Лабораторная работа №2: Создание пользовательских элементов управления в Windows Forms

Цель работы:

- изучить приемы создания пользовательских элементов управления с использованием технологий Windows Forms;
- познакомиться с примерами использования наследования, полиморфизма и композиции объектов на практике.

Задание:

Разработать приложение в Windows Forms с пользовательскими элементами управления:

Галерея (компонент для просмотра изображений) на основе класса `HoverButton`.

a) галерея с hover-эффектом. При наведении курсора на изображение появляется её подпись и другие декоративные элементы. По клику на изображение загружается следующая картинка.

b) аналогичная галерея круглой формы (должна быть наследником класса из пункта a).

Галерея должна быть универсальной и совместимой с любым набором изображений.

Дополнительное задание в зависимости от <u>последней цифры пароля</u>:

Вариант 1: К реализованному hover-эффекту добавить дополнительно увеличение размера галереи при наведении курсора и возвращение исходного размера, когда курсор покидает область галереи.

Вариант 2: Реализовать возможность не по клику, а по двойному клику на изображение загружать следующую картинку.

Вариант выбирается исходя из последней цифры пароля от личного кабинета по следующей таблице:

	Последняя цифра пароля									
	0	1	2	3	4	5	6	7	8	9
Номер варианта	1	2	1	2	1	2	1	2	1	2

Результат выполнения работы предоставить в виде:

- архив с проектом (если размер архива больше 2 Мбайт, то рекомендуется загрузить проект на https://github.com/ или на другое общедоступное хранилище и предоставить ссылку);
- отчет по лабораторной работе в формате Microsoft Word, который содержит следующие разделы:

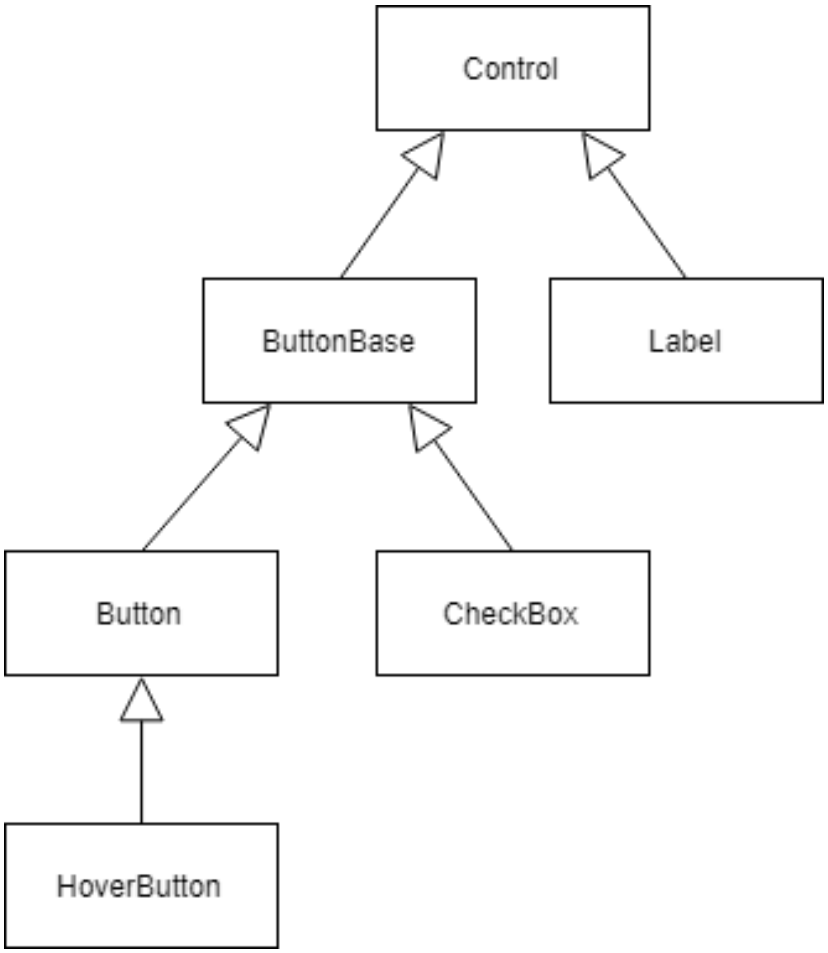
1. титульный лист;
2. задание на лабораторную работу;
3. краткое описание разработанных программ и используемых алгоритмов со скриншотами выполнения;
4. вывод с результатами работы.

Копирование исходного кода программ не допускается. Проекты с одинаковым исходным кодом засчитываться не будут.

Краткие теоретические сведения:

Создание кнопки с анимацией (hover-эффект) в Windows Forms

Иерархия элементов управления в Windows Forms представлена следующей диаграммой (данная диаграмма является неполной, на ней представлены лишь некоторые элементы управления, тем не менее, она иллюстрирует основные взаимосвязи между классами).

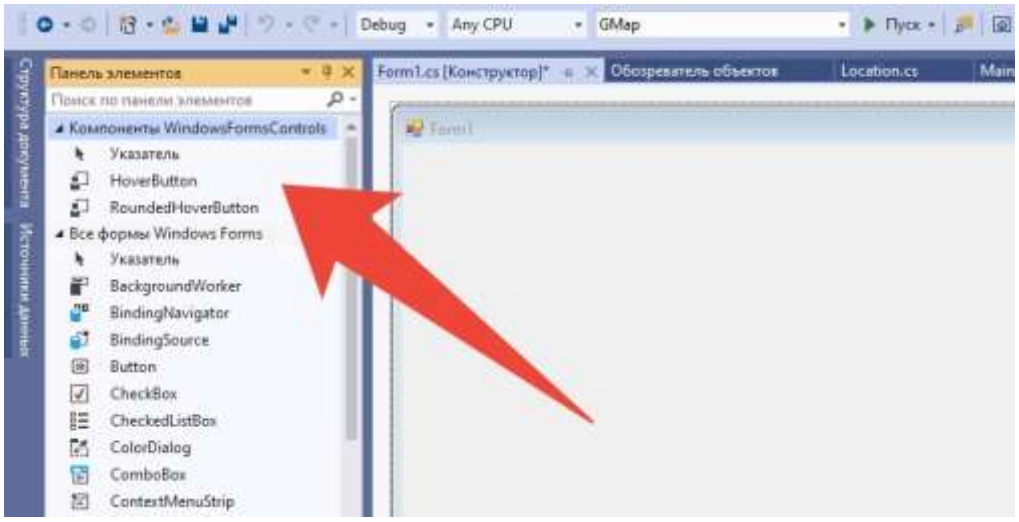


Все элементы управления являются наследниками базового класса `Control`, который реализует общую логику всех элементов управления (отображение, взаимодействие с пользователем и др.).
Подклассы `Button`, `CheckBox` и `Label` представляют собой конкретные элементы управления – кнопка, флажок, заголовок. Промежуточный класс `ButtonBase` реализует общую логику для схожих компонентов `Button` и `CheckBox`.

Как правило, при создании пользовательского элемента управления расширяют один из уже существующих компонентов. Создайте проект «Приложение Windows Forms (.NET Framework)». Добавьте новый класс, расширяющий функционал стандартного класса `Button`:

```
class HoverButton : Button {


}
```

После сборки проекта (Сборка – Собрать решение или F6) компонент будет доступен в панели элементов. Расположите компонент на окне с помощью визуального редактора:



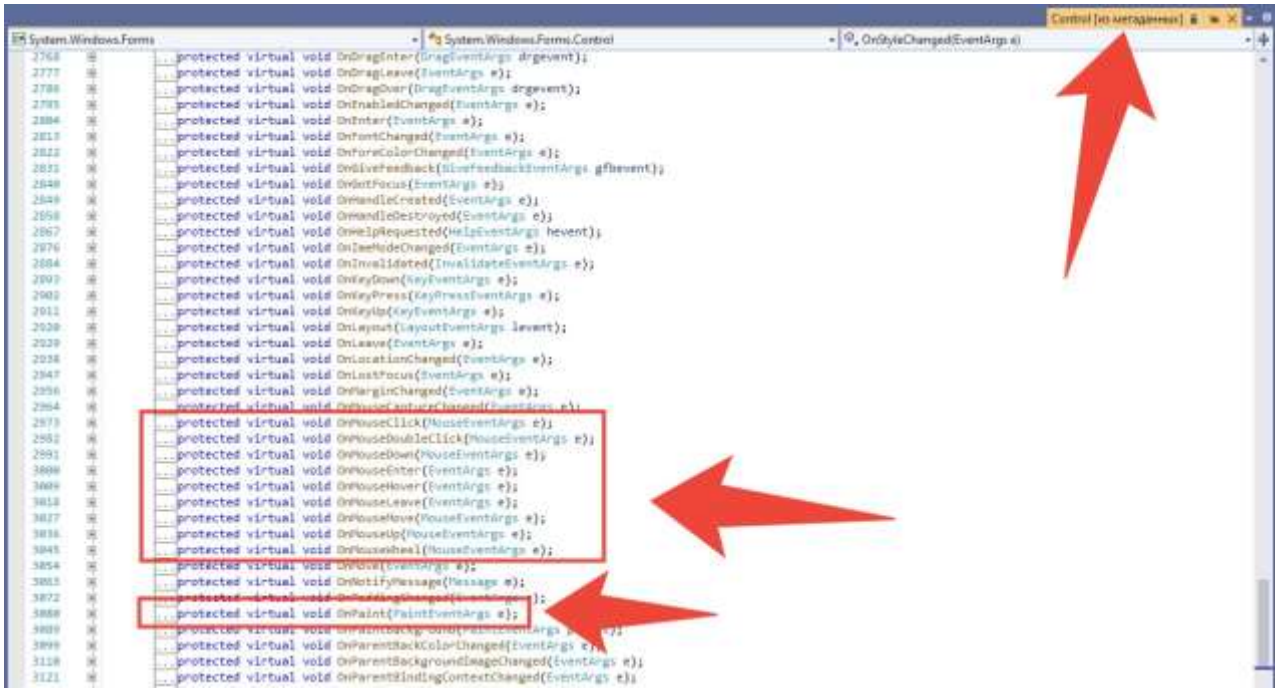
В конструкторе можно задать свойства по умолчанию, например, цвет текста и шрифт.

```
public HoverButton() : base()
{
       ForeColor = Color.White;
       Font = new Font("Microsoft YaHei UI",
                       20.25F,
                       FontStyle.Bold,
                       GraphicsUnit.Point,
                       0);
}
```

Hover-эффект представляет собой реакцию на наведение курсора мыши в область кнопки. Таким образом, необходимо изменить поведение кнопки при наведении на нее курсора.

Для решения подобного рода задач хорошо подходит **полиморфизм.** Одной из разновидностей полиморфизма является переопределение виртуальных методов.

В базовом классе Control определены виртуальные методы для управления компонентом, в том числе для управления мышью и для отрисовки:



Переопределим метод `OnPaint`, отвечающий за отрисовку. Для этого добавим в класс `HoverButton` метод `OnPaint` с такой же сигнатурой, но с модификатором `override`:

```
protected override void OnPaint(PaintEventArgs pe)
{
    base.OnPaint(pe);
}
```

Сейчас метод вызывает только базовую реализацию. Для отрисовки кнопки определенного цвета необходимо добавить следующий код:

```
private Color color = Color.SkyBlue;
...
protected override void OnPaint(PaintEventArgs pe)
{
    base.OnPaint(pe);
    // отрисовка прямоугольника
    pe.Graphics.FillRectangle(new SolidBrush(color), ClientRectangle);
    // отрисовка текста
    pe.Graphics.DrawString(Text, Font, new SolidBrush(ForeColor), ClientRectangle);
}
```

Таким образом, можно управлять цветом кнопки, меняя значение поля `color`.

Один из вариантов hover-эффекта – изменение цвета кнопки при наведении мыши. В базовом классе `Control` есть методы, определяющие нахождение курсора мыши в области элемента управления: `OnMouseEnter` (курсор наведен на элемент управления) и `OnMouseLeave` (курсор перемещен в другую область). При наведении курсора на кнопку сделаем её более темной, а при перемещении курсора в другую область вернем исходный цвет:

```
protected override void OnMouseEnter(EventArgs e)
{       base.OnMouseEnter(e);
        color = Color.Blue;
}
```

```
protected override void OnMouseLeave(EventArgs e)
{     base.OnMouseLeave(e);
      color = Color.SkyBlue;
}
```

Во время выполнения это будет выглядеть так:

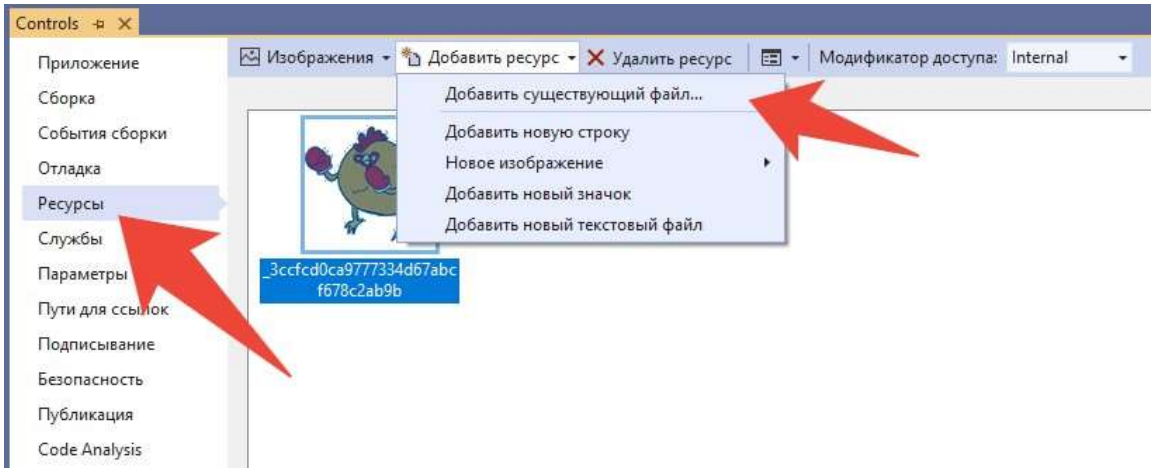
Для достижения других эффектов можно аналогичным образом дополнить реализацию других методов, например `OnMouseDown` (переключение кнопки вниз), `OnMouseUp` (переключение кнопки вверх), `OnMouseWheel` (прокрутка колесика). Для изменения формы кнопки можно переопределить метод `OnResize`, отвечающий за расчёт границ элемента управления. Например, можно сделать кнопку овальной:

```
protected override void OnResize(EventArgs e)
{
    base.OnResize(e);
    GraphicsPath graphicsPath = new GraphicsPath();
    graphicsPath.AddEllipse(new Rectangle(0, 0, Width - 1, Height - 1));

    Region = new Region(graphicsPath);

}
```

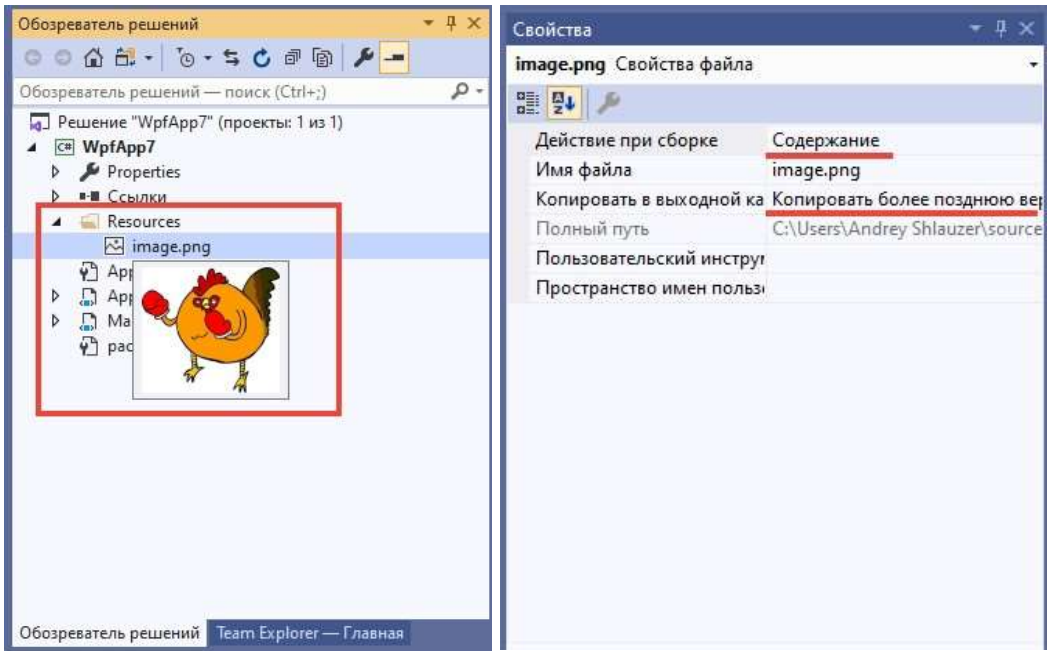
Во время выполнения это будет выглядеть так:

Работа с изображениями:

В приложениях Windows Forms и WPF с изображением удобно работать как с ресурсом. Чтобы загрузить изображение, перейдите в свойства проекта (вкладка «Ресурсы») и добавьте файл с изображением в ресурсы.



После добавления изображения в ресурсы оно появится в обозревателе решений. Выделите его и установите в панели свойств такие же значения атрибутов, как на скриншоте (это нужно для того, чтобы изображение копировалось в выходную директорию с .exe-файлом в процессе сборки).



Затем можно обращаться к изображению из кода. В стандартном классе `Resources` будет сгенерировано одноименное поле для получения картинки. Пример получения и отрисовки картинки в Windows Forms может выглядеть так:

```
protected override void OnPaint(PaintEventArgs pe)
{
    base.OnPaint(pe);
    // получение картинки из ресурсов
    Bitmap bitmap = new Bitmap(Resources.image);
    // отрисовка картинки в точке (0,0)
    pe.Graphics.DrawImage(bitmap, 0, 0);
}
```

Если поместить этот код в классе `HoverButton`, то картинка будет установлена в качестве фона.

Список литературы:

1) Герберт Шилдт "C# 4.0: полное руководство"
2) Эндрю Троелсен "Язык программирования C# 5.0 и платформа .NET 4.5"
3) Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7 https://metanit.com/sharp/tutorial/
4) Руководство по WPF https://metanit.com/sharp/wpf/
5) C# 5.0 и платформа .NET 4.5 http://professorweb.ru/my/csharp/charp_theory/level1/infocsharp.php
6) https://github.com/Microsoft/WPF-Samples