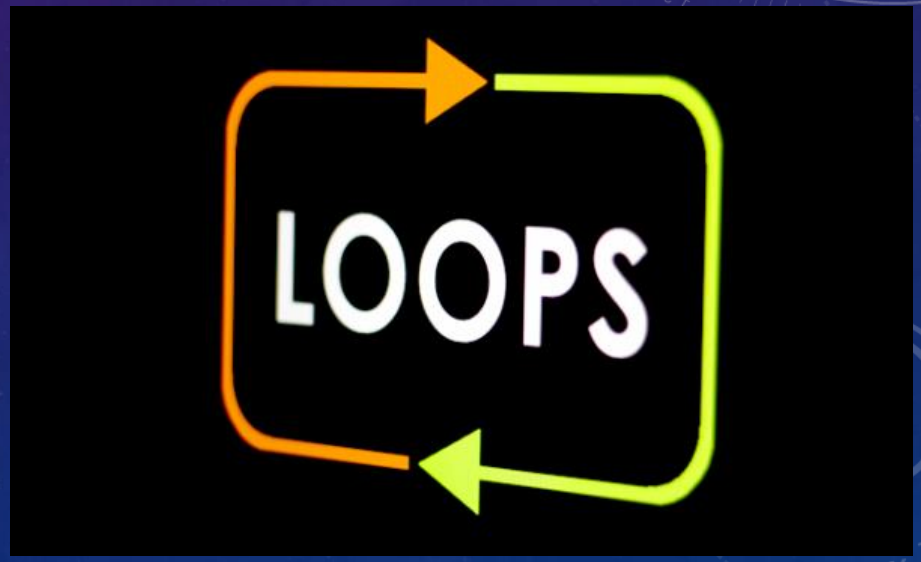
LOOPING CONSTRUCTS & ITERATION

VISUAL BASIC

TABLE OF CONTENTS

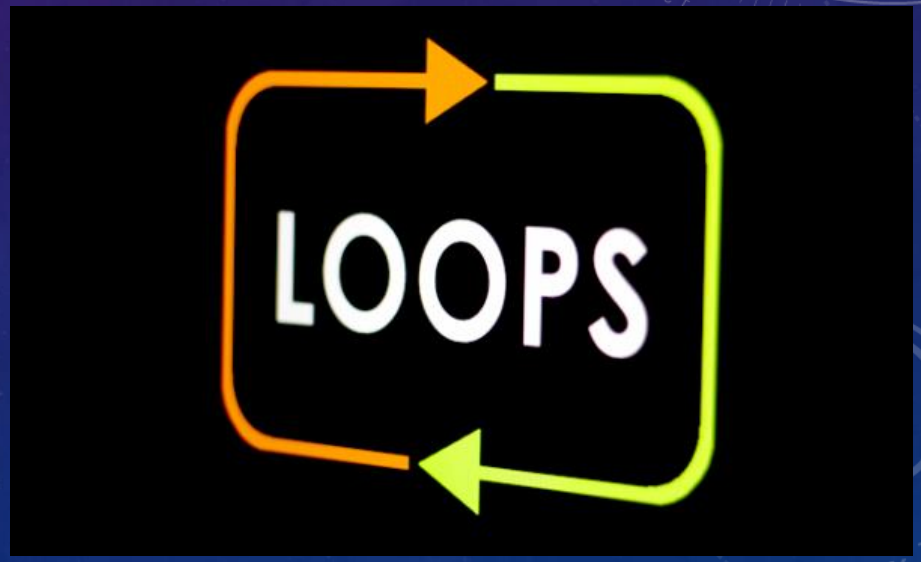
LECTURE CONTEXT

- Understand the power of the second structured programming technique (iteration)
- See why we need looping constructs to create powerful, flexible programs
- Be familiar with several looping constructs used in Visual Basic and their usage
- Understand what is meant by nesting and the type of problem that requires its use

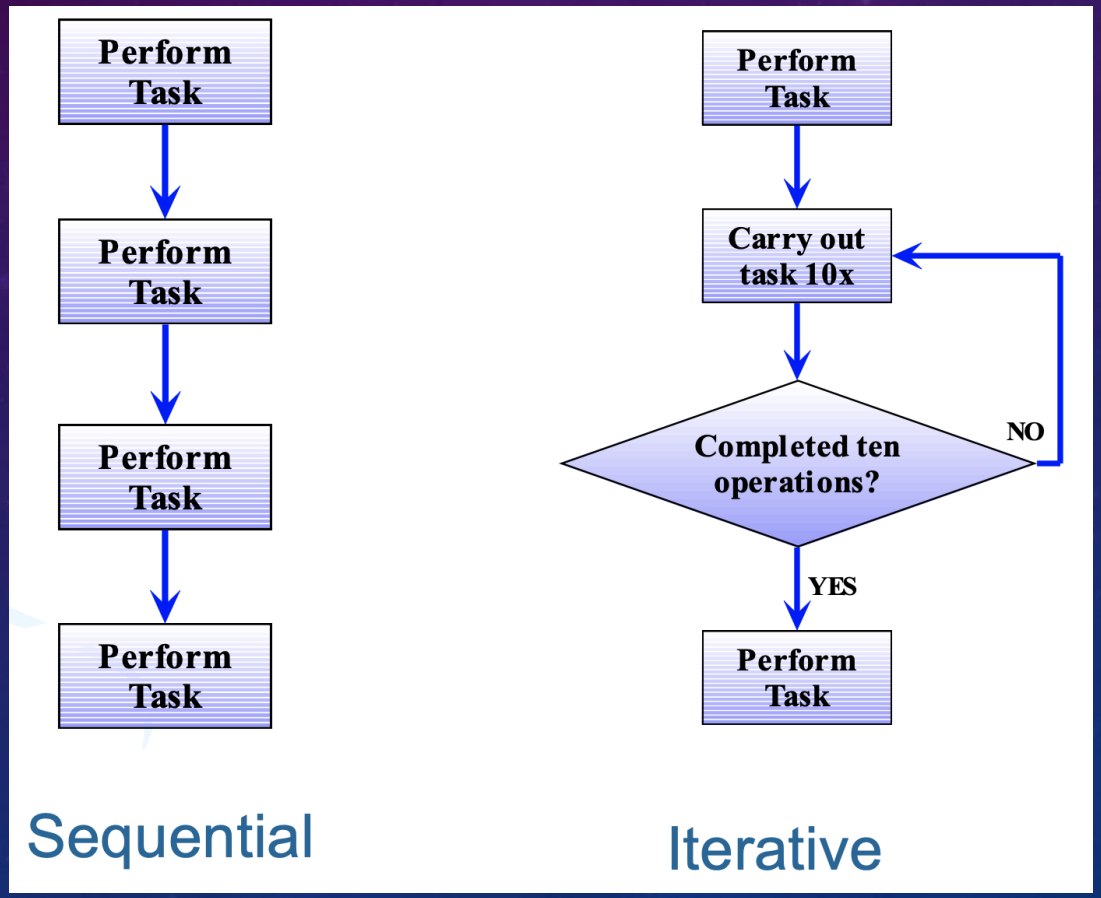


LECTURE CONTEXT

- The most powerful construct of the three available to a programmer is iteration – the ability to repeat an operation for a either a predetermined number of cycles or until some stimulus occurs or condition is met.
- Write a few lines instead of thousands!! This is particularly useful in engineering when one is often processing groups of data stored in arrays (matrices).
- Programming languages typically offer several iterative constructs with slightly differing functionality for varying tasks and VB2010 is no exception.



TYPICAL FLOW SEQUENCES



WHILE LOOP

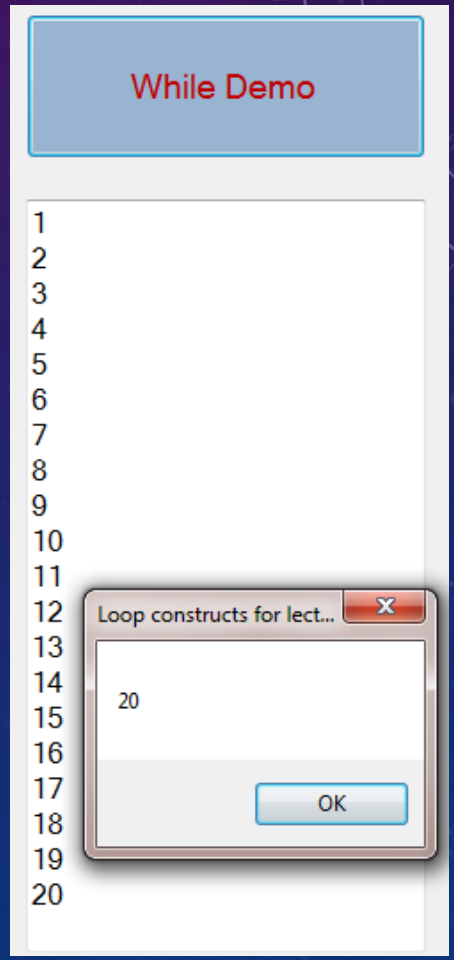
- Repeated execution until some condition is met
- Comparisons for all constructs x>y, x<y, x=y, x<>y
- Begins with **While** keyword, ends with **End While**
- Test performed before code is run
- If condition false before loop entry, code not run
- If condition never met in loop then it runs until program is terminated - use *exit sub x*

```
While (condition)
        Insert user code here
End While 'go back to start of loop until condition met
```

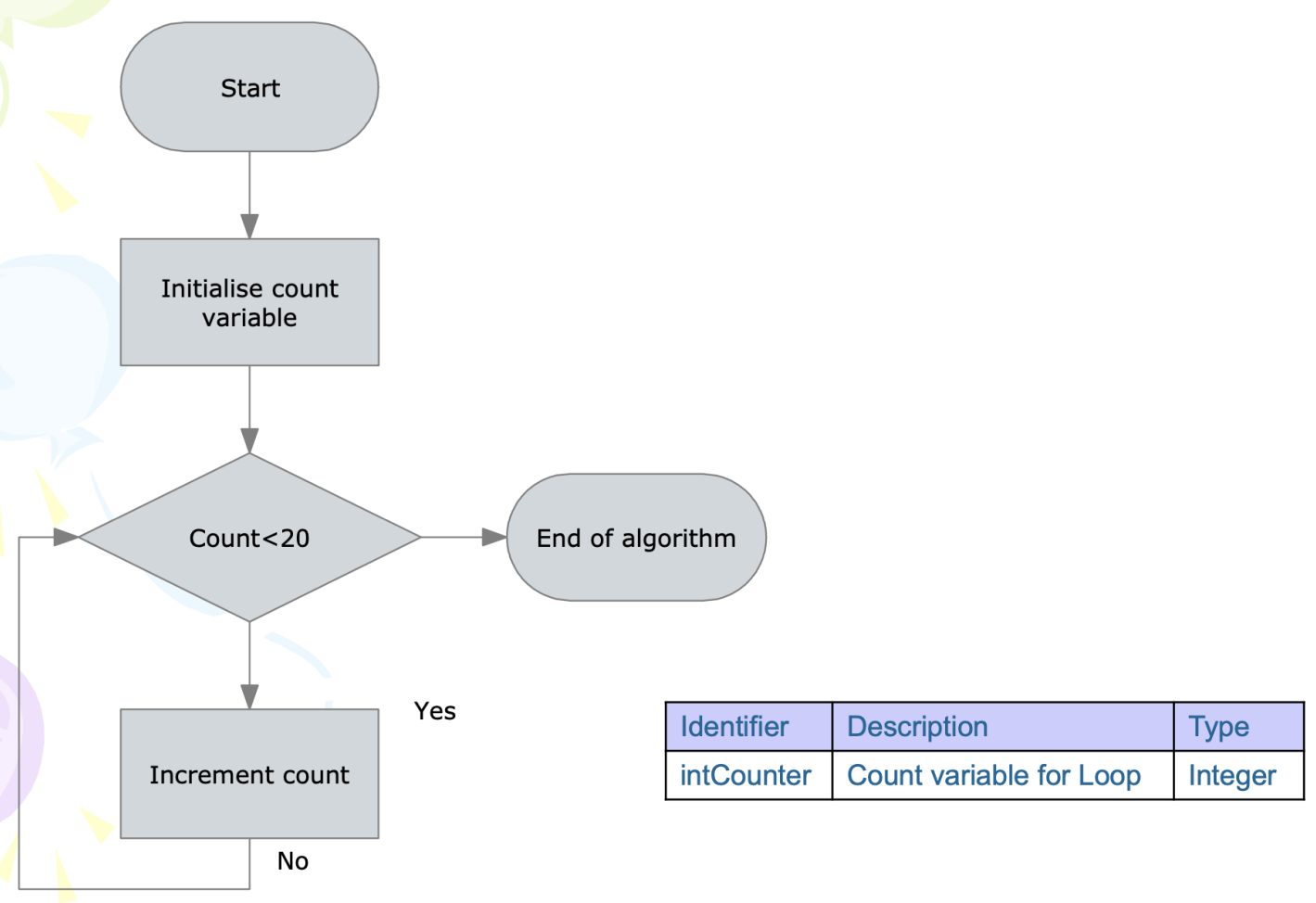
WHILE LOOP

- May prematurely leave the loop if desired using Exit keyword:
- Exit While

```
Private Sub While_Demo(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim intCounter As Integer  'declare variable
    intCounter = 0              ' Initialise variable

    While intCounter < 20      ' Test value
        intCounter = intCounter + 1 'Increment variable
        txtDisplayResult.Text = txtDisplayResult.Text & intCounter & vbCrLf 'added for clarity in lecture
    End While   'End While loop when intCounter > 19

    MsgBox(intCounter)  ' Prints total count in message box

End Sub
```



WHILE LOOP



Identifier	Description	Type
intCounter	Count variable for Loop	Integer

DO LOOP

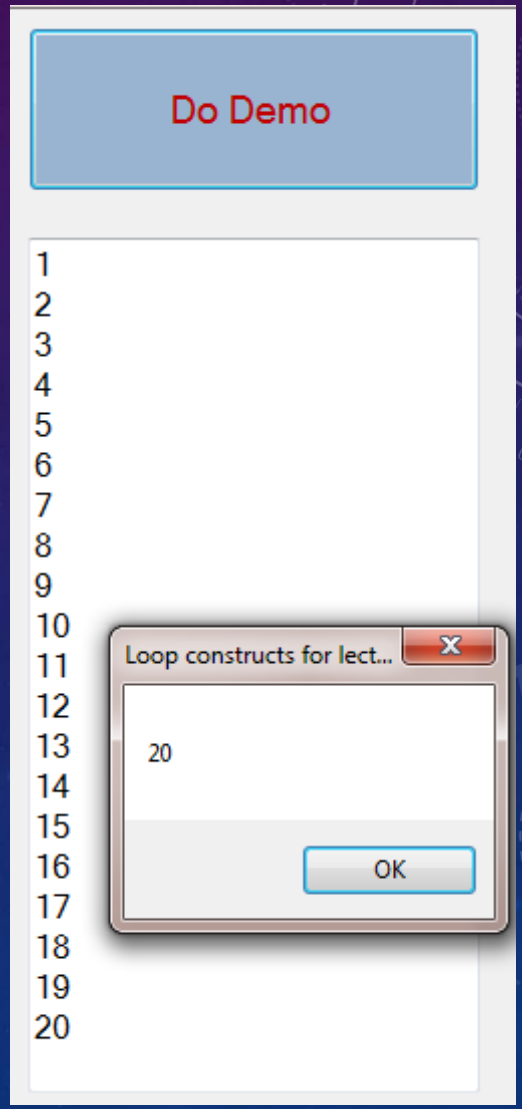
- The test is performed at **end** of the loop
- Will always run at least **once,** even when the condition was met before the code is reached
- *We may prematurely exit using **Exit Do***
- Begins with the **Do** keyword and ends with **Loop While** or **Loop Until**
- Similar to do while loops in other programming languages

Do
Insert user code here
Loop While (until) (test condition met/not met)

DO LOOP

- May prematurely leave the loop if desired using Exit keyword:
- Exit Do

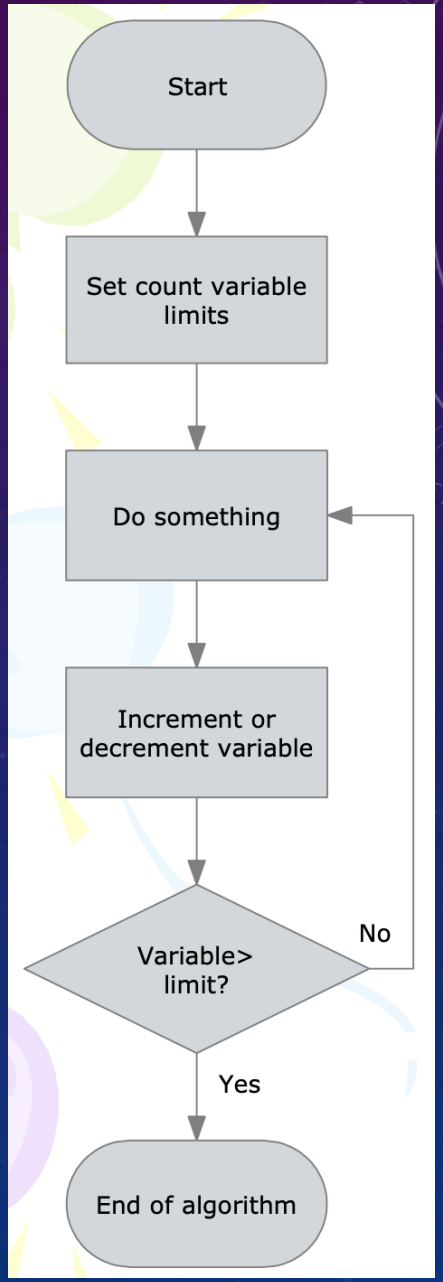
```
Private Sub Do_Demo(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    Dim intCounter As Integer    'declare variable
    intCounter = 0               'Initialize variable to 0 (what happens when =20?)

    Do
        intCounter = intCounter + 1 ' Increment variable
        txtDoResult.Text = txtDoResult.Text & intCounter & vbCrLf 'added for clarity in lecture
    Loop Until intCounter > 19  ' Test value of Counter

    MsgBox(intCounter)  ' Prints 20 in message box

End Sub
```



FLOWCHART - FOR NEXT LOOP

When the loop is exited, the count variable is one more count than the upper limit as the variable increment/decrement takes place on the ‘Next’ instruction.

Identifier	Description	Type
intCounter	Count variable for Loop	Integer



FOR NEXT STEPPING

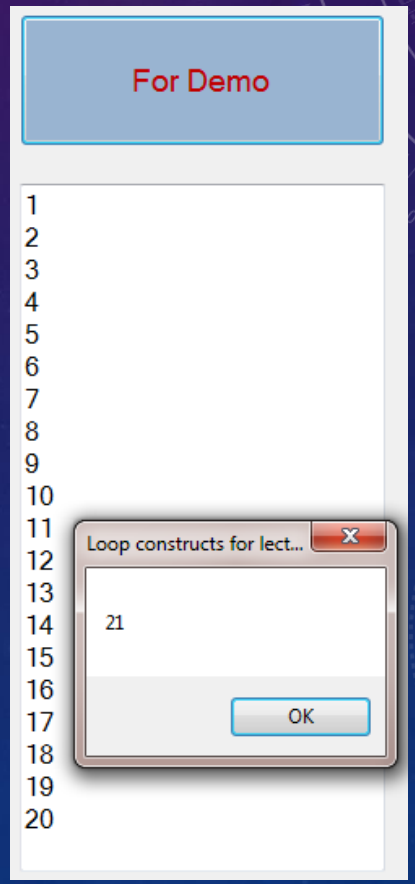
- To decrement use **Step** argument with negative values *eg. step -1*
- Step sets size of increment – can be decimals e.g. step 0.1
- *User code will be run 10 times*

```
For intCounter = 18 to 0 Step -2
    Insert user code here
Next
```

FOR NEXT LOOP

- May prematurely leave the loop if desired using Exit keyword:
- Exit While

```
Private Sub For_Demo(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    Dim intCounter As Integer    'declare variable

    For intCounter = 1 To 20
        txtForResult.Text = txtForResult.Text & intCounter & vbCrLf 'added for clarity in lecture
    Next

    MsgBox(intCounter)  ' Prints 21 in message box
End Sub
```



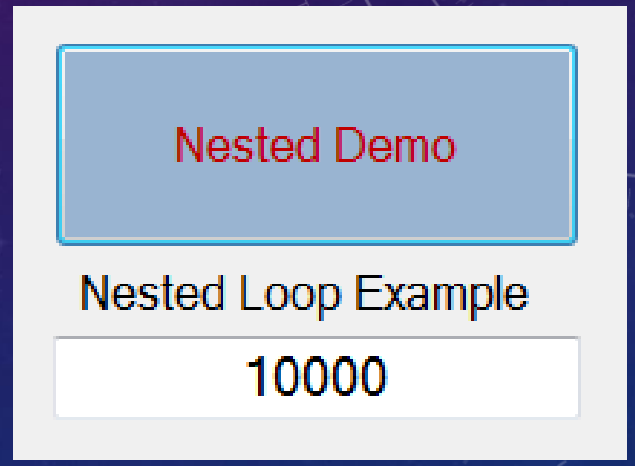
PROBLEMS WITH LOOPING

- When iterating, processor load = 100%
- Windows cannot perform house-keeping tasks – update screen etc.
- Correct way to handle is multi-threading in your programs: can do, but complicated!!
- Application.DoEvents releases system for housekeeping during looping

```
Do
Some code repeatedly
Application.DoEvents 'let windows do its thing Loop
While (condition not met)
```

NESTED LOOPS

```
Private Sub Nested_Demo(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
    Dim intInner, intOuter, intTotal As Integer

    For intOuter = 0 To 99
        For intInner = 0 To 99
            intTotal = intTotal + 1   'count number of times this line is run
        Next
    Next

    txtNestedLoop.Text = intTotal
End Sub
```

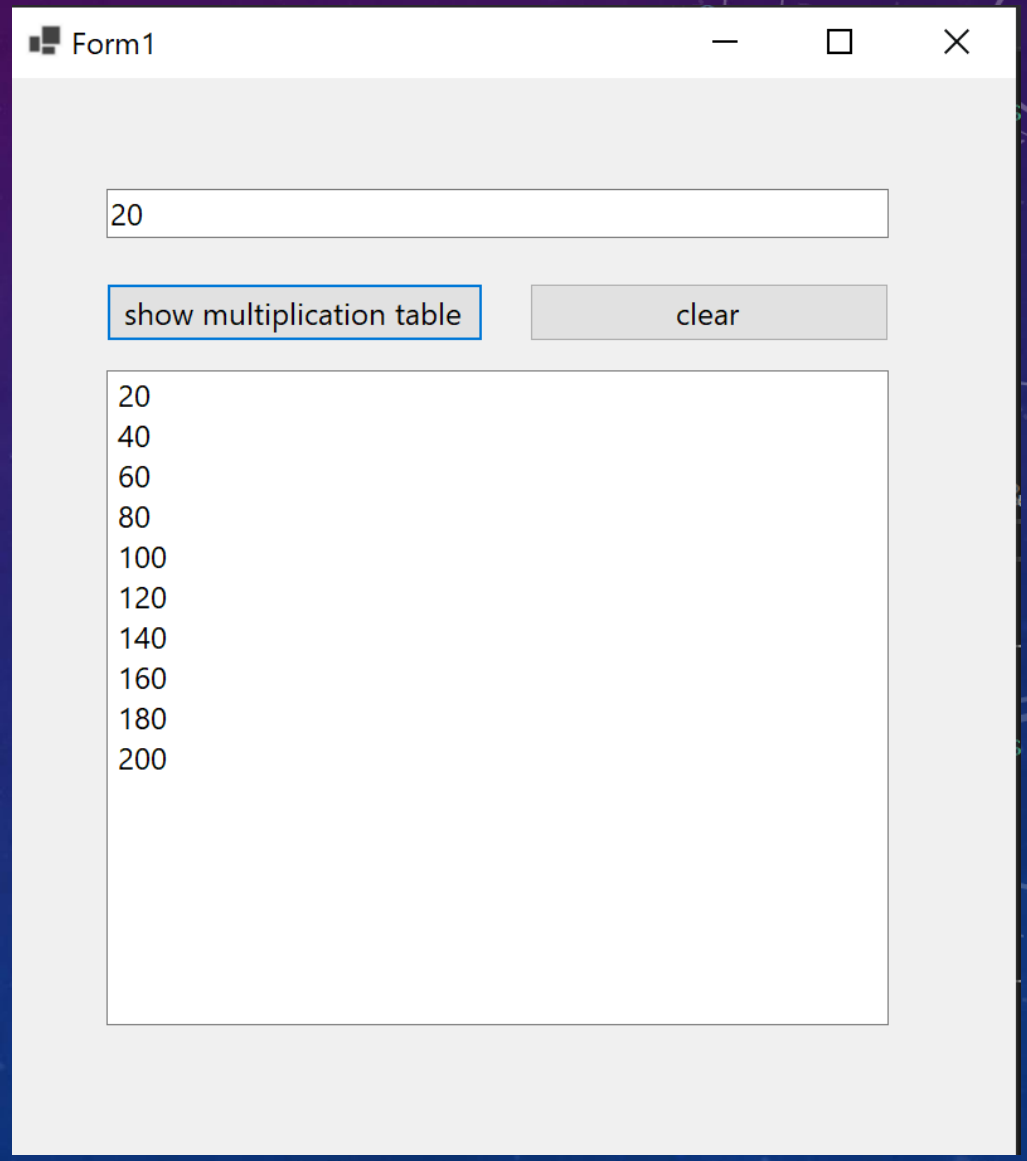


Loops around 100 x 100 = 10,000 times!

EXERCISE (15 – 20 MINUTES)

To create a mini application

1. We are going to create a multiplication table
2. When I click the multiplication table it should generate the 1st 10 numbers of the multiplication table for this number.
3. When I click clear it should clear all the values inside the table.
4. We assume only valid numbers to be added into the textbox.



HEADS UP FOR YOUR EXERCISE

You need to convert the datatype inside the textbox from string to integer in order for this project to work.

Do use the reference link below to help you get started.

https://www.convertdatatypes.com/Convert-Single-to-Integer-in-VB6-VBA.html