

INF 154

LECTURE 8



ARRAYS, LISTS and INDEXES



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihalefi

After this lecture you should ...



1. Be able to declare and use an array
2. Be able to use a For loop to step through the elements of an array.
3. Be able to work with the Lines property of a RichTextBox as a list of indexed elements.
4. Be able to use the ImageList control as an indexed list of images.
5. Be able to work with the items of a ListBox as a list of indexed elements.



Introduction to arrays



Introduction to arrays

- Programming problems often require the use of a group of similar variables.
- For instance: assume you want to get the average marks of 5 students you have in class.
- You could declare 5 variables:

```
double mark1;  
double mark2;  
double mark3;  
double mark4;  
double mark5;
```

- You can then use the 5 variables as follows:

```
Average = (mark1 + mark2 + mark3 + mark4 + mark5) / 5;
```

- **But what if you want to do it for 400 students?**



Introduction to arrays

- An array is a group (or list) of contiguous memory locations that have the same name and the same type.
- To refer to a particular element in an array we specify the position number (index or subscript) of the element in the list.
- NB: The first element is at position 0 (zero), NOT 1.

classMarks

| | |
|------|---------------|
| 50.0 | classMarks[0] |
| 40.0 | classMarks[1] |
| 70.0 | classMarks[2] |
| 60.0 | classMarks[3] |
| 80.0 | classMarks[4] |

Declaring an array

The array called **score** can be declared as follows:

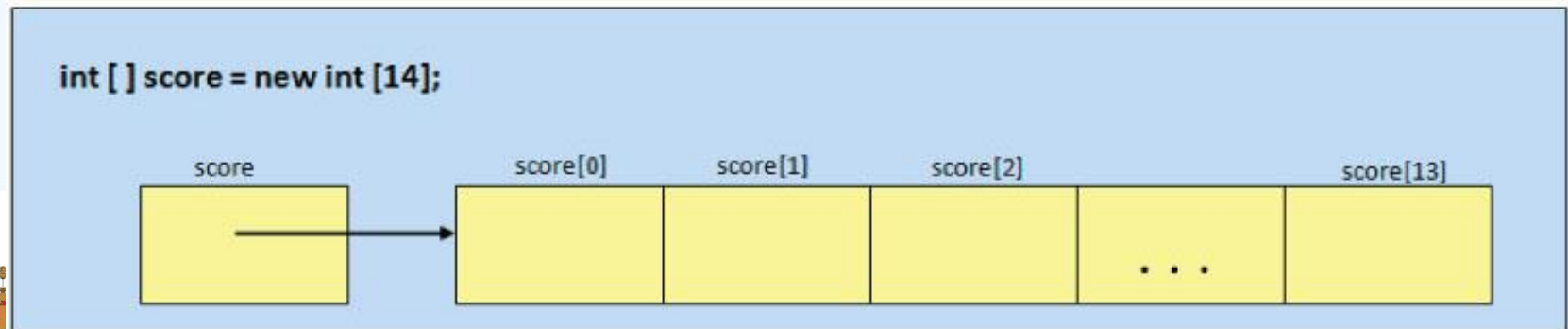
```
int [] score = new int[14];
```

OR

```
int [] score;
```

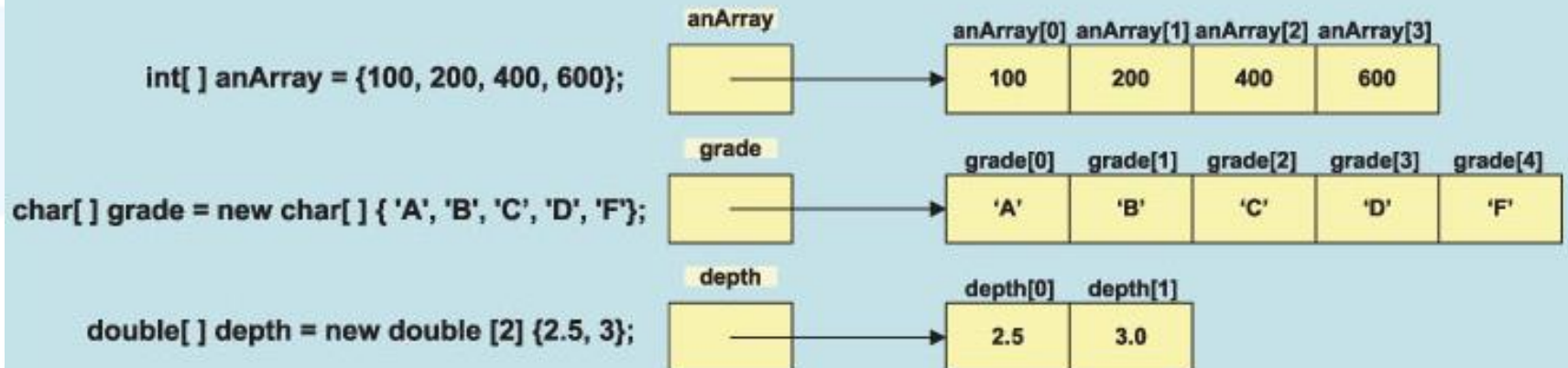
```
int size = 14;
```

```
score = new int[size];
```



Declaring an array

```
const int size = 15;  
string [ ] lastName = new string [25];  
double [ ] cost = new double [1000];  
double [ ] temperature = new double [size];  
int [ ] score;  
score = new int [size + 15];
```



Using an array

The following code will create an array (numbers) with five integer elements and assign the values 10, 20, 30, 40, 50 to them.

```
int[] numbers = new int[5];  
  
for (i = 0; i < 5; i++)  
{  
    numbers[i] = (i+1) * 10;  
}
```

The for loop's counter is used as the index to the array elements. The index goes from 0 to 4

Using an array (calculating an average)

```
double[] mark = new double[5];  
double sum = 0;  
double average;
```

Declare the
array

```
mark[0] = 50.0;  
mark[1] = 40.0;  
mark[2] = 70.0;  
mark[3] = 60.0;  
mark[4] = 80.0;
```

Assign values to
the elements

```
for (int i = 0; i < 5; i++)  
{  
    sum += mark[i];  
}
```

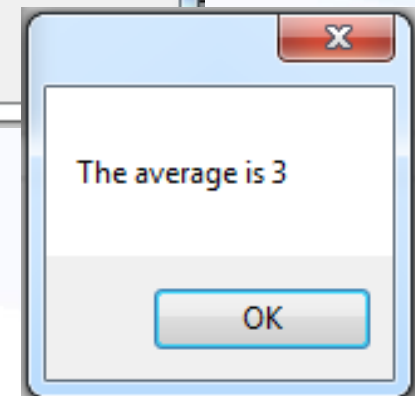
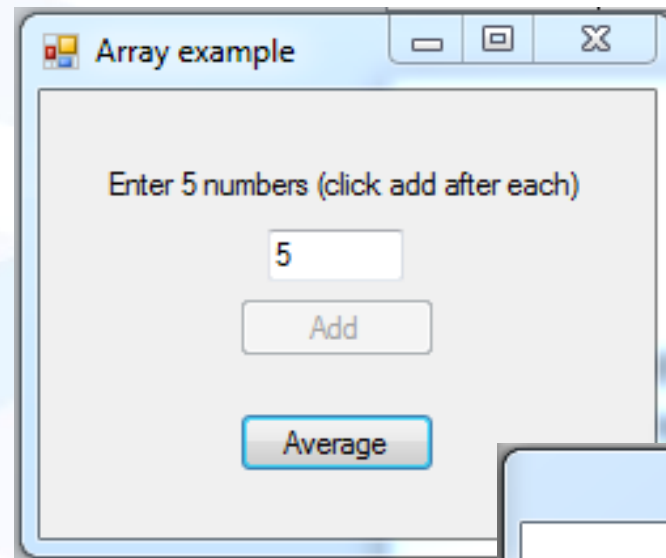
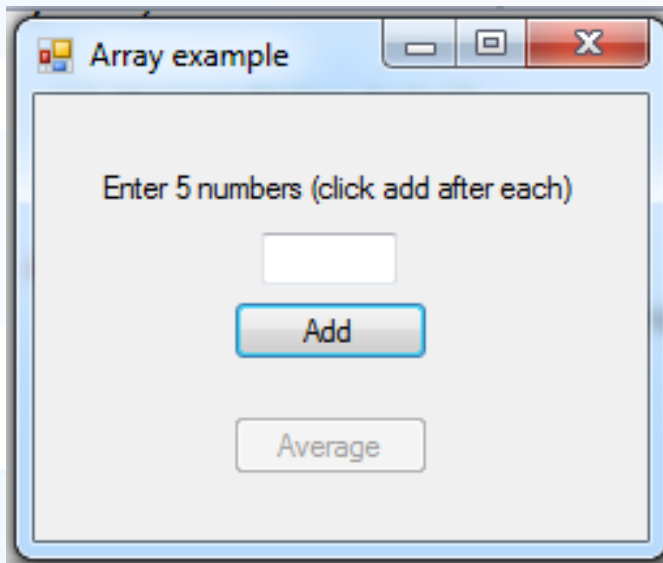
Use the array
elements

```
average = sum / 5;  
lblOutput.Text = "Average mark is: " +  
    Convert.ToString(average));
```



Theory Exercise 1

Change the program on the previous slide so that the five numbers to be stored in the array are provided by the user.



The Form level array and Add event handler

```
// Declare a Form level array
int[] numbers = new int[5];
int i = 0;
```

The array is declared on Form level so that both event handlers can use it

1 reference

```
private void btnAdd_Click(object sender, EventArgs e)
{
    numbers[i] = Convert.ToInt32(txtInput.Text);
    i++;

    txtInput.Clear(); // Clear text box
    txtInput.Focus(); // Put cursor in text box

    // if 5 numbers entered, disable Add button
    if (i == 5)
    {
        btnAdd.Enabled = false;
        btnAverage.Enabled = true;
    }
}
```

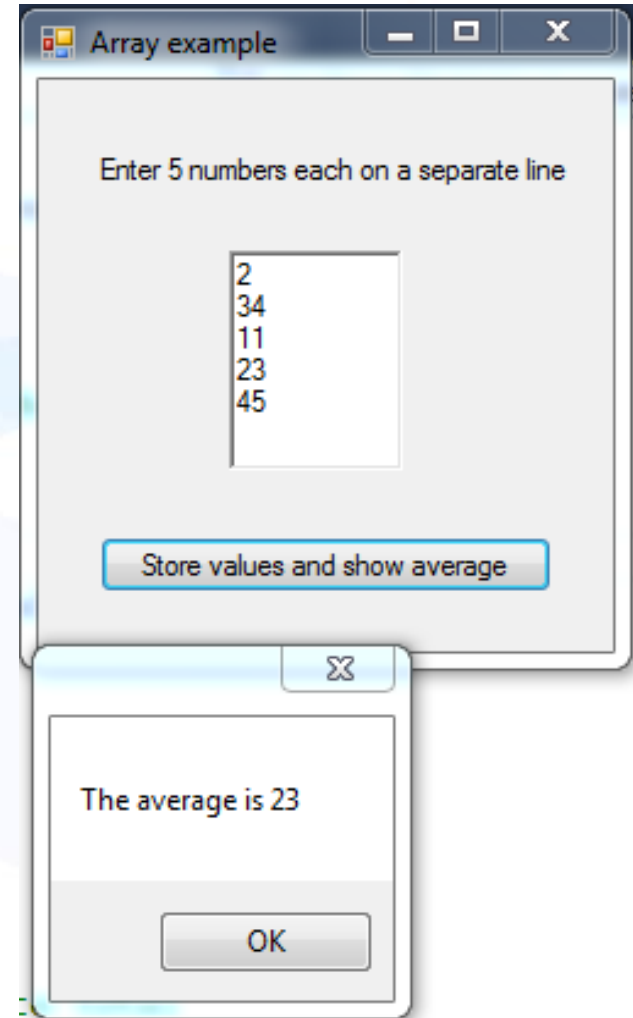
Make the interface usable

The btnAverage event handler

```
private void btnAverage_Click(object sender, EventArgs e)
{
    int sum = 0;
    double average;
    // Calculate average
    for (int i = 0; i < 5; i++)
    {
        sum += numbers[i]; // Accumulate total
    }
    average = sum / 5.0;
    MessageBox.Show("The average is " +
                    Convert.ToString(average));
}
```

Theory Exercise 2

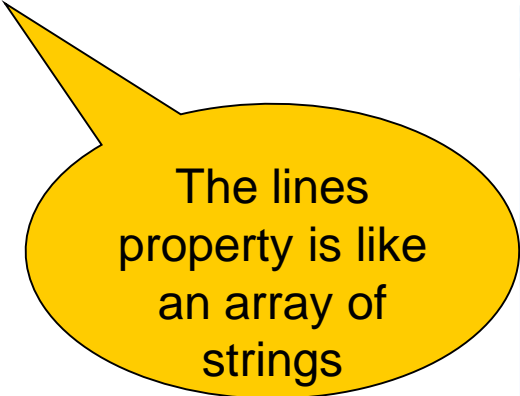
Change the previous program so that the user enters all five values into a RichTextBox and the values are then transferred to the array.



```
private void btnStore_Click(object sender, EventArgs e)
{
    int[] numbers = new int[5];

    // Load the lines in the RichTextBox into the Array
    for (int i = 0; i < 5; i++)
    {
        // Transfer values to array
        numbers[i] = Convert.ToInt32(rtxInput.Lines[i]);
    }

    int sum = 0;
    double average;
    // Calculate average
    for (int i = 0; i < 5; i++)
    {
        sum += numbers[i]; // Accumulate total
    }
    average = sum / 5.0;
    MessageBox.Show("The average is " +
                    Convert.ToString(average));
}
```

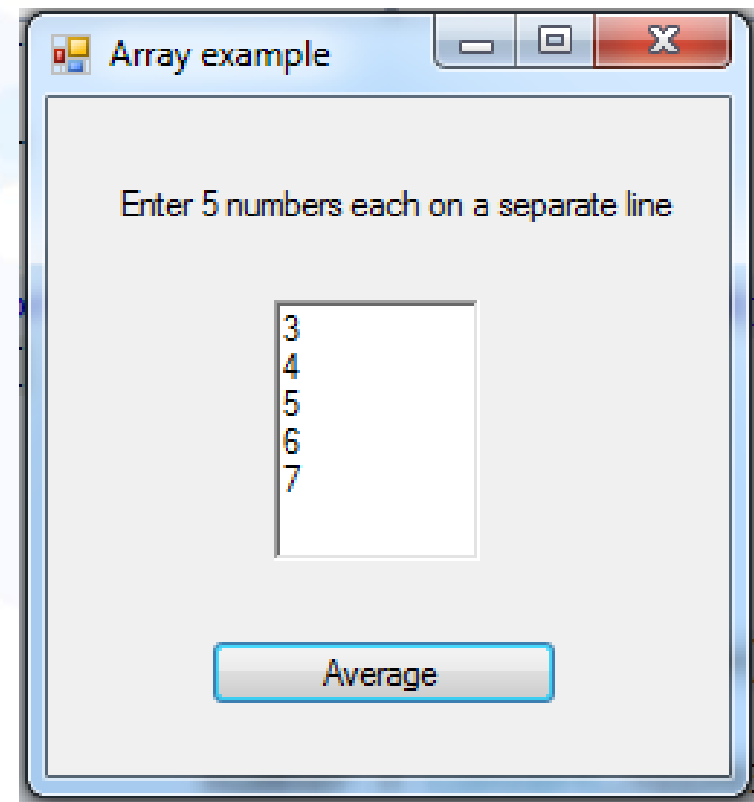


The lines
property is like
an array of
strings

Theory Exercise 3

Change the previous program so that the user enters all five values into a RichTextBox and the average is calculated without using an array.

Note: The Lines property of the RichTextBox is already a form of an array, so we can use it like an array.



The Lines property as an array

```
private void btnStore_Click(object sender, EventArgs e)
{
    int sum = 0;
    int number = 0;
    double average;

    // Calculate average
    for (int i = 0; i < 5; i++)
    {
        number = Convert.ToInt32(rtxInput.Lines[i]);
        sum += number; // Accumulate total
    }
    average = sum / 5.0;
    MessageBox.Show("The average is " +
                    Convert.ToString(average));
}
```



Summary of Array Basics

- Data structure that may contain any number of variables
- Variables must be of same type
- Single identifier (name) given to entire structure
- Individual variables are called elements
- Elements are accessed through an index
 - Index also called a subscript
 - Elements are sometimes referred to as indexed or subscripted variables

Summary of Array Basics

- First index for all arrays is 0
- Last element of all arrays is always referenced by an index with a value of the length of the array minus one
- Specify which element to access by specifying the index enclosed in square brackets after the array identifier (array name) , e.g. **Score[0]**



Working with ListBox items

- The Items property of the ListBox control is a list of items that can be accessed individually through an index.
- The index of the first item in the ListBox is 0.
- The statement

`lstNames.Items[0] = "Clark Kent";`

stores the string "Clark Kent" in the first item of the ListBox named lstNames.

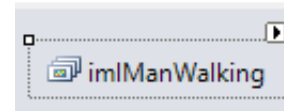
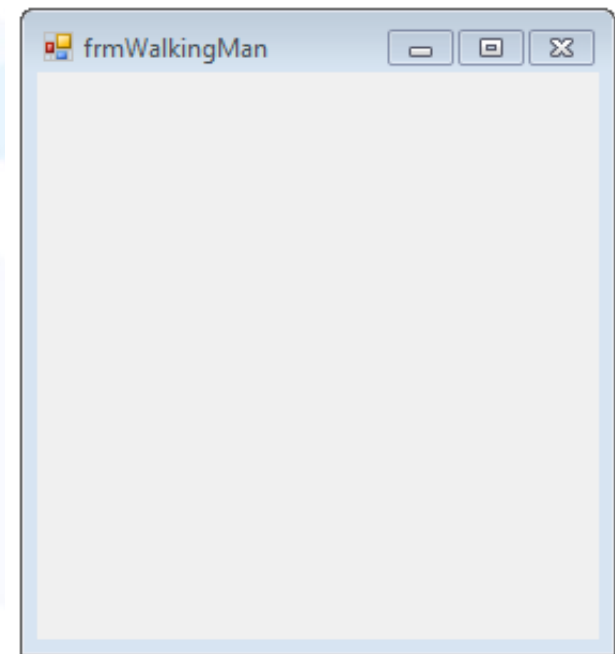
- A for loop can be used to step through all the items in a ListBox, using the counter variable as the index (see exercises to do at home at the end of this presentation).

The ImageList



The ImageList control (Exercise 4)

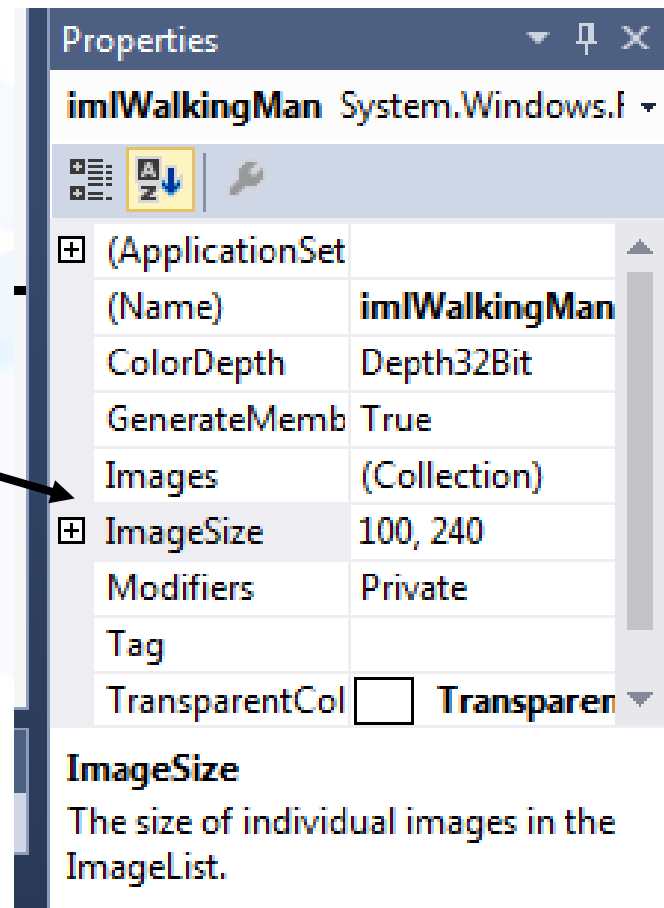
- The **ImageList** control is used to store a collection of images for use by other controls.
- Its **Images** property is an array of images where you can keep a collection of images for use in a program.
- Add an ImageList to the Form and name it imlManWalking.
- Go to clickUP and download the six “Man” images from this week’s Lecture Notes folder. Store them on the Desktop.



The ImageList control

To add images to the ImageList, click on its **Images** property to open the Images Collection Editor. Use the Add button to copy images from wherever you have stored them.

Change the size of the images to 100, 240



The ImageList control

- Add a PictureBox control name pbxMan to the form.
- Change the **SizeMode** property to **StretchImage** to make the image fit into the picture box.
- One can place an image from the ImageList into the PictureBox's Image property. For example:

```
pbxMan.Image = imlManWalking.Images[3];
```

- We want an image to be loaded into the PictureBox when the form comes up (see next slide)

The FormLoad Event

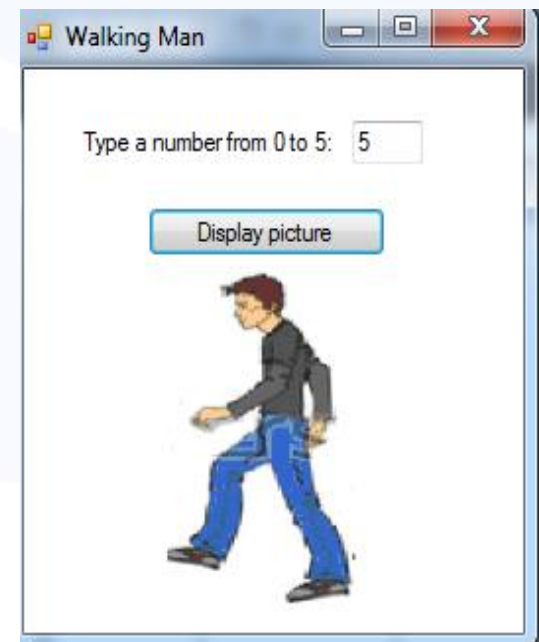
- Whenever a new form is displayed, its Load event is triggered.
- Double-click anywhere in the form to open its Load event handler.
- Now add the following line of code to that event handler.

```
private void Form1_Load(object sender, EventArgs e)
{
    pbxMan.Image = imlManWalking.Images[3];
}
```



Exercise 4 continued

- Change the walking man form to allow the user to choose an image to display. The interface should appear as below.
 - Let the user enter a number from 1 to 6 and then display the corresponding picture in the PictureBox when the user clicks the button.
 - If an invalid number is entered, display an error message.



Exercise 4 continued

The button's event handler looks as follows:

```
private void btnDisplay_Click(object sender,
                                EventArgs e)
{
    int num = Convert.ToInt32(txtNumber.Text);
    pbxMan.Image = imlWalkingMan.Images[num];
}
```

The user's input is the index of the element to be retrieved from the Images list.

Exercise 5 - Animation

- Display the six images of the walking man one after the other every tenth of a second
- When it comes to the end of the images it starts again at the first image
- Use a button to start and stop the animation
- Make the form's background white
- We will need a variable that lies outside of the event handlers (on Form level) to keep count of which image is displayed.
- A Timer will execute every tenth of a second to load a new image.

Exercise 5 - Animation

```
private void btnStartStop_Click(object sender, EventArgs e)
{
    tmrWalkingMan.Enabled = !tmrWalkingMan.Enabled;
}
```

```
int num = 0; // Form level variable
```

```
private void tmrWalkingMan_Tick(object sender, EventArgs e)
{
    // Change index value
    if (num == 5)
        num = 0;
    else
        num++;

    // Load new image in the
    pbxMan.Image = imlWalkingMan.Images[num];
}
```



Working with list elements

Exercises to work through at home

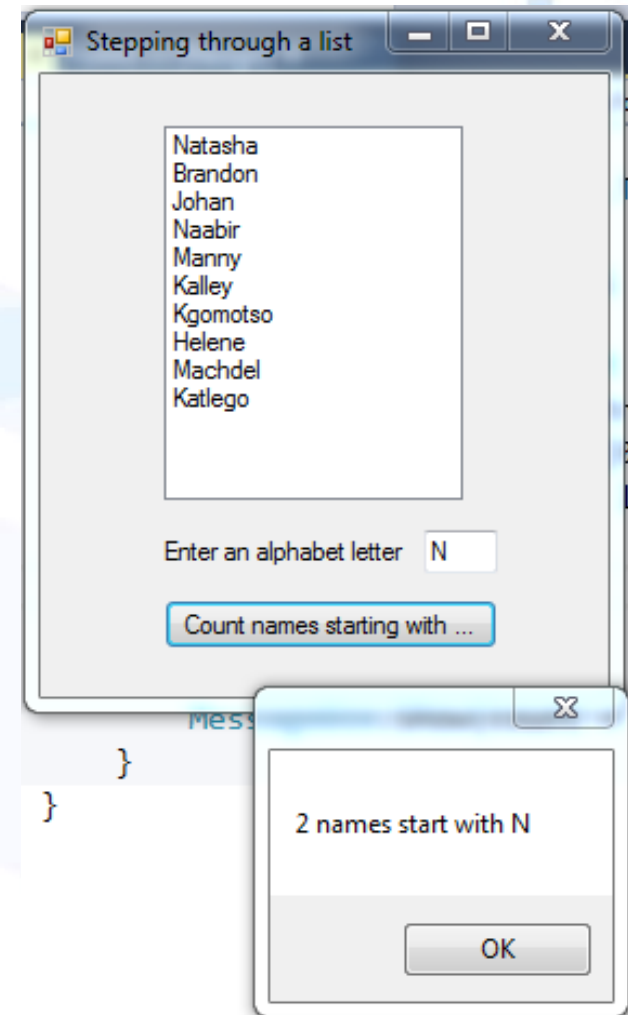


Working with list elements

In this program we have list of 10 names.
The user provides a letter and the program must then count how many names start with that letter.

The solution does the following:

- Step through the list of names one by one.
- For each name, get the first letter.
- Compare the letter with the user's input.
- If equal, add 1 to a counter.



Working with list elements

```
string name;  
char letterToFind, firstLetter;  
int count = 0;  
  
letterToFind = Convert.ToChar(txtLetter.Text);  
  
for (int i = 0; i < 10; i++) // counter goes from 0 to 9  
{  
    // retrieve name at position i  
    name = Convert.ToString(lstAchievers.Items[i]);  
    firstLetter = name[0]; // get the first letter of the name  
    if (firstLetter == letterToFind)  
    {  
        count++;  
    }  
}  
MessageBox.Show(count + " names start with " + letterToFind);
```

This is the i-th
element in the
Items list.

A string is also like
an array – an
array of chars

There are a few things that could go wrong here –
can you spot them?



Working with list elements

The things that could go wrong are:

1. There may be fewer than 10 names in the list, then the for loop won't work. To fix:

```
for (int i = 0; i < lstAchievers.Items.Count; i++)
```

2. The names may not start with a capital letter or the user may not enter a capital letter. To fix:

```
if (Char.ToUpper(firstLetter) == Char.ToUpper(letterToFind))
```

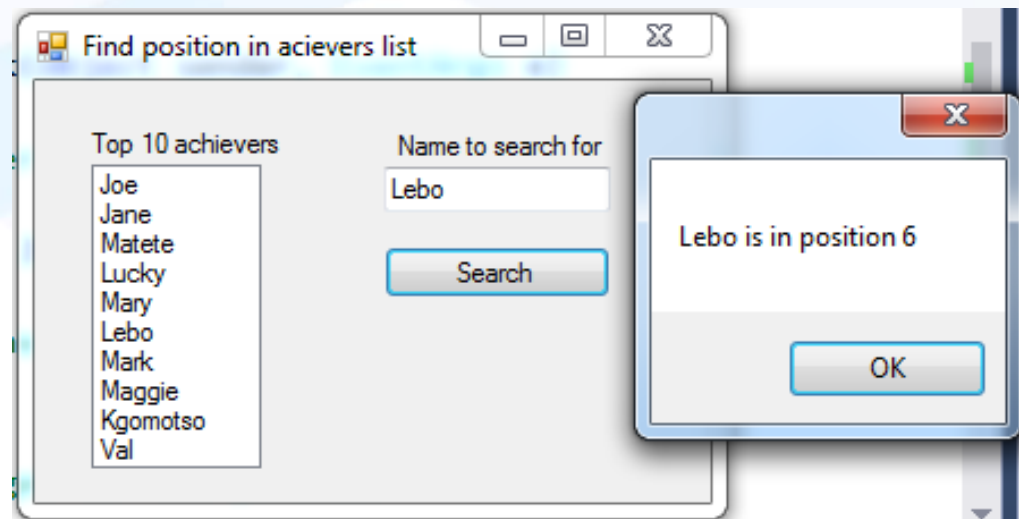
3. The input may be empty. To fix:

```
if (txtInput.Text != "")
```


Working with list elements

At the end of Lecture 6's slides we gave you the following program:
The top ten INF154 students' names are listed from first to tenth in a list box. The program allows you to enter a name and then displays that person's position in the list.

We will use a While loop to go through the list and test each name to see if it is the required one. When we find the name the position will be displayed.



Working with list elements

This was the solution

```
private void btnSearch_Click(object sender, EventArgs e)
{
    int index = 0; // Number of name in the list, starting at 0

    while (index < 10 && (string)lstNames.Items[index] != txtName.Text)
    {
        index++; // Go to the next name in the list
    }

    if (index >= 10) // name not in list
        MessageBox.Show("The name is not in the list");
    else // name found
    {
        index++; // Add one to get correct position because first item is 0

        // Display the person's position - add 1 to index
        MessageBox.Show(txtName.Text + " is in position " + index);
    }
}
```

Here we are
retrieving a element
from the list with an
index