

**Тема 4. Работа с элементами
управления. Списки, комбинированные
поля, чекбоксы и радиокнопки.
Контейнеры и управление
расположением компонентов.**

Цель занятия:

Изучить работу с элементами управления, такими как списки, комбинированные поля, чекбоксы и радиокнопки, основными контейнерами и механизмами управления расположением компонентов.

Учебные вопросы:

- 1. Чекбоксы (CheckBox).**
- 2. Радиокнопки (RadioButton).**
- 3. Списки (ListBox).**
- 4. Комбинированные поля (ComboBox).**
- 5. Элементы управления для работы с датой и временем.**
- 6. Основные контейнеры в Windows Forms.**
- 7. Управление расположением компонентов.**
- 8. Примеры использования контейнеров.**

1. Чекбоксы (CheckBox).

Назначение и использование:

- Выбор одного или нескольких вариантов.
- Использование для включения/выключения опций.

Основные свойства:

- Text: текст рядом с чекбоксом.
- Checked: состояние чекбокса (выбран/не выбран).
- ThreeState: поддержка третьего состояния (неопределенное).

Основные методы:

- Check(): установить состояние "выбран".
- Uncheck(): установить состояние "не выбран".

Основные события:

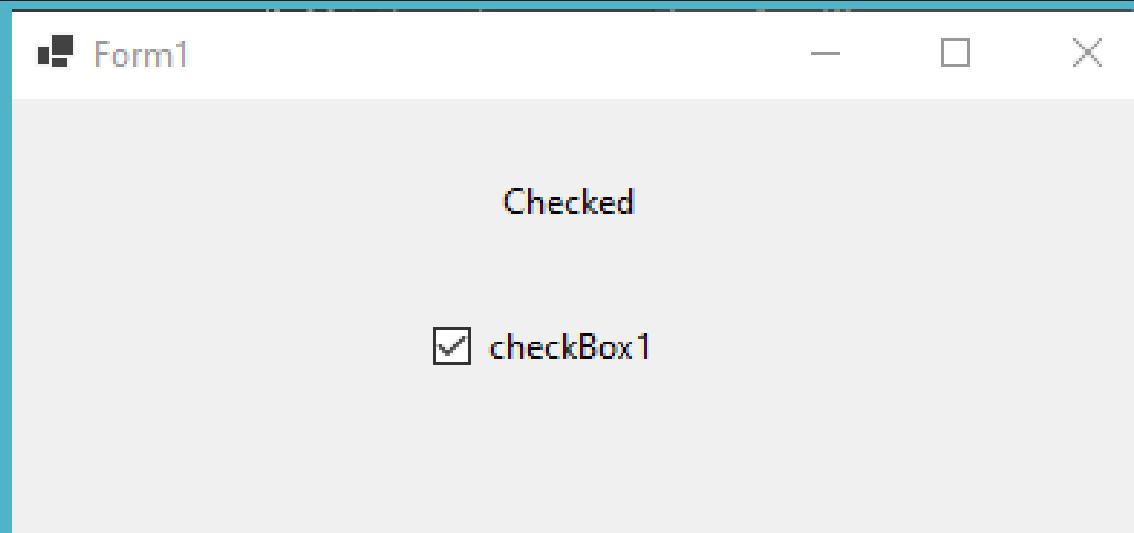
- CheckedChanged: изменение состояния чекбокса.

Пример использования:

- Создание группы чекбоксов.
- Обработка изменения состояния.

Пример обработки события CheckedChanged
(изменение состояния чекбокса).

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked)
        label1.Text = "Checked";
    else
        label1.Text = "Unchecked";
}
```



2. Радиокнопки (RadioButton).

Назначение и использование:

- Выбор одного варианта из группы.
- Использование для взаимоисключающих опций.

Основные свойства:

- Text: текст рядом с радиокнопкой.
- Checked: состояние радиокнопки (выбрана/не выбрана).
- AutoCheck: автоматическое снятие выбора с других радиокнопок в группе.

Основные методы:

- Check(): установить состояние "выбрана".
- Uncheck(): установить состояние "не выбрана".

Основные события:

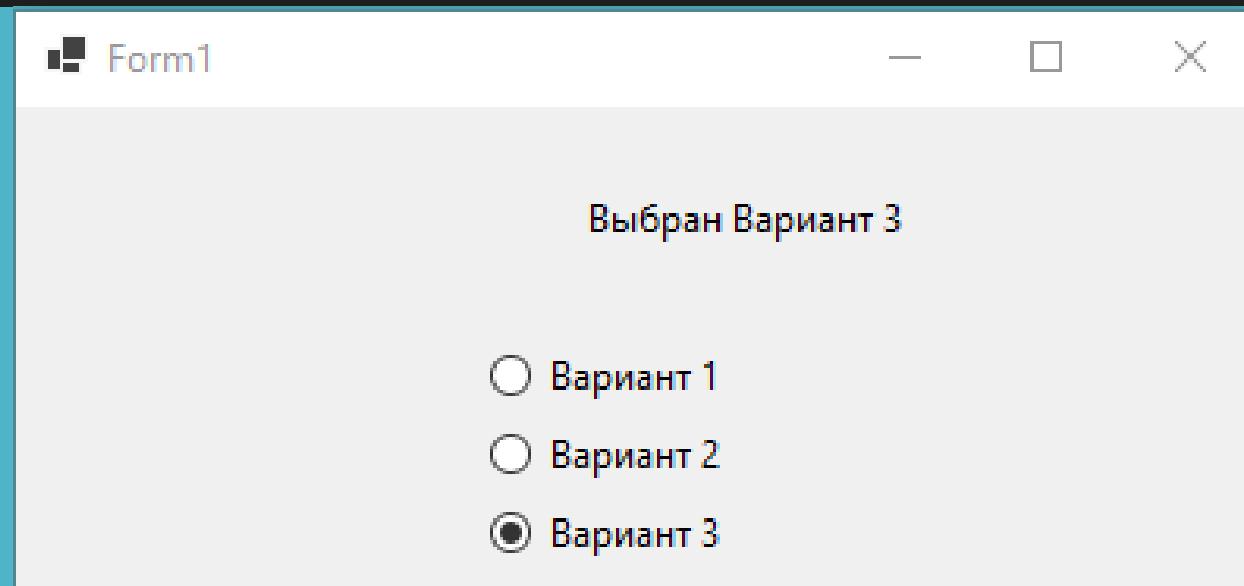
- CheckedChanged: изменение состояния радиокнопки.

Пример использования:

- Создание группы радиокнопок.
- Обработка выбора радиокнопки.

Пример обработки события CheckedChanged (изменение состояния радиокнопки).

```
private void radioButton_CheckedChanged(object sender, EventArgs e)
{
    // Проверяем, какая радиокнопка выбрана
    if (radioButton1.Checked)
        label1.Text = "Выбран Вариант 1";
    else if (radioButton2.Checked)
        label1.Text = "Выбран Вариант 2";
    else
        label1.Text = "Выбран Вариант 3";
}
```



3. Списки (ListBox).

Назначение и использование:

- Отображение списка элементов.
- Поддержка множественного выбора.

Основные свойства:

- Items: коллекция элементов.
- SelectedItem: выбранный элемент.
- SelectedIndex: индекс выбранного элемента.
- SelectionMode: режим выбора (один или несколько элементов).
- MultiColumn: отображение в несколько колонок.

Основные методы:

- Items.Add(): добавление элемента.
- Items.Remove(): удаление элемента.
- Items.Clear(): очистка списка.

Основные события:

- SelectedIndexChanged: изменение выбранного элемента.
- SelectedValueChanged: изменение значения выбранного элемента.

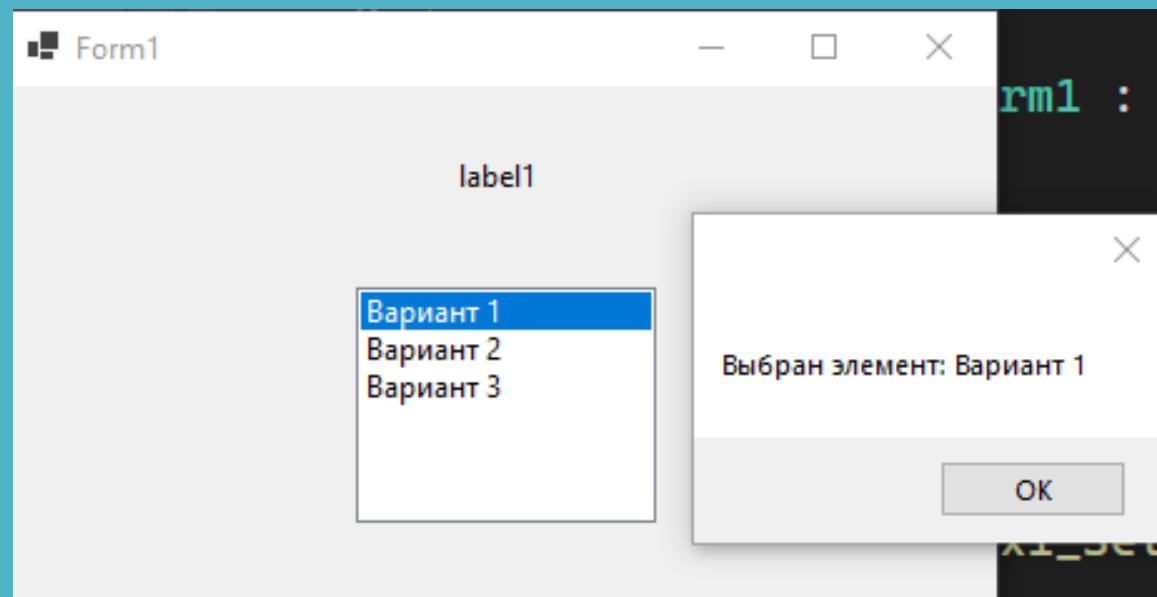
Пример использования:

- Добавление элементов в список.
- Обработка выбора элемента.

Пример обработки события SelectedIndexChanged (изменение выбранного элемента).

```
Событие: 1
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    // Проверяем, выбран ли элемент
    if (listBox1.SelectedItem != null)
    {
        // Получаем выбранный элемент
        string selectedItem = listBox1.SelectedItem.ToString();

        // Выводим сообщение с выбранным элементом
        MessageBox.Show("Выбран элемент: " + selectedItem);
    }
}
```



4. Комбинированные поля (ComboBox).

Назначение и использование:

- Выпадающий список для выбора одного элемента.

Основные свойства:

- Items: коллекция элементов.
- SelectedItem: выбранный элемент.
- SelectedIndex: индекс выбранного элемента.
- DropDownStyle: стиль списка (простой, выпадающий, выпадающий с редактированием).
- Text: текст в поле ввода.

Основные методы:

- Items.Add(): добавление элемента.
- Items.Remove(): удаление элемента.
- Items.Clear(): очистка списка.

Основные события:

- SelectedIndexChanged: изменение выбранного элемента.
- TextChanged: изменение текста в поле ввода (если ComboBox редактируемый).

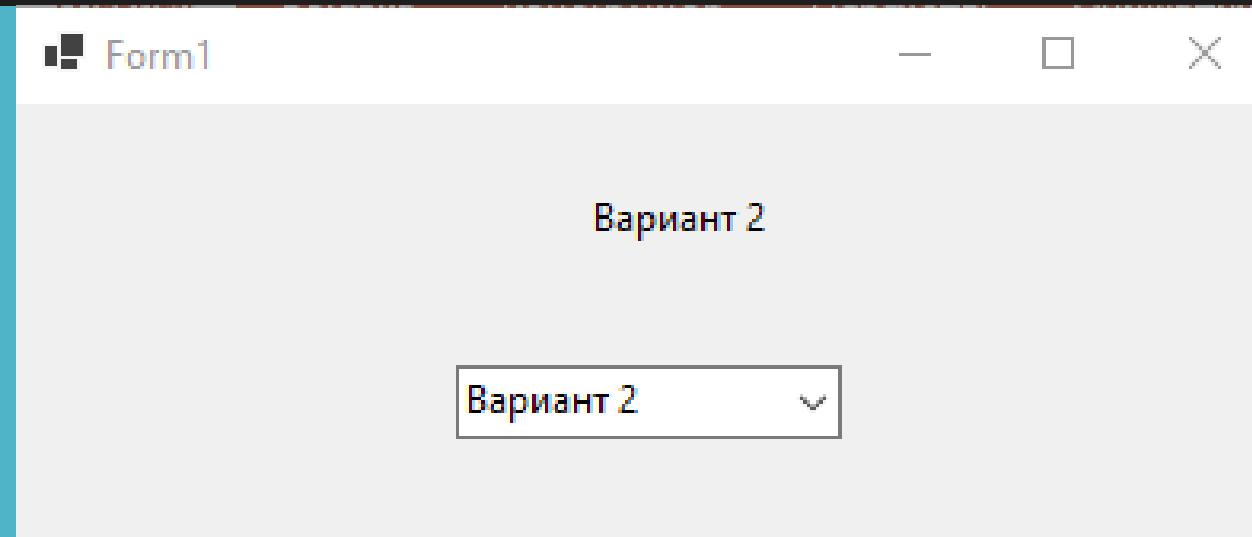
Пример использования:

- Добавление элементов в ComboBox.
- Обработка выбора элемента.

Пример обработки события SelectedIndexChanged (изменение выбранного элемента).

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    // Проверяем, выбран ли элемент
    if (comboBox1.SelectedItem != null)
    {
        // Получаем выбранный элемент
        string selectedItem = comboBox1.SelectedItem.ToString();

        // Выводим сообщение с выбранным элементом
        label1.Text = selectedItem;
    }
}
```



5. Элементы управления для работы с датой и временем.

DateTimePicker

Назначение:

DateTimePicker позволяет пользователю выбирать дату и/или время из выпадающего календаря или вводить их вручную.

Поддерживает различные форматы отображения даты и времени.

Основные свойства:

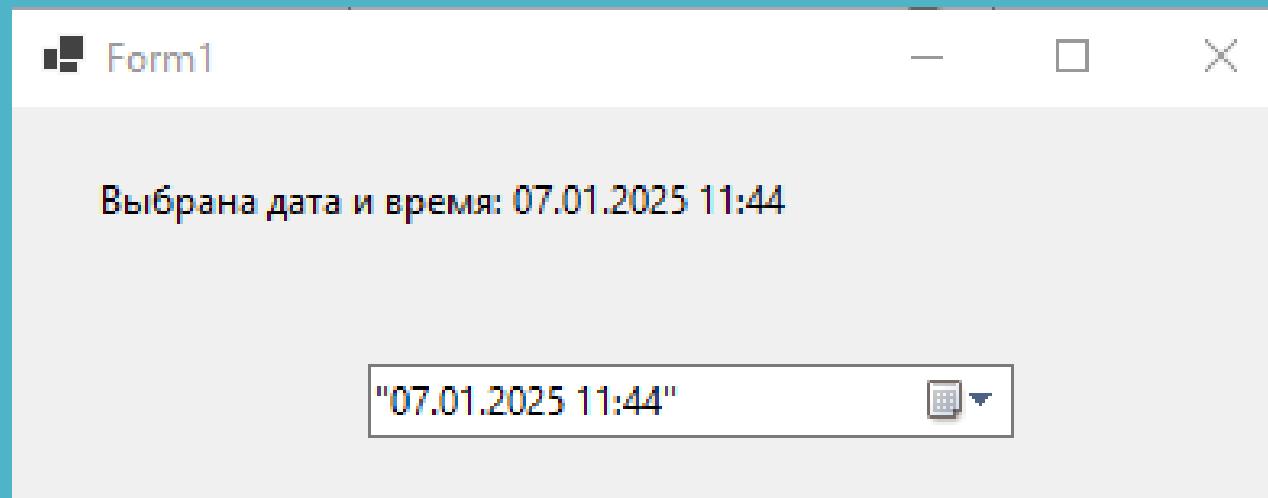
- **Value**: текущая выбранная дата и время.
- **Format**: формат отображения даты и времени (например, Short, Long, Time, Custom).
- **CustomFormat**: пользовательский формат (например, dd.MM.yyyy HH:mm).
- **ShowUpDown**: отображение стрелок для изменения значений (вместо выпадающего календаря).
- **MinDate** и **MaxDate**: минимальная и максимальная допустимая дата.

Основные события:

- **ValueChanged**: происходит при изменении выбранной даты или времени.
- **CloseUp**: происходит при закрытии выпадающего календаря.

Пример обработки события DateSelected (выбор даты/времени)

```
Символ: 1
private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
{
    DateTime selectedDateTime = dateTimePicker1.Value;
    label1.Text = "Выбрана дата и время: " + selectedDateTime.ToString("dd.MM.yyyy HH:mm");
}
```



MonthCalendar

Назначение:

- MonthCalendar позволяет пользователю выбирать дату или диапазон дат из календаря.
- Подходит для выбора одной даты или диапазона дат (например, для бронирования).

Основные свойства:

- SelectionStart: начальная дата выбранного диапазона.
- SelectionEnd: конечная дата выбранного диапазона.
- SelectionRange: выбранный диапазон дат.
- MaxSelectionCount: максимальное количество дней, которые можно выбрать.
- MinDate и MaxDate: минимальная и максимальная допустимая дата.

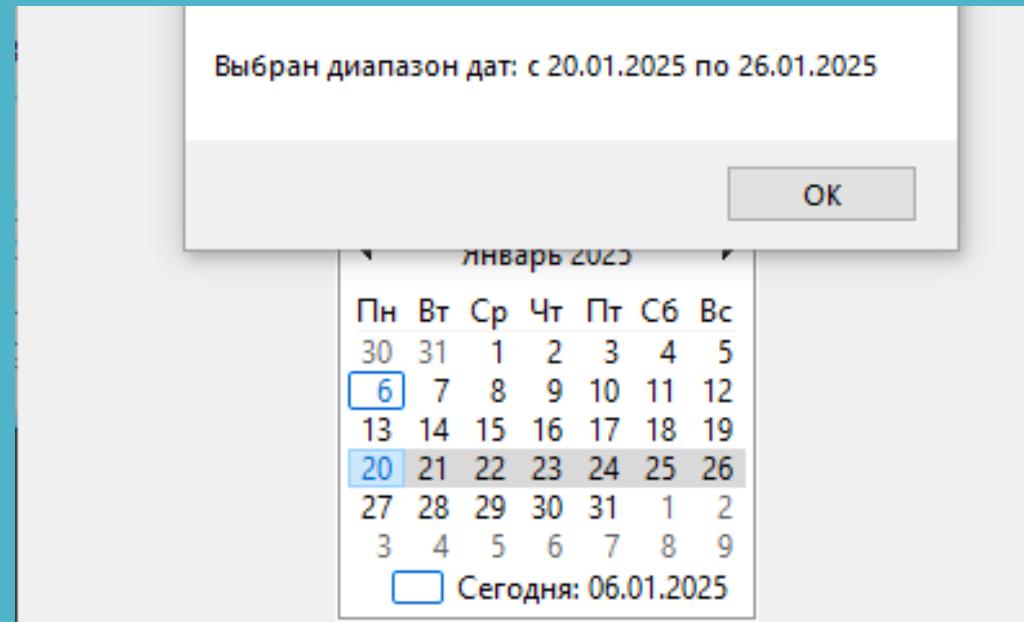
Основные события:

- DateChanged: происходит при изменении выбранной даты.
- DateSelected: происходит при выборе даты пользователем.

Пример обработки события DateSelected (выбор диапазона дат)

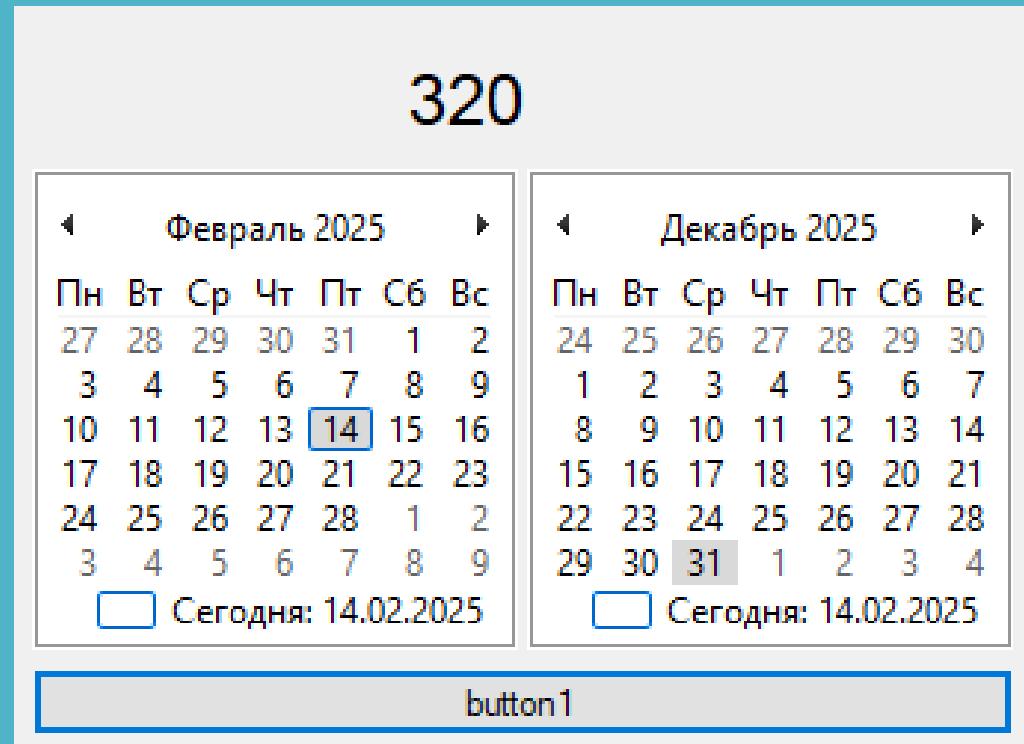
```
Семейка 1
private void monthCalendar1_DateSelected(object sender, DateRangeEventArgs e)
{
    DateTime startDate = monthCalendar1.SelectionStart;
    DateTime endDate = monthCalendar1.SelectionEnd;

    if (startDate == endDate)
        MessageBox.Show("Выбрана дата: " + startDate.ToString("dd.ММ.yyyy"));
    else
        MessageBox.Show("Выбран диапазон дат: с " + startDate.ToString("dd.ММ.yyyy")
            + " по " + endDate.ToString("dd.ММ.yyyy"));
}
```



Пример для диапазона дат больше месяца.

```
private void button1_Click(object sender, EventArgs e)
{
    DateTime start = monthCalendar1.SelectionStart;
    DateTime end = monthCalendar2.SelectionEnd;
    //label1.Text = start.ToString("d") + " - " + end.ToString("d");
    label1.Text = (end - start).Days.ToString("d");
}
```



6. Основные контейнеры в Windows Forms.

Контейнеры в Windows Forms — это специальные элементы управления, которые используются для группировки и организации других элементов управления (кнопок, текстовых полей, меток и т.д.).

Они выступают в роли "родительских" элементов, внутри которых размещаются "дочерние" элементы.

Основные задачи контейнеров:

1. Группировка элементов управления
2. Управление расположением элементов
3. Создание сложных и адаптивных интерфейсов
4. Упрощение управления элементами
5. Улучшение читаемости и поддержки кода

В Windows Forms существует несколько стандартных контейнеров, которые используются для группировки и управления расположением элементов управления.

Каждый из этих контейнеров имеет свои особенности и применяется в зависимости от задач. Рассмотрим основные контейнеры:

1. Panel – простая панель

Описание:

Panel — это простой контейнер, который используется для группировки элементов управления. Он не имеет визуальных эффектов, таких как рамка или заголовок, но может быть настроен с помощью свойств (например, цвет фона или границы).

Основные свойства:

BackColor – цвет фона панели.

BorderStyle – стиль границы (None, FixedSingle, Fixed3D).

AutoScroll – автоматическая прокрутка, если содержимое не помещается в панель.

2. GroupBox – группа с заголовком

Используется для логической группировки связанных элементов управления (например, радиокнопок или флаjkов).

Основные свойства:

Text – заголовок группы.

FlatStyle – стиль отображения рамки (Standard, Flat, Popup, System).

3. TabControl – контейнер с вкладками

TabControl позволяет создавать интерфейсы с вкладками. Каждая вкладка (TabPage) является отдельным контейнером, который может содержать свои элементы управления.

Основные свойства:

TabPages – коллекция вкладок.

SelectedIndex – индекс активной вкладки.

4. FlowLayoutPanel – автоматическое расположение в потоке

FlowLayoutPanel автоматически располагает элементы управления в потоке (горизонтально или вертикально). Это полезно для создания адаптивных интерфейсов.

Основные свойства:

FlowDirection – направление расположения элементов (LeftToRight, RightToLeft, TopDown, BottomUp).

WrapContents – перенос элементов на новую строку/столбец, если они не помещаются.

5. TableLayoutPanel – табличное расположение

TableLayoutPanel позволяет размещать элементы управления в таблице с заданным количеством строк и столбцов. Это полезно для создания сложных макетов.

Основные свойства:

RowCount – количество строк.

ColumnCount – количество столбцов.

RowStyles и ColumnStyles – настройки стилей строк и столбцов (размеры, автоматическое растяжение).

Различия между TableLayoutPanel и FlowLayoutPanel

Характеристика	TableLayoutPanel	FlowLayoutPanel
Принцип размещения	Размещает элементы в виде таблицы с фиксированным числом строк и столбцов	Располагает элементы последовательно в строку или столбец
Автоматическая адаптация	Можно задавать фиксированные размеры строк/столбцов или использовать автоматическое масштабирование	Автоматически переносит элементы на новую строку/столбец, если не хватает места
Гибкость компоновки	Хорошо подходит для сложных макетов, где важно точное позиционирование элементов	Подходит для динамического размещения элементов, когда их количество заранее неизвестно
Поддержка относительных размеров	Поддерживает настройку ширины/высоты в процентах	Все элементы имеют свои собственные размеры и размещаются без относительных привязок
Применение в адаптивных интерфейсах	Используется, когда требуется точная табличная структура (например, формы, настройки)	Хорош для панелей инструментов, динамических списков и адаптивных интерфейсов

6. SplitContainer – разделение области

SplitContainer разделяет область на две части, между которыми можно изменять размеры. Это полезно для создания интерфейсов с разделенными областями (например, список файлов и область просмотра).

Основные свойства:

Orientation – ориентация разделения (Horizontal или Vertical).

Panel1 и Panel2 – две разделенные области.

7. Управление расположением компонентов.

Управление расположением компонентов — это важный аспект разработки пользовательских интерфейсов.

В Windows Forms для этого используются различные свойства и контейнеры, которые позволяют гибко настраивать размещение элементов управления на форме.

Рассмотрим основные механизмы управления расположением.

1. Свойство Dock – закрепление элементов

Свойство Dock позволяет закрепить элемент управления относительно краев контейнера (формы или другого контейнера). Элемент "прилипает" к указанному краю и растягивается вдоль него.

Возможные значения:

None – элемент не закреплен (по умолчанию).

Top – закрепление к верхнему краю.

Bottom – закрепление к нижнему краю.

Left – закрепление к левому краю.

Right – закрепление к правому краю.

Fill – заполнение всего доступного пространства.

2. Свойство Anchor – привязка элементов

Свойство Anchor позволяет привязать элемент управления к одному или нескольким краям контейнера. При изменении размеров контейнера элемент будет сохранять расстояние до указанных краев.

Возможные значения:

Top – привязка к верхнему краю.

Bottom – привязка к нижнему краю.

Left – привязка к левому краю.

Right – привязка к правому краю.

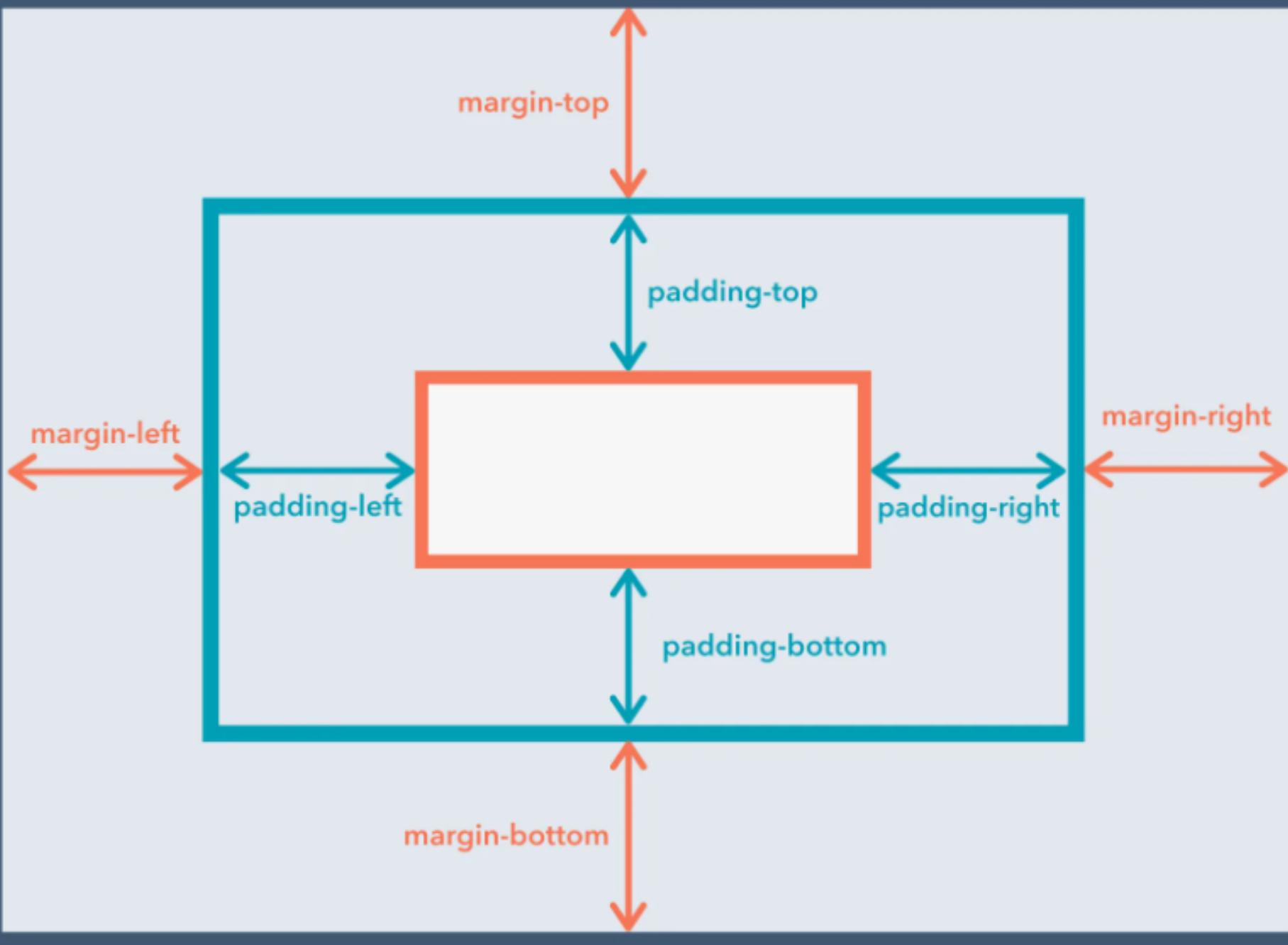
3. Свойство AutoSize – автоматическое изменение размеров

Свойство AutoSize позволяет элементу управления автоматически изменять свои размеры в зависимости от содержимого.

4. Свойства Padding и Margin – отступы

Padding – отступы внутри элемента управления (между границей элемента и его содержимым).

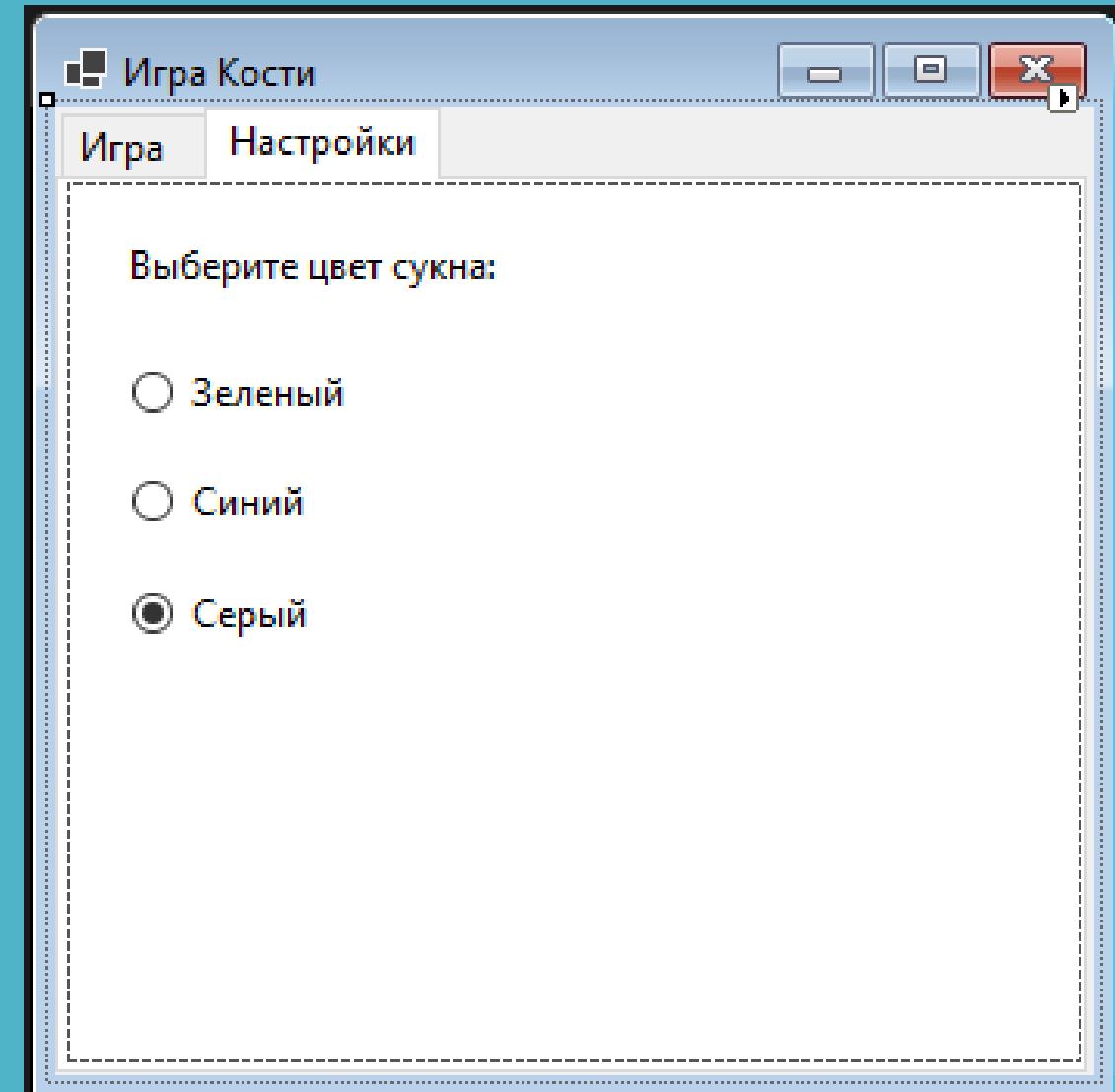
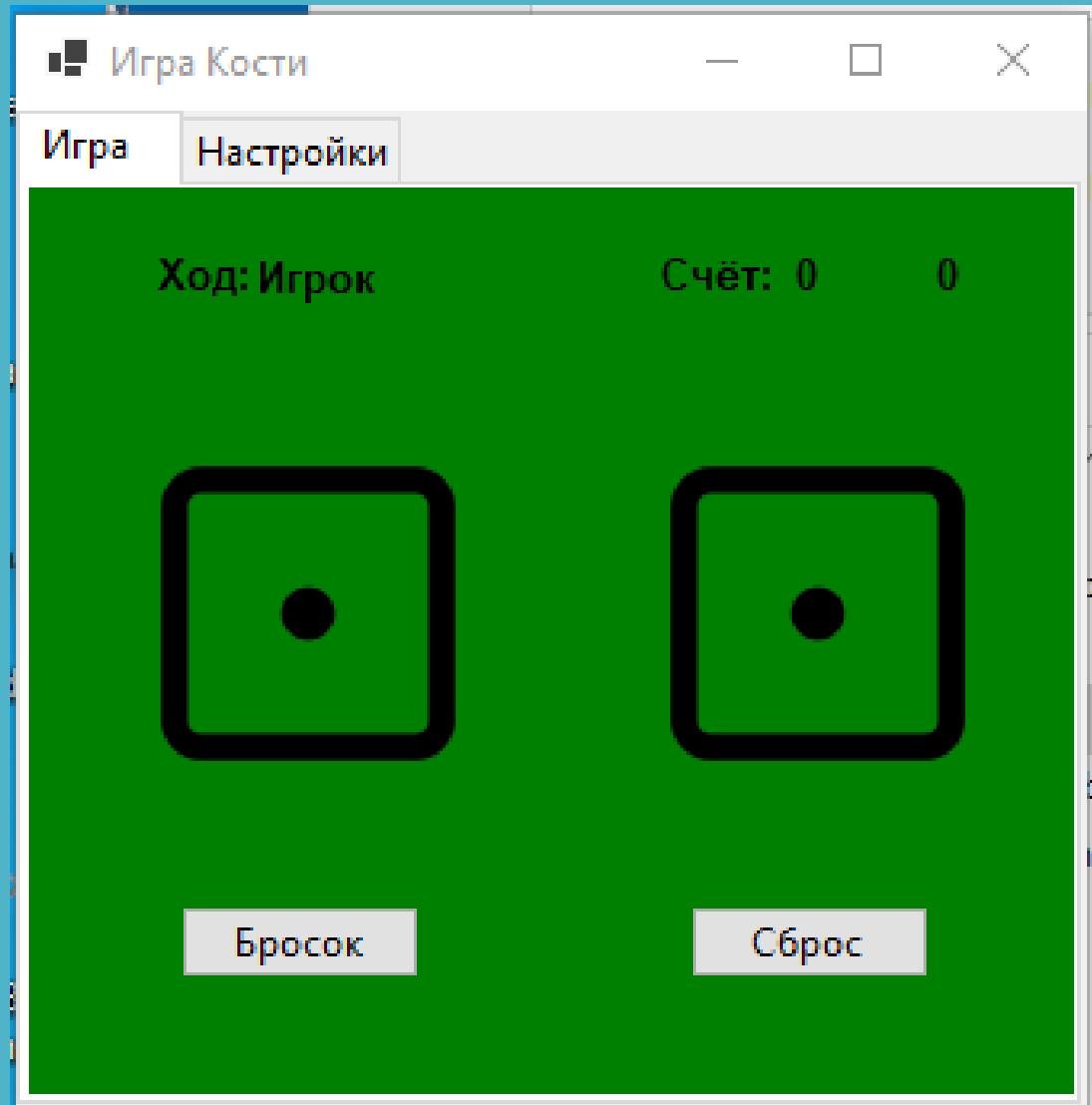
Margin – отступы вокруг элемента управления (между элементом и другими элементами или краями контейнера).



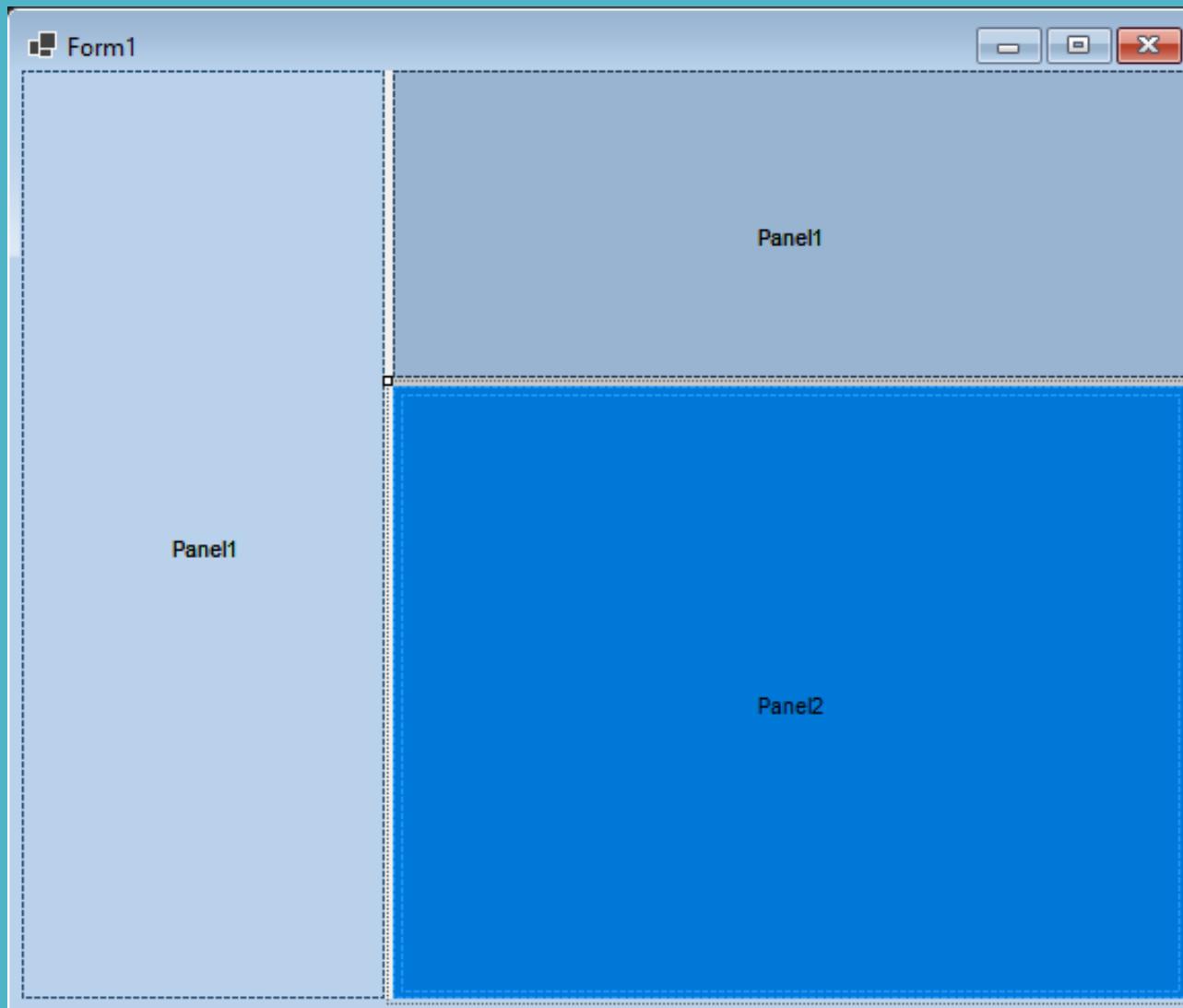
8. Примеры использования контейнеров.

- Создание форм с вкладками.
- Разделение формы на несколько областей.
- Автоматическое расположение элементов в потоке или таблице.

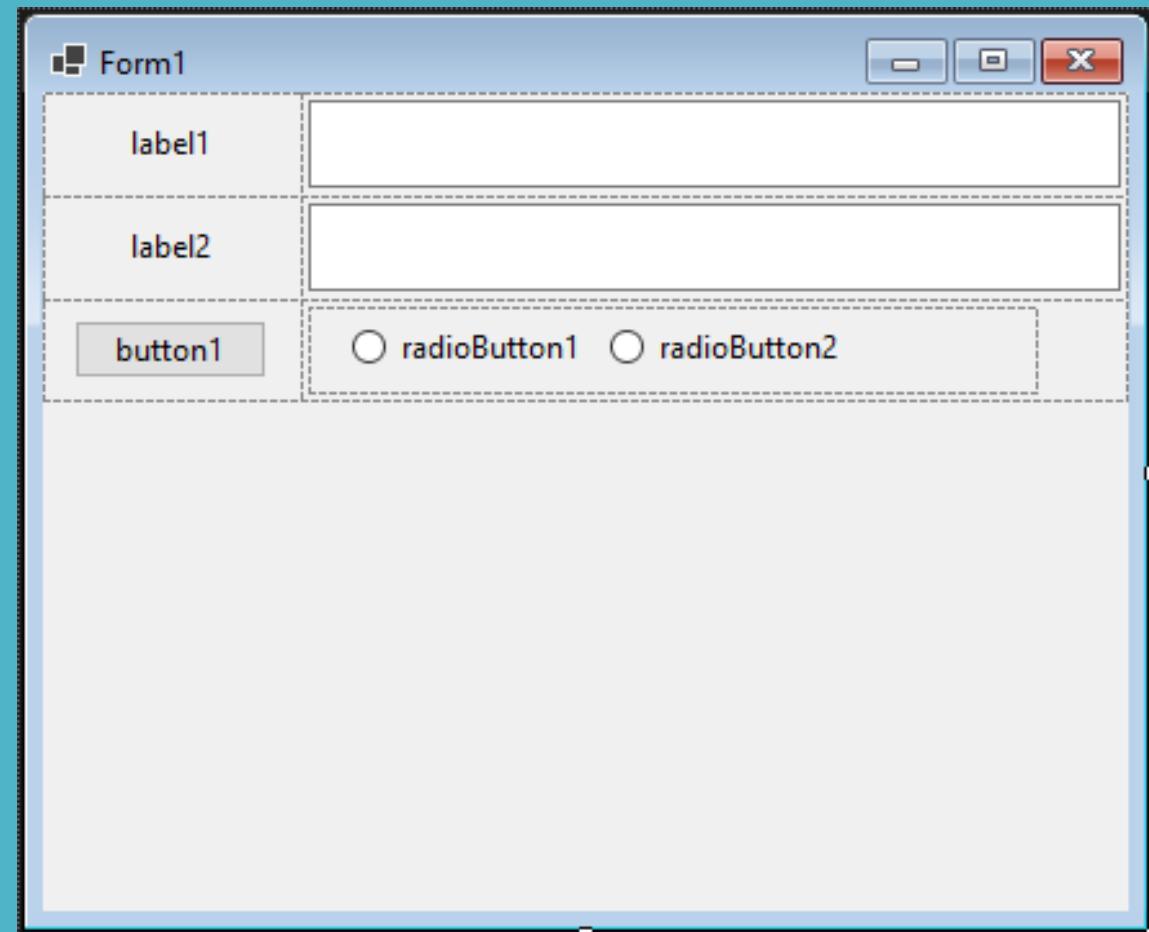
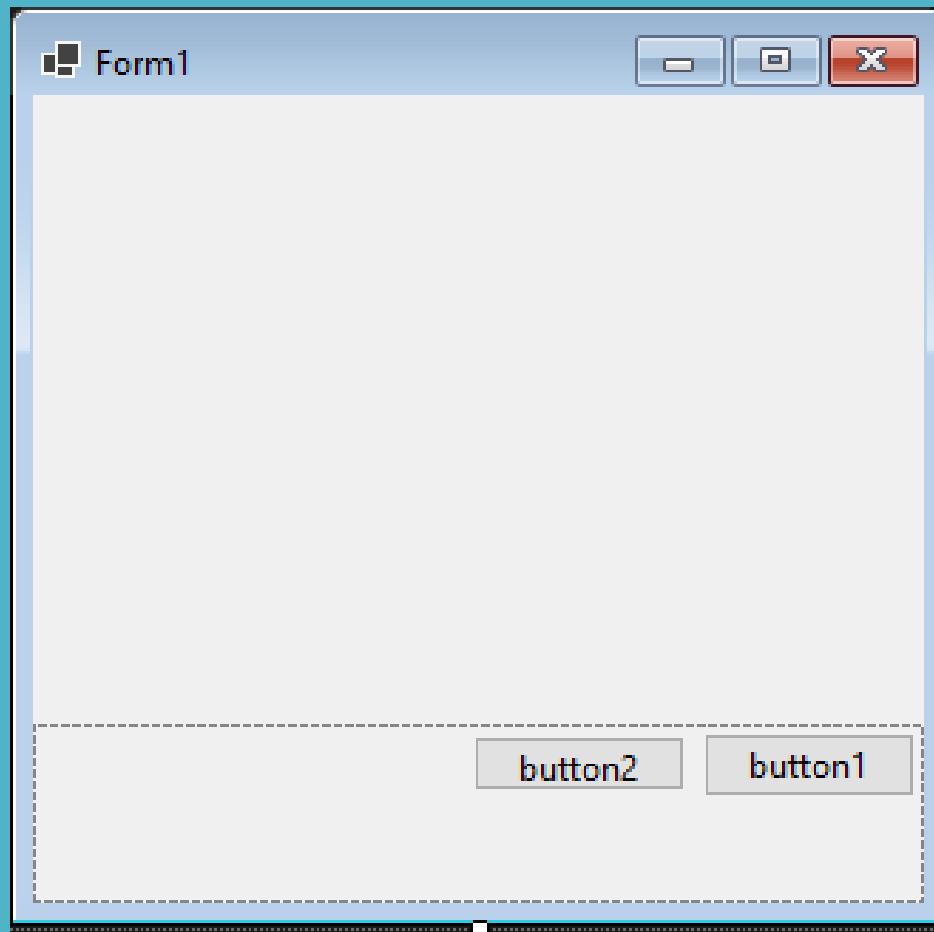
Пример (TabControl)



Пример (SplitContainer)



Пример (TableLayoutPanel и FlowLayoutPanel)



Список литературы:

1. [Видеокурс C#](#).
2. [Видеокурс C# Windows Forms](#)
3. [Metanit](#)
4. <https://metanit.com/sharp/windowsforms/3.1.php>
5. [Видеокурс C# Windows Forms](#)
6. [FlowLayoutPanel и TableLayoutPanel в проектах Windows Forms](#)

Материалы лекций:

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

Домашнее задание (Работа с элементами управления):

Задание 1. Работа с CheckBox и RadioButton.

Создайте Windows Forms-приложение с формой, на которой:

- Разместите три CheckBox, имитирующих выбор дополнительных услуг (например, "Доставка", "Подарочная упаковка", "СМС-уведомление").
- Разместите группу из трех RadioButton, позволяющих выбрать способ оплаты ("Наличные", "Карта", "Перевод").
- Добавьте Label, который будет отображать выбранные опции при нажатии на кнопку "Показать выбор".

Что нужно реализовать:

- При изменении состояния CheckBox обновлять список выбранных услуг.
- Гарантировать, что в группе RadioButton можно выбрать только один вариант.
- Выводить информацию о выборе в Label при нажатии на кнопку.

Задание 2. Анкета.

Создайте небольшую анкету с элементами управления:

- Имя (TextBox).
- Дата рождения (DateTimePicker).
- Пол (RadioButton).
- Увлечения (несколько CheckBox).
- Кнопку "Отправить", которая будет отображать введенные данные в MessageBox.

Задание 3: Приложение для подсчета доходности банковского вклада с использованием MonthCalendar.

Создайте приложение на Windows Forms, которое позволяет пользователю рассчитать доходность банковского вклада.

Приложение должно рассчитать итоговую сумму вклада с учетом процентов и отобразить результат.

Входные данные:

Начальная сумма вклада (initialAmount) (TextBox).

Годовая процентная ставка (annualInterestRate) (TextBox).

Дата начала вклада (startDate) (MonthCalendar).

Дата окончания вклада (endDate) (MonthCalendar).

Расчет:

Вычислите количество дней между startDate и endDate.

Рассчитайте доходность по формуле:

$$\text{Итоговая сумма} = \text{Начальная сумма} \times \left(1 + \frac{\text{Годовая ставка} \times \text{Количество дней}}{365 \times 100} \right)$$

Задание 4. Форма регистрации

Цель: Создать форму регистрации, используя TableLayoutPanel.

Требования:

Использовать TableLayoutPanel с 5 строками и 2 столбцами.

В ячейках разместить элементы управления:

Label "Имя" → TextBox

Label "Фамилия" → TextBox

Label "Email" → TextBox

Label "Пароль" → TextBox (с UseSystemPasswordChar = true)

Кнопка "Регистрация" (на всю ширину таблицы)

Сделать адаптивный интерфейс (при изменении размеров окна поля ввода должны растягиваться).

 Подсказка:

Используйте Dock = Fill для полей ввода.

Для кнопки "Регистрация" установите ColumnSpan = 2.