

Лабораторная работа 1. Введение в разработку форм

Цель работы

Изучение методов построения форм Windows и получение навыков по настройке форм, созданию непрямоугольных и наследуемых (производных) форм.

Упражнение 1. Настройка прямоугольной формы Windows

Формы Windows — это основной компонент пользовательского интерфейса. Формы предоставляют контейнер, который содержит элементы управления, меню и позволяет отображать приложение в уже привычной и единообразной модели. Формы могут реагировать на события мыши и клавиатуры, поступающие от пользователя, и выводить на экран данные для пользователя с помощью элементов управления, которые содержатся в форме.

Формы Windows содержат множество свойств, позволяющих настраивать их внешний вид и поведение. Просматривать и изменять эти свойства можно в окне **Properties** конструктора при разработке, а также программно во время выполнения приложения.

В следующей таблице перечислены некоторые свойства форм Windows, отвечающие за внешний вид и поведение приложения:

Свойство	Описание
Name	Задает имя классу Form , показанному в конструкторе. Данное свойство задается исключительно во время разработки
BackColor	Указывает цвет фона формы
Enabled	Указывает, может ли форма принимать ввод от пользователя. Если свойству Enabled задано значение False , все элементы управления формы также блокируются
ForeColor	Указывает цвет переднего плана формы, то есть цвет выводимого текста. Если отдельно не указать значение свойства ForeColor элементов управления формы, они примут то же значение
FormBorderStyle	Указывает вид и поведение границы и строки заголовка формы Значения свойства: None - Форма не имеет границы, не может быть минимизирована или развернута до максимальных размеров и у нее нет экранной кнопки управления окном и кнопки справки FixedSingle - Форма имеет тонкую границу, и размеры формы нельзя изменить во время выполнения. Форма

Свойство	Описание
	может быть минимизирована, развернута до максимальных размеров, и иметь кнопку справки или кнопку управления окном, что определяется остальными свойствами Fixed3D - Форма имеет объемную границу, и размеры формы нельзя изменить во время выполнения. Форма может быть минимизирована, развернута до максимальных размеров, и иметь кнопку справки или кнопку управления окном, что определяется остальными свойствами
	FixedDialog - Форма имеет тонкую границу, и размеры формы нельзя изменить во время выполнения. У формы нет экранной кнопки управления окном, но может быть кнопка справки, что определяется остальными свойствами. Фому можно минимизировать и развернуть до максимальных размеров
	Sizable - Форма имеет настройки по умолчанию, но они могут изменяться пользователем. Форма может быть минимизирована, развернута до максимальных размеров, и иметь кнопку справки, что определяется остальными свойствами
	FixedToolWindow - Форма имеет тонкую границу, и размеры формы нельзя изменить во время выполнения. Форма содержит только кнопку закрытия
SizeMode	Форма имеет тонкую границу, и размеры формы могут быть изменены пользователем. Форма содержит только кнопку закрытия
Location	Когда свойству StartPosition задано значение Manual , это свойство указывает исходное положение формы относительно верхнего левого угла экрана
MaximizeBox	Указывает, есть ли у формы кнопка MaximizeBox
MinimumSize	Устанавливает минимальный размер формы, который задать этому свойству размер 0; 0, у формы не будет верхнего ограничения размера
MinimizeBox	Указывает, есть ли у формы кнопка MinimizeBox
Opacity	Устанавливает уровень непрозрачности или прозрачности формы от 0 до 100%. Форма, непрозрачность которой составляет 100%, полностью непрозрачна, а форма, имеющая 0 % непрозрачности, наоборот, полностью прозрачна

Свойство	Описание
Size	Принимает и устанавливает исходный размер формы
StartPosition	Указывает положение формы в момент ее первого выводения на экран
Text	Указывает заголовок формы
TopMost	Указывает, всегда ли форма отображается поверх всех остальных форм, свойству TopMost которых не задано значение True
Visible	Указывает, видима ли форма во время работы
WindowState	Указывает, является ли форма минимизированной, развернутой до максимальных размеров, или же при первом появлении ей задан размер, указанный в свойстве Size

Создание нового проекта

1. Откройте Visual Studio и создайте новый проект Windows Forms. Проект откроется с формой по умолчанию с именем **Form1** в конструкторе.
2. Выберите форму в конструкторе. Свойства формы отображаются в окне **Properties**.
3. В окне **Properties** задайте свойствам значения, как указано ниже:

Свойство	Значение
Text	Trey Research
FormBorderStyle	Fixed3D
StartPosition	Manual
Location	100; 200
Opacity	75%

4. Перетащите три кнопки из **Toolbox** в форму и разместите их так, как вам будет удобно.
5. Поочередно выберите каждую кнопку и в окне **Properties** задайте свойству кнопок **Text** значения **Border Style**, **Resize** и **Opacity**.
6. Для кнопки **Border Style** задайте свойство **Anchor – Top, Left**.

Реализация обработчиков событий

7. В конструкторе дважды щелкните кнопку **Border Style**, чтобы открыть окно с кодом обработчика события **Button1 Click**. Добавьте в этот метод следующую строку кода:

```
this.FormBorderStyle = FormBorderStyle.Sizable;
```

8. Возвратитесь в окно конструктора, дважды щелкните кнопку **Resize** и добавьте следующую строку:

```
this.Size = new Size(300, 500);
```

9. Возвратитесь в окно конструктора, дважды щелкните кнопку **Opacity** и добавьте следующую строку:

```
this.Opacity = 1;
```

Запуск готового решения

10. Для построения решения выберите меню **Build (Построение)**, далее команду **Build Solution (Построить решение)**. При наличии ошибок исправьте их и снова постройте решение. В дальнейшем при необходимости выбора последовательности действий очередность команд будет описываться, например, так: **Build | Build Solution**.

11. Нажмите **Ctrl + F5** или выберите **Debug (Отладка) | Start Without Debugging (Запуск без отладки)**, чтобы запустить приложение. Щелкайте каждую кнопку и наблюдайте, как изменяется вид формы.

12. Измените поочередно расположение левой и верхней границ формы и сравните поведение кнопок внутри формы. Обратите внимание, что расстояние до этих границ от кнопки **Border Style** остается постоянным. Почему?

Упражнение 2. Создание непрямоугольной формы Windows

В этом упражнении вы создадите треугольную форму Windows.

1. Откройте Visual Studio и создайте новый проект Windows Forms. Проект откроется с формой по умолчанию с именем **Form1** в конструкторе.

2. В окне Properties задайте свойству **FormBorderStyle** значение **None**, а свойству **BackColor** значение **Red**. В этом случае форму легче будет увидеть при тестировании приложения.

3. Перетащите кнопку из Toolbox в левый верхний угол формы. Задайте свойству **Text** кнопки значение **Close Form**.

4. Дважды щелкните кнопку Close Form и добавьте в обработчик события **Button1 Click** следующий код:

```
this.Close();
```

5. В конструкторе дважды щелкните форму, чтобы открыть обработчик события **Form1 Load**. Добавьте в этот метод следующий код (он задает области формы треугольную форму указанием многоугольника с тремя углами):

```
System.Drawing.Drawing2D.GraphicsPath myPath =  
    new System.Drawing.Drawing2D.GraphicsPath();  
myPath.AddPolygon(new Point[] { new Point(0, 0),  
    new Point(0, this.Height),  
    new Point(this.Width, 0) });  
Region myRegion = new Region(myPath);  
this.Region = myRegion;
```

6. Постройте и запустите приложение. Появится треугольная форма.

Упражнение 3. Создание наследуемой формы

Если у вас имеется уже готовая форма, которую вы собираетесь использовать в нескольких приложениях, удобно создать наследуемую (производную) форму. В этом упражнении вы создадите новую форму и унаследуете ее от существующей базовой формы, а затем измените производную форму, настроив ее для конкретной работы.

1. Откройте проект из предыдущего упражнения. Базовой формой для создания производной будет треугольная форма.
 2. Для кнопки **Close Form** задайте свойство **Modifiers** как **protected**.
 3. Добавьте производную форму: меню **Project (Проект) | Add Windows Form...**(Добавить форму Windows), в окне **Categories (Категории)** укажите **Windows Form**, в окне **Templates (Шаблоны)** выберите **Inherited Form (Наследуемая форма)**.
 4. В окне **Add New Item** в поле **Name** укажите название формы: nForm.cs и нажмите **Add** для добавления формы.
 5. В появившемся окне **Inheritance Picker**, в котором отображаются все формы текущего проекта, выберите базовую форму Form1 и нажмите **OK**.
 6. Постройте проект.
 7. Откройте форму nForm в режиме конструктора. Проверьте, что она имеет треугольную форму и свойства базовой формы и элемента управления наследованы.
 8. Настройте свойства производной формы:
 - для кнопки:
 - свойство **Text** – Hello!!!
 - свойство **BackColor** – Brown
 - для формы: свойство **BackColor** – Blue
 9. Постройте проект.
 10. Задайте производную форму в качестве стартовой, указав в функции **Main** следующий код:
- ```
Application.Run(new nForm());
```

11. Постройте и запустите приложение. Должна открыться производная форма со своими свойствами. Проверьте, наследуется ли закрытие формы кнопкой.

#### **Упражнение 4. Создание MDI-приложения**

В этом упражнении Вы создадите MDI-приложение с родительской формой, загружающей и организующей дочерние формы. Также Вы познакомитесь с элементом управления **MenuStrip**, который позволяет создать меню формы.

#### **Создание нового проекта с базовой формой**

1. Создайте новый проект Windows Forms, укажите имя MdiApplication.
2. Переименуйте файл Form1.cs на ParentForm.cs.
3. Для формы задайте следующие свойства:

| Name | ParentForm  |
|------|-------------|
| Size | 420; 320    |
| Text | Parent Form |

4. Проверьте, что произошли изменения в функции Main так, чтобы форма ParentForm стала стартовой.

5. Откройте файл ParentForm.cs в режиме конструктора.
6. Для свойства формы **IsMdiContainer** задайте значение **True**. Таким способом эта форма будет определена как родительская форма MDI.

### **Создание меню для работы с формами**

#### **7. Создайте пункт меню File:**

- a. Откройте ПИ **Toolbox**, добавьте на форму ЭУ **MenuStrip** и задайте для его свойства **Name** значение **MdiMenu**.
- b. Выделите меню в верхней части формы и задайте имя первого пункта меню **&File**.
- c. Для свойства **Name** пункта меню **File** задайте значение **FileMenuItem**.
- d. Раскройте меню **File**.
- e. Выделите элемент, появившийся под элементом **File**, и задайте его как **&New**.
- f. Для свойства **Name** пункта меню **New** задайте значение **NewMenuItem**.
- g. Выделите элемент, появившийся под элементом **New**, и задайте его как **&Exit**.
- h. Для свойства **Name** пункта меню **Exit** задайте значение **ExitMenuItem**.
- i. Дважды кликните левой кнопкой мыши по пункту меню **Exit** для создания обработчика события **Click**.
- j. В обработчик события **Click** для пункта меню **Exit** добавьте следующий код:  
`this.Close();`

#### **8. Создайте пункт меню Window:**

- a. Переключитесь в режим конструктора.
- b. Выделите второй пункт меню справа от **File** и задайте его значением **&Window**.
- c. Для свойства **Name** пункта меню **Window** задайте значение **WindowMenuItem**.
- d. Раскройте меню **Window**.
- e. Выделите элемент, появившийся под элементом **Window**, и задайте для его свойства **Text** значение **&Cascade**.
- f. Для свойства **Name** пункта меню **Cascade** задайте значение **WindowCascadeMenuItem**.
- g. Выделите элемент, появившийся под элементом **Cascade**, и задайте для его свойства **Text** значение **&Tile**.
- h. Для свойства **Name** пункта меню **Tile** задайте значение **WindowTileMenuItem**.
- i. Дважды кликните левой кнопкой мыши по пункту меню **Cascade** для создания обработчика события **Click**:  
`this.LayoutMdi (System.Windows.Forms.MdiLayout.Cascade);`

- j. Вернитесь в режим конструктора и дважды кликните левой кнопкой мыши по пункту меню **Tile**.
  - k. В обработчик события **Click** для пункта меню **Tile** добавьте следующий код:
- ```
this.LayoutMdi(System.Windows.Forms.MdiLayout.TileHorizontal);
```
9. Реализуйте список открытых окон в меню **Window**:
 - a. В конструкторе выберите компонент **Mdimenu**. Укажите в свойстве **MdiWindowListItem** имя пункта, созданного для этого – **WindowMenuItem**.

Создание дочерней формы

10. Создайте дочернюю форму:
 - a. Выберите пункт меню **Project | Add Windows Form**.
 - b. Задайте имя формы **ChildForm.cs**.
 - c. Для свойства **Text** формы задайте значение **Child Form**.
 - d. На ПИ **Toolbox** дважды кликните левой кнопкой мыши по ЭУ **RichTextBox** и задайте для его свойства **Name** значение **ChildTextBox**.
 - e. Для свойства **Dock** ЭУ **RichTextBox** задайте значение **Fill**.
 - f. Удалите существующий текст (если он есть) для свойства **Text** ЭУ **RichTextBox** и оставьте его пустым.
 - g. На ПИ **Toolbox** дважды кликните левой кнопкой мыши по ЭУ **MenuStrip**.
 - h. Для свойства **Name** ЭУ **MenuStrip** задайте значение **ChildWindowMenu**.
 - i. Выделите меню в верхней части формы и наберите текст **F&ormat**.
 - j. Для свойства **Name** пункта меню **Format** задайте значение **FormatMenuItem**, для свойства **MergeAction** установите значение **Insert**, а свойству **MergeIndex** – 1. В этом случае меню **Format** будет располагаться после **File** при объединении базового и дочерних меню.
 - k. Выделите элемент, появившийся под элементом **Format**, и наберите текст **&Toggle Foreground**.
 - l. Для свойства **Name** пункта меню **Toggle Foreground** задайте значение **ToggleMenuItem**.
 - m. Дважды кликните левой кнопкой мыши по пункту меню **Toggle Foreground** и добавьте следующий код в обработчик события **Click**:

```
if (ToggleMenuItem.Checked)
{
    ToggleMenuItem.Checked = false;
    ChildTextBox.ForeColor = System.Drawing.Color.Black;
}
else
{
    ToggleMenuItem.Checked = true;
}
```

```
    ChildTextBox.ForeColor = System.Drawing.Color.Blue;
}
```

Отображение дочерней формы

11. Отобразите дочернюю форму в родительской форме:
 - a. Откройте ParentForm.cs в режиме конструктора.
 - b. Дважды кликните левой кнопкой мыши по кнопке **New** в меню **File** для создания обработчика события **Click**.
 - c. Добавьте следующий код для обработчика события **Click** для пункта меню **New**:

```
ChildForm newChild = new ChildForm();
newChild.MdiParent = this;
newChild.Show();
```

Работа с приложением

12. Проверьте работу приложения:

- a. Постройте и запустите приложение.
- b. Когда появится родительская форма, выберите пункт меню **File | New**.
- c. В родительском окне появится новая дочерняя форма. Обратите внимание на то, дочернее меню сливаются с родительским и пункты меню упорядочиваются в соответствие со свойством **MergeIndex**, установленным ранее.
- d. Наберите какой-нибудь текст в дочернем окне и воспользуйтесь пунктом меню **Format** для изменения цвета шрифта текста.
- e. Откройте еще несколько дочерних окон.
- f. Выберите пункт меню **Window | Tile**. Обратите внимание на то, что дочерние окна выстраиваются в упорядоченном порядке.
- g. Закройте все дочерние окна.
- h. Обратите внимание на то, что, когда закроется последнее дочернее окно, меню родительской формы изменится, и оттуда исчезнет пункт **Format**.
- i. Для закрытия приложения выберите пункт меню **File | Exit**.

13. Обратите внимание, что заголовок у дочерних окон одинаковый.

При создании нескольких документов, например в Microsoft Word, они называются **ДокументN**, где N — номер документа. Реализуйте эту возможность:

- a. Откройте код родительской формы и в классе ParentForm объяйте переменную **openDocuments**:

```
private int openDocuments = 0;
```

- b. К свойству **Text** дочерней формы добавьте счетчик числа открываемых документов (в коде обработчика события **Click** для пункта меню **New**):

```
newChild.Text = newChild.Text + " " + ++openDocuments;
```

14. Запустите приложение. Теперь заголовки новых документов содержат порядковый номер.

Дополнительное упражнение

Для углубления знаний о добавлении и настройке форм Windows выполните следующие задания.

Задание 1. Создайте пользовательскую форму, которая во время выполнения будет иметь овальное очертание. Данная форма должна содержать функциональность, дающую возможность пользователю закрывать ее во время выполнения.

Рекомендация: при разработке формы в виде эллипса используйте следующий код:

```
// Добавление эллипса, вписанного в прямоугольную форму  
// заданной ширины и высоты  
myPath.AddEllipse(0, 0, this.Width, this.Height);
```

Задание 2. Создайте приложение с двумя формами и установите вторую форму как стартовую. Сделайте так, чтобы при запуске стартовая форма разворачивалась до максимальных размеров и содержала функциональность, дающую возможность пользователю открыть первую форму, отображающуюся в виде ромба зеленого цвета с кнопкой (в центре ромба) закрытия формы с надписью GREENPEACE.

Лабораторная работа 2. Работа с элементами управления

Цель работы

Изучение способов использования элементов управления и получение навыков по обработке событий.

Упражнение 1. Обработка событий Click и MouseMove

Элементы управления – это компоненты, объединяющие графический интерфейс с предварительно разработанной функциональностью. Элементы управления представляют собой многократно используемые блоки кода, предназначенные для выполнения определенных задач. Все элементы управления являются производными базового класса **Control**, а значит, тоже используют различные свойства, задающие размер, расположение и другие основные аспекты элементов управления.

Выполнив первое упражнение этого задания, вы создадите простое приложение, отслеживающее события мыши, которые происходят у конкретного элемента управления.

Размещение на форме элементов управления

1. Создайте новое Windows приложение. Назовите его **WinQuestion**.
2. Расположите на форме две кнопки **Button** и надпись **Label**, разместите их по-своему усмотрению.
3. Установите следующие свойства элементов управления и формы:

Объект	Свойство	Значение
Form1	FormBorderStyle	Fixed3D
	Size	350; 200
	Text	Насущный вопрос
label1	Text	Вы довольны своей зарплатой?
Button1	Name	btnyes
	Text	Да
Button2	Name	bttnno
	Text	Нет

4. Щелкните дважды по кнопке "Да". В обработчике этой кнопки добавьте следующий код:

```
MessageBox.Show("Мы и не сомневались, что Вы так  
думаете!");
```

5. Выделите кнопку "Нет". В окне **Properties** переключитесь в окно событий и дважды щелкните в поле **MouseMove**.

6. В обработчике этого события добавьте код для связывания движения мыши с координатами кнопки и указания координат, куда кнопка будет перемещаться:

```
bttnno.Top -= e.Y;  
bttnno.Left += e.X;  
if (bttnno.Top < -10 || bttnno.Top > 100)  
    bttnno.Top = 60;  
if (bttnno.Left < -80 || bttnno.Left > 250)  
    bttnno.Left = 120;
```

7. Запустите приложение и нажмите на каждую из кнопок.

Упражнение 2. Работа со списками

Основными элементами управления списком являются **ListBox**, **ComboBox** и **CheckedListBox**. Несмотря на некоторые отличия во внешнем виде и разные функциональные возможности, они одинаково формируют и представляют списки данных и включают в себя коллекцию **Items**, которая систематизирует элементы, содержащие один из этих элементов управления.

ListBox — самый простой элемент управления списка. Он служит главным образом для отображения простого списка элементов в пользовательском интерфейсе, по которому легко перемещаться.

CheckedListBox — помечаемый список. Является разновидностью простого списка. Его дополнительное достоинство — в наличии чекбоксов рядом с каждым элементом списка. Пользователь имеет возможность отметить один или несколько элементов списка, выставив напротив его флажок.

ComboBox — выпадающий список. Постоянно на форме представлено только одно значение этого списка. При необходимости пользователь может раскрыть список и выбрать другое интересующее его значение или ввести собственное.

Создание приложения, использующее список

1. Создайте новый проект Windows Forms, укажите имя TestList.

2. Добавьте на форму следующие элементы управления:

- a. **GroupBox**,
- b. **CheckedListBox** (поместите в **GroupBox**)
- c. **ComboBox**
- d. три элемента **Button**.

3. Установите следующие свойства формы и элементов управления:

Объект	Свойство	Значение
Form1	FormBorderStyle	Fixed3D
	Text	Работа со списками
	Size	410;310
groupBox1	Text	Список участников
CheckedListBox	Name	memberList
ComboBox	Name	peopleList
	Text	
Button1	Name	buttonAdd
	Text	Добавить
Button2	Name	buttonDelete
	Text	Удалить
Button3	Name	buttonSort
	Text	Сортировать

4. Проинициализируйте элемент управления **ComboBox** списком предполагаемых участников. Для этого в окне свойств peopleList выберите свойство **Items**. Откройте окно **String Collection Editor**, нажав на кнопку с тремя точками в поле **Items**. Добавьте в окно Ф.И.О. нескольких участников. Нажмите OK.

5. Добавьте обработчики для кнопок **Добавить** и **Удалить**, два раза щелкнув левой кнопкой мыши по каждой из кнопок.

6. В тело обработчика события кнопки **Добавить** вставьте следующий код:

```
if (peopleList.Text.Length != 0)
{
    memberList.Items.Add(peopleList.Text);
}
else MessageBox.Show("Выберите элемент из списка или введите новый");
```

7. Для реализации удаления элементов из списка введите код в тело обработчика события кнопки **Удалить**:

```
while (memberList.CheckedIndices.Count > 0)
    memberList.Items.RemoveAt(memberList.CheckedIndices[0]);
```

8. Для реализации сортировки элементов введите код в тело обработчика события кнопки **Сортировать**:

```
memberList.Sorted = true;
```

9. Откомпилируйте и запустите приложение. Заполните список участников, выбирая их из элемента **ComboBox**. Запишите новые данные в этот элемент и добавьте их в список. Отсортируйте список участников.

Упражнение 3. Создание и использование элемента управления ToolStrip

ToolStrip – это элемент управления, разработанный с целью упрощения создания пользовательских панелей инструментов, которые выглядят и работают, как панели инструментов Microsoft Office и Microsoft Internet Explorer. Используя элемент управления **ToolStrip**, вы можете быстро разрабатывать легко настраиваемые панели инструментов профессионального вида.

Добавление на форму шаблона панели инструментов

1. Откройте проект MdiApplication.
2. Откройте форму **ParentForm** в режиме конструктора.
3. Добавьте на форму ЭУ **ToolStrip**.

4. На форме откройте выпадающий список ЭУ **ToolStrip** и выберите **button** – добавится элемент **toolStripButton1**. В панели инструментов он представлен в виде кнопки с рисунком, обозначающим функцию, которую этот элемент содержит.

5. Снова откройте выпадающий список ЭУ **ToolStrip** и выберите **Separator** – добавится элемент, который отделяет одни элементы панели инструментов от других.

6. Справа от разделителя добавьте еще две кнопки – элементы **toolStripButton2** и **toolStripButton3**.

7. В итоге вы должны получить три кнопки, отделенные одним разделителем.

Отображение рисунка на элементах панели управления

8. Выберите первую кнопку. Убедитесь, что в окне Properties свойству **DisplayStyle** задано значение **Image**.

9. В окне Properties выберите изображение элемента управления, щелкнув свойство **Image** и выбрав соответствующее изображение или путь к нему в диалоговом окне **Select Resource**. Если у Вас есть готовые файлы подходящих изображений, то выберите их, в противном случае укажите любой рисунок из папки *Мои рисунки*. Набор готовых изображений можно найти в графических файлах в папке `\Microsoft Visual Studio 9.0\Common7\VS2008ImageLibrary\`.

10. Повторите предыдущие два пункта для остальных кнопок.

Создание графических изображений кнопок панели инструментов

11. Если Вас не устраивает вид готовых рисунков, то можно для придания кнопкам графических изображений самостоятельно их создать. Для работы с изображениями в среде разработки существует **Image Editor**. Он позволяет создать изображения с использованием простейших инструментов.

12. Для создания файла с изображением выберите меню **File | New | File...**. В появившемся окне **New File** выберите тип файла

“Файл точечного рисунка”, нажмите **Open**. Появится пустое изображение с дополнительной панелью управления. В основном меню появится новый пункт меню **Image**. Для отображения панели с палитрой компонент выберите в меню пункт **Image | Show Colors Window (Показать окно выбора цвета)**.

13. Создайте по своему усмотрению изображение для кнопки **New**, например, в виде белого листа и сохраните изображение в файл с именем **Icon_New.bmp** в каталог с решением.

14. Повторите действия для создания иконок для других кнопок, например, в виде нарисованных распылителем букв **C** и **T**. Сохраните изображения в каталог с решением в файлы с именами **Icon_Windows_Cascade.bmp** и **Icon_Windows_Title.bmp** соответственно.

15. Выполните действия для указания новых изображений этим кнопкам.

16. Сохраните и запустите проект. Проверьте вид и работоспособность кнопок.

Добавление обработчиков событий для кнопок

17. Добавьте обработчик события **Click** объекта **toolStrip1**, щелкнув два раза указателем мыши по имени события **Click** на закладке событий в окне свойств. В программу добавится функция **toolStrip1_ItemClicked** как обработчик события, происходящего при нажатии кнопки на панели инструментов.

18. В окне **Properties** для **toolStripButton1** в свойстве **Tag** запишите **NewDoc**. Аналогично укажите для **toolStripButton2** и **toolStripButton3** для свойства **Tag** значения **Cascade** и **Title** соответственно.

19. Укажите для кнопок всплывающие подсказки в свойстве **ToolTipText**: Create new document, Windows cascade и Windows title.

20. В обработчике события **Click** объекта **toolStrip1_ItemClicked** добавьте код, который будет реализовывать различные действия в зависимости от нажимаемой кнопки:

```
switch(e.ClickedItem.Tag.ToString())
{
    case "NewDoc":
        ChildForm newChild = new ChildForm();
        newChild.MdiParent = this;
        newChild.Show();
        newChild.Text = newChild.Text+" "+++openDocuments;
        break;
    case "Cascade":
        this.LayoutMdi (System.Windows.Forms.MdiLayout.Cascade);
        break;
    case "Title":
        this.LayoutMdi
        (System.Windows.Forms.MdiLayout.TileHorizontal);
        break;
}
```

21. Откомпилируйте и запустите приложение. Проверьте работоспособность кнопок.

Упражнение 4. Использование элемента управления StatusStrip

Элемент управления **StatusStrip** применяется в программах для вывода информации в строку состояния — небольшую полоску, расположенную внизу приложения. В этом упражнении вы добавите к приложению **MdiApplication** строку состояния, на которой показывается вариант ориентации окон и выводится текущая дата.

1. Откройте проект **MdiApplication**.
2. Увеличьте размер формы **ParentForm** до значения (450;350).
3. Добавьте на форму **ParentForm** элемент управления **StatusStrip**.
4. Удалите содержимое поля свойства **Text**.
5. Щелкните на кнопку выпадающего списка панели и выберите **StatusLabel**. Добавится элемент **toolStripStatusLabel1** – первая панель для отображения.
6. Создайте еще одну панель аналогичным способом – **toolStripStatusLabel2** и установите им следующие свойства:

Объект	Свойство	Значение
Первая панель	Text	Status
	Name	spWin
Вторая панель	Text	Data
	Name	spData

7. Для отображения информации на первой панели вставьте в соответствующие обработчики команд меню и кнопок на панели инструментов следующую строку кода:

- a. Для каскадной ориентации:
`spWin.Text="Windows is cascade";`
- b. Для горизонтальной ориентации:
`spWin.Text="Windows is horizontal";`
8. Для отображения даты на второй панели в конструкторе формы **ParentForm** добавьте код:

```
public ParentForm()
{
    InitializeComponent();
    // Свойству Text панели spData устанавливается текущая дата
    spData.Text =
        Convert.ToString(System.DateTime.Today.ToString("G"));
}
```

9. Откомпилируйте и запустите приложение. Проверьте работоспособность панели состояния.

Упражнение 5. Работа с контейнерными элементами управления

Контейнерные элементы управления — это специализированные элементы управления, выступающие в роли настраиваемого вместилища для других элементов управления. К контейнерным элементам управления относятся **Panel** и **GroupBox**. Они предоставляют форме логические и

физические подразделы, которые могут группировать другие элементы управления в единообразные подгруппы пользовательского интерфейса. Например, элемент управления **GroupBox** содержит в себе набор связанных элементов управления **RadioButton**. Контейнерные элементы управления помогут вам создать ощущение стиля или информационного потока в пользовательском интерфейсе и позволят согласованно управлять элементами управления, которые содержатся в них.

Выполнив это упражнение, вы научитесь создавать формы Windows с использованием различных контейнерных элементов управления.

Создание проекта с возможностью группировки элементов на вкладках

1. Откройте Visual Studio и создайте новый проект Windows Forms. Назовите его WinContainer.

2. Перетащите из Toolbox в форму элемент управления **TabControl**. В окне Properties задайте свойству **Dock** значение **Fill**.

3. В окне Properties выберите свойство **TabPages**, чтобы открыть TabPage Collection Editor. Добавьте вкладки так, чтобы их стало всего пять. Задайте свойствам **Text** этих пяти элементов управления **TabPage** значения **GroupBox**, **Panel**, **FlowLayoutPanel**, **TableLayoutPanel** и **SplitContainer**. Щелкните OK.

Настройка контейнерного элемента **GroupBox**

4. В форме выберите вкладку **GroupBox**. Перетащите элемент управления **GroupBox** из Toolbox в элемент управления **TabPage**.

5. Перетащите в **GroupBox** два элемента управления **RadioButton**.

6. Добавьте на вкладку **GroupBox** вне элемента управления **GroupBox** кнопку (элемент управления **Button**). Для кнопки свойство **Text** сделайте пустым, а свойству **Name** укажите значение **but**.

7. Дважды кликните по кнопке и добавьте код обработчика события установки надписи на кнопке в зависимости от выбранного переключателя (**RadioButton**):

```
if (radioButton1.Checked == true)
    this.but.Text = "First";
else if (radioButton2.Checked == true)
    this.but.Text = "Second";
```

Настройка элемента **Panel**

8. Выберите в форме вкладку **Panel**. Перетащите элемент управления **Panel** из Toolbox в элемент управления **TabPage**. Для элемента **Panel** задайте свойству **Dock** значение **Fill**.

9. Перетащите четыре элемента управления **Button** из Toolbox в элемент управления **Panel**.

10. Свойству **AutoScroll** установите значение **True**, в этом случае элемент управления **Panel** будет отображать полосы прокрутки, если элементы находятся за пределами видимых границ.

Настройка элемента FlowLayoutPanel

11. Выберите в форме вкладку **FlowLayoutPanel**. Перетащите элемент управления **FlowLayoutPanel** из **Toolbox** в элемент управления **TabPage**. Задайте значение **Fill** свойству **Dock** элемента управления **FlowLayoutPanel**.

12. Перетащите четыре элемента управления **Button** из **Toolbox** в элемент управления **Panel**. Обратите внимание на размещение добавляемых элементов: по умолчанию порядок следования элементов управления в *FlowLayoutPanel* — слева направо. Это значит, что элементы управления, расположенные в *FlowLayoutPanel*, будут находиться в левом верхнем углу и размещаться вправо до тех пор, пока не достигнут края панели. Такое поведение контролируется свойством *FlowDirection*, которому может быть задано четыре значения заливки в *FlowLayoutPanel*: *LeftToRight* — по умолчанию, *RightToLeft* — справа налево, *TopDown* — сверху вниз и *BottomUp* — снизу вверх.

13. Дважды щелкните кнопку **button5** и добавьте в обработчик события **button5_Click** следующий код:

```
flowLayoutPanel1.SetFlowBreak(button6, true);
```

Настройка элемента TableLayoutPanel

14. Выберите конструктор формы (если это необходимо). В форме выберите вкладку **TableLayoutPanel**. Перетащите элемент управления **TableLayoutPanel** из **Toolbox** в **TabPage**. Задайте свойству **CellBorderStyle** (определяет вид ячеек таблицы и их поведение) значение **Inset**, а свойству **AutoScroll** — **True**.

15. Перетащите элемент управления **Button** из **Toolbox** в левую верхнюю ячейку элемента управления **TableLayoutPanel**.

16. Дважды щелкните **Button9** и добавьте в обработчик события **Button9_Click** следующий код:

```
Button aButton = new Button();
tableLayoutPanel1.Controls.Add(aButton, 1, 1);
```

Настройка элемента SplitContainer

17. В конструкторе выберите вкладку **SplitContainer**. Перетащите элемент управления **SplitContainer** из **Toolbox** в **TabPage**. Задайте свойству **BorderStyle** значение **Fixed3D**.

18. Перетащите два элемента управления **Button** из **Toolbox** в **Panell1** элемента управления **SplitContainer**. Задайте свойствам **Text** этих кнопок значения *Fix/Unfix Panell1* и *Fix/Unfix Splitter*. Измените размеры кнопок так, чтобы отображался текст.

19. Добавьте кнопку в **Panell2** и задайте свойству **Text** значение *Collapse/Uncollapse Panell1*. Измените размеры кнопок так, чтобы отображался текст.

20. Дважды щелкните кнопку *Fix/Unfix Panell1* и добавьте в обработчик события **Click** следующий код:

```
if (splitContainer1.FixedPanel == FixedPanel.Panell1)
    splitContainer1.FixedPanel = FixedPanel.None;
```

```
        else
            splitContainer1.FixedPanel = FixedPanel.Panel1;
```

21. Дважды щелкните кнопку *Fix/Unfix Splitter* и добавьте в обработчик события *Click* следующий код

```
splitContainer1.IsSplitterFixed =
! (splitContainer1.IsSplitterFixed);
```

22. Дважды щелкните кнопку *Collapse/Uncollapse Panel1* и добавьте в обработчик события *Click* следующий код:

```
splitContainer1.Panel1Collapsed =
! (splitContainer1.Panel1Collapsed);
```

23. Постройте и запустите приложение.

24. На вкладке *GroupBox* поочередно выбирайте переключатели следите за изменением надписи на кнопке.

25. На вкладке *Panel* измените размер формы с помощью мыши. Проверьте, появились ли полосы прокрутки.

26. На вкладке *FlowLayoutPanel* измените размер формы с помощью мыши. Просмотрите, что автоматически изменилась компоновка. Щелкните кнопку **button5** и проверьте, прервалась ли последовательность на элементе управления **button6** (это было реализовано вызовом метода *SetFlowBreak*).

27. На вкладке *TableLayoutPanel* щелкните кнопку **button9**, добавится новая кнопка.

28. На вкладке *SplitContainer* измените размеры формы, а также размеры каждой панели, передвинув *Splitter*. По очереди щелкайте каждую кнопку и смотрите, как это отражается на возможности элемента управления изменять свои размеры.

Упражнение 6. Элементы с поддержкой отображения текста

Выполнив это упражнение, вы научитесь использовать элемент управления **LinkLabel** в форме и настраивать его так, чтобы открывалось диалоговое окно, запрашивающее у пользователя имя.

Настройка элемента управления LinkLabel

1. Откройте Visual Studio и создайте новый проект Windows Forms. Назовите его **WinLinkLabel**.

2. Перетащите два элемента управления **LinkLabel** из *Toolbox* в форму.

3. В окне *Properties* для первого элемента задайте свойству **Text** значение **Open Form**, для второго – значение **Microsoft**.

Создание диалоговой формы

4. В меню *Project* выберите *Add Windows Form* и добавьте в свой проект новую форму Windows с именем **Form2**.

5. В конструкторе перетащите два элемента управления **Button** в форму **Form2**. Задайте свойству **Text** этих кнопок значения **Accept** и **Cancel**. Расположите их в правом нижнем углу формы.

6. Задайте свойству **DialogResult** кнопки **Accept** значение **OK**, а свойству **DialogResult** кнопки **Cancel** — значение **Cancel**.

7. Перетащите два элемента управления **TextBox** из **Toolbox** в форму.

8. Задайте свойству **Modifiers** каждого элемента управления **TextBox** значение **Internal**. Свойство **Text** для них оставьте пустым.

9. Перетащите два элемента управления **Label** из **Toolbox** в форму и разместите их рядом с элементом управления **TextBox**.

10. Задайте свойствам **Text** элементов управления **Label** значения **&First Name** и **&Last Name**.

11. Проверьте, что свойству **UseMnemonic** всех надписей установлено значение **True**.

12. Установите свойство **TabIndex** в окне **Properties**, как показано ниже.

Элемент управления	Индекс закладки
label1	0
textBox1	1
label2	2
textBox2	3
button1	4
button2	5

Реализация обработчика события вызова диалогового окна

13. Выберите в конструкторе закладку для формы **Form1**. Дважды щелкните первый элемент управления **linkLabel1** для создания обработчика события **linkLabel1_LinkClicked**. Добавьте следующий код:

```
 DialogResult aResult;
Form2 aForm = new Form2();
aResult = aForm.ShowDialog();
if (aResult == System.Windows.Forms.DialogResult.OK)
{
    MessageBox.Show("Your name is " + aForm.textBox1.Text + "
" + aForm.textBox2.Text);
}
linkLabel1.LinkVisited = true;
```

Реализация обработчика события вызова веб-страницы

14. Выберите в конструкторе закладку для формы **Form1**. Дважды щелкните второй элемент управления **linkLabel2** для создания обработчика события **linkLabel2_LinkClicked**. Добавьте следующий код:

```
System.Diagnostics.Process.Start("www.limtu.com");
linkLabel2.LinkVisited = true;
```

15. Постройте и запустите приложение.

16. Щелкните элемент управления **linkLabel – Open Form**, чтобы открыть форму. Введите соответствующую информацию в поля ввода и проверьте кнопки **Accept** и **Cancel**.

17. Щелкните элемент управления **linkLabel2 – Microsoft**, чтобы открыть сайт известного учебного центра.

Упражнение 7. Элементы с поддержкой редактирования текста

TextBox — это основной элемент управления, с помощью которого можно принимать вводимый пользователем текст а также отображать текст для пользователя. Существует возможность создавать как поля, отображающие многострочный текст, так и поля отображающие знак пароля вместо реально введенного текста.

Элемент управления **MaskedTextBox** — это видоизмененный элемент управления **TextBox**, позволяющий задавать предварительно установленный шаблон для принятия пользовательского ввода или отказа от него. С помощью свойства **Mask** можно указать обязательные или необязательные символы либо тип вводимых символов (буквы или цифры) и применить форматирование для отображения строк.

1. Откройте выполненное вами в предыдущем упражнении решение **WinLinkLabel**.

2. Отобразите конструктор для формы *Form2*.

3. Добавьте элемент управления **TextBox** на форму под расположенными ранее элементами. Перетащите элемент управления **Label** в форму и разместите слева от этого элемента.

4. Задайте свойству **Text** элемента управления **Label** значение **Address**.

5. Для элемента управления **TextBox** задайте следующие свойства:

Свойства	Значение	Комментарий
Multiline	True	многострочный
WordWrap	False	переход слова с одной строки на другую
ScrollBars	Both	отображение полос прокрутки

6. Измените размеры элемента управления **TextBox** так, чтобы он вмещал адрес. При необходимости увеличьте размеры формы и переместите кнопки **Accept** и **Cancel**.

7. Перетащите элементы управления **MaskedTextBox** и **Label** из **Toolbox** на форму и разместите их под ранее введенные элементы.

8. Свойству **Text** элемента управления **Label** задайте значение **Phone Number**.

9. Задайте значение **(999)-000-0000** свойству **Mask** элемента управления **MaskedTextBox**.

10. Задайте значение **Internal** свойству **Modifiers** для последних элементов управления **TextBox** и **MaskedTextBox**.

11. Откройте окно кода формы *Form1*.

12. В обработчике события **linkLabel1_LinkClicked** добавьте в блок **if**, расположенный под кодом, который вы добавили в предыдущем упражнении, следующий код

```
MessageBox.Show("Your address is " + aForm.textBox3.Text);  
MessageBox.Show("Your phone number is " +  
aForm.maskedTextBox1.Text);
```

13. Постройте и запустите приложение. Введите в текстовое поле свой телефон. Проверьте, что номер отображается согласно требуемому формату.

Упражнение 8. Добавление и удаление элементов управления в режиме работы приложения

При размещении на форме элемента управления в режиме дизайна, среда создает код, описывающий этот элемент. Если назначить в обработчике заданного элемента управления генерацию аналогичного кода, то в запущенном приложении можно будет добавлять на форму или удалять элементы, активизируя этот обработчик.

Для работы с элементами управления используется объект **ControlsCollection**, содержащий ряд методов, основные из которых будут использованы в данном упражнении.

1. Создайте новое приложение и назовите его RegistrationForm.
2. Добавьте на форму три надписи, два текстовых поля, кнопку, элементы CheckBox и GroupBox
3. Установите следующие значения свойств формы и элементов управления:

Объект	Свойство	Значение
Form1	FormBorderStyle	Fixed3D
	Text	Регистрация
	Size	400;310
label1	Location	30;10
	Text	Выберите тип регистрации
label2	Location	16; 32
	Text	Name
label3	Location	16; 64
	Text	PIN
button1	Location	80; 248
	Text	Регистрация
textBox1	Location	96; 32
	Text	
	Size	184; 20
textBox2	Location	96; 64
	Size	184; 20
	Text	
checkBox1	Location	40; 40
	Size	232; 24
	Text	Расширенные возможности
groupBox1	Text	Введите регистрационные данные
	Location	16; 80
	Size	344; 144

4. Для реализации возможности добавления и удаления элементов в процессе выполнения программы реализуйте обработчик события CheckedChanged: щелкните дважды на элементе checkBox1 и добавьте следующий код:

```
if (checkBox1.Checked == true)
{
    Label lbl = new Label();
    lbl.Location = new System.Drawing.Point(16, 96);
    lbl.Size = new System.Drawing.Size(32, 23);
    lbl.Name = "label1";
    lbl.TabIndex = 2;
    lbl.Text = "PIN2";
    groupBox1.Controls.Add(lbl);
    TextBox txt = new TextBox();
    txt.Location = new System.Drawing.Point(96, 96);
    txt.Size = new System.Drawing.Size(184, 20);
    txt.Name = "textboxx";
    txt.TabIndex = 1;
    txt.Text = "";
    groupBox1.Controls.Add(txt);
}
else {
}
```

5. Откомпилируйте и запустите приложение. Проверьте, что при установке флашка в ЭУ checkBox “Расширенные возможности” на форме появляется надпись и поле ввода для дополнительных данных.

6. Для удаления ЭУ с формы могут применяться методы: **Clear** (удаление всех элементов из коллекции), **Remove** (удаление элемента из коллекции) и **RemoveAt** (удаление элемента по заданному индексу). В тело оператора else добавьте код для удаления ЭУ по индексу:

```
int lcv;
lcv = groupBox1.Controls.Count; // определяется количество
while (lcv > 4)
{
    groupBox1.Controls.RemoveAt(lcv - 1);
    lcv -= 1;
}
```

7. Запустите приложение. Убедитесь, что при включении “Расширенные возможности” дополнительные элементы появляются на форме, а при выключении – исчезают.

Упражнение 9. Проверка вводимых значений. События KeyPress и Validating. Элемент управления ErrorProvider

При внесении значений параметров пользователем во многих случаях требуется проверять вводимый текст по заданным критериям. Например, регистрационный номер, телефон не должны содержать букв, поле имени – цифр. В этом упражнении рассматриваются реализации проверок, которые можно осуществлять, используя встроенные события текстового поля.

Использование события KeyPress

1. Откройте приложение RegistrationForm.

2. Выделите поочередно текстовые поля **TextBox1** и **TextBox2**, в окне **Properties** создайте обработчики события **KeyPress**, возникающего при нажатии любой клавиши в поле.

3. В тело обработчика события **KeyPress** для текстового поля **TextBox1** укажите следующий код (для элемента **TextBox1** недопустимыми значениями будут цифры):

```
if (char.IsDigit(e.KeyChar) )  
{  
    e.Handled = true;  
    MessageBox.Show("Поле Name не может содержать цифры");  
}
```

4. Для элемента **TextBox2**, наоборот, недопустимыми значениями будут буквы, в обработчике события **KeyPress** для текстового поля **TextBox2** укажите код:

```
if (!char.IsDigit(e.KeyChar) )  
{  
    e.Handled = true;  
    MessageBox.Show("Поле PIN не может содержать буквы");  
}
```

5. Откомпилируйте и запустите приложение. Попробуйте ввести в поле **Name** цифры, в поле **PIN** – буквы.

6. Для защиты текстового поля, появляющегося при установке галочки в чекбоксе "Расширенные возможности", необходимо вручную определить событие **KeyPress**. В обработчике события **CheckedChanged** для элемента **CheckBox1** укажите код:

```
txt.KeyPress+= new  
System.Windows.Forms.KeyPressEventHandler(this.textBox2_KeyPress);
```

7. Запустите и протестируйте приложение.

Применение события **Validating**

Событие **KeyPress** блокирует часть клавиатуры. Другим способом проверки является событие **Validating**, позволяющее работать с клавиатурой, но блокирующее другие действия пользователя.

8. Закомментируйте обработчик элемента **TextBox2**.

9. В режиме дизайна формы в окне **Properties** элемента **TextBox2** создайте обработчик события **Validating** и запишите следующий код:

```
if(textBox2.Text == "")  
{  
    e.Cancel=false;  
}  
else  
{  
    try  
    {  
        double.Parse(textBox2.Text);  
        e.Cancel = false;  
    }  
    catch  
    {  
        e.Cancel = true;  
        MessageBox.Show("Поле PIN не может содержать буквы");  
    }  
}
```

```
    }  
}
```

10. Запустите приложение. При переключении фокуса ввода или нажатии на кнопку "регистрация" происходит событие **Validating**.

Применение элемента управления ErrorProvider

Элемент управления **ErrorProvider** удобно применять, когда нужно выводить небольшую иконку в случае ошибки ввода.

11. В режиме дизайна из окна **ToolBox** перенесите на форму элемент управления **ErrorProvider**.

12. В коде формы в обработчике `textBox1_KeyPress` добавьте следующую строку:

```
errorProvider1.SetError(textBox1, "Must be letter");
```

13. Запустите приложение. При ошибке ввода появляется мигающая иконка уведомления, при наведении на нее всплывает поясняющее сообщение об ошибке.