Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский Государственный Университет»

Практическая работа №5

Настройка командных элементов управления, а также элементов с поддержкой отображения

Подготовил:

Старший преподаватель

кафедры ПОВТАС

Резак Е.В.

Хабаровск. 2021

**Тема**: Настройка командных элементов управления, а также элементов с поддержкой отображения текста.

**Цель работы**:

1. Изучить теорию. Подготовить конспект по теории.

3. Выполнить задания.

**Настройка командных элементов управления, а также элементов с поддержкой отображения текста.**

**Элемент управления Button.**

Элемент управления Button - это один из наиболее известных элементов управления Toolbox. Он является главным элементом управления, обеспечивающим командное взаимодействие между пользователем и пользовательским интерфейсом. Элемент управления Button может отображать небольшую надпись на кнопке и реагировать на щелчки пользователя. Он дает визуальную подсказку, когда его щелкают мышью, и предоставляет обработчик события, который дает возможность пользователю напи¬сать код, выполняемый при щелчке Button Элемент управления Button предоставляет несколько свойств, благодаря которым можно настроить его внешний вид и поведение.

***Основные свойства элемента управления Button***.

|  |  |
| --- | --- |
| AutoEllipsis | Обеспечивает автоматическую обработку текста, который выходит за ширину кнопки |
| DialogResult | Задает значение DialogResult, которое можно связать с такой кнопкой, как DialogResult.ОК или DialogResult.Cancel |
| FlatAppearance | Определяет стили, управляющие внешним видом и поведением кнопки, когда свойству FlatStyle задано значение Flat |
| FlatStyle | Задает визуальный стиль кнопки, когда пользователь наводить указатель мыши на кнопку и щелкает ее |
| Text | Задает текст на кнопке |
| TextAlign | Указывает, как будет выровнен текст на кнопке |

**Реагирование на щелчки мыши.**

Основная функция элемента управления Button в пользовательском интерфейсе заключается в том, чтобы реагировать на щелчки пользователя. Обычно код выполняется щелчком мыши элемента управления Button. Например, у вас есть кнопка О К, которая после предоставления пользователем необходимой информации приводит к продолжению выполнения приложения, или у вас есть кнопка Cancel, возвращающая выполнение на шаг назад.

С помощью обработчика события Button.Click вы можете написать код, который будет выполняться после щелчка мышью кнопки. Это метод, при котором принимается щелчок мышью кнопки, а затем выполняется соответствующий код.

***Написание кода для обработчика события Button.Click***

1. В конструкторе дважды щелкните кнопку, для которой необходимо написать код. Visual Studio автоматически генерирует объявление метода с именем ButtonClick и самостоятельно добавляет код, чтобы сконфигурировать метод для обработки события Button.Click. Visual Studio отображает новый метод Button Click, который будет выполнен, когда пользователь щелкнет кнопку.

2. Напишите в этом методе соответствующий код. После запуска программы код будет выполняться щелчком кнопки. Следующий код демонстрирует полный пример метода Button Click.

private void button1\_Click(object sender, EventArgs e)

{

MessageBox.Show("The Button has been clicked!");

}

**Реагирование на другие щелчки мыши.**

Несмотря на то что обработчик события ButtonClick полезен при реагировании на простые щелчки мыши, вы также можете внести соответствующие настройки, чтобы кнопка или другой элемент управления реагировал на другие щелчки мыши, например, на щелчок правой кнопкой мыши. Благодаря событию MouseDown вы сможете реагировать на эти щелчки.

Экземпляр класса MouseClickEventArgs — один из аргументов обработчика события Button\_MouseDown. Данный аргумент содержит подробную информацию о положении и типе щелчка мыши, благодаря чему с его помощью можно различать правые, левые и двойные щелчки или другие виды взаимодействия с мышью.

|  |  |
| --- | --- |
| Button | Указывает щелкнутую кнопку мыши. Возможные значения — Left, Right, Middle, None, XButtonl или XButton2 |
| Clicks | Задает, сколько раз была щелкнута кнопка |
| Delta | Указывает, сколько раз было прокручено колесо мыши |
| Location | Указывает текущее положение указателя мыши |
| X | Указывает координату X указателя мыши |
| Y | Указывает координату Y указателя мыши |

С помощью значений, предоставленных экземпляром MouseClickEventArgs, вы можете определить щелкнутую кнопку и положение колеса мыши. Помните, что, если щелкнуть элемент управления не левой кнопкой, а какой-либо другой, он визуально не среагирует (щелчок в пользовательском интерфейсе), что обычно происходит с кнопкой.

***Реагирование на различные щелчки мыши***

1. Выберите в конструкторе элемент управления Button, для которого необходимо написать код.

2. Чтобы просмотреть события элемента управления Button, щелкните в окне Properties кнопку с изображением молнии.

3. Чтобы в Visual Studio сгенерировать и отобразить обработчик события Button\_ MouseDown, в окне Properties дважды щелкните ячейку, расположенную рядом с MouseDown.

4. Напишите код в этом обработчике события, который реагирует на щелчки нужных комбинаций кнопок мыши. Следующий пример демонстрирует, как различать правую и левую кнопки мыши.

private void button1\_MouseDown(object sender, MouseEventArgs e)

{

switch (e.Button)

{

case MouseButtons Left:

MessageBox.Show("The left button was clicked”); break;

case MouseButtons.Right:

MessageBox.Show("The right button was clicked"); break; default:

MessageBox.Show("Some other button was clicked"); break;

}

}

**Свойства FlatStyle и FlatAppearance.**

Свойство FlatStyle определяет вид кнопки — трехмерный и рельефный или плоский. Для того чтобы кнопка имела плоский вид, необходимо свойству FlatStyle задать значение Flat. Когда FlatStyle присвоено значение Flat, свойство FlatAppearance определяет внешний вид и поведение кнопки в пользовательском интерфейсе. Свойство FlatAppearance является экземпляром структуры, которая содержит свойства.

|  |  |
| --- | --- |
| BorderColor | Задает цвет границы кнопки |
| BorderSize | Задает размер границы кнопки |
| MouseDownBackColor | Задает цвет кнопки, когда ее щелкают левой кнопкой мыши |
| MouseOverBackColor | Задает цвет кнопки, когда на нее наводят указатель мыши |

Когда свойству FlatStyle задано значение Flat, у пользователя меньше встроенных визуальных подсказок для взаимодействия с кнопкой. Обеспечить дополнительные подсказки можно, задав соответствующие значения свойству FlatAppearance. Следующая процедура описывает, как задается цвет фона кнопки, когда на нее наведен указатель мыши.

***Изменение цвета фона кнопки при наведении на нее указателя мыши***

1. В окне Properties задайте свойству FlatStyle значение Flat.

2. Откройте свойство Flat Appearance.

3. В свойстве FlatAppearance задайте свойству MouseOverBackColor тот цвет кнопки, который она должна иметь при наведении на нее указателя мыши.

**Кнопки принятия и отмены.**

При создании диалоговых форм, как правило, в форме создается кнопка Accept или Cancel, что обеспечивает соответствующее установление свойства DialogResult формы при щелчке кнопки. Создать кнопки Accept или Cancel можно с помощью свойства DialogResult элемента управления Button.

***Создание кнопки Accept или Cancel***

1. Перетащите кнопку из Toolbox в форму и задайте свойству Text соответствующее значение (например, значение Accept для кнопки Accept).

2. В окне Properties задайте свойству DialogResult значение ОК для кнопки Accept или значение Cancel для кнопки Cancel.

3. Дважды щелкните кнопку в конструкторе, чтобы открыть окно кода.

4. Закройте форму в обработчике события ButtonjClick, как показано ниже:

this.Close();

Если при вызове формы с помощью метода ShowDialog он автоматически возвра¬тит диалоговый результат, связанный со щелкнутой кнопкой. Следующий пример демонстрирует, как можно использовать эту форму. Предположительной форме дано имя DialogForm.

// C#

Dialog aForm = new DialogForm();

System Window.Forms.DialogResult aResult; aResult = aForm. ShowDialogO;

if (aResult == DialogResult.Ok)

{

// Действия

}

else

{

// Иные действия

}

**Элемент управления Label.**

Элемент управления Label в основном используется для отображения пользователю текстовой информации только для чтения. Так, для отображения информационной строки рядом с элементом управления часто применяют надписи, например, First Name — рядом с элементом управления TextBox, предназначенным для имени пользователя. Надписи также можно использовать с целью указания клавиш быстрого вызова для других элементов управления.

Текст, отображаемый в надписи, задается в ее свойстве Text. Чтобы надпись автоматически подстраивала свои размеры под размеры текста, ее свойству AutoSize нужно задать значение True. Если AutoSize присвоено значение False, задать размер надписи можно, сместив края элемента управления в конструкторе.

Элементы управления Label можно использовать для указания клавиш доступа для других элементов управления. Клавиши доступа — это клавиши, которые, при их нажатии вместе с клавишей Alt, перемещают фокус на нужный элемент управления. Следующая процедура описывает, как с помощью элемента управления Label указывается клавиша доступа для другого элемента управления.

***Указание клавиши доступа***

1. Перетащите элемент управления Label из Toolbox в форму рядом с элементом управления, для которого вы хотите указать клавишу доступа (например, элемент управления TextBox).

2. В окне Properties задайте свойству Text описательное имя элемента управления. Пос-тавьте перед буквой, которую вы хотите использовать в качестве клавиши доступа, знак амперсанда (&). Например, чтобы использовать клавишу F в качестве клавиши доступа, необходимо свойству Text надписи задать значение & First Name.

3. В окне Properties задайте свойству UseMnemonic значение True (по умолчанию).

4. В окне Properties задайте свойству Tablndex значение, меньше того, что задано свойству Tablndex элемента управления, для которого вы указываете клавишу доступа. Проверьте, чтобы у двух элементов управления были разные значения свойства Tablndex.

**Элемент управления LinkLabel.**

Элемент управления LinkLabel позволяет создавать в форме гиперссылку, которая открывает веб-страницу или выполняет какое-либо другое действие, если ее щёлкнуть мышью. LinkLabel содержит различные свойства, позволяющие его настраивать.

***Основные свойства элемента управления LinkLabel***

|  |  |
| --- | --- |
| ActiveLinkColor | Задает цвет активных ссылок |
| LinkArea | Обозначает область элемента управления LinkLabel, которая функционирует как ссылка |
| LinkBehavior | Указывает поведение ссылки |
| LinkColor | Задает цвет ссылки |
| LinkVisited | Указывает, посещали ли данную ссылку |
| VisitedLinkColor | Задает цвет посещенным ссылкам |

**Указание цвета ссылки.**

Вы можете указать цвет ссылки, которую отображает элемент управления LinkLabel, установив его свойства. Свойство LinkColor указывает цвет ссылки до того, как ее щёлкнули мышью. Свойство ActiveLinkColor представляет цвет ссылки, когда ее щелкают мышью, а свойство VisitedLinkColor — цвет ссылки, которую уже посетили. Помните: для того чтобы ссылка была того же цвета, который указан в свойстве VisitedLinkColor, необходимо задать значение True свойству LinkVisited элемента управления LinkLabel. Все эти свойства можно задать во время разработки в окне Properties.

**Определение поведения ссылки.**

Базовое поведение ссылки определяется свойством LinkBehavior. У свойства LinkBe- havioreсть четыре возможных значения: SystemDefault, AlwaysUnderline, HoverUnderline и NeverUnderline. Поведение, указываемое каждым отдельным значением, говорит само за себя. Значение SystemDefault означает, что элемент управления LinkLabel покажет то же поведение ссылки по умолчанию, что указано системой. Значение AlwaysUnderline означает, что ссылка подчеркивается всегда. Аналогичным образом, если элементу управления задано значение NeverUnderline, ссылка никогда не будет подчеркиваться, а в случае значения HoverUnderline — ссылка будет подчеркиваться только при наведении на нее указателя мыши. Как правило, свойство LinkBehavior задается во время разработки в окне Properties.

**Открытие формы или веб-страницы с помощью элемента управления LinkLabel.**

Новая форма или веб-страница открывается с помощью обработчика события LinkLa- bel\_LinkClicked. Таким же образом вы можете задать значение True свойству LinkVisited, в результате чего цвет ссылки будет соответствовать цвету свойства VisitedLinkColor. Следующая процедура демонстрирует, как с помощью элемента управления LinkLabel можно открыть новую форму или веб-страницу.

***Открытие формы или веб-страницы с помощью элемента управления LinkLabel.***

1. Задайте свойству Text элемента управления LinkLabel значение, определяющее назначение ссылки (например, Shipping Form или Microsoft Web Site).

2. Для создания обработчика события LinkLabel\_LinkClicked дважды щелкните элемент управления LinkLabel в конструкторе.

3. Напишите соответствующий код, чтобы открыть новую веб-страницу или вывести на экран новую форму. Задайте свойству LinkVisitedэлемента управления LinkLabel значение True Пример приведен ниже. Он предполагает, что в проекте вы дали элементу управления LinkLabel имя LinkLabell, а форме — ShippingForm.

// Открытие новой формы

ShippingForm.Show();

// Открытие нового сайта в Internet Explorer

System.Diagnostics.Process.Start("www.microsoft.com");

// Установка свойства LinkVisited в True

linkLabell.LinkVisited = true;

**Создание и настройка элементов управления с поддержкой редактирования текста.**

**Элемент управления TextBox.**

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 TextBox предоставляет свойства, позволяющие конфигурировать его поведение.

|  |  |
| --- | --- |
| AutoCompleteCustomSource | Содержит коллекцию строк для автозаполнения, когда свойству AutoCompleteMode задано значение, отличное от None, а свойству AutoCompleteSource задано значение Custom |
| AutoCompleteMode | Задает элементу управления режим AutoComplete. Возможные значения: None, Append, Suggest или SuggestAppend |
| AutoCompleteSource | Задает источник данных для автозаполнения. Свойству можно задать множество источников системы или пользовательский источник, предоставляемый свойством AutoCompleteCustomSource |
| CharacterCasing | Указывает регистр символов в элементе управления TextBox. Возможные значения: Normal, Upper или Lower |
| Lines | Возвращает массив строк, представляющих отдельные линии поля. Это свойство, в частности, необходимо, когда свойству MultiLine задано значение True. Помните, что линия описана как строка, на конце которой расположен символ возврата каретки, и не имеет ничего общего с видимыми линиями пользовательского интерфейса, которые можно увидеть, когда свойству Wordwrap задано значение True |
| MaxLength | Указывает максимальное число символов, которое может быть введено в TextBox |
| MultiLine | Указывает, может TextBox содержать одну строку текста или несколько |
| PasswordChar | Указывает, что вместо фактического текста в TextBox должен отображаться знак пароля |
| ReadOnly | Указывает, можно ли редактировать текст |
| Text | Задает текст, который содержит TextBox |
| UseSystemPasswordChar | Указывает, нужно ли использовать пароль, заданный на уровне системы, вместо реально введеного в TextBox текста |
| Wordwrap | Указывает, переходят ли слова автоматически с одной строки на другую, когда свойству MultiLine задано значение True |
| ScrollBars | Обозначает полосы прокрутки, которые будут отображаться в TextBox, когда свойству MultiLine задано значение True |

Основная задача элемента управления TextBox состоит в том, чтобы предоставить контейнер для редактируемого текста. Пользователи могут вводить текст в поля или редактировать текстовые данные, отображаемые приложением. Текст, который содер-жится в свойстве TextBox, доступен при использовании свойства Text. Текст в элементе управления TextBox можно редактировать, если свойству Readonly задано значение по умолчанию False. Если свойству Readonly задано значение True, пользователь не может редактировать отображаемый текст.

**Создание многострочного элемента управления TextBox.**

Элементы управления TextBox являются однострочными по умолчанию, но вы можете создать многострочный TextBox, задав свойству MultiLine значение True. Это позволит изменить размеры элемента управления TextBox как в вертикальном направлении, так и в горизонтальном.

Когда свойству MultiLine задано значение True, важно обратить внимание на несколько других свойств. Свойство Lines предоставляет массив строк, который содержит отдельные строки элемента управления TextBox. Свойство ScrollBars указывает, будут ли отображены полосы прокрутки элемента управления TextBox и какие именно — горизонтальные, вертикальные или и те, и другие. Свойство Wordwrap указывает, будут ли слова автоматически переходить с одной строки на другую. Помните, если свойству Wordwrap задано значение True, полосы прокрутки отображаться не будут, даже если свойству ScrollBars задано значение Horizontal.

**Создание элемента управления TextBox для ввода пароля.**

С помощью свойства PasswordChar или UseSystemPasswordChar можно создать поле, принимающее текстовый ввод, но вместо реально введенного текста отображается маскирующий символ, представляя введенный пользователем текст невозможным для прочтения наблюдателю. Чаще всего такой метод используют при создании поля для ввода пароля. Если PasswordChar присвоен какой-либо символ (например, звездочка), то этот символ будет отображаться при каждом введении пользователем символа в поле. Обратите внимание, что реально введенные пользователем символы будут со¬храняться в свойстве Text — изменится только их отображение в пользовательском интерфейсе. Кроме того, свойству UseSystemPasswordChar можно задать значение True, при котором весь пользовательский ввод визуально представляется в виде сим¬вола-пароля, заданного на уровне системы. Если UseSystemPasswordChar равен True, a PasswordChar присвоен какой-либо символ, использоваться будет символ-пароль, заданный на уровне системы.

**Элемент управления MaskedTextBox**

Элемент управления MaskedTextBox — это видоизмененный элемент управления Text- Box, 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.

|  |  |
| --- | --- |
| AllowPromtAsInput | Указывает возможен ли ввод приглашающего символа |
| AsiiOnly | Указывает, только ли символы ASCII допускаются при вводе. Если свойству задано значение TRUE, допустим ввод лишь букв a-z, A-Z |
| BeepOnError | Указывает, посылает ли элемент управления MaskedTextBox системный звуковой сигнал при каждом неверном вводе символа |
| CutCopyMaskFormat | Определяет, учтены ли литеральные и приглашающие символы при вырезании или копировании текста |
| HidePromtOnLeave | Указывает, скрыты ли приглашающие символы при потере фокуса элементом управления MaskedTextBox |
| InsertKeyMode | Задает элементу управления MaskedTextBox режим вставки текста |
| Mask | Определяет маску ввода для элемента управления MaskedTextBox |
| RejectlnputOnFirstFailure | Задает значение, обозначающее, прекращается ли синтаксический анализ ввода пользователя после первого же неверного символа |
| PromptChar | Задает приглашающий символ |
| ResetOnPrompt | Указывает, как обрабатывать введенный символ, соответствующий приглашающему символу |
| ResetOnSpace | Определяет, как обрабатывать введенный символ пробела |
| SkipLiterals | Указывает, вводить повторно или пропускать литеральные символы в маске |
| TextMaskFormat | Указывает, включены ли литеральные и приглашающие символы в текст, возвращаемый свойством Text |

**Свойство Mask.**

Свойство Mask — самое важное свойство элемента управления MaskedTextBox. Оно позволяет определять строку, которая представляет собой обязательный формат строки ввода в MaskedTextBox. Класс MaskedTextProvider, связанный с элементом управления MaskedTextBox, предоставляет механизм синтаксического разбора, который анализирует формат Mask.

|  |  |
| --- | --- |
| Элемент маски | Описание |
| **0** | Представляет обязательную цифру от 0 до 9 |
| **9** | Представляет необязательную цифру от 0 до 9 |
| **#** | Представляет необязательную цифру от 0 до 9 или пробел. Плюс (+) и минус (-) также допускаются |
| **L** | Представляет обязательную букву верхнего или нижнего регистра (A-Z, a-z) |
| **?** | Представляет необязательную букву верхнего или нижнего регистра (A-Z, a-z) |
| **&** | Представляет обязательный символ. Если свойству AsciiOnly задано значение True, данный элемент ведет себя как элемент L |
| **C** | Представляет необязательный символ. Если свойству AsciiOnly задано значение True, данный элемент ведет себя как элемент [?] |
| **A,a** | Представляет необязательный буквенно-цифровой символ. Если свойству AsciiOnlyзадано значение True, элемент примет только буквы A-Z и a-z |
| **.** | Десятичный символ-заполнитель. Представляет десятичный символ. Реально используемый символ станет десятичным символом, заданным свойством FormatProvider элемента управления |
| **,** | Тысячный символ-заполнитель. Представляет разделитель тысяч. Реально используемый символ станет разделителем тысяч, заданным свойством FormatProxnder элемента управления |
| **:** | Представляет разделитель времени. Реально используемый символ станет разделителем времени, заданным свойством FormatProvider элемента управления |
| **/** | Представляет разделитель даты. Реально используемый символ станет разделителем даты, заданным свойством FormatProvider элемента управления |
| **$** | Представляет символ валюты. Реально используемый символ станет символом валюты- заданным свойством FormatProvider элемента управления |
| **<** | Переместить вниз. Переводит все последующие символы в нижний регистр |
| **>** | Переместить вверх. Переводит все последующие символы в верхний регистр. |
| **\** | Превращает символ маски в литеральный символ. С помощью двойного слеша (\\) литеральный символ можно превратить в символ маски |
| **|** | Блокирует предыдущие перемещения вверх и вниз |
| Все остальные символы | Все остальные символы остаются без изменений в элементе управления MaskedTextBox, и пользователь не может их переместить или удалить |

Создать маску для маскирующего поля можно посредством создания строки, состоящей из символов, описанных в табл. 2-8. Установка свойства Mask элемента управления MaskedEditBox ограничивает возможный ввод форматом, который определяется строкой маски.

|  |  |  |
| --- | --- | --- |
| **Строка маски** | **Вводимый текст** | **Отображаемый текст** |
| (999)-000-0000 | 1234567890 | (123)-456-7890 |
| 00/00/0000 | 07141969 | 07/14/1969 (помните, что реально отображаемый разделитель даты будет определен свойством FormatProvider элемента управления) |
| $99,999.00 | 1234567 | $12,345.00 (помните, что реальный валютный символ, тысячный и десятичный разделитель будут определены свойством FormatProvider элемента управления) |
| LL>L|LLL<LL | abcdABCD | abCdABcd |

**Настройка элемента управления MaskedTextBox для ввода пользователя.**

Кроме свойства Mask у элемента управления MaskedTextBox есть несколько свойств, влияющих на поведение элемента управления при вводе пользователя. Свойство Ascii- Only определяет, только ли символы ASCII допускаются при вводе. Если свойству задано значение True, допустим ввод лишь букв a-z и A -Z— все остальные вводы отклоняются. Вы можете настроить элемент управления так, чтобы пользователь уведомлялся об ошибке, задав свойству ВеерОnЕтгог значение True. Свойство SkipLiterals определяет: вводить повторно (если присвоено значение False) или пропускать литеральные символы в элементе управления MaskedTextBox (если присвоено значение True).

Свойство RejectlnputOnFirstFailure управляет обработкой текста, вставленного в элемент управления MaskedTextBox. Когда вставляется строка, не соответствующая формату Mask, элемент управления MaskedTextBox отклонит всю строку, если свойству RejectlnputOnFirstFailure задано значение True. В случае значения False элемент управления MaskedTextBox примет все соответствующие формату Mask символы.

Свойство Prompt задает символы, отображаемые элементами управления MaskedTextBox, когда нет ввода для указанной позиции. Значение по умолчанию свойства Prompt — знак подчеркивания (\_). Свойства AllowPromptAsInput и ResetOnPrompt управляют интерпретацией приглашающего символа при его вводе. Если свойству ResetOnPrompt задано значение True, приглашающие символы будут приняты, Mask будет задана позиция символа, а курсор переместится на следующую позицию. Если свойству ResetOnPrompt задано значение False, a AllowPromptAsInput присвоено значение True, тогда приглашающий символ будет рассматриваться как обычный ввод. Если обоим свойствам задано значение False, приглашающий символ будет отклонен. Свойство ResetOnSpace управляет обработкой пробелов точно так же, как свойство ResetOnPrompt управляет обработкой приглашающих символов.

**Управление текстом в элементе управления MaskedTextBox**

Текст, показанный в элементе управления MaskedTextBox, необязательно доступен пользователю при вырезании и вставке либо приложению, когда текстом управляют программно. Свойство CutCopyMaskFormat определяет, как обрабатывается текст в элементе управления MaskedTextBox, когда пользователь вырезает или копирует этот текст. По умолчанию ему задано значение IncludeLiterals, означающее, что при вырезании или копировании текста литеральные символы маски учитываются, а приглашающие — нет. Кроме того, этому свойству можно задать значение ExcludePromptAndLiterals — исключающее как литеральные символы, так и приглашающие; IncludePrompt — учитывающее приглашающие символы, но исключающее литеральные, а также IncludePromptAndLiterals — учитывающее и приглашающие, и литеральные символы. Свойство TextMaskFormat предоставляет те же значения и функционирует таким же образом по отношению к возвращаемому свойством Text тексту.

**Практикум.**

**Упражнение 1.**

**Создание диалоговой формы**

1. Перетащите элемент управления LinkLabel из Toolbox в форму.

2. В окне Properties задайте свойству Text значение Open Form.

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

4. В конструкторе перетащите два элемента управления Button в форму Form2. Задайте свойству Text этих кнопок значения Accept и Cancel.

5. Задайте свойству DialogResult кнопки AcceptButton значение ОК, а свойству Dia- logResult кнопки Cancel — значение Cancel.

6. Перетащите два элемента управления TextBox из Toolbox в форму.

7. Только для С#: Задайте свойству Modifiers каждого элемента управления TextBox значение Internal.

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

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

10. Задайте свойству UseMnemonic всех надписей значение True.

11. Установите свойство Tablndex в окне Properties, как показано ниже.

|  |  |
| --- | --- |
| **Элемент управления** | **Индекс закладки** |
| Label1 | 0 |
| TextBox1 | 1 |
| Label2 | 2 |
| TextBox2 | 3 |
| Button1 | 4 |
| Button2 | 5 |

12. Выберите в конструкторе закладку для формы Forml. Дважды щелкните элемент управления LinkLabel для создания обработчика события LinkLabel\_LinkClicked. Добавьте следующий код:

// C#

DialogResult aResult;

Form2 aForm = new Form2();

aResult = aForm.ShowDialog();

if (aResult == System.Windows.Forms.DialogResult.OK)

{

MessageBox.Show("Your name is" + aForm.textBoxl.Text +" "+ a Form.tеxtBox2.Text);

}

linkLabell.LinkVisited = true;

13. Для запуска приложения нажмите клавишу F5. Щелкните элемент управления LinkLabel, чтобы открыть форму. Проверьте клавиши доступа и кнопки Accept и Cancel.

**Упражнение 2.**

**Добавление элементов управления с поддержкой отображения текста.**

1. В Visual Studio загрузите выполненное вами на занятии 2 решение или готовое решение занятия 2, представленное на прилагаемом диске в папке кода.

2. В Solution Explorer дважды щелкните Form2, чтобы открыть конструктор для Form2.

3. Перетащите элемент управления TextBox из Toolbox в форму. Перетащите элемент управления Label в форму и разместите рядом с элементом управления TextBox.

4. Задайте свойству Text элемента управления Label значение Address.

5. Задайте свойству Multiline элемента управления TextBox значение True, свойству Wordwrap — значение False, а свойству ScrollBars — значение Both. Измените размеры элемента управления TextBox так, чтобы он вмещал адрес. Перетащите элементы управления MaskedTextBox и Label из Toolbox в форму.

6. Свойству Text элемента управления Label задайте значение Phone Number.

7. Задайте значение (999)-000-0000 свойству Mask элемента управления MaskedText-Box.

8. Только для С#. Задайте значение Internal свойству Modifiers элементов управления TextBox и MaskedTextBox.

9. В Solution Explorer щелкните правой кнопкой мыши Forml и выберите View Code.

10. В обработчике события LinkLabell\_LinkClicked добавьте в блок If, расположенный под кодом, который вы добавили на занятии 2, следующий код:

// C#

MessageBox.Show("Your address is ” + aForm.textBox3.Text);

MessageBox.Show("Your phone number is " + aForm.maskedTextBoxI.Text);

11. Запустите приложение, нажав клавишу F5, и протестируйте его.

**Упражнение 3.**

**Обработка событий Click и MouseMove**

Элементы управления - это компоненты, объединяющие графический интерфейс с предварительно разработанной функциональностью. Элементы управления представляют собой многократно используемые блоки кола, предназначенные для выполнения определенных задач. Все элементы управления являются производными базового класса Control, а значит, тоже используют различные свойства, задающие размер, расположение п 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 Hutton и надпись Label, разместите их по-своему усмотрению.

3. Установите следующие свойства элементов управления и формы:

|  |  |  |
| --- | --- | --- |
| Объект | Свойство | Значение |
| Form 1 | FormBorderStyle | Fixed3D |
| Size | 350: 200 |
| Text | Насущный вопрос |
| label 1 | Text | Вы довольны своей зарплатой? |
| Button 1 | Name | btnyes |
| Text | Да |
| Button2 | Name | btnno |
| 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:

btnno.Top -= е.Y; bt.nno.Left += е.Х;

if (btnno.Тор < -10 || btnno.Тор > 100) btnno.Тор = 60;

if (btnno.Left < -80 || btnno.Left > 250) btnno.Left = 120;

7. Запустите приложение и нажмите на каждую из кнопок.

**Упражнение 4.**

**Работа со списками.**

Основными элементами управления списком яв.ляются List Box, ComboBox п 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 простои 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.

|  |  |  |
| --- | --- | --- |
| 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 people List 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 ОК.

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.Checkedlndices.Count > 0) memberList.Items.RemoveAt(memberList.Checkedlndices[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 5.**

**Создание и использование элемента управления Tool Strip**

***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 toolStripButtonI. 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 изображении можно найти в графических файлах в папке \Microsoft Visual Studio 9.0\Common7\V S2008ImageLibrary\.

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 New7, 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 С н Т. 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 toolStripl, щелкнув два раза указателем мыши по имени события Click на закладке событий в окне свойств. В программу добавится функция toolStripMtemClicked как обработчик события, происходящего при нажатии кнопки на панели инструментов.

18. В окне Properties для toolStripButtonl в свойстве Tag запишите New Doc. Аналогично укажите для toolStripButton2 и toolStripButtonS 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 toolStripMtemClicked добавьте код, который будет реализовывать различные действия в зависимости от нажимаемой кнопки:

switch(е.Clickedltem.Tag.ToString())

{

case "NewDoc":

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

newChild.Show();

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

break;

case "Cascade": this.LayoutMdiSystem.Windows.Forms.MdiLayout.Cascade); break;

case "Title":this.LayoutMdi(System.Windows.Forms.

MdiLayout.TileHorizontal);

break;

}

21. Откомпилируйте и запустите приложение. Проверьте работоспособность кнопок.

**Упражнение 6.**

**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 toolStriрStatusLabel1 - первая панель для отображения.

6. Создайте еще одну панель аналогичным способом – toolStripStatusLabel1 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

а. Для каскадной ориентации:

spWin.Text="Windows is cascade";

б. 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.ToLongDateString());

}

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

**Упражнение 7.**

**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.

***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 переключате.ля (RadioButton):

if (radioButtonl.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 FlowLavoutPanel.***

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

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 Flow Direct ion, которому может быть задано четыре значения заливки в FlowLayoutPanel: LeftToRight — по умолчанию, RightToLeft — справа налево, TopDown — сверху вниз и Bottom Up — снизу-вверх.

13. Дважды щелкните кнопку button? и добавьте в обработчик события button5\_Click следующий код:

flowLayoutPanell.SetFlowBreak(button6, true);

***Настройка элемента TableLavoutPanel***

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{);

tableLayoutPanell.Controls.Add(aButton,1,1);

***Настройка элемента SplitContainer***

17. В конструкторе выберите вкладку SplitContainer. Перетащите элемент управления SplitContainer из Toolbox в Tab Page. Задайте свойству BorderStvle значение Fixed3D.

18. Перетащите два элемента управления Button из Toolbox в Panel1 элемента управления SplitContainer. Задайте свойствам Text этих кнопок значения Fix/Unfix Panel1 и Fix/Unfix Splitter. Измените размеры кнопок так, чтобы отображался текст.

19. Добавьте кнопку в Рапе 12 и задайте свойству Text значение Collapse/Uncollapse Panel]. Измените размеры кнопок так, чтобы отображался текст.

20. Дважды щелкните кнопку Fix/UnfixPanel1 и добавьте в обработчик события Click следующий код:

if (SplitContainer1.FixedPanel == FixedPanel.Panell) SplitContainer1.FixedPanel = FixedPanel.None;

else

SplitContainer1.FixedPanel = FixedPanel.Panell;

21. Дважды щелкните кнопку Fix/UnfixSplitter и добавьте в обработчик события Click следующий код:

SplitContainer1.PanellCollapsed = !(SplitContainer1.PanellCollapsed);

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 buttonS и проверьте, прервалась ли последовательность на элементе управления buttonb (это было реализовано вызовом метода SetFloxvBreak).

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 8.**

**Элементы с поддержкой отображения текста**

***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 ОК, 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 Tablndex в окне Properties, как показано ниже.

|  |  |
| --- | --- |
| Элемент управления | Индекс закладки |
| label 1 | 0 |
| textBox 1 | 1 |
| label2 | 2 |
| textBox2 | 3 |
| button 1 | 4 |
| button2 | 5 |

***Реализация обработчика события вызова диалогового окна***

13. Выберите в конструкторе закладку для формы Forml. Дважды щелкните первый элемент управления linkLabell для создания обработчика события linkLabell\_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.textBoxl.Text + "

" + aForm.textBox2.Text);

}

linkLabell.LinkVisited = true;

***Реализация обработчика события вызова веб-страницы***

14. Выберите в конструкторе закладку для формы Forml. Дважды щелкните второй элемент управления HnkLabel2 для создания обработчика события linkLabel2\_LinkClicked. Добавьте следующий код:

System.Diagnostics.Process.Start("https://www.youtube.com/watch?v=gZEipqcBI00");

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 linkLabe2 - Microsoft, чтобы открыть сайт известного учебного центра.

**Упражнение 9.**

**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) п 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.textВохЗ.Text);

MessageBox.Show{"Your phone number is " +

aForm.maskedTextBoxl.Text);

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

**Упражнение 10.**

**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 ControlsColleetion, 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 |
| Form 1 | FormBorderStyle | Fixed3D |
| Text | Регистрация |
| Size | 400;310 |
| label 1 | Location | 30; 10 |
| Text | Выберите тип регистрации |
| label2 | Location | 16; 32 |
| Text | Name |
| label3 | Location | 16; 64 |
| Text | PIN |
| button 1 | Location | 80; 248 |
| Text | Регистрация |
| textBoxl | Location | 96; 32 |
| Text |  |
| Size | 184; 20 |
| textBox2 | Location | 96; 64 |
| Size | 184; 20 |
| Text |  |
| checkbox 1 | Location | 40; 40 |
| Size | 232; 24 |
| Text | Расширенные возможности |
| groupВox 1 | 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 (checkBoxl.Checked == true)

{

Label lbl = new Label ();

lbl.Location = new System.Drawing.Point(16, 96);

lbl.Size = new System.Drawing.Size(32, 23);

lbl.Name = "labelll";

lbl.Tablndex = 2;

lbl.Text = "PIN2";

groupBoxl.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.Tablndex = 1; txt.Text =

groupBoxl.Controls.Add(txt);

}

else {

}

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

6. Для удаления ЗУ 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 lev;

lev = groupBoxl.Controls.Count;// определяется количество while (lev > 4)

{

groupBoxl.Controls.RemoveAt(lev - 1);

lev -= 1;

}

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

**Упражнение 11.**

**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 TextBoxl 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 TextBoxl укажите следующий код (для элемента TextBoxl недопустимыми значениями будут цифры):

if (char.IsDigit(е.KeyChar) )

{

е.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 по.ля TextBox2 укажите код:

if (!char.IsDigit(е.KeyChar) )

{ " е.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 д.ля элемента CheckBoxl укажите код:

txt.KeyPress += new System.Windows.Forms.KeyPressEventHandler (this.textBox2\_KeyPress);

7. Запустите п 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 =="")

{

е . 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 textBox l\_KeyPress добавьте следующую строку:

errorProvider1.SetError(textBoxl, "Must be letter");

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