Repetition Structures

There are two types repetition structures, code that repeats the execution of the next block of code in specific situations. They are FOR loops and DO loops.

FOR LOOPS: These are counted loops. The pattern for writing FOR loops is:

```
FOR counter = StartValue To EndValue [Step increment]
        Block of code to repeat
NEXT [counter]
```

The data in square brackets is optional, is not required to be in the code.

If you do not specify the step value, the default behavior is to add 1 to the counter.

On the last line, NEXT is required, and tells the computer to change the counter to the next value. It is not necessary to list the name of the counter variable, but it is good programming practice, as it is a form of documentation. If there many lines in the block of code that is repeating, it can be very useful to be reminded of the name of the counter at the bottom.

When a FOR loop starts, the counter is assigned the StartValue and the block of code runs. When the computer gets to NEXT, it adds the step value to the counter. Then it checks to see if the counter is beyond the end value. The end value is like a boundary or fence – the counter may get to the fence and even sit on it, but it may not go beyond the fence. The counter may not hit the final value exactly, and that's okay. Look at some examples:

```
FOR j = 1 to 5 step 1
        ... code ...
Next
```

Loop 1: j = 1, this is not beyond the end value of 5 so it runs, adds 1 to j when it gets to "Next"
Loop 2: j = 2, this is not beyond the end value of 5 so it runs, adds 1 to j when it gets to "Next"
Loop 3: j = 3, this is not beyond the end value of 5 so it runs, adds 1 to j when it gets to "Next"
Loop 4: j = 4, this is not beyond the end value of 5 so it runs, adds 1 to j when it gets to "Next"
Loop 5: j = 5, this is not beyond the end value of 5 so it runs, adds 1 to j when it gets to "Next"
Loop 6: j = 6, this is beyond the end value of 6, so it does not run, loop is done
The loop executed 5 times, the final value of the counter is 6. Note that the final iteration was loop #5, when the counter was the same value as the end value.

```
FOR j = 1 to 10 step 2
        ... code ...
Next
```

Loop 1: j = 1, this is not beyond the end value of 10 so it runs, adds 2 to j when it gets to "Next"
Loop 2: j = 3, this is not beyond the end value of 10 so it runs, adds 2 to j when it gets to "Next"
Loop 3: j = 5, this is not beyond the end value of 10 so it runs, adds 2 to j when it gets to "Next"

Loop 4: j = 7, this is not beyond the end value of 10 so it runs, adds 2 to j when it gets to “Next”
Loop 5: j = 9, this is not beyond the end value of 10 so it runs, adds 2 to j when it gets to “Next”
Loop 6: j = 11, this is beyond the end value of 10, so it does not run, loop is done
The loop executed 5 times, the final value of the counter is 11. Note that the final iteration was loop #5, when the counter was less than the end value. The counter does not have to equal the end value exactly in the last iteration of the loop.

If there is a question about how many times a For loop repeats, write out a desk check like above. The first iteration is the initial value of the counter.

One of the best reasons to use a loop is the use of the counter in the code of the loop.

```
Dim intAnswer, j as Integer
For j = 1 To 5
      intAnswer += j
Next j
```

Desk check:
Loop 1: j = 1, that is not beyond the end value of 5 so it runs
intAnswer += j resolves to intAnswer = 0 + 1 = 1
j increments by 1, j = 2
Loop 2: j = 2, that is not beyond the end value of 5 so it runs
intAnswer += j resolves to intAnswer = 1 + 2 = 3
j increments by 1, j = 3
Loop 3: j = 3, that is not beyond the end value of 5 so it runs
intAnswer += j resolves to intAnswer = 3 + 3 = 6
j increments by 1, j = 4
Loop 4: j = 4, that is not beyond the end value of 5 so it runs
intAnswer += j resolves to intAnswer = 6 + 4 = 10
j increments by 1, j = 5
Loop 5: j = 5, that is not beyond the end value of 5 so it runs
intAnswer += j resolves to intAnswer = 10 + 5 = 15
j increments by 1, j = 6
Loop 6: j = 6, that is beyond the end value of 5 so the loop is done
When the loop ends, j = 6, intAnswer = 15, and it iterated 5 times.

Error: you can write a FOR loop so that it never ends. This is most often because the Step value does not move the counter closer to the end value.
Look at this example: For counter = 15 to 1 step 2 – the counter needs to decrease to reach the end value, but the step is increasing the counter. A desk check would show this.
Loop 1: counter = 15, it is not beyond the end value of 1 so it runs, adds 2 to the counter
Loop 2: counter = 17, it is not beyond the end value of 1 so it runs, adds 2 to the counter
Loop 3: counter = 19 .... the counter is not getting closer and closer to the end value, and will never reach the point where the loop ends.

DO LOOPS: These are conditional loops. There are 4 different versions of Do loops. The loop repeats WHILE a condition is true or UNTIL a condition becomes true. That condition is tested either at the top, before entering the loop, or at the bottom, after completing the loop.

Do While, pretest	Do Until, pretest
Tested variable = initial value DO While (condition is true) ... code ... Tested variable = some new value LOOP	Tested variable = initial value DO Until (condition is true) ... code ... Tested variable = some new value LOOP
Do While, post-test	**Do Until, post-test**
Tested variable = initial value DO ... code ... Tested variable = some new value LOOP While (condition is true)	Tested variable = initial value DO ... code ... Tested variable = some new value LOOP Until (condition is true)

In both Do loops and For loops, the tested variable is initialized, changed, and tested. The code to do that work differs based on the type of loop.

	For Loop	Do Loop
Initialize tested variable	For counter = **initial value** to end value	Tested variable = initial value (above the loop)
Change tested variable	Next (adds step value to counter, no additional explicit coding needed)	Tested variable = some new value (must be explicitly written in the code block of the loop)
Test variable	For counter = initial value to **end value** (if counter is beyond the end value, loop ends)	While condition is true or Until condition is true (tested at top or bottom of loop)

Do loops are not limited to control variables that are numbers. The loop iterates while a condition is true or until a condition becomes true. When testing for true values, use the relational operators (=, <>, >, <, <=, >=), exactly like IF selection structures. Here are some examples of the conditional statements using numbers, strings, and Boolean variables.

Do While intAnswer < 100
Do While intAnswer > 1 and intInput < 25
Do Until blnGoodData = true
Do While blnGoodData = false
Do While strState = “UT”
Do Until strCity = “Kaysville”

Look at a desk check of a Do loop.

```
Dim intCounter, intAnswer as Integer;
intCounter = 3
intAnswer = 0
Do While  intCounter < 6
     intCounter += 1
     intAnswer += intCounter
Loop
lblResult.Text = "The answer is " & intAnswer.ToString
```

Loop 1: intCounter = 3, intAnswer = 0, it is true that intCounter < 6, loop iterates
intCounter += 1 resolves to intCounter = 3 + 1 = 4
intAnswer += intCounter resolves to intAnswer = 0 + 4 = 4
Loop 2: intCounter = 4, intAnswer = 4, it is true that intCounter < 6, loop iterates
intCounter += 1 resolves to intCounter = 4 + 1 = 5
intAnswer += intCounter resolves to intAnswer = 4 + 5 = 9
Loop 3: intCounter = 5, intAnswer = 9, it is true that intCounter < 6, loop iterates
intCounter += 1 resolves to intCounter = 5 + 1 = 6
intAnswer += intCounter resolves to intAnswer = 9 + 6 = 15
Loop 4: intCounter = 6, intAnswer = 15, it is false that intCounter < 6, loop ends
Next line of code after the bottom of the loop executes and changes the text in the label
intAnswer = 15, so the label displays "The answer is 15"

Be sure to keep track of the sentinel value, the variable being tested in the condition. In the example above, it's easy to look at the value of intAnswer in loop 2, which was set to 9, and think the loop must end now because that value is not less than 6. But the condition is testing intCounter, not intAnswer.

Another frequently seen issue with checking a Do loop is figuring out when to stop. If the condition is still valid, the loop iterates, even though the sentinel value is changed inside that loop. In the example above, in loop 3, the intCounter variable changes to 6 – that does not stop the loop at that point. The loop will stop when it tests its condition again. So once in the loop, all lines of code in the block of code will execute fully. Then the condition is tested to decide whether there should be another loop.

Error: It is very easy to write an infinite Do loop by forgetting to write the code that changes the control variable. In a For loop, the control variable auto-magically changes each time the keyword "Next" executes. In a Do loop, you must write a specific line of code to change the control variable.