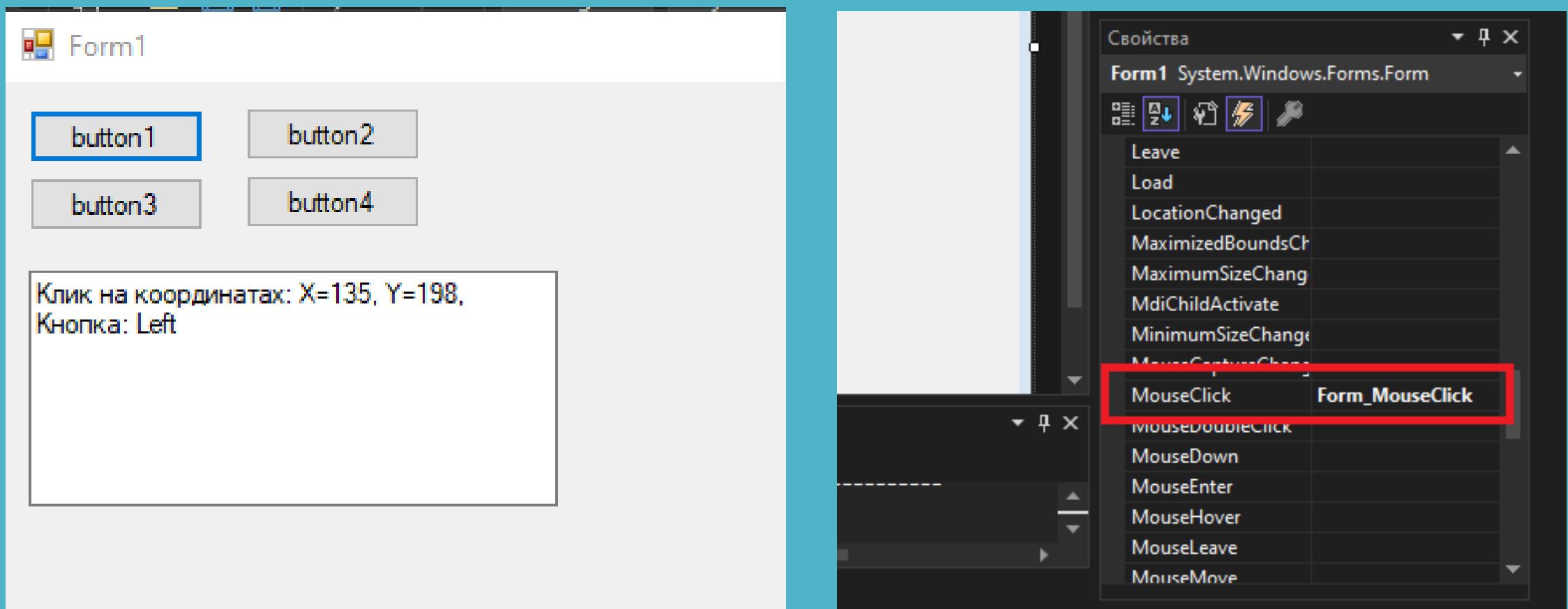
Назначение: Этот параметр содержит **дополнительные данные о событии**. Базовый класс EventArgs обычно не содержит данных, но для некоторых событий используются его производные классы, которые предоставляют дополнительную информацию.

Примеры:

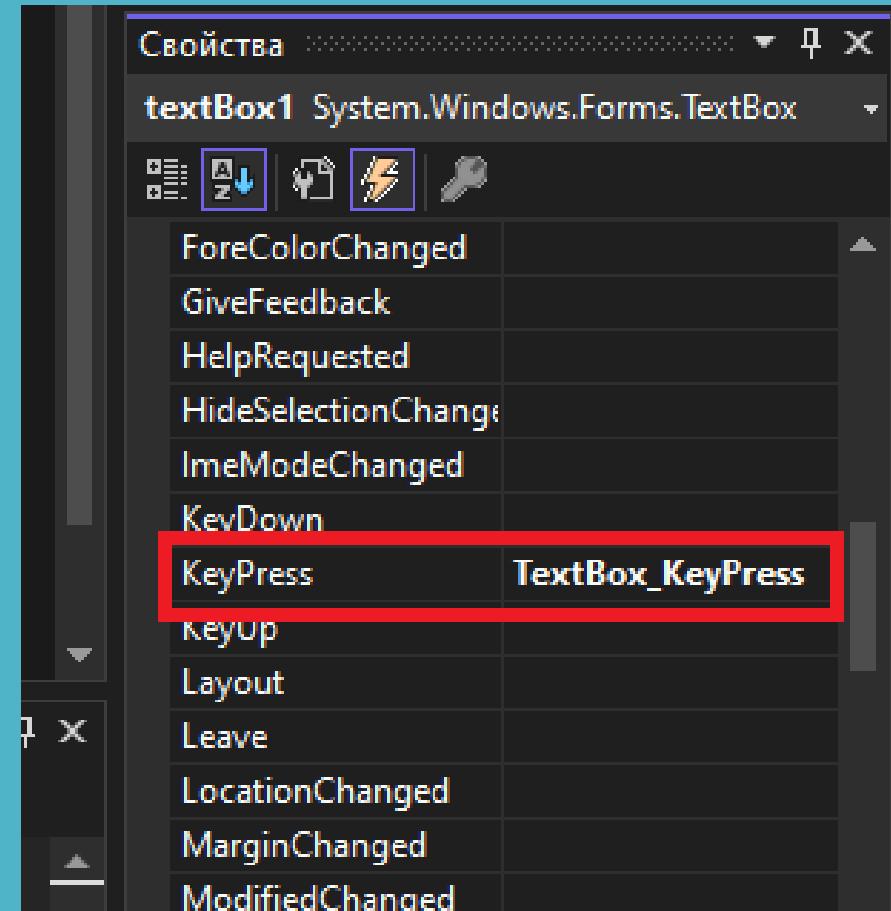
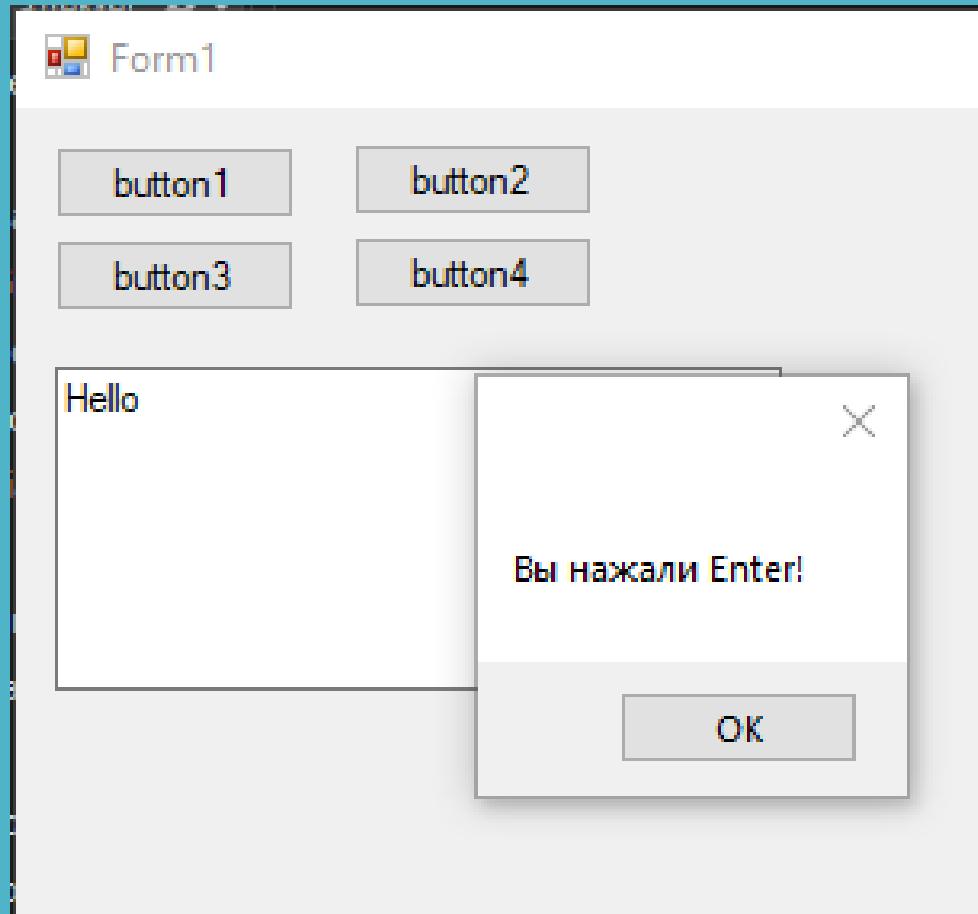
- Для события MouseClick используется MouseEventArgs, который содержит информацию о позиции курсора и кнопке мыши.
- Для события KeyPress используется KeyPressEventArgs, который содержит информацию о нажатой клавише.

Примеры:

```
private void Form_MouseClick(object sender, MouseEventArgs e)
{
    textBox1.Text = $"Клик на координатах: X={e.X}, Y={e.Y}, Кнопка: {e.Button}";
}
```



```
private void TextBox_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)Keys.Enter)
    {
        MessageBox.Show("Вы нажали Enter!");
    }
}
```



Зачем нужны эти параметры?

Гибкость: **Один обработчик может быть использован для нескольких элементов управления.** Например, если у вас есть несколько кнопок, вы можете использовать один метод для обработки события Click всех кнопок, а затем определить, какая именно кнопка была нажата, через параметр sender.

Дополнительные данные: Параметр e позволяет получить контекст события, например, координаты мыши, нажатую клавишу или другие данные, связанные с событием.

Список литературы:

1. [Создание меню ToolStrip](#)
2. [Контекстное меню ContextMenuStrip](#)
3. [Окно сообщения MessageBox](#)
4. [OpenFileDialog и SaveFileDialog](#)

Видеоуроки:

1. [Видеокурс C#.](#)
2. [MenuStrip](#)
3. [ContextMenuStrip](#)
4. [Видеокурс C# Windows Forms](#)
5. [Metanit](#)

Материалы лекций:

<https://github.com/ShViktor72/Education2025>

Домашнее задание :

Задание 1: Создание главного меню

Создайте новое приложение Windows Forms.

Добавьте на форму элемент управления ToolStrip.

Создайте главное меню с пунктами:

- Файл (с подпунктами: "Новый", "Открыть", "Сохранить", "Выход").
- Правка (с подпунктами: "Копировать", "Вставить", "Вырезать").
- Справка (с подпунктом: "О программе").

Реализуйте обработчики событий для каждого пункта меню:

- При выборе "Новый" выводите сообщение: "Создан новый документ".
- При выборе "Открыть" откройте диалоговое окно OpenFileDialog и выведите путь к выбранному файлу в MessageBox.
- При выборе "Сохранить" откройте диалоговое окно SaveFileDialog и выведите путь к сохраненному файлу в MessageBox.
- При выборе "Выход" закройте приложение.

Для пунктов "Копировать", "Вставить", "Вырезать" добавьте соответствующие действия (можно просто выводить сообщения).

При выборе "О программе" отобразите MessageBox с информацией о вашем приложении.

Задание 2: Создание контекстного меню

Добавьте на форму элемент управления TextBox.

Создайте контекстное меню (ContextMenuStrip) для TextBox с пунктами:

- "Копировать"
- "Вставить"
- "Вырезать"

Реализуйте функционал для каждого пункта меню:

- При выборе "Копировать" скопируйте выделенный текст в буфер обмена.
- При выборе "Вставить" вставьте текст из буфера обмена в TextBox.
- При выборе "Вырезать" скопируйте выделенный текст в буфер обмена и удалите его из TextBox.

Добавьте разделитель между пунктами меню.

Задание 3: Работа с диалоговыми окнами

Добавьте на форму кнопку "Выбрать цвет".

При нажатии на кнопку откройте диалоговое окно ColorDialog и измените цвет фона формы на выбранный пользователем цвет.

Добавьте на форму кнопку "Выбрать шрифт".

При нажатии на кнопку откройте диалоговое окно FontDialog и измените шрифт и цвет текста в TextBox на выбранные пользователем параметры.

Задание 4: Создание панели инструментов и строки состояния

Добавьте на форму элемент управления ToolStrip.

Создайте панель инструментов с кнопками:

"Новый" (с иконкой, если возможно).

"Открыть" (с иконкой, если возможно).

"Сохранить" (с иконкой, если возможно).

Реализуйте обработчики событий для кнопок:

При нажатии на "Новый" выводите сообщение: "Создан новый документ".

При нажатии на "Открыть" откройте диалоговое окно OpenFileDialog.

При нажатии на "Сохранить" откройте диалоговое окно SaveFileDialog.

Добавьте на форму элемент управления StatusStrip.

В строке состояния отобразите:

Текущее время (обновляйте его каждую секунду).

Сообщение "Готово" при запуске приложения.

Задание 5: Дополнительное задание Добавьте на форму кнопку "Выбрать папку".

При нажатии на кнопку откройте диалоговое окно FolderBrowserDialog и выведите путь к выбранной папке в MessageBox.

Добавьте на форму кнопку "Сообщение".

При нажатии на кнопку отобразите MessageBox с вопросом: "Вы уверены, что хотите выйти?" и кнопками "Да" и "Нет". Если пользователь выберет "Да", закройте приложение.