INF 154
LECTURE 7

REPETITION 2 – THE FOR LOOP, NESTED LOOPS and the TIMER

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

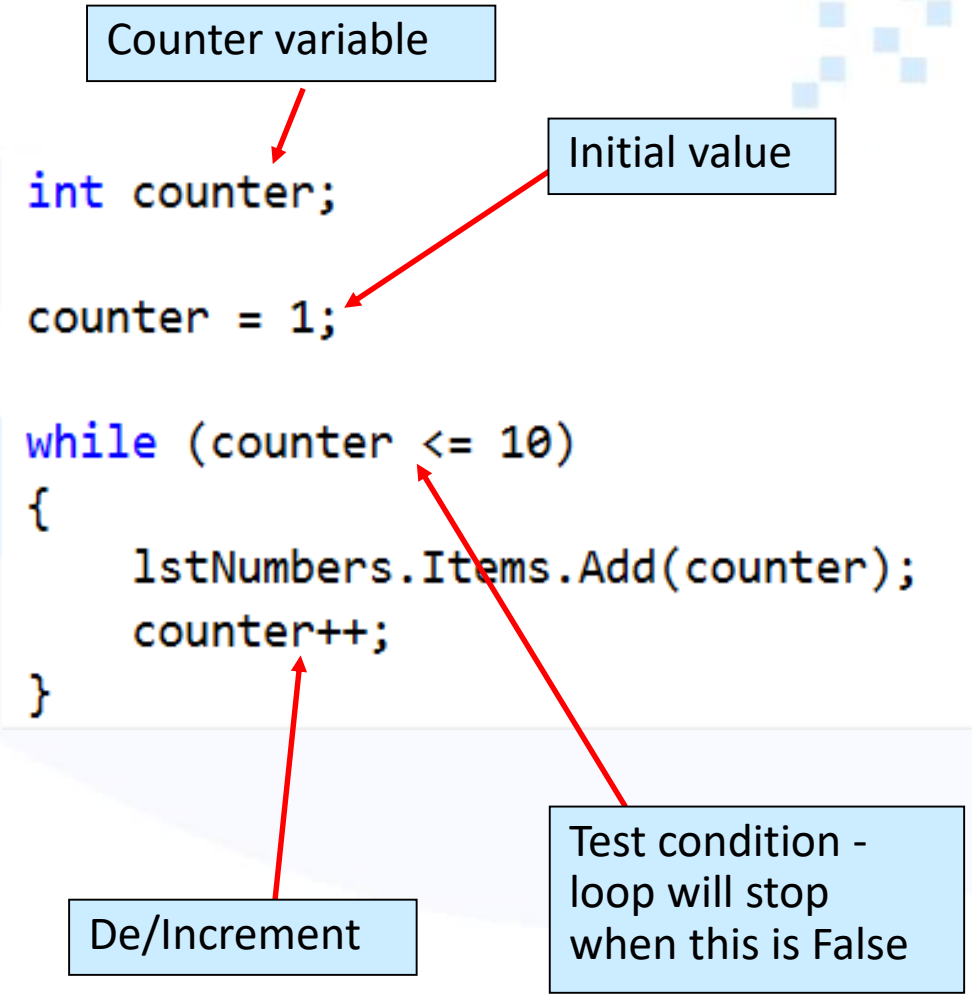
After this lecture you should …

1. Be able to use a simple For loop in a program
2. Be able to use a nested While and For loops
3. Be able to use a Timer in a program.

Counter-controlled **While** loops (reminder)

The essentials of counter-controlled loops are:

- You have a counter variable (can have any name)
- The counter variable has an initial value
- **INSIDE** the loop you increment (or decrement) the counter variable
- You need to test the final value of the counter variable to stop the loop



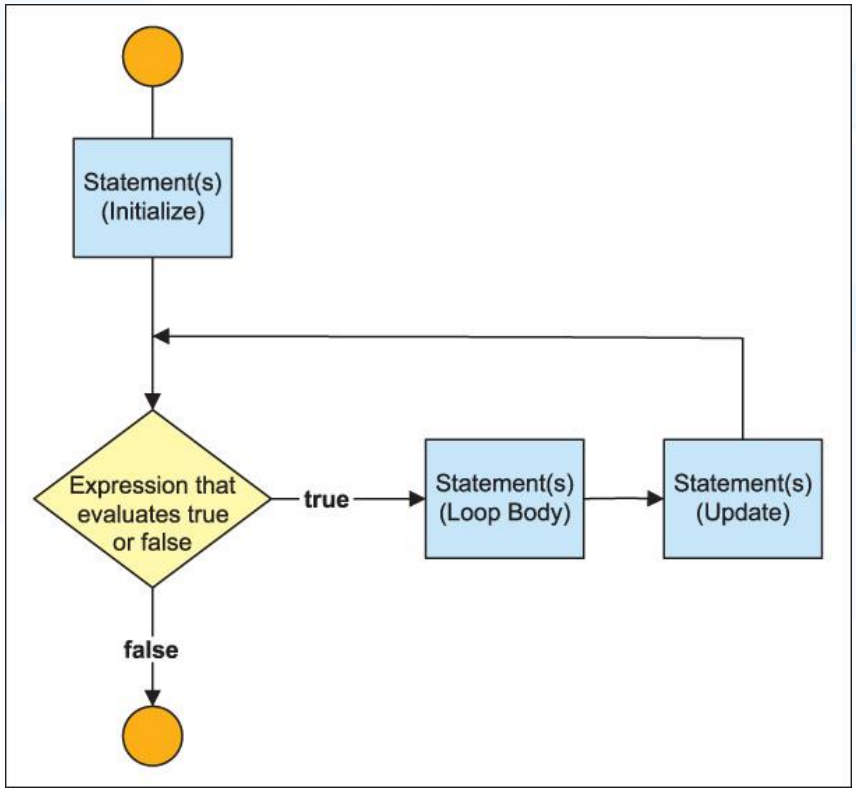
Counter-controlled **for** loops

The **For** loop

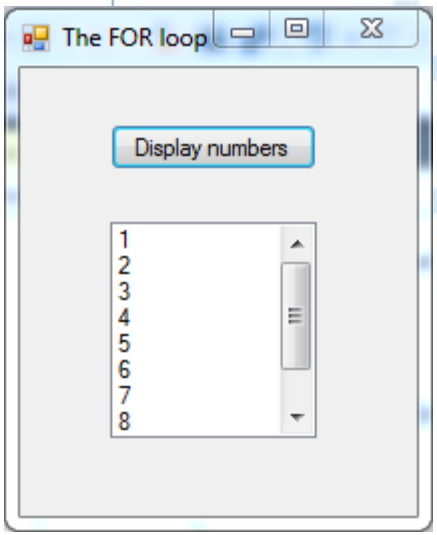
General format of the for loop:

```
for (initialize_counter; condition; update_counter)
{
    statements;
}
```

NOTE: The **for** loop is always counter-controlled



The **For** loop



Example: Add the numbers 1 to 10 to a list box

Counter variable

Initial value

```
private void btnDisplayNumbers_Click(object sender, EventArgs e)
{
    int counter;

    for (counter = 1; counter <= 10; counter++)
    {
        lstNumbers.Items.Add(counter);
    }
}
```

De/Increment

Test final value

The **For** loop

Example: Calculate the sum of the numbers 1 to 100.

```
private void btnDisplayNumbers_Click(object sender
{
    int sum = 0;

    for (int counter = 1; counter <= 100; counter+
    {
        sum += counter;
    }

    MessageBox.Show("The sum is: " + sum);
}
```

Sum accumulation variable. Initialised to 0.

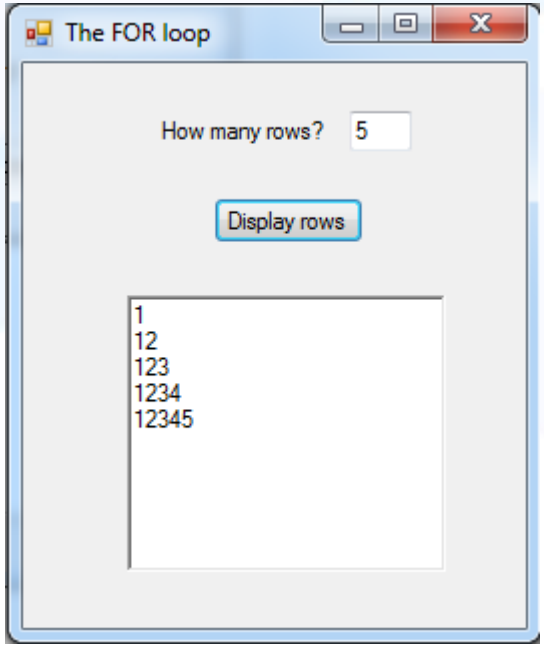
Counter goes from 1 to 100

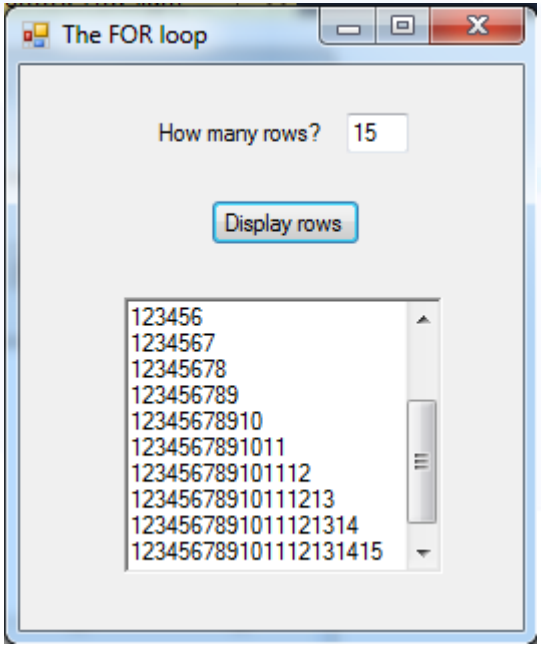
Counter added to sum with each iteration

Note that the counter variable can be declared here

The **For** loop – Theory exercise 1

- Create a program that requests a number of rows to display in a RichTextBox.
- It then displays as many rows: on the first row it displays 1, then 12, then 123, etc. until it has displayed the required number of rows.





The **For** loop – Theory exercise 1

```
private void btnDisplayRows_Click(object sender, EventArgs e)
{
    int noOfRows;
    string line = "";  // variable to create lines of output

    // Get number of rows to display
    noOfRows = Convert.ToInt32(txtNoOfRows.Text);

    // Clear output box
    rtxOutput.Clear();

    for (int i = 1; i <= noOfRows; i++)
    {
        line += Convert.ToString(i);   // build line on previous one
        rtxOutput.Text += line + "\n"; // add new line to the output text
    }
}
```

This adds the counter value to the previous value of line

This adds the new line to the text already in the Text property.

Nested loops

Nested loops

- Like If statements, loop statements can also be nested.
- You can place any combination of **for** and **while** loops within each other.
- A for loop nested in another would have the following structure:

```
for ( initialise; condition; increment )
{
    statement(s);
    for ( initialise; condition; increment )
    {
        statement(s);
    }
    statement(s);
}
```

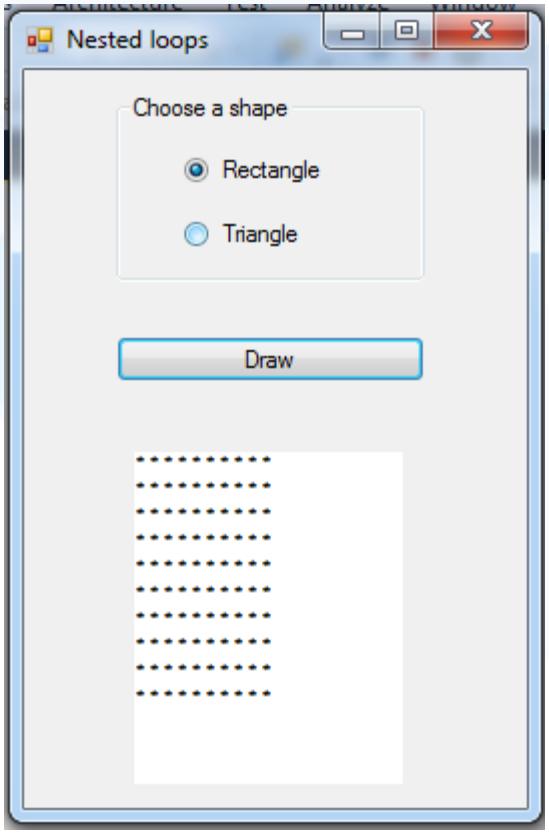
Nested loops

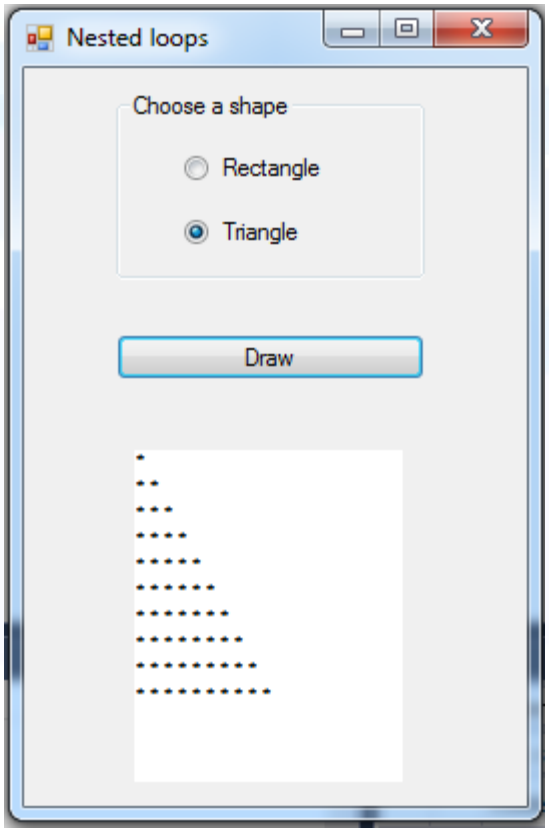
- A for loop nested in a while loop would have the following structure:

```
while (condition)
{
    statement(s);
    for ( initialise; condition; increment )
    {
        statement(s);
    }
    statement(s);
}
```

Nested loops – Theory exercise 2

- Create a program that draws one of two shapes in a RichTextBox (as shown in the images below).
- It strings together the * character to create the shapes using nested for loops.





Nested loops – Theory exercise 2

```
for (int i = 1; i <= 10; i++)
{
    line = "";
    for (int j = 1; j <= 10; j++)
    {
        line += "* ";
    }
    rtxOutput.Text += line + "\n";
}
```

Code for rectangle

Code for triangle

```
for (int i = 1; i <= 10; i++)
{
    line = "";
    for (int j = 1; j <= i; j++)
    {
        line += "* ";
    }
    rtxOutput.Text += line + "\n";
}
```

Nested loops – Theory exercise 3

Can you work out what the output of the following will be?

```
int inner, outer;

for (outer = 1; outer <= 3; outer++)
{
    for(inner = 10; inner > 5; inner--)
    {
      lstOutput.Items.Add(outer + "  " + inner)
    }
}
```

Repetition with the Timer

Repetition with the **Timer** control

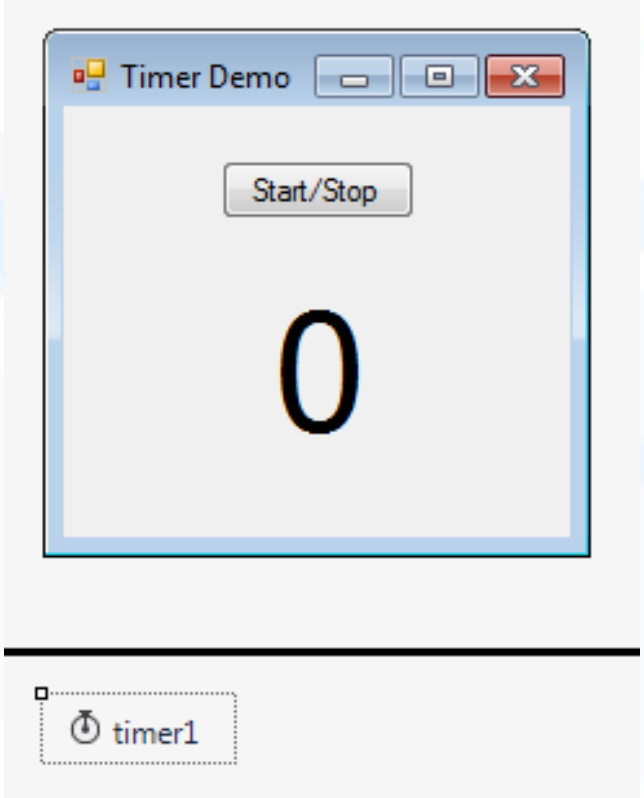
- The purpose of the **Timer** control is to trigger an event at regular intervals.
- The length of the intervals is defined by the **Interval** property, whose value is in milliseconds, i.e. 1000 = 1 second.
- The timer must be **Enabled** or **Start**ed to start working.
- When the control is enabled, the **Tick** event is raised every interval. This is where you would add code to be executed.
- It is a non-visible control.

The Timer control – Exercise 4

Create a form with a large Label with Text property set to “0” and a Button with the Text “Start/Stop”.

Also place a Timer control on the form.

When the user clicks the Button, the Timer is enabled and it adds 1 to the Text in the label every 1 second. When the user clicks the menu item again the counter stops.



The Timer control – Exercise 4

We need the following two event handlers:

```
private void btnStartStop_Click(object sender, EventArgs e)
{
    // if the timer is enabled, disable it and vice versa
    if (tmrCount.Enabled)
        tmrCount.Stop();
    else
        tmrCount.Start();
}
```

When the button is clicked the timer must either be enabled or disabled, depending on the state it is currently in.

```
private void tmrCount_Tick(object sender, EventArgs e)
{
    // Get the current counter value and add 1
    int count = Convert.ToInt32(lblCounter.Text);
    lblCounter.Text = Convert.ToString(count + 1);
}
```

In the timer's Tick event the number in the label is increased by 1. This event handler is executed repeatedly only while the Timer is enabled.

The Timer control – Exercise 4

An alternative way to implement the counter is to declare a counter variable outside the Tick event handler that is incremented inside the event handler.

```
int count = 0;

private void tmrCount_Tick(object sender, EventArgs e)
{
      // Add 1 to the counter and display it in the label.
      count++;
      lblCounter.Text = Convert.ToString(count + 1);
}
```

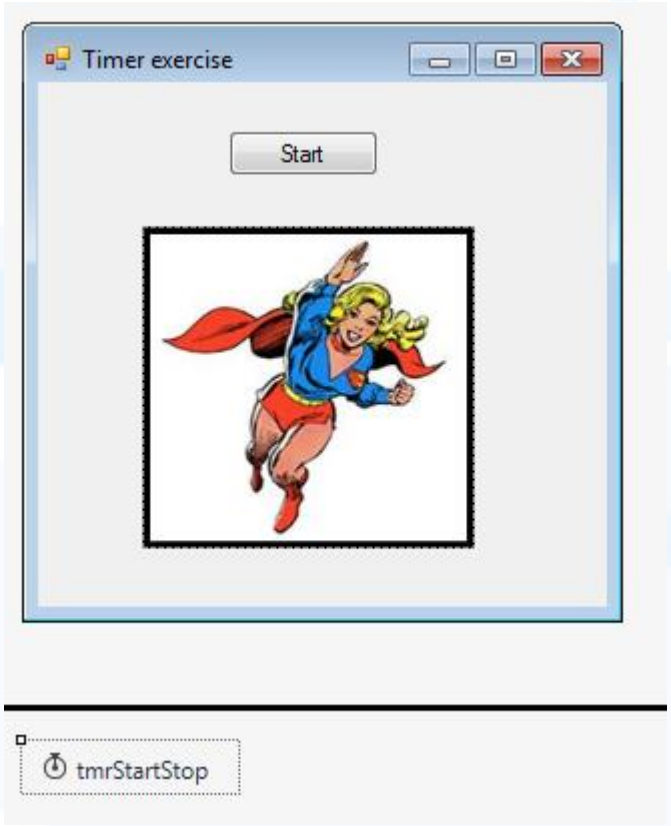
We have to declare the counter outside the event handler for counting to continue on. If we declare it inside the event handler it will be reset every time the event handler is triggered.

The Timer control – Home exercise

- Create a form with a PictureBox, a Timer and a Button.
- Set the following properties for the Timer in the Properties Window:

(Name)	tmrStartStop
Enabled	False
GenerateMemb	True
Interval	1000

- Place any picture in the PictureBox and change the Button's Text to "Start".
- When the user clicks Start, the image starts flashing (showing and disappearing). Also, the Text on button changes to "Stop".
- If the user clicks Stop, the image stops flashing and the Button's Text changes to "Start".



The Timer control – Home exercise

- Double-click on the Button and change its event handler as follows:

```
private void btnShowHide_Click(object sender, EventArgs e)
{
    if (tmrStartStop.Enabled)
    {
      // Stop the timer and change button's text
      tmrStartStop.Stop();
      btnShowHide.Text = "Start";
    }
    else
    {
      // Start the timer and change button's text
      tmrStartStop.Start();
      btnShowHide.Text = "Stop";
    }
}
```

The Timer control – Home exercise

- Double-click the Timer and change its event handler as follows :

```
private void tmrStartStop_Tick(object sender, EventArgs e)
{
    // if  image is visible make it invisible and vice versa
    pbxSuper.Visible = !pbxSuper.Visible;
}
```