ПРАКТИЧЕСКАЯ РАБОТА №1

Данная практическая работа является ознакомительной, внимательно читайте и постарайтесь **САМОСТОЯТЕЛЬНО** выполнить данную работу. Ваша задача самим научиться и понять!

Из этого краткого руководства вы узнаете, как создать новое приложение Windows Forms с помощью Visual Studio. После создания первоначального приложения вы научитесь добавлять элементы управления и обрабатывать события. По завершении работы с этим руководством у вас будет простое приложение, добавляющее имена в список.

В этом руководстве описано следующее:

- Создание приложения Windows Forms;
- Добавление элементов управления на форму;
- Обработка событий элемента управления для предоставления функциональных возможностей приложения;
- Запустите СВОЕ первое приложение.

Предварительные требования:

- Visual Studio 2019 версии 16.8 или более поздней;
- Выберите рабочую нагрузку "**Рабочий стол Visual Studio**"; (смотрите дальше по заданию)
- Выберите **Отдельные компоненты .NET 5**; (смотрите дальше по заданию)

Немного теории

Большинство форм создается путем добавления элементов управления на поверхность формы для определения пользовательского интерфейса. Элемент управления — это компонент в форме, используемый для вывода информации или ввода данных пользователем.

Основной способ добавления элемента управления в форму — с помощью конструктора Visual Studio, но вы также можете управлять элементами управления в форме во время выполнения с помощью кода.

Добавление с помощью конструктора

Для создания форм в Visual Studio используется конструктор форм. На панели "Элементы управления" представлен список всех элементов управления, доступных для приложения. Элементы управления можно добавлять с панели двумя способами:

1. Добавление элемента управления двойным щелчком ***Краткое видео (кликни)***

2. Добавление элемента управления путем рисования ***Краткое видео (кликни)***

Добавление с помощью кода

Элементы управления можно создавать и добавлять в форму во время выполнения с помощью коллекции **Controls** формы. Эту коллекцию можно также использовать для удаления элементов управления из формы.

```
Label label1 = new Label()
{
    Text = "&First Name",
    Location = new Point(10, 10),
    TabIndex = 10
};

TextBox field1 = new TextBox()
{
    Location = new Point(label1.Location.X, label1.Bounds.Bottom + Padding.Top),
    TabIndex = 11
};

Controls.Add(label1);
Controls.Add(field1);
```

Рисунок 1. Добавление и помощью кода (**попробуйте перевести код, чтобы понять, что он делает**)

Upd. Код выше Следующий код добавляет и размещает два элемента управления, Label и TextBox

Создание приложения Windows Forms

Первым шагом в создании нового приложения является запуск Visual Studio и создание приложения на основе шаблона.

1. Запустите Visual Studio.
2. Выберите Создать новый проект. Создать карусель Добавьте описание

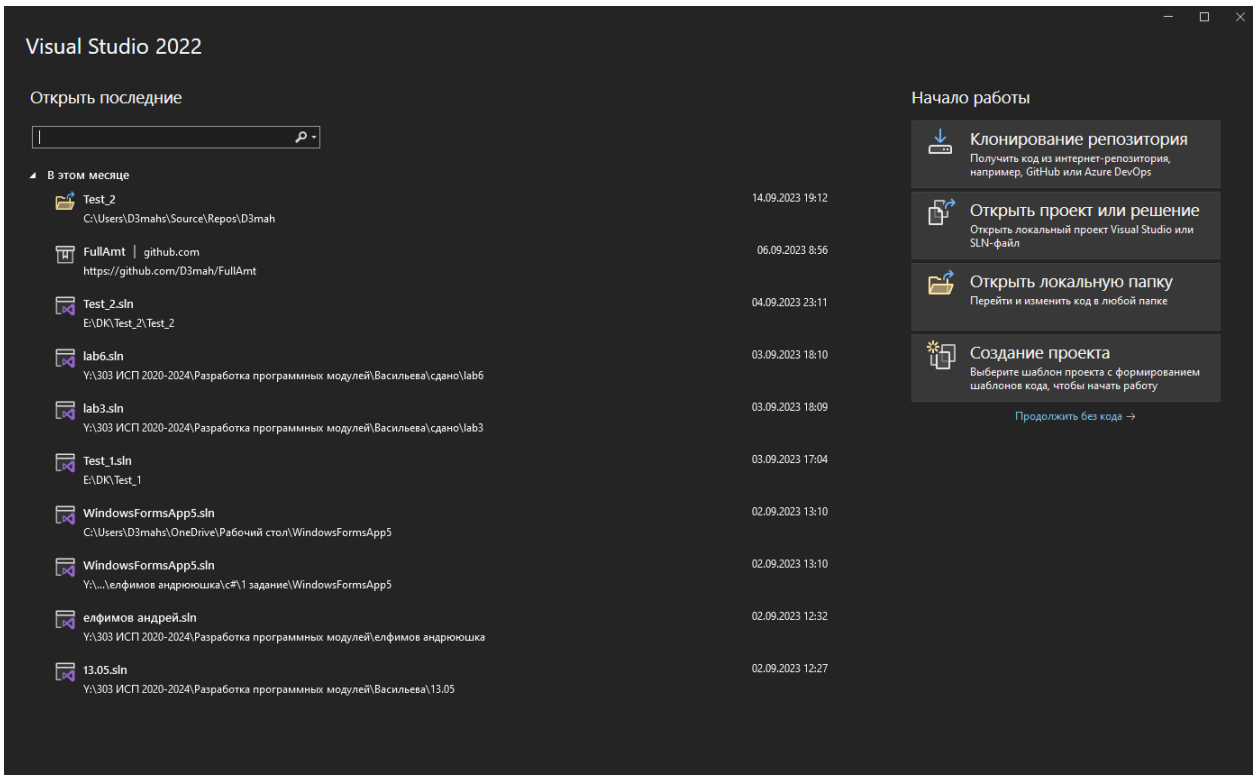

Рисунок 2. Создание проекта

3. В поле Поиск шаблонов введите **winforms** и нажмите клавишу ВВОД.
4. В раскрывающемся списке язык кода выберите **C#** или **Visual Basic**.
5. В списке шаблонов выберите Приложение **Windows Forms (.NET)** и затем щелкните Далее.

Рисунок 3. Создание приложения

6. В окне **Настроить** новый проект задайте в качестве имени проекта значение Names и щелкните **Создать**.

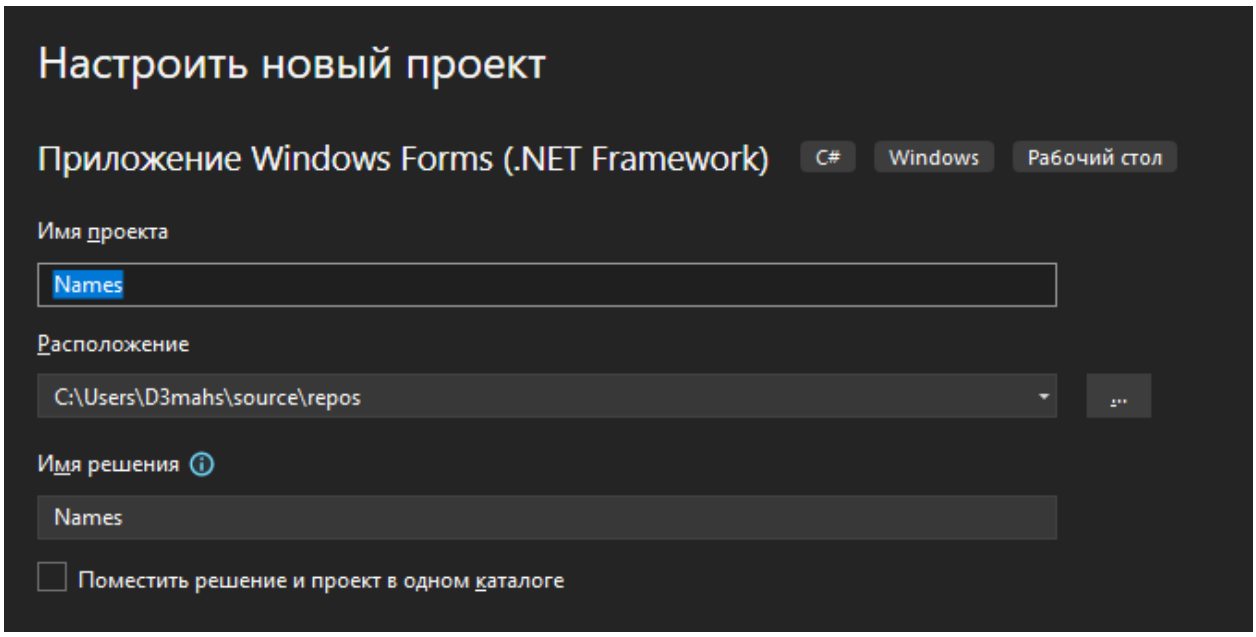

Рисунок 4. Приложение Names

После создания приложения Visual Studio должен открыть панель конструктора для формы по умолчанию Form1. Если конструктор форм не отображается, дважды щелкните форму в области Обозреватель решений, чтобы открыть окно конструктора.

Важные элементы среды Visual Studio

Поддержка Windows Forms в Visual Studio состоит из четырех важных компонентов, с которыми вы будете взаимодействовать при создании приложения.

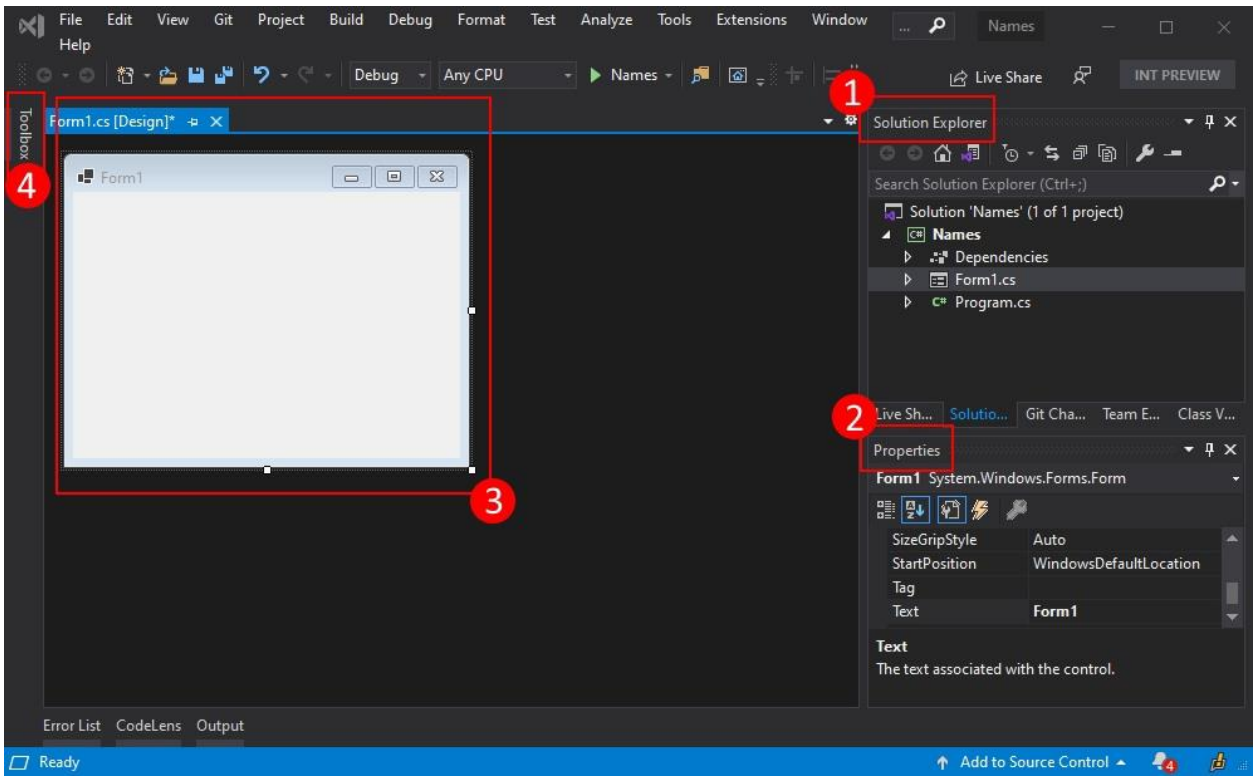

Рисунок 5. Важные элементы среды VS

Обозреватель решений - Все файлы проекта, код, формы и ресурсы отображаются в этой области.

Properties (Свойства) - На этой панели отображаются параметры свойств, которые можно настроить в зависимости от выбранного элемента. Например, если выбрать элемент в Обозревателе решений, отобразятся параметры свойств, связанные с файлом. Если выбрать объект в конструкторе, отобразятся параметры элемента управления или формы.

Конструктор форм - Это конструктор для формы. Он является интерактивным, и на него можно перетаскивать объекты из панели элементов. Выбирая и перемещая элементы в конструкторе, можно визуально создавать пользовательский интерфейс для приложения.

Панель элементов - Панель элементов содержит все элементы управления, которые можно добавить на форму. Чтобы добавить элемент управления на текущую форму, дважды щелкните элемент управления или перетащите его.

UPD. ВАЖНО, при первом открытии приложения, у вас может быть скрыты свойства, обозреватель решений, панель элементов, для того чтобы их отобразить в вашем проекте необходимо выполнить следующие шаги:

Свойства – Вид – Окна свойств (или нажать F4);

Панель элементов – Вид – Панель элементов (или ctrl+alt+x)

И так далее, «покапатайтесь» во-вкладке «Вид» и настройте рабочую область, так как нужно **ВАМ.**

Добавление элементов управления на форму

Открыв конструктор форм Form1, используйте панель Область элементов, чтобы добавить на форму следующие элементы управления:

1. Метка (*попадос, там все названия элементов на английском, переводим и используем, тут подскажу* ***label***)
2. Кнопка (*как на английском кнопка, вспоминаем*)
3. Listbox (*повезло так и подписано*)
4. Текстовое поле (*сами*)

Вы можете расположить и изменить размер элементов управления в соответствии со следующими настройками. Либо визуально перенесите их, чтобы они соответствовали следующему снимку экрана, либо щелкните каждый элемент управления и настройте параметры в области Свойства. Можно также щелкнуть область заголовка формы, чтобы выбрать форму.

ВАЖНО

ВСЕ СЛЕДУЮЩИЕ СКРИНЫ БУДУТ «***ONLY ENGLISH***»
(будем учить)

Объект	Параметр	Значение
Form	Текст	Names
	Размер	268, 180
Label	Расположение	12, 9
	Текст	Names
Listbox	Имя	lstNames
	Расположение	12, 27
	Размер	120, 94
текстовое поле;	Имя	txtName
	Расположение	138, 26
	Размер	100, 23
Button	Имя	btnAdd
	Расположение	138, 55
	Размер	100, 23
	Текст	Add Name

Рисунок 6. Настройка элементов на форме

Upd. Используйте свойства для задания размеров, расположения и текста

Вы должны получить в конструкторе форму, которая выглядит следующим образом.

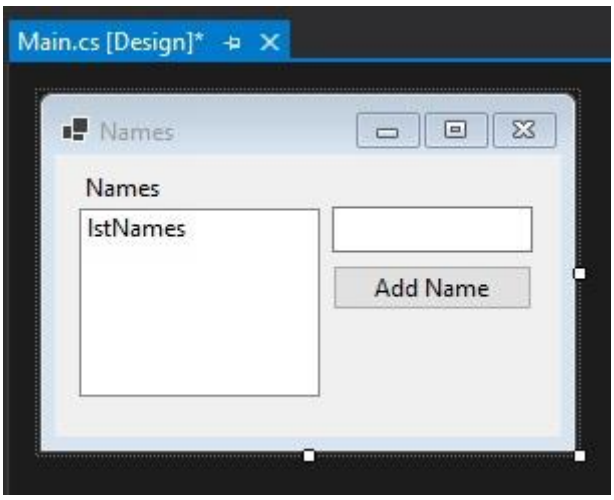

Рисунок 7. Готовая форма

Обработка событий (если понятным языком, то обработка вашей программой действий пользователя)

Теперь, когда в форме есть все элементы управления, необходимо обрабатывать события элементов управления, чтобы реагировать на вводимые пользователем данные. Открыв конструктор форм, выполните следующие действия.

1. Выберите в форме элемент управления "**Кнопка**".
2. В области Свойства щелкните значок события, чтобы вывести список событий кнопки. Добавьте описание
3. Найдите событие Click и дважды щелкните его, чтобы создать обработчик событий. Это действие добавляет следующий код в форму:

```
private void btnAdd_Click(object sender, EventArgs e)
{

}
```

Рисунок 8. Событие клик

Код, помещаемый в этот обработчик, будет добавлять имя, заданное элементом управления ***TextBox txtName***, в элемент управления ***ListBox lstNames***. Однако мы хотим, чтобы имя удовлетворяло двум условиям: указанное имя не должно быть пустым, и его еще не должно быть в списке.

В следующем примере кода показано добавление имени в элемент управления ***lstNames***.

```
private void btnAdd_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(txtName.Text) && !lstNames.Items.Contains(txtName.Text))
        lstNames.Items.Add(txtName.Text);
}
```

Рисунок 9. Код на кнопку (переведите и вам станет легче)

Запустите приложение

Теперь, когда у нас есть код события, можно запустить приложение, нажав клавишу F5 или выбрав пункт меню Отладка > Начать отладку. Отобразится форма, и вы можете ввести имя в текстовое поле, а затем добавить его, нажав кнопку.

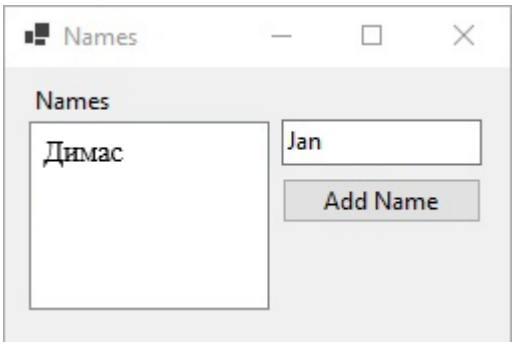

Рисунок 10. Работа программы

Как обрабатывать сообщения ввода с клавиатуры в форме (Windows Forms .NET)

Windows Forms предоставляет возможность обработки сообщений клавиатуры на уровне формы, прежде чем они достигнут элемента управления.

Обработка сообщения клавиатуры

Обработайте событие **KeyPress** или **KeyDown** активной формы и задайте для свойства формы **KeyPreview** значение **true**. Это приведет к тому, что форма будет получать события клавиатуры до того, как они достигнут каких-либо элементов управления на форме. В следующем примере кода обрабатывается событие **KeyPress** посредством обнаружения всех цифровых клавиш и использования 1, 4 и 7.

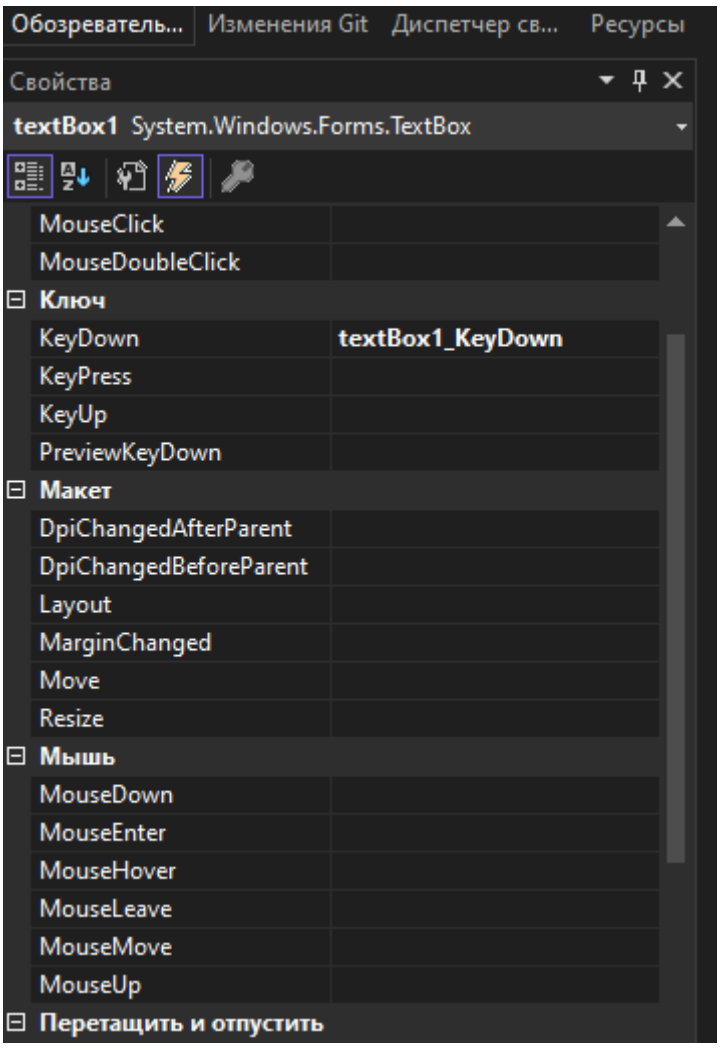

Рисунок 11. Где находится keypress,keyDown, keyPrewiew

Следующий код (стараемся перевести и понять, что делаем)

```
void Form1_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar >= 48 && e.KeyChar <= 57)
    {
        MessageBox.Show($"Form.KeyPress: '{e.KeyChar}' pressed.");

        switch (e.KeyChar)
        {
            case (char)49:
            case (char)52:
            case (char)55:
                MessageBox.Show($"Form.KeyPress: '{e.KeyChar}' consumed.");
                e.Handled = true;
                break;
        }
    }
}
```

Рисунок 12. Код Формы KeyPress

Что делает написанный вами код? Подумайте исходя из кода

Upd. данный код позволяет отловить нажатие клавиш 1,4 и 7 и вывести диалоговое сообщение о данном событии.

Управление указателями мыши (Windows Forms .NET)

Указатель мыши, который иногда называют курсором, является точечным рисунком, указывающим на экране точку фокуса для ввода данных пользователем с помощью мыши. В этом разделе приводится обзор указателя мыши в **Windows Forms** и описываются некоторые способы его изменения и управления им.

Указатель мыши представлен классом **Cursor**, и каждый элемент **Control** имеет свойство **Control.Cursor**, которое задает для него указатель. Класс **Cursor** содержит свойства, описывающие указатель, например свойства **Position** и **HotSpot**, а также методы, которые могут изменять внешний вид указателя, например методы **Show**, **Hide** и **DrawStretched**.

В следующем примере курсор скрывается при наведении курсора на кнопку:

```
private void button1_MouseEnter(object sender, EventArgs e) =>
    Cursor.Hide();

private void button1_MouseLeave(object sender, EventArgs e) =>
    Cursor.Show();
```

Рисунок 12. Указатель мыши

Управление указателем мыши

Иногда может потребоваться ограничить область, в которой можно использовать указатель мыши, или изменить расположение мыши. Можно получить или задать текущее расположение мыши с помощью свойства **Position** объекта **Cursor**. Кроме того, можно ограничить область, в которой можно использовать указатель мыши, задав свойство **Clip**. По умолчанию областью действия является весь экран.

В следующем примере указатель мыши помещается между двумя кнопками при их нажатии:

```
private void button1_Click(object sender, EventArgs e) =>
    Cursor.Position = PointToScreen(button2.Location);

private void button2_Click(object sender, EventArgs e) =>
    Cursor.Position = PointToScreen(button1.Location);
```

Рисунок 13. Управление указателем мыши

Изменение указателя мыши

Изменение указателя мыши является важным способом предоставления обратной связи пользователю. Например, указатель мыши можно изменить в обработчиках событий **MouseEnter** и **MouseLeave**, чтобы сообщить пользователю о том, что выполняются вычисления, и ограничить взаимодействие с пользователем в элементе управления. Иногда указатель мыши изменяется из-за системных событий, например, когда приложение вовлечено в операцию перетаскивания.

Основной способ изменения указателя мыши заключается в присвоении свойству **Control.Cursor** или **DefaultCursor** элемента управления нового объекта **Cursor**. Примеры изменения указателя мыши см. в примере кода в описании класса **Cursor**. Кроме того, класс **Cursors** предоставляет набор объектов **Cursor** для многих различных типов указателей, например указатель в виде руки.

В следующем примере курсор мыши для кнопки меняется на руку:

```
button2.Cursor = System.Windows.Forms.Cursors.Hand;
```

Рисунок 14. Изменение курсора при наведении (нажатии)

Программный

Задайте для свойства **Text** строку, содержащую амперсанд (**&**) перед буквой, которая будет использоваться в качестве клавиши доступа.

```
button1.Text = "&Print";
```

Рисунок 15. Свойство Text для button

Использование метки для переключения на элемент управления

Несмотря на то, что переключиться на метку невозможно, имеется возможность переключиться на следующий элемент управления в последовательности перехода в форме. Каждому элементу управления присваивается значение свойства **TabIndex**, как правило, в порядке возрастания последовательности. Когда свойству **Label.Text** назначается клавиша доступа, переключение выполняется на следующий элемент управления в последовательности перехода.

Используя пример из раздела Программный, если для кнопки задан не текст, а изображение принтера, для переключения на кнопку можно использовать метку.

```
label1.Text = "&Print";
label1.TabIndex = 9
button1.TabIndex = 10
```

Рисунок 16. Метка для переключения

Отображение символа амперсанда

При задании текста или заголовка для элемента управления, который интерпретирует амперсанд (&) как клавишу доступа, используйте два последовательных амперсанда (&&) для отображения одного символа амперсанда.

Например, текст кнопки "Print && Close" отображается в заголовке Print & Close следующим образом:

```
button1.Text = "Print && Close";
```

Рисунок 17. Одинарный амперсанд

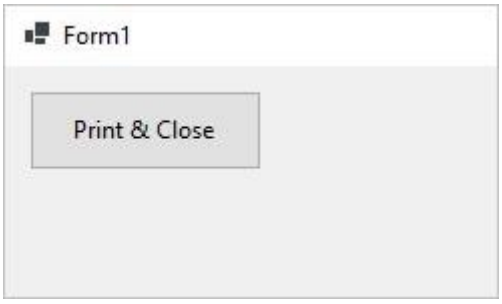

Рисунок 18. Одинарный амперсанд

Кастомизируйте свое приложение, «поиграйтесь» с различными свойствами «цветом, шрифтом, размерами и.т.д», попробуйте добавить дополнительные элементы и «заставить» их взаимодействовать с вашим приложением.

В конце не забудьте отправить готовый (рабочий) проект, в свой репозиторий гитхаба (вы с ним уже разбирались), для этой практической работы создайте отдельную ветку назовите которую «Pr_1», проверьте отправился ли ваш проект, и только тогда проверю вашу работу.

Всем удачи!