TEST 1 Overview

- Test will cover topics weeks 1-5
- Topics include:
  - Week 1
    - Broad overview of Computer Components
    - Algorithm design
    - Structure Theorem
    - Steps in writing an algorithm
    - Desk checking an algorithm
  - Week 2
    - Objects and Classes
    - Properties and Behaviours
    - VB.NET Controls
    - VB.NET IDE
    - Built a simple splash screen
  - Week 3
    - TOE Charts
    - Designing User Interface
    - In-built functions: Val(), Format()
    - Classes Convert and Random
    - Variables/Data Types
    - Literals and Constants
    - Option Explicit and Refs

Data Verification

- Checking that data is complete and relevant
- Usually performed prior to computer processing

Data Validation

- Performed by programmer
- Confirms the validity of the data inputs before processing
- Checks that
  - Data is present (not NULL)
  - Data is of the correct type
  - Data is within valid range of values
- Processing must not proceed with invalid data
- Programs must not crash as a result of invalid data inputs
Tools for data validation

- Decision structures
  - Test inputs
  - Control processing
- Type conversion functions
  - Val()
  - Convert.ToInt32(), Convert.ToSingle() etc.
- String conversion functions
  - aString.ToUpper(), aString.ToLower() etc.
- Data testing functions
  - IsNumeric()

Errors

- Programming is a human activity
  - Humans are error-prone
- Errors in coding are inevitable
  - Hopefully minimal occurrences
- Types of programming errors:
  - Syntax errors
  - Run-time errors
  - Logical errors

Syntax Errors

- Each programming language has specific syntax
  - Rules for constructing statements, expressions, code files, etc.
- Breaking the rules causes a syntax error
- Example:
  ```
  Public Sub message(birthYear as Integer)
  If birthYear > 1990 Then
    lblMessage.Text = "You are young"
  End Sub
  ```
  This will cause a syntax error because End If was expected before EndSub appears.

Run-time Errors

- Errors due to circumstances at time program is run
- Caused by program trying to do something "illegal"
- Examples:
  - Trying to divide by zero
  - Trying to open a non-existent file for reading
  - Running out of computer memory
- Usual response is program will crash and die:
  - "An exception has been caught ...
- Sometimes we can "trap" or capture these circumstances in order to gracefully recover from problem
  - Exception Handling (Try Catch blocks) – later in the unit

Logical Errors

- Program appears to run successfully to completion
- Result does not seem right
- Error due to malformed logical expressions
- Example:
  ```
  Write a loop that asks for numbers until 0 is entered, then print the average of those numbers.
  ```

Example Logical Error

```vbnet
Private Sub Button1_Click(...) Handles Button1.Click
  Dim sngNumber As Single, intCount As Integer = 0
  Dim sngTotal As Single = 0, sngResult As Single
  ' Repeatedly ask for a number
  ' and add it to the total
  Do
    sngNumber = Val(InputBox("Type a number"))
    sngTotal = sngTotal + sngNumber
    intCount = intCount + 1
  Loop Until (sngNumber = 0)
  ' Determine average, and print result
  sngResult = sngTotal / intCount
  lblResult.Text = sngResult.ToString()
End Sub
```
Debugging Tools in VB.NET

- Visual Basic provides tools for debugging
- Ensure the 'Debug' option is selected in the drop-down box beside the play button.
- When you click 'Play', or select 'Start' from the Debug menu, your code begins running in debugging mode.
  - You can interrupt it (break it) by selecting 'Break All' from menu, or
  - You can specify where you want the code to break.
  - If an error occurs you are given the option to break at the offending line of code.

Breakpoints
- Points in the code at which you want to interrupt normal program execution
- Indicated by a red dot beside the code, and red-highlighted line
- Able to be pre-set by clicking where the red dot should appear.

Watches
- Allow you to 'watch' how a variable or expression changes as execution progresses
- Right-click on variable and select "Add Watch" (when interrupted)
- Watches-window shows all watches, and updates whenever a watched variable changes.
- Select "Watch Window" from "View" menu if not visible

Step Through The Code
- Code-Stepping useful to examine control flow
  - E.g. See what branch of an IF THEN ELSE is executed
  - See how many times loops are executed
- To step through the code statement by statement:
  - After reaching a breakpoint, press F11 (or select Debug->Step Into) to run the line and stop again.
  - To continue running without stepping, press F5 or Play button

Example Logical Error

Private Sub Button1_Click(...) Handles Button1.Click
Dim sngNumber As Single, intCount As Integer = 0
Dim sngTotal As Single, sngResult As Single
' Repeatedly ask for a number
' and add it to the total
' and increment count of component values
Do
  sngNumber = Val(InputBox("Type a number"))
  sngTotal = sngTotal + sngNumber
  intCount = intCount + 1
Loop Until (sngNumber = 0)
' Determine average, and print result
sngResult = sngTotal / (intCount-1)
lblResult.Text = sngResult.ToString()
End Sub

Because '0' was treated as a component, it caused count to be incremented.
So we need to exempt it from the calculation.

Errors and Debugging - Summary

- Types of Errors
  - Syntax, runtime, logical
- Debugging Tools in VB.NET:
  - Code editor built into IDE reports syntax errors
  - Add Breakpoints (F9)
  - Stepping through code with (F11) or (F10)
  - Use watches to see specific values of variables
- Reading
  - Unit Guide, Study Guide 5, p89-93
  - Zak, p97-99