Chapter 6
More About Loops and Decisions

6.1 Combining Loops With `If-Then` Statements

By combining loops and decision structures, programs become much more complex.

Loops can be nested inside selection structures and selections can be nested inside loops.

Loops can be nested inside other loops.

Using multiple combinations of these structures allows for limitless possibilities!

Exiting a Loop Early

One reason to nest a selection structure inside a loop is to allow the loop to end before the test condition has been met for any reason. For example:

- If the user has entered an incorrect value that would cause an error
- If the user has entered a required response and the program can continue without further iterations

Example: Exiting the Loop When There's No More Money

```
1     Declare Cost, Total, Max As Float
2     Declare Count As Integer
3     Set Total = 0
4     Write "Enter the maximum amount you want to spend: $ "
5     Input Max
6     For (Count = 1; Count < 11; Count++)
7           Write "Enter the cost of an item: "
8           Input Cost
9           Set Total = Total + Cost
10          If Total > Max Then
11                Write "You have reached your spending limit."
12                Write "You cannot buy this item or anything else."
13                Set Total = Total – Cost
14                Exit For
15          End If
16          Write "You have bought " + Count + " items."
17    End For
18    Write "Your total cost is $ " + Total
```


Example: The Guessing Game using a `Do…While` loop

```
1     Declare SecretNumber, Count, Guess As Integer
2     Write "Enter a secret number: "
3     Input SecretNumber
4     Clear Screen
5     Set Count = 1
6     Do
7           Write "Guess the secret number: "
8           Input Guess
9           If Guess == SecretNumber Then
10                Write "You guessed it!"
11                Exit Loop
12          Else
13                Write "Try again"
14          End If
15          Set Count = Count + 1
16    While Count <= 5
```


Example: Combining Loops, Decisions, and Validating Data

Computing valid square roots

```
1     Declare Number, Root As Float
2     Declare Response As Character
3     Write "Do you want to find the square root of a number?"
4     Write "Enter 'y' for yes, 'n' for no: "
5     While Response == "y"
6           Write "Enter a positive number: "
7           Input Number
8           If (Number >= 0) Then
9                 Set Root = Sqrt(Number)
10                Write "The square root of " + Number + " is: " + Root
11          Else
12                Write "Your number is invalid."
13          End If
14          Write "Do you want to do this again?"
15          Write "Enter 'y' for yes, 'n' for no: "
16          Input Response
17    End While
```

6.2 Combining Loops and Decisions in Longer Programs

The example in the next slide shows how to keep track of how many positive numbers and how many negative numbers are input by a user.

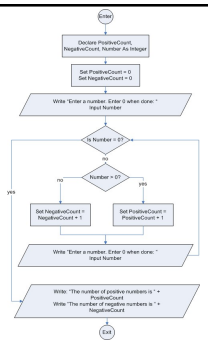
Uses for this type of program:

- Embed in a larger program to track various types of entries
- Use by a college to enter demographic information on students
- Use by a business to track number of items purchased that were above or below a certain cost

Example: Keeping Track of User Input

```
1     Declare PositiveCount, NegativeCount, Number As Integer
2     Set PositiveCount = 0
3     Set NegativeCount = 0
4     Write "Enter a number. Enter 0 when done: "
5     Input Number
6     While Number != 0
7           If Number > 0 Then
8                 Set PositiveCount = PositiveCount + 1
9           Else
10                Set NegativeCount = NegativeCount + 1
11          End If
12          Write "Enter a number. Enter 0 when done: "
13          Input Number
14    End While
15    Write "Positive numbers entered: " + PositiveCount
16    Write "Negative numbers entered: " + NegativeCount
```


Flowchart using a selection structure with a loop to keep track of the number of positive and negative numbers entered



The `Length_Of()` Function

The `Length_Of()` function takes a string or a string variable inside the parentheses and returns the number of characters in that string.

- **`MyLength`** = `Length_Of("Hello")` assigns the value of `5` to **`MyLength`** because `"Hello"` has five characters.
- **`MyLength`** = `Length_Of("Good-bye!")` assigns the value of `9` to **`MyLength`** because the string has nine characters, including the hyphen and exclamation point.

If **`Name`** = `"Hermione Hatfield"` then:

- **`MyLength`** = `Length_Of(`**`Name`**`)` assigns the value of `17` to **`MyLength`**.

If **`TheSpace`** = `" "` then:

- **`MyLength`** = `Length_Of(`**`TheSpace`**`)` assigns the value of `1` to **`MyLength`** because a space counts as one character.

The `Print` Statement and the `newline` Indicator <NL>

- The **`Write`** statement indicates output to the screen with the assumption that each new **`Write`** statement would begin on a new line.
- The **`Print`** statement indicates output to the screen, including the ability to concatenate variables and text. However, until the **`newline indicator`** is used, it is assumed that output from any subsequent **`Print`** statements will be on the same line. The newline indicator is **`<NL>`**.

Code	Code
`Write "Hi"`	`Print "Hi"`
`Write "Ho"`	`Print "Ho" <NL>`
`Write "Done"`	`Print "Done"`

Display	Display
`Hi`	`HiHo`
`Ho`	`Done`
`Done`	

Using the `Length_Of()` Function for Formatting

```
1       Declare Name As String
2       Declare Symbol, Choice As Character
3       Declare Number, Count As Integer
4       Set Count = 0
5       Write "Enter your name: "
6       Input Name
7       Write "Choose one of the following symbols: * or # "
8       Input Symbol
9       Write "Do you want a space between each symbol? Enter 'Y' for yes, 'N' for no"
10      Input Choice
11      Set Number = Length_Of(Name)
12      Print Name <NL>
13      While Count <= Number
14              If Choice == "y" OR Choice == "Y" Then
15                      Print Symbol + " "
16                      Set Count = Count + 2
17              Else
18                      Print Symbol
19                      Set Count = Count + 1
20              End If
21      End While
```

6.3 Random Numbers

- **Random numbers** are numbers whose values form an unpredictable sequence.
- They have many interesting applications in programming.
- One major use is to provide an element of chance in **computer games.**
- They are also used to **simulate situations** or **processes** in business, mathematics, engineering, and other disciplines.

The `Random()` Function

- Most programming languages contain a function that is used to generate a sequence of random numbers.
- The name of this function and the way it works varies from language to language.
- We define a function of the following form: **`Random()`** which generates a random number from **`0.0`** to **`1.0`** (including **`0.0`** but not **`1.0`**).
- To increase the range of random numbers generated, multiply **`Random()`** by any value.
- To change the spread of the range, add an integer to the values generated.
- Use the **`Int()`** or **`Floor()`** function to generate random integers.

Examples of Random Numbers Using the `Random()` Function

- If `Random() = 0.3792`, then `Random() * 10 = 3.7920`
- If `Random() = 0.0578`, then `Random() * 10 = 0.5780`

Multiplying `Random()` by `10` generates random numbers between `0.0` and `10.0`, not including `10.0`.

To generate only integer values, use the `Floor()` or `Int()` function:

- If `Random() = 0.3792`, then `Floor(Random() * 10) = 3`
- If `Random() = 0.0578`, then `Floor(Random() * 10) = 0`

To change the range of integers, add a number to the result. For example, adding `1` to the previous result will generate random numbers between `1` and `10`.

- If `Random() = 0.3792`, then `(Floor(Random() * 10) + 1) = 4`
- If `Random() = 0.0578`, then `(Floor(Random() * 10) + 1) = 1`

Generating Random Numbers in Various Ranges

If **NewNumber** is an integer variable, then:

- **NewNumber** = Floor(Random() * 10) + 1 will result in a random number between 1 and 10 (inclusive)
- **NewNumber** = Floor(Random() * 100) + 1 will result in a random number between 1 and 100 (inclusive)
- **NewNumber** = Floor(Random() * 10) + 4 will result in a random number between 4 and 13 (inclusive)
- **NewNumber** = Floor(Random() * 2) will result in either 0 or 1
- **NewNumber** = Floor(Random() * 2) + 1 will result in either 1 or 2

After examining these examples, we can conclude that, to generate a sequence of **N** random integers beginning with the integer **M**, use:

```
Floor(Random() * N) + M
```


Example: Flipping a Coin

```
1   Declare Number As Integer
2   Declare Response As Character
3   Write "Do you want to flip a coin? Enter 'y' for yes, 'n' for no:"
4   Input Response
5   While Response == "y"
6         Set Number = Floor(Random() * 2)
7         If Number == 1 Then
8               Write "Heads"
9         Else
10              Write "Tails"
11        End If
12        Write "Flip again? Enter 'y' for yes, 'n' for no: "
13        Input Response
14  End While
```


Example: Winning at Dice: What number should I bet on?

Possible ways to roll the 11 possible sums

1 + 1 = 2	2 + 1 = 3	3 + 1 = 4	4 + 1 = 5	5 + 1 = 6	6 + 1 = 7
1 + 2 = 3	2 + 2 = 4	3 + 2 = 5	4 + 2 = 6	5 + 2 = 7	6 + 2 = 8
1 + 3 = 4	2 + 3 = 5	3 + 3 = 6	4 + 3 = 7	5 + 3 = 8	6 + 3 = 9
1 + 4 = 5	2 + 4 = 6	3 + 4 = 7	4 + 4 = 8	5 + 4 = 9	6 + 4 = 10
1 + 5 = 6	2 + 5 = 7	3 + 5 = 8	4 + 5 = 9	5 + 5 = 10	6 + 5 = 11
1 + 6 = 7	2 + 6 = 8	3 + 6 = 9	4 + 6 = 10	5 + 6 = 11	6 + 6 = 12

Possible ways to roll a 5: (1,4), (4,1), (2,3), (3,2)

Possible ways to roll an 8: (2,6), (6,2), (3,5), (5,3), (4,4)

Example: Winning at Dice: What number should I bet on?

```
1     Declare FiveCount, EightCount, K, Die1, Die2, Sum As Integer
2     Set FiveCount = 0
3     Set EightCount = 0
4     For (K = 1; K <= 1000; K++)
5           Set Die1 = Floor(Random() * 6) + 1
6           Set Die2 = Floor(Random() * 6) + 1
7           Set Sum = Die1 + Die2
8           If Sum == 5 Then
9                 Set FiveCount = FiveCount + 1
10          End If
11          If Sum == 8 Then
12                Set EightCount = EightCount + 1
13          End If
14    End For
15    Write "Number of times sum was 5: " + FiveCount
16    Write "Number of times sum was 8: " + EightCount
```


The Algorithm for a Pseudorandom Number

- A computer doesn't understand, "Pick any number between 1 and 20."
- A computer must receive instructions from a program.
- Random numbers are often produced by means of a **mathematical algorithm.**
- A mathematical algorithm is a formula that instructs the computer how to pick some number in a specified range.
- An algorithm requires some beginning value to manipulate.
- This starting number is the **seed value.**

The Seed for a Pseudorandom Number

- If the same seed is used each time, the numbers generated are not really random. Such numbers are called **pseudorandom.**
- If the starting value of the algorithm never changes, the same sequence of numbers will be produced each time the program is executed. This may be useful for debugging purposes.
- The programmer must force the computer to use a different seed on each run so that the random numbers produced will be unpredictable.
  - use a seed that is not predetermined, like the number of milliseconds since the beginning of the current year
  - it will only occur once a year
  - forces a random number generator to start with a different seed each time

6.4 Nested Loops

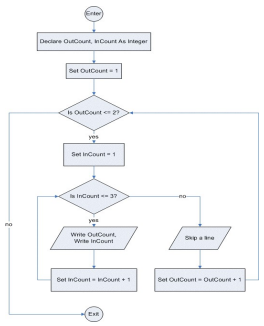
- When one loop is contained entirely within another we say that they are **nested loops**.
  - The larger loop is called the **outer loop**
  - The one lying within it is called the **inner loop**
- It is often difficult to follow the logical sequence of steps when nested loops are implemented.
- It is very important to be able to walk through (**desk check**) the pseudocode with paper and pencil, carefully writing down the values of each variable at each step.

Example

```
1   Declare OutCount As Integer
2   Declare InCount As Integer
3     For (OutCount = 1; OutCount <=2; OutCount++)
4        For (InCount = 1; InCount <=3; InCount++)
5           Write OutCount + ": " + InCount
6        End For(InCount)
7        Write " "
8     End For(OutCount)
```

OutCount	InCount	output
1	1	
1	2	
1	3	
1	4	
2	1	
2	2	
2	3	
2	4	
3		

Flowchart for the nested loops on the previous slide



Nesting Other Kinds of Loops:

Any style of loop may be nested. Each nested loop must be indented to indicate which statements are controlled by which looping construct.

```
Do
     Code Statements
     While <condition>
          More code may be here
          For (initial condition; test; increment)
               more code here
          End For
          More code may be here
     End While
     More code may be here
While <condition>
```


Example: Drawing Squares

```
1     Declare Count1, Count2, Side As Integer
2     Declare Symbol As Character
3     Write "Choose a symbol (any character from the keyboard):"
4     Input Symbol
5     Write "Enter the length of a side of the square: "
6     Input Side
7     Set Count1 = 1
8     Set Count2 = 1
9     While Count1 <= Side
10          While Count2 <= Side
11               Print Symbol
12               Set Count2 = Count2 + 1
13          End While
14          Print <NL>
15          Set Count2 = 1
16          Set Count1 = Count1 + 1
17    End While
```


Mind Games Workout #1

```
1     Declare X, Y, Z As Integer
2     For (X = 1; X < 4; X++)
3          Write "Pass Number " + X
4          For (Y = 1; Y < 10; Y+3)
5               Set Z = X + Y
6               Write X + " + " + Y + " = "+ Z
7          End For(Y)
8     End For(X)
```

X	Y	Z	output
1	?	?	Pass Number 1
1	1	2	1 + 1 = 2
1	4	5	1 + 4 = 5
1	7	8	1 + 7 = 8
1	10	8	
2	10	8	Pass Number 2
2	1	3	2 + 1 = 3
2	4	6	2 + 4 = 6
2	7	9	2 + 7 = 9
2	10	9	
3	10	9	Pass Number 3
3	1	4	3 + 1 = 4
3	4	7	3 + 4 = 7
3	7	10	3 + 7 = 10
3	10	10	
4	10	10	

Mind Games: Workout # 2

```
1   Declare X, Y, Z As Integer
2   Declare Count1 As Integer
3   Declare Count2 As Integer
2  Set Y = 3
3  Set Count1 = 1
4  Do
5       Set X = Count1 + 1
6       Set Count2 = 1
7       Write "Pass Number " + Count1
8       While Count2 <= Y
9          Set Z = Y * X
10         Write "X = " + X + ", Y = " + Y + ", Z = " + Z
11         Set X = X + 1
12         Set Count2 = Count2 + 1
13      End While
14      Set Count1 = Count1 + 1
15 While Count1 < Y
```

X	Y	Z	Count1	Count2	output
?	3	?	1	?	
2	3	?	1	1	Pass Number 1
2	3	6	1	1	X = 2, Y = 3, Z = 6
3	3	6	1	2	
3	3	9	1	2	X = 3, Y = 3, Z = 9
4	3	9	1	3	
4	3	12	1	3	X = 4, Y = 3, Z = 12
5	3	12	1	4	
And so on...					

Mind Games: Workout #3

```
1     Declare A, B, C As Integer
2     Set A = 1
3     Write "Cheers!"
4     While A < 3
5           Set C = A
6           Write A
7           Set B = 1
8           While B < 4
9                 Set C = C + A
10                Write C
11                If (A == 1 AND C >= 4) Then
12                      Write "Let's do this some more!"
13                Else
14                      If (A == 2 AND C >= 8) Then
15                            Write "Who do we appreciate?"
16                      End If
17                End If
18                Set B = B + 1
19          End While
20          Set A = A + 1
21    End While
```


Mind Games: Workout #3 Follow Along…

A	B	C	output
1	?	?	Cheers!
1	?	1	1
1	1	1	
1	1	2	2
1	2	2	
1	2	3	3
1	3	3	
1	3	4	4
1	3	4	Let's do this some more!
1	4	4	
2	1	2	2
2	1	4	4
2	2	4	
2	2	6	6
2	3	6	
2	3	8	8
2	3	8	Who do we appreciate?
2	4	8	
3	4	8	