

Выполнить: Разработайте приложение, в котором при нажатии на кнопку изображение будет перемещаться по экрану (например, по горизонтали).



Форма «Анимация»

| объект | свойство name |
|-------------------------------------|------------------|
| форма | |
| контейнер для картинки (PictureBox) | pct |
| кнопка (button) | btnStart |
| кнопка (button) | btnExit |
| таймер (timer) | tmr |

1. Создайте новый проект и расположите на новой форме элементы управления *PictureBox* (назовите его *pct*) и два элемента *Button*: *btnStart* и *btnExit*, как показано на рисунке. Создайте также элемент управления *Timer* (*tmr*), который будет передвигать *PictureBox* через определенные промежутки времени.

Элемент *Timer* скрытый, он появляется ниже дизайна формы.

2. В окне свойств элемента *PictureBox* выберите свойство *Image* и в открывшемся диалоговом окне отметьте пункт *Local Resource* (Локальный ресурс). Нажмите кнопку *Import* и выберите любой файл картинки.
3. Для того чтобы *PictureBox* перемещался вправо, необходимо увеличивать его свойство *Left* – координату левого края элемента управления. Поэтому запрограммируйте событие *Tick* элемента *Timer* следующим образом:

```
private void tmr_Tick(object sender, EventArgs e)
{
    pct.Left = pct.Left + 20;
}
```

4. Теперь необходимо запустить *Таймер*. Это можно сделать нажатием кнопки *Старт*. Поместите следующий код в обработку события *Click* кнопки *Старт*:

```
private void btnStart_Click(object sender, EventArgs e)
{
    tmr.Enabled = true;
    tmr.Interval = 50;
}
```

5. Запустите и отладьте приложение. Сохраните его.
6. Исправьте приложение так, чтобы *Графическое окно (PictureBox)* не выходило за край формы (останавливалось бы у правого края окна), а при нажатии на кнопку *Старт* перемещалось бы в исходное положение.
7. Измените приложение так, чтобы при первом нажатии на кнопку *Старт* графическое окно начинало двигаться, а при повторном – останавливалось. При этом должна меняться надпись на кнопке: *Старт* – при запуске, *Стоп* – при остановке.

Задание №2

Выполнить: Создайте приложение *Windows Forms Application*, которое вычисляет сумму ряда $x + x/2 + x/3 + x/4 + \dots$ для $(|x| < 1)$ и количество слагаемых в этой сумме. Расчет производится пока приращение не будет меньше заданной точности.

[Solution and Project name: Lab6_2, form name Lab6_2.cs]

Пример выполнения:

| элемент управления | значение свойства | значение свойства |
|--------------------|-------------------|-------------------|
| | name | text |
| form | Lab7 | Сумма ряда |
| button | btnCalc | Вычислить |
| textbox 1 | txtX | |
| textbox 2 | txtPrecision | |
| Label | lblResult | |
| Labels | | |

1. *Внимание!* Свойства *name* всех элементов управления должны быть заданы в соответствии с таблицей.
2. Создайте новый проект (Файл → Создать → Проект → Windows Forms Application), дайте ему имя — *Lab7*; свойство *name* формы должно быть *Lab7* (окно Свойства window → (Name)).
3. Расположите элементы управления, как на рисунке.
4. Запрограммируйте событие *Click* для кнопки. Для того чтобы это сделать дважды щелкните по кнопке на форме.
5. Объявите переменные для хранения значений: *количество слагаемых, суммы и само слагаемое*:

```
private void btnCalc_Click(object sender, EventArgs e)
{
    // здесь ваш код
    int counter=0; // количество слагаемых
    double sum=0; // сумма
    double summand=0.0; // слагаемое
    double x;
    //...
}
```

6. Для вычисления суммы ряда будем использовать цикл `do..while`. Каждое слагаемое — это результат операции деления `x` на `counter`. Для того, чтобы избежать появления ошибки в случае неверного значения для `x`, будем использовать процедуру `TryParse()`:

```
//...
do
{
    counter++;
    if (double.TryParse(txtX.Text, out x)){
        summand = double.Parse(txtX.Text) / counter;
    }
    sum += summand;
} while (Math.Abs(summand) > double.Parse(txtPrecision.Text));
```

7. Результат будем размещать в метке `lblResults`. Для этого после цикла добавьте код:

```
lblResult.Text = "сумма = " + sum + ", количество = " + counter;
```

8. Запустите приложение, используйте *запятую* в качестве плавающей точки, например: 0,3.
9. Изменим условие цикла так, чтобы при вводе некорректного значения для *приращения*, не возникала бы ошибка. Будем использовать логическую переменную. Добавьте код непосредственно перед циклом:

```
bool f = false;
    if (double.TryParse(txtPrecision.Text, out precision)) {
        f = true;
    }
```

переменная `f` — индикатор того, правильно ли введено значение для приращения. Если значение верное, то `f = true`.

10. Изменяем условие цикла:

```
//..
while (f && Math.Abs(summand) > precision);
```

Проверяем значение «индикатора» *f*. Если *true*, то приращение задано верно, можем продолжить выполнение тела цикла. Ну и кроме того слагаемое должно быть больше приращения (по условию задания).

11. В том случае, если пользователь вводит некорректное значение, будем информировать его об этом, используя класс *MessageBox* и оператор *if*. Измените код вывода результатов:

```
if (f) {  
    lblResult.Text = "сумма = " + sum + ", количество = " + counter;  
}  
else  
{  
    MessageBox.Show("введите числовое значение!");  
}
```

12. Запустите приложение.