Lecture Outline

Repetition (looping)
- The For Next Loop
- The Do While Loop
- The Repeat Until Loop

Control Structures
- The Structure Theorem (from Week 1): any computer program can be built from just three control structures:
  - Sequence
  - Selection
  - Repetition – This Study Guide
- Used to determine (control) the order in which the steps in an algorithm are to be executed

Repetition Structures/Loops
- Some problems require a sequence of tasks to be processed repeatedly, e.g.:
  - apply same processing to a set of different data items
  - processing until and target point is reached
- Repetition structures (aka loops, iteration structures) cater for this

Repetition structures have …
- A block of statements that are to be repeated
- A condition that will determine if the loop processing should stop
  - the condition is evaluated once in each iteration
- A variable that will change each time the loop is processed
  - tested in the condition
The Pre-Test Loop

- Condition is tested PRIOR to the execution of the statement block
- If the condition is TRUE:
  - Perform the statement block (perform one iteration)
  - Go back to the condition (to check again)
- If the condition is FALSE:
  - “Exit” from the loop (do not perform any more iterations)
- The loop may potentially not be executed:
  - if condition fails on first test

```
number = 0
DOWHILE number < 10
   display number
   number = number + 1
ENDDO
```

```
DOWHILE a condition evaluates to TRUE
statement block
ENDDO
```

DO WHILE Loops

- If the condition evaluates to FALSE initially, no iterations are performed
- To ensure at least one iteration, variables tested in condition need to be initialised prior to the loop
- At least one variable tested in the condition must change within the statement block
  - otherwise endless loop ("infinite" loop)

```
DOWHILE - Algorithm
```

```
DOWHILE - Problem Example
```

```
DOWHILE - Solution Algorithm
```

DO WHILE Loops

- If the condition evaluates to FALSE initially, no iterations are performed
- To ensure at least one iteration, variables tested in condition need to be initialised prior to the loop
- At least one variable tested in the condition must change within the statement block
  - otherwise endless loop ("infinite" loop)

```
DOWHILE - Algorithm
```

```
DOWHILE - Problem Example
```

```
DOWHILE - Solution Algorithm
```

Defining Diagram

```
<table>
<thead>
<tr>
<th>Input</th>
<th>Process</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>fahrenheit</td>
<td>Read fahrenheit</td>
<td>calculate celsius</td>
</tr>
<tr>
<td></td>
<td>Display celsius</td>
<td></td>
</tr>
</tbody>
</table>

Sometimes called an IPO chart (for Input, Processing, Output)
DOWHILE - Desk Check

Input data:

Test value 1: \( f_{\text{temp}} = 32 \),
\[ c_{\text{temp}} = (f_{\text{temp}} - 32) \times \frac{5}{9} = 0 \]
Expected outcome: \( c_{\text{temp}} = 0 \)

Test value 2: \( f_{\text{temp}} = 50 \)
\[ c_{\text{temp}} = (f_{\text{temp}} - 32) \times \frac{5}{9} = 10 \]
Expected outcome: \( c_{\text{temp}} = 10 \)

DO WHILE – Desk Checking

<table>
<thead>
<tr>
<th>Statement</th>
<th>Temp_count</th>
<th>f_temp</th>
<th>c_temp</th>
<th>DOWHILE condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DO WHILE</td>
<td>TRUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compute</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display</td>
<td>Display</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DO WHILE</td>
<td>TRUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display</td>
<td>END</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Common Traps with Loops

- Forgetting to give initial values to loop counters
- Forgetting to change the condition variables inside the loop

The Post-Test Loop

- The condition is tested at the END of each iteration of the loop
  - At least one iteration will be performed
  - If the condition evaluates to FALSE, another iteration will be executed
  - If the condition is TRUE, exit from the loop
- Expressed by:
  - `REPEAT ... UNTIL` loops (for algorithms)
  - `Do ... Loop Until` (for Visual Basic .NET)

REPEAT UNTIL - Algorithm

REPEAT
<table>
<thead>
<tr>
<th>statement block</th>
</tr>
</thead>
</table>
UNTIL a condition evaluates TRUE

For example:
number = 0
REPEAT
<table>
<thead>
<tr>
<th>Display number</th>
</tr>
</thead>
</table>
Increment number by 1
UNTIL number = 10
A Repeat Until in an algorithm becomes a DO ... LOOP UNTIL in VB.NET code:

Do
    Statements
Loop Until condition

Counted loops
- Execute the statement block a pre-determined number of times
  - Number of iterations known in advance
- A control variable keeps count of the number of repetitions
  - No need to change this explicitly in code
- May use i, j & k as control variables (historical)
  - Meaningful names better
- Do ... End Do (algorithms)
- FOR...NEXT loops (VB.NET)

Counted Loops - Problem Example
Fahrenheit - Celsius conversion (Robertson, p.52)
Everyday, a weather station receives 15 temperatures expressed in Fahrenheit. A program is to be written which will accept each Fahrenheit temperature, convert it to Celsius and display the converted temperature to the screen. After 15 temperatures have been processed, the words 'All temperatures processed' are to be displayed on the screen.

Counted Loops - Solution Algorithm

Fahrenheit_Celsius_conversion
DO temperature_count = 1 to 15
    Prompt operator for f_temp
    Get f_temp
    Compute c_temp = (f_temp - 32) * 5/9
    Display c_temp
END DO
Display 'All temperatures processed' to the screen
END

FOR NEXT – VB.NET Syntax
For counter_var = m to n
    statement block
Next counter_var

DO ... END DO - Algorithm

FOR...NEXT – VB.NET Syntax
For counter_var = m to n step s
    statement block
Next counter_var
Hints for Programming with Loops

- Minimise the steps inside any loop
  - Keep It Simple!
- Make each loop perform one function
- Document (Comment) the loop by:
  - Stating the conditions for which the loop executes, or
  - Explicitly stating the termination conditions
- Enter the loop from only one location
- Leave the loop only by satisfying the exit condition
  - Structured programming requires this

Repetition – Summary

- Pre-test loops: Do While ...
- Post-test loops: Repeat ... Until
- Counted loops: For ... Next

You have now learnt all the elements of the Structure Theorem.

Questions/Reading

- Questions
  - Hammer in the key concepts
  - Pull out those questions
  - List the three types of loops and when they should be used.
  - Do you know the syntax of each loop?
- Reading
  - Unit Guide, Study Guide 5
  - Zak, Chpt 5
  - Robertson Chpt 5

TEST 1 (Next week) Overview

- Test will cover topics weeks 1-5
- Topics include:
  - Week 1
    - Algorithm design
    - Structure Theorem
    - Steps in writing a program
    - Desk checking an algorithm
  - Week 2
    - Properties and Behaviours
    - VB Objects and Controls
  - Week 3
    - TOE Charts
    - In-built functions: Val(), Format()”
    - Classes Convert and Random
    - Variables/Data Types
    - Literals and Constants
  - Week 4
    - Selection
    - If statements
    - Case statements
  - Week 5
    - Repetition (looping)
    - The For Next Loop
    - The Do While Loop

TEST 1 Overview

- Test will be 50 mins long
- Counts for 10% of total mark
- Question types
  - Short answer written responses
  - Writing pseudo code
  - Develop TOE chart + User Interface
  - VB.NET coding questions
- Wishing you all the success you deserve!
- Reminder TEST 1 starts at 10am next week!