Лабораторная работа № 7

Введение в разработку форм

Цель лабораторной работы: изучение методов построения форм Windows и получение навыков по настройке форм, созданию непрямоугольных и наследуемых (производных) форм.

Формы Windows — это основной компонент пользовательского интерфейса. Формы предоставляют контейнер, который содержит элементы управления, меню и позволяет отображать приложение в 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 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. Форму 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
	SizableToolWindow - 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**
MaximumSize	Устанавливает максимальный размер формы. Если задать этому свойству размер 0; 0, у формы не будет верхнего ограничения размера
MinimizeBox	Указывает, есть ли у формы кнопка **MinimizeBox**
MinimumSize	Устанавливает минимальный размер формы, который пользователь может задать
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
Size	Принимает и устанавливает исходный размер формы

StartPosition	Указывает положение формы в момент ее первого выведения 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 **Buttonl 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.

Создание непрямоугольной формы 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.

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:
   a. для кнопки:
      i. свойство **Text** – Hello!!!
      ii. свойство **BackColor** – Brown
   b. 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.

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	ParenForm

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, появившейся 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, появившейся 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 **Format**.
    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;
}
```

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 сливается 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. Реализуйте эту возможность автонумерации заголовков:
    a. Откройте код родительской формы и в классе **ParentForm** объявите 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 **CLOSE**.