

VBScript: Condition Statement

In this chapter, you will learn how to use the following VBScript functions to World Class standards:

1. **Writing Math Equations in VBScripts**
2. **Beginning a New VBScript Program**
3. **Adding Copyright Statement to a Program**
4. **Using a Message Box to Communicate the Copyright and More**
5. **Declaring Variables in a Program with the Dimension Statement**
6. **Setting Variables in a Program**
7. **Determining a String Length with the Function Len**
8. **Using a Loop with the Do While Function**
9. **Examining a Single Character at a Time with the Left Function**
10. **Changing a Character to ASCII with the ASC Function**
11. **Testing a Case with the IF – Then Function**
12. **Removing a Single Character at a Time with the Mid Function**
13. **Adding One to the Counter in the Loop**
14. **Using More IF – Then Functions**
15. **Ending the Program**
16. **Saving the Program**
17. **Running the Program**

Writing Condition Statements in VBScript

When a person comes to an intersection in a road, they have a decision to make whether to turn right or to turn left or maybe just go straight ahead. In programming, we need to make the same type of decision based upon a test. Back at the road, if we wanted to go to the store, possibly we would turn to the right. If we wanted to visit a friend, then maybe we will choose the left turn or finally we want to head on home and we go straight. Back to the program, in this chapter, we will learn how to write if-then statements to test whether a password meets the complexity level and the string length to meet the strong password criteria.

The definition of a strong password is that the string of characters needs to meet three of the following conditions. At least one character needs to be uppercase. At least one character needs to be lower case. At least one character needs to be a number. At least one character should be a special character such as question mark, exclamation mark, dash or And sign (? , ! , - , &). The password should be at least seven characters in length. Meeting the length requirement and three out of the four types of characters conditions make the string of text a strong password.

A method that we can use in doing the test is to remove a single character at a time from the password string and make four tests, one for each condition. By passing a single test, we will change a variable named for that particular test to a “1” or On state. For example, if our password is “WorldClassCAD” and we remove the capital **W**, in one test we would say if the **W** falls between the capital A and capital Z, then we will change the variable named Uppercase from zero to one. Below, we show an example of this code.

```
'Test for uppercase  
If holder>=65 and holder<= 90 then uppercase = 1 End If
```

We will test all four conditions on each letter, knowing that only one condition will change. As we continue with our testing in another example, we will test the next letter, the **o** from the password “WorldClassCAD”. In one test for the letter **o**, we would say if the **o** falls between the lowercase a and lowercase z, then we will change the variable named Lowercase from zero to one. As we continue testing through the password, we can see only the Uppercase and Lowercase variables will change to the On state and the other two variables, Special and Number will remain zero or Off. Even though the password “WorldClassCAD” has thirteen characters and is strong in length, however this password is not strong because it has only met two out of four character conditions.

We will use **if-then** statements to construct the text to announce whether the string of characters does or does not meet the strong password criteria. If we find a password to be lacking in one or more conditions, we will use the same type of decision statement to construct a sentence like **"Make at least one character uppercase"**. We will concatenate sentences together then broadcast them using a message box. This program is practical and helps us to develop specialized skills and since there are multiple lines of code in this project that gives us a better chance to remember this useful skill. The **if-then** statement will become a cornerstone function and every one of our programs throughout a career will likely use this function.

We still will use message boxes, declare variables, and assign values to variables just as we did in the previous chapters. We want to start the learning process using the skills we learned in chapter 2 and 3, so in this chapter, we will start first by using those message and input boxes. So let's get started.

Beginning a New VBScript Program

With the EditPad Lite program open, select **File** on the Menu Bar and select **Save**. We will save the file in the folder on either the desktop or the My Documents folder called "VBScripts". After locating the folder, name the file "password checker.vbs". The file extension *vbs* as can be easily seen means Visual Basic Script. We can run a finished VBScript file with the *vbs* extension at the command line by going to the Start button, select Run, and then Browse to locate the script program. When the plain text is compiled at runtime, the program will execute our commands in precise order.

If we want to change the font type or size, select Options on the Menu Bar and pick **Font**. The Font window will appear and we can select the font, font style and size. We will select Arial Narrow, Regular font style and 12 point size for this textbook. We suggest that programmers select text shapes that are easy to read. We are using the narrow style font to eliminate the need for word wrapping.

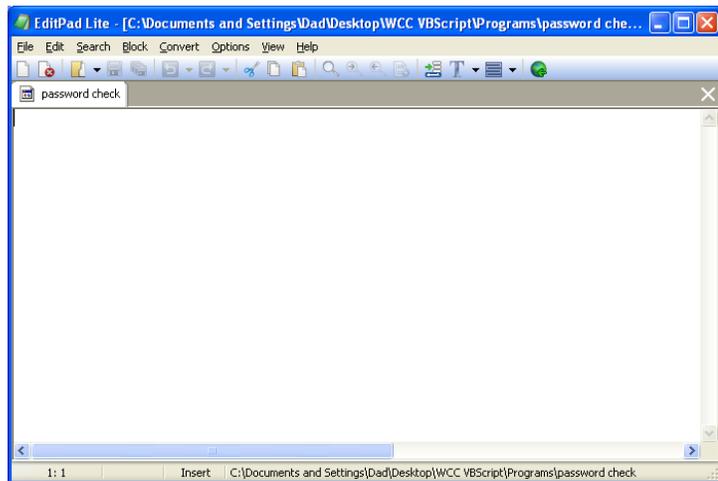


Figure 4.1 – The EditPad Lite Script Editor

Adding a Copyright Statement to a Program

At the beginning of a new program, we will expect to see an explanation or any special instructions in the form of comments such as copyright, permissions or other legal notices to inform programmers what are the rules dealing with running the code. Comments at the opening of the code could help an individual determine whether the program is right for their application or is legal to use. The message box is a great tool when properly utilized to inform someone if they are breaking a copyright law when running the code.

As in Visual Basic, the single quote character (‘) will precede a comment. When the code is compiled, comments are ignored.

Begin the password checker.vbs script with:

```
'password checker.vbs copyright (c) 2007 by charles w. robbins
```

Then add a short description for the program:

```
' this program will check the password against complexity requirements and length
```

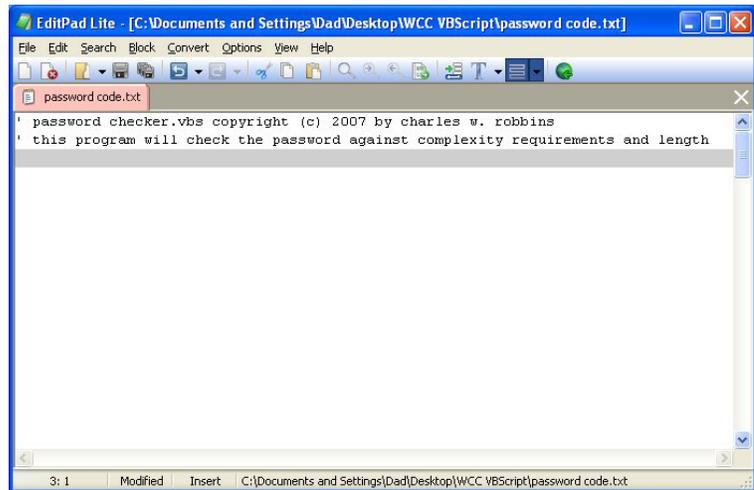


Figure 4.2 – Adding a Copyright Statement

Using a Message Box to Communicate the Copyright and More

The comments we placed in the first two lines of the program will inform the individual opening and reading the code, but those user that may run the application without checking, the message box is a great tool to alert the client to the rules of the program and what will the application do.

In our code add another comment:

```
' alert the user with a message box
```

The function MsgBox will launch a message box in Windows. The text or as programmers state “string” will be enclosed in quotes.

Type the following line of code:

```
MsgBox "Password_checker.vbs copyright (c) 2007 by Charles W. Robbins. This program will check the password against complexity requirements and length."
```

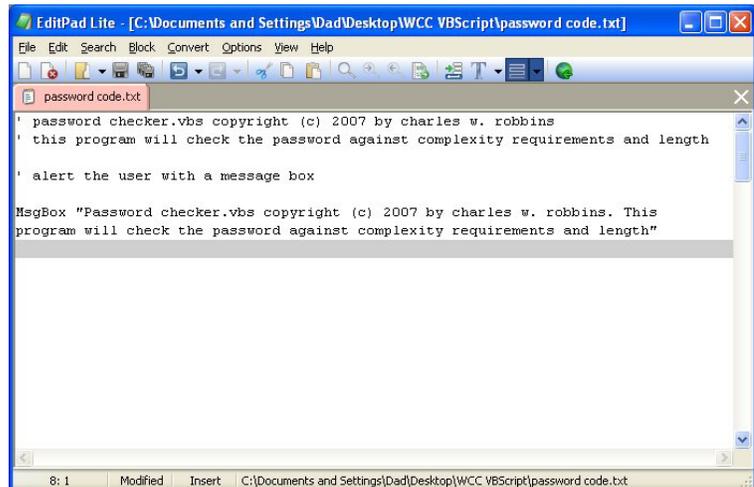


Figure 4.3 – Adding a Message Box

Declaring Variables in a Program with the Dimension Statement

When we are going to use a number, text string or object that may change throughout the life of the code, we create a variable to hold the value of that changing entity. In Visual Basic, the dimension statement is one of the ways to declare a variable at the script or procedure level. The other two ways are the Private and Public statements, which we will use in later chapters.

In our program, we will set fourteen variables to hold answers. The variables number, uppercase, lowercase, special and length will be used to test the password in characters. The variable password will hold the password typed by the user when running the program. The variable length will hold the integer or whole number that communicates how many characters are in the password. The variable character holds the single character removed from the left hand side of the password. The variable holder contains an ASCII number equal to the character. The variable counter holds a whole number used for counting each step through the programming loop. The variable check holds the sum of the four conditions. Remember, we pass the strong password test when the variable check is greater or equal to three in the variable length is greater or equal to seven. The variables msg1, msg2, msg3 and msg4 hold sentences that we can use to communicate to the user what they need to do to fix their password choice.

Type the following code:

```
'Declare variable
dim number
dim uppercase
dim lowercase
dim special
dim password
dim length
dim character
dim holder
dim counter
dim check
dim msg1
dim msg2
dim msg3
dim msg4
```

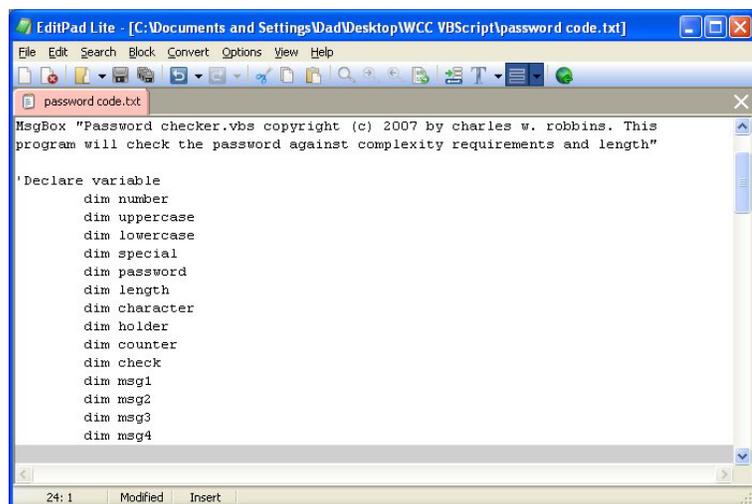
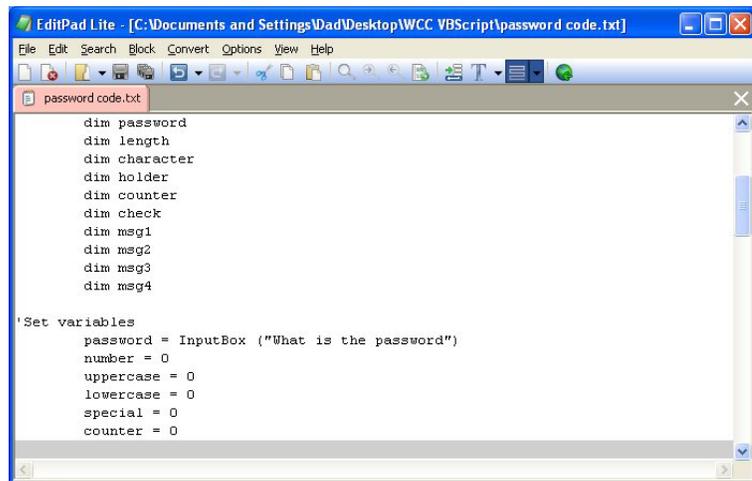


Figure 4.4 – Declaring Variables with Dim Statements

Notice that the variable name should be a word or a phrase without spaces that represents the value that the variable contains. If we want to hold a value of one's date of birth, we can call the variable, DateofBirth. The keywords Date and Birth are in sentence case with the first letter capitalized. There are no spaces in the name. Some programmers use the underscore character (_) to separate words in phrases. This is acceptable, but a double underscore (__) can cause errors if we do not detect the repeated character.

Setting Variables in a Program with an InputBox

Next, we will set the variables using the equal function (=) and an InputBox. When this section of the code is run, we want a window to appear with the prompt “**What is the password**” so the user can input the password they are planning to use in the blank text box. Now, we have four variables representing the cases and we set them to zero (0) at the beginning of the program. The counter for the loop is also set to zero.



```
dim password
dim length
dim character
dim holder
dim counter
dim check
dim msg1
dim msg2
dim msg3
dim msg4

'Set variables
password = InputBox ("What is the password")
number = 0
uppercase = 0
lowercase = 0
special = 0
counter = 0
```

Figure 4.5 – Setting the Variables in the Script

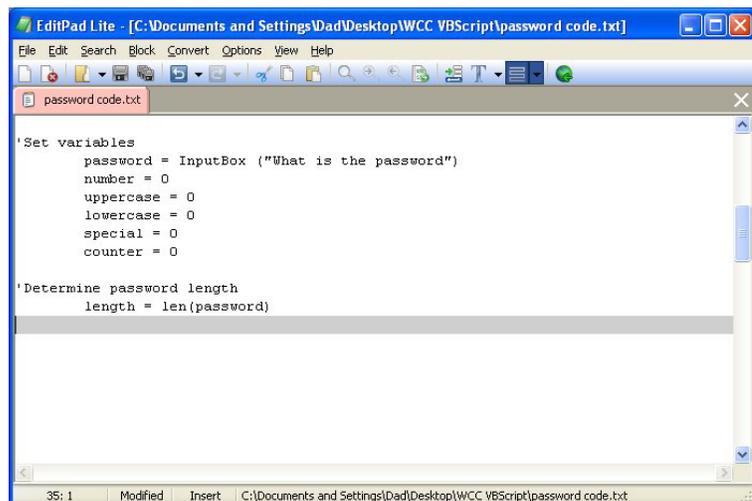
So type the following code in the program to set the variables.

```
'Set variables
password = InputBox ("What is the password")
number = 0
uppercase = 0
lowercase = 0
special = 0
counter = 0
```

Determining a String Length with the Function Len

When we want to find how many characters are in a text string then we will use the **len** function. In a simple case, like with the word “test”, the answer the len(“test”) is 4. When there is a space in the text string, like “easy coding” then len(“easy coding”) is 11. The space counts in the computation of the text string length. Type the following code in the program.

```
'Determine password length
length = len(password)
```



```
'Set variables
password = InputBox ("What is the password")
number = 0
uppercase = 0
lowercase = 0
special = 0
counter = 0

'Determine password length
length = len(password)
```

Figure 4.6 – Setting the Variables in the Script

The following is an extract from the VBScript Quick Reference for the Len function.

Function	Name	Description
Len	Length	Returns the number of characters in a text string as an integer
Examples		
If the word typed is "test"	length = len(password)	Answers 4
If the numbers that are typed are "923975"	length = len(password)	Answers 6
If the characters that are typed are "N01Graalb"	length = len(password)	Answers 9

We find a number of characters in the string so we know how many times we must go through a while loop to test each character.

Using a Loop with the Do While Function

Many years ago I brought a class of students through the steps of creating while loops in their computer programs. In that exercise, I had the students create a basement stairs completely from scratch using a visual program. The problem involved some mathematics, the knowledge of selections sets, and of course the while loops. I would have to say that most of the students really struggled through the exercise with me. My approach was too difficult. My challenge was to find a technique to train powerful programming functions and simultaneously allowing the programming student to concentrate on coding.

The next group going through the lesson plan for while loops at the college, I still used a step with a run of 10 inches in a rise of the 8 inches. We repeated the single step ten times to construct a simple looking stairs. We drew a ball and bounced it down the stairs using the while loop. They enjoyed the simplicity of the assignment and went on to make very nice looking animations. In our first while loop in VBScript, we are going to remove a single letter and test the character four times. Sounds pretty effortless and through simplicity we learn how to use another useful tool.

When we are using a Do While Loop function in a VBScript, right after the words Do While we will place a test statement that will be used by the Do While function each time to determine whether to enter or exit the loop. In our example and in most of our programs, we will use the counter. We set the variable counter previously to zero. We know the number of characters in the password which is held in the variable "length". The test is simple. We will stay in the loop as long as the variable counter is less than length.

If the password is 13 characters long, the first time into the loop the condition is ($< 0\ 13$) which is true so all of the expression inside the while loop will be read in the program. The second time into the loop the condition is ($< 1\ 13$) which is also true so all the loop continues to run. The third time into the loop the condition is ($< 2\ 13$) which is also true and the loop continues. The fourth time into the loop the condition is ($< 3\ 13$) which is also true and this seems to be going on and on.

```

EditPad Lite - [C:\Documents and Settings\Dad\Desktop\WCC VBScript\password code.txt]
File Edit Search Block Convert Options View Help
password code.txt
' Set variables
password = InputBox ("What is the password")
number = 0
uppercase = 0
lowercase = 0
special = 0
counter = 0

' Determine password length
length = len(password)

Do While counter < length

```

Figure 4.7 – Starting a Loop

In a classroom we go through every step on our first while loop. By this time many students do not think this will ever end. The thirteenth time into the loop the condition is ($< 12\ 13$) which is also true, because 12 is less than 13. Now on fourteenth time into the loop the condition is ($< 13\ 13$) which is false and so the while loop will not execute and the next expression in the code will be read.

The following is an extract from the VBScript Quick Reference for the Do While and $<$ function.

Function	Name	Description
Do While	While Loop	Will execute a subroutine inside a loop if an initial test is passed
Examples		
Using a counter	Do While counter < length Loop	If the length equals 13 then the loop will continue 13 times
Using a question	Do While question="yes" Loop	Will continue as long as question equals "yes"

Using the Left Function to Remove a Character from a String

Once inside the loop, we need to extract a single letter at a time from the password in order to test the letter against the strong password criteria. We will do this operation by using the **left** function. The left function is set up as shown below.

**'Check password
character = left(password,1)**

```

'EditPad Lite - [C:\Documents and Settings\Dad\Desktop\WCC VBScript\password code.txt]
File Edit Search Block Convert Options View Help
password code.txt
'Set variables
password = InputBox ("What is the password")
number = 0
uppercase = 0
lowercase = 0
special = 0
counter = 0

'Determine password length
length = len(password)

Do While counter<length

'Check password
character = left(password,1)
holder = asc(character)
    
```

Figure 4.8 – Check the Password

We type the name **character**, which is a variable name that will hold a single character and then an equal sign. After entering the **left** function, comes an open parentheses, then the variable name password which hopefully contains seven or more characters. Now to retrieve the first character, we will type one in the first character will be taken from the password. The line of code ends with a closed parentheses.

Someday, we may want to use this function for other purposes so we should know that we can modify this same line of code to

character = left(password,3)

And then the variable **character** will contain the first three letters of the password text string.

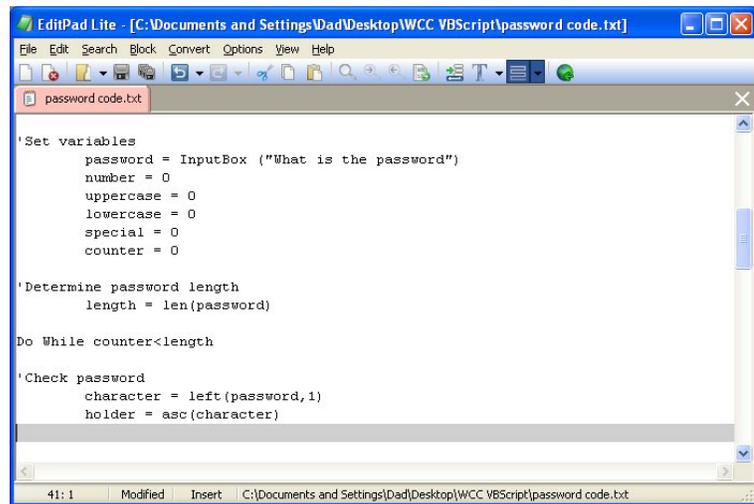
The following is an extract from the VBScript Quick Reference for the Left function.

Function	Name	Description
Left	Left	Will execute a subroutine inside a loop if an initial test is passed
Examples		
Extract 1 letter from the password "test"	character = left(password,1)	Returns the first letter of the password "t"
Extract 3 letters from the password "test"	character = left(password,3)	Returns the first three letters of the password "tes"

Changing a Character to ASCII with the ASC Function

Another easy function to learn to use is the ASC or ASCII tool. We named the variable holder because we are just translating the keyboard character extracted from the password to a numeric code. American Standard Code for Information Interchange or ASCII format has the number that designation for every keyboard character. Type the following code.

holder = asc(character)



```

'EditPad Lite - [C:\Documents and Settings\Dad\Desktop\WCC VBScript\password code.txt]
File Edit Search Block Convert Options View Help
password code.txt
' Set variables
password = InputBox ("What is the password")
number = 0
uppercase = 0
lowercase = 0
special = 0
counter = 0

'Determine password length
length = len(password)

Do While counter < length

'Check password
character = left(password, 1)
holder = asc(character)
    
```

Figure 4.9 – The ASCII Function

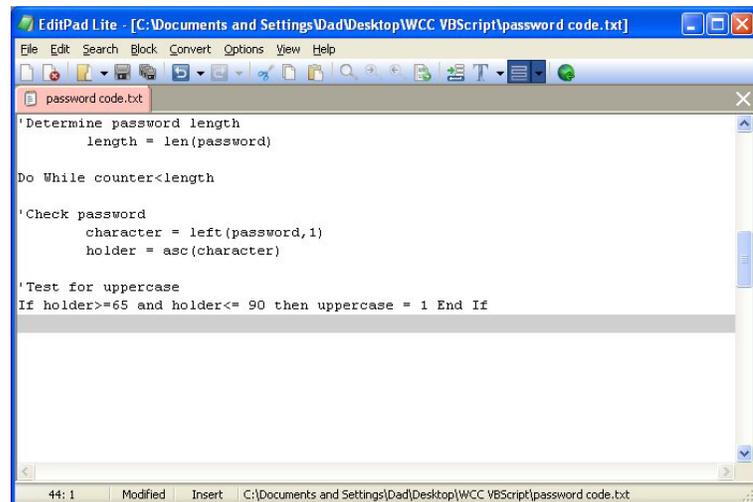
The reason we are changing the extracted character from the password to a numeric value is so that we can be easily compare this translated value to a range of numeric values that represent uppercase letters, lowercase letters, numbers and special characters. In the ASCII appendix of this text, we can see that the numeric range of uppercase letters is from 65 to 90. In the same table, we see that numeric arrange for lowercase letters to be from 97 to 122. Again, the table shows the range zero to nine on the keyboard to be from 48 to 57. For special characters, we will use an exact numbers to check the single extracted character. This is a very easy technique to use in programming.

The following is an extract from the VBScript Quick Reference for the ASC function.

Function	Name	Description
ASC	ASCII	Returns the ASCII numeric code for a single text string character
Examples		
The uppercase letter "A"	holder = asc(character)	Returns 65
The lowercase letter "a"	holder = asc(character)	Returns 97
The number "1"	holder = asc(character)	Returns 49
The special character "?"	holder = asc(character)	Returns 63

Testing a Case with the If -Then Function

Whenever we are confronted with making a choice between two or more options in a computer program, then the **if-then** function becomes a very popular solution to this challenge. The **if-then** function will execute the statements within the then section of the **if-then** expression when the logical test is true. The **if-then** function also will execute the else section of the **if-then** expression when the logical test is false.



```

' Determine password length
length = len(password)

Do While counter < length

' Check password
character = left(password, 1)
holder = asc(character)

' Test for uppercase
If holder >= 65 and holder <= 90 then uppercase = 1 End If

```

Figure 4.10 – The If -Then Function

The **if-then** function is arranged to work in a more complex fashion than other VBScript tools. It is initially, and after that an expression containing the logical test is written right after the **if**. The logical expression tests for a true or false response. In this program, the test is whether the ASCII number in the variables **holder** is greater or equal to 65 and less or equal to 90. If the answer is true, then the variable **uppercase** is assigned the value of one. If the answer is false, then the variable **uppercase** remains a zero as assigned earlier in the program.

So type the following expression in the routine:

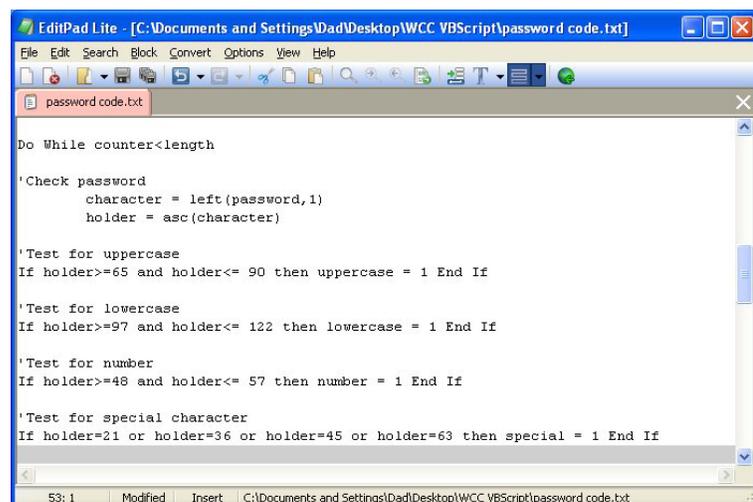
```

' Test for uppercase
If holder >= 65 and holder <= 90 then uppercase = 1 End If

```

In our first **if-then** statement, we are not going to use the **else** section of the function. If we needed to execute a statement for a false return, we would use the **else** section of the function.

In this program, we get to practice our first **if-then** statements another three times. Type the following lines of code to test the single extracted character for lowercase, number and special character conditions.



```

Do While counter < length

' Check password
character = left(password, 1)
holder = asc(character)

' Test for uppercase
If holder >= 65 and holder <= 90 then uppercase = 1 End If

' Test for lowercase
If holder >= 97 and holder <= 122 then lowercase = 1 End If

' Test for number
If holder >= 48 and holder <= 57 then number = 1 End If

' Test for special character
If holder = 21 or holder = 36 or holder = 45 or holder = 63 then special = 1 End If

```

Figure 4.11 – More If -Then Function

'Test for lowercase

If holder>=97 and holder<= 122 then lowercase = 1 End If

'Test for number

If holder>=48 and holder<= 57 then number = 1 End If

'Test for special character

If holder=21 or holder=36 or holder=45 or holder=63 then special = 1 End If

The following is an extract from the VBScript Quick Reference for the **if-then** function.

Function	Name	Description
if	If Statement	The if function will execute the functions within the then section of the if expression when the logical test is true and within the else section of the if expression when the logical test is false
Example		
If-then statement with just a then section with a logical test equally true holder = 70	If holder>=65 and holder<= 90 then uppercase = 1 End If	Uppercase = 1
If statement with just a then section with a logical test equally false holder = 60	If holder>=65 and holder<= 90 then uppercase = 1 End If	Uppercase = 0
If statement with a then and or else section with a logical test equally false holder = 60	If holder>=65 and holder<= 90 then uppercase = 1 Else uppercase = 0 End If	Uppercase = 0

The test case in the **if-then** statement has an **And** function. The test case is

holder>=65 and holder<= 90

where the ASCII number holding variable has to be greater then or equal to 65 and less than or equal to 90. When we us the **And** function, both cases have to be true for the case statement to be true.

Removing Single Character at a Time with the Mid Function

The **mid** function will work wonders for us if we know how to use this text handling function.

By typing a comma and a number after the variable **password**, we can extract just about any part of a text string that we wish. In this case we will extract all the characters after the first character. When we leave the next integer or whole number off after the 2 in the line of code:

password = mid(password,2)

then all remaining characters are extracted

```

'Check password
  character = left(password,1)
  holder = asc(character)

'Test for uppercase
If holder>=65 and holder<= 90 then uppercase = 1 End If

'Test for lowercase
If holder>=97 and holder<= 122 then lowercase = 1 End If

'Test for number
If holder>=48 and holder<= 57 then number = 1 End If

'Test for special character
If holder=21 or holder=36 or holder=45 or holder=63 then special = 1 End If

password = mid(password,2)
  
```

Figure 4.12 – Using the Mid Function

The following is an extract from the VBScript Quick Reference for the **Mid** function.

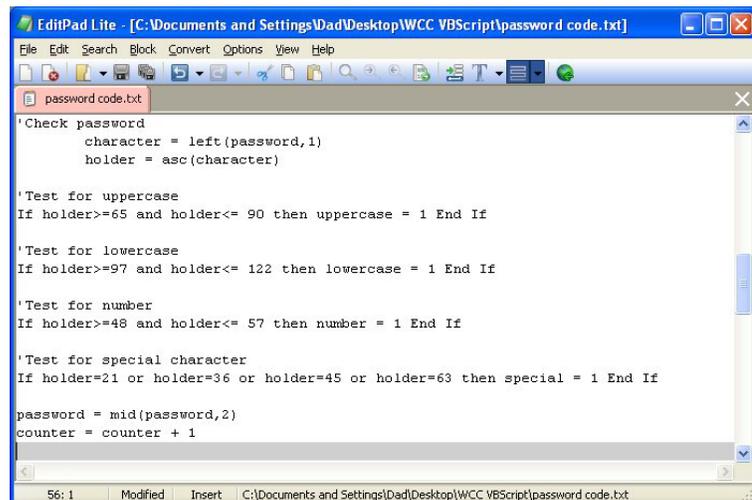
Function	Name	Description
Mid	Mid	Will return a text string based upon a position number and character count
Examples		
Extract the remaining text string after the first letter password = "test"	password = mid(password,2)	Returns the first letter of the password "est"
Extract the next two characters after the first letter password = "test"	password = left(password,2,2)	Returns the first three letters of the password "es"

Adding One to the Counter in the Loop

Now, we need to discuss the properties of a programming loop. The easiest technique we have demonstrate this concept is to take a small group of students and form a circle. We identify the beginning of the circle to be one person and we give them the value of 0 and they say out load “zero”. Each person passes the marker around clockwise until a student hands the first person the same marker back. Now in order to make a basic programming loop function properly, we will add one to the beginning value, so the first person announces “one” and continues to pass the marker around the group again. The next time the first person receives the marker they have figured out the game by now, and states “two”. To allow this to become common place in their programming skill set, typically while we are discussing the writing of the **Adding** expression, we allow them to continue until the entire group is tired of passing the marker in a circle and hearing the count by ones, but finally someone will ask how to stop the loop. That will be another discussion with another function, but remember where you heard about loops first, using the **adding** function. Okay, we can stop counting, now.

The next expression we will place in the code will add one to the variable **counter**. We will use the adding function to add 1 to the counter, so the variable **counter** will now be 1.

```
counter = counter + 1
```



```
File Edit Search Block Convert Options View Help
password code.txt
'Check password
  character = left(password,1)
  holder = asc(character)

'Test for uppercase
If holder>=65 and holder<= 90 then uppercase = 1 End If

'Test for lowercase
If holder>=97 and holder<= 122 then lowercase = 1 End If

'Test for number
If holder>=48 and holder<= 57 then number = 1 End If

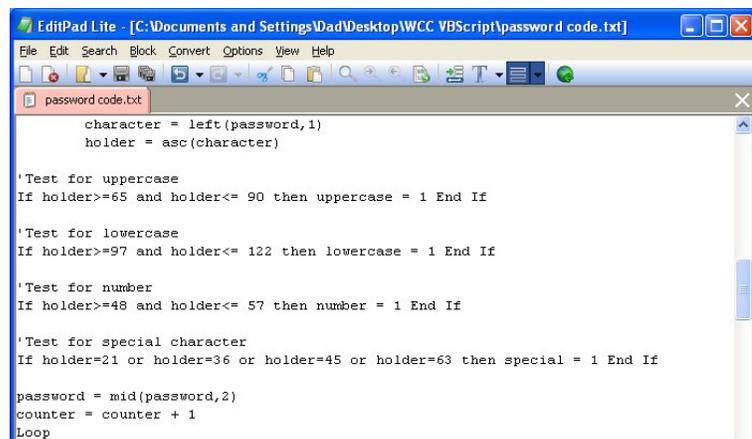
'Test for special character
If holder=21 or holder=36 or holder=45 or holder=63 then special = 1 End If

password = mid(password,2)
counter = counter + 1

56: 1 Modified Insert C:\Documents and Settings\Dad\Desktop\WCC VBScript\password code.txt
```

Figure 4.13 – Adding One to the Counter

At the end of the Do While loop type Loop as shown in Figure 4.14. At this point of the program the computer will return to the initial starting point and the test in the Do While case will be run again. The loop will continue until the case is false.



```
File Edit Search Block Convert Options View Help
password code.txt
  character = left(password,1)
  holder = asc(character)

'Test for uppercase
If holder>=65 and holder<= 90 then uppercase = 1 End If

'Test for lowercase
If holder>=97 and holder<= 122 then lowercase = 1 End If

'Test for number
If holder>=48 and holder<= 57 then number = 1 End If

'Test for special character
If holder=21 or holder=36 or holder=45 or holder=63 then special = 1 End If

password = mid(password,2)
counter = counter + 1
Loop
```

Figure 4.14 – Ending the Loop

The following is an extract from the VBScript Quick Reference for the **Adding** function.

Function	Name	Description
+	Adding	The addition function will add two or more numbers
Examples		
Using integers and counter = 0	counter = counter + 1	Answer: counter = 1

Using More If-Then Functions

After exiting the Do While loop, we will check the password to see if the text string meets the four criteria. Type the following code as shown in Figure 4.15.

```
'Check the password
check = uppercase + lowercase + number + special
```

Our strategy is to have the variable check equal to 3 or more, which means that the password has met the 3 out of 4 types of characters.

```
password = mid(password,2)
counter = counter + 1
Loop
'Check the password
check = uppercase + lowercase + number + special
```

Figure 4.15 – Adding the Check Variables

Next, we will test for the password length. We captured this piece of information earlier before entering the loop. If the text string is less than seven characters long, then we will construct a message recommending at least a seven character text string.

```
if length < 7 then msg1 = "Make the
password at least 7 characters
long." End If
```

```
password = mid(password,2)
counter = counter + 1
Loop
'Check the password
check = uppercase + lowercase + number + special

if length < 7 then msg1 = "Make the password at least 7 characters long."
End If
```

Figure 4.16 – If – Then Statement to Build Statements

We will now check each value of the test variables to see if they are equal to zero and if they are, we will make a recommendation to change the password for this consideration.

Type the following code where we will assign a sentence to a message named msg1, msg2, msg3 and msg4. We will use the messages in the last message box informing the user of the results of the program.

```

EditPad Lite - [C:\Documents and Settings\Dad\Desktop\WCC VBScript\password code.txt]
File Edit Search Block Convert Options View Help
password code.txt
password = mid(password,2)
counter = counter + 1
Loop

'Check the password
check = uppercase + lowercase + number + special

if length < 7 then msg1 = "Make the password at least 7 characters long.
" End If
if uppercase = 0 then msg1 = "Make at least one character uppercase.
" End If
if lowercase = 0 then msg2 = "Make at least one character lowercase.
" End If
if number = 0 then msg3 = "Make at least one character a number.
" End If
if special = 0 then msg4 = "Make at least one character a special character.
" End If

```

Figure 4.17 – If – Then Statement to Build Statements

if uppercase = 0 then msg1 = "Make at least one character uppercase. " End If
if lowercase = 0 then msg2 = "Make at least one character lowercase. " End If
if number = 0 then msg3 = "Make at least one character a number. " End If
if special = 0 then msg4 = "Make at least one character a special character. " End If

In the last if-then statement, we will use and else section to announce that the password is not strong and to construct the messages from the previous section to what changes should be made. In this statement the case is

check>=3 and length>=7

and if the answer is True then the

"Password is strong"

```

EditPad Lite - [C:\Documents and Settings\Dad\Desktop\WCC VBScript\password code.txt]
File Edit Search Block Convert Options View Help
password code.txt
'Check the password
check = uppercase + lowercase + number + special

if length < 7 then msg1 = "Make the password at least 7 characters long.
" End If
if uppercase = 0 then msg1 = "Make at least one character uppercase.
" End If
if lowercase = 0 then msg2 = "Make at least one character lowercase.
" End If
if number = 0 then msg3 = "Make at least one character a number.
" End If
if special = 0 then msg4 = "Make at least one character a special character.
" End If

if check>=3 and length>=7 then MsgBox "Password is strong" else MsgBox "Password
is not strong. " & msg1 & msg2 & msg3 & msg4 End If

```

Figure 4.18 – If – Then Statement to Build Statements

Type the following code as shown in Figure 4.18.

if check>=3 and length>=7 then MsgBox "Password is strong" else MsgBox "Password is not strong. " & msg1 & msg2 & msg3 & msg4 End If

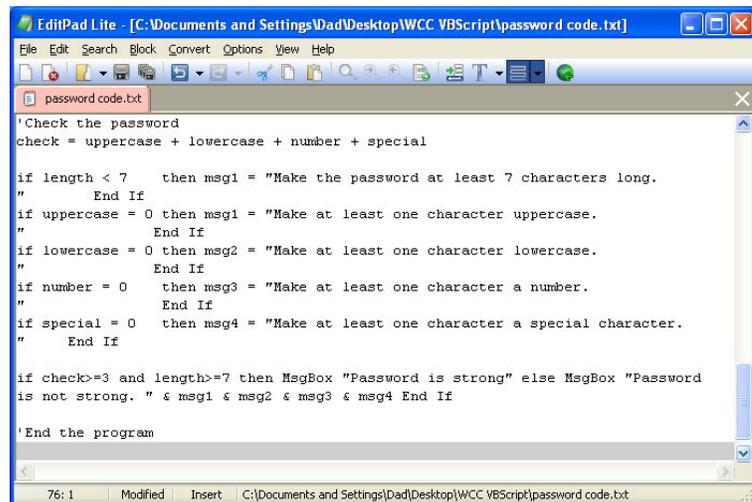
We will use & to concatenate the four test messages to the first sentence “Password is not strong.”

Ending the Program

To end this program, we will type a comment saying so. In the future, this will be more elaborate, but for now we will just get used to announcing the natural divisions of the script.

Type the following code:

'End of program

A screenshot of the EditPad Lite application window. The title bar reads "EditPad Lite - [C:\Documents and Settings\Dad\Desktop\WCC VBScript\password code.txt]". The menu bar includes "File", "Edit", "Search", "Block", "Convert", "Options", "View", and "Help". The main text area contains the following VBScript code:

```
'Check the password
check = uppercase + lowercase + number + special

if length < 7 then msg1 = "Make the password at least 7 characters long.
"
End If
if uppercase = 0 then msg1 = "Make at least one character uppercase.
"
End If
if lowercase = 0 then msg2 = "Make at least one character lowercase.
"
End If
if number = 0 then msg3 = "Make at least one character a number.
"
End If
if special = 0 then msg4 = "Make at least one character a special character.
"
End If

if check>=3 and length>=7 then MsgBox "Password is strong" else MsgBox "Password
is not strong. " & msg1 & msg2 & msg3 & msg4 End If

'End the program
```

The status bar at the bottom shows "76: 1 Modified Insert C:\Documents and Settings\Dad\Desktop\WCC VBScript\password code.txt".

Figure 4.19 – Ending the Program

Programs creating and placing text in a message box are very easy to write once we have achieved writing the first program with these new functions. There are additional exercises for simple routines in the appendixes of this manual. Written below is the entire Password_checker.vbs code for your benefit.

```
' Password_checker.vbs copyright (c) 2007 by Charles W. Robbins
' this program will check the password against complexity requirements and length

' alert the user with a message box

MsgBox "Password_checker.vbs copyright (c) 2007 by Charles W. Robbins. This program will
check the password against complexity requirements and length"

'Declare variable
dim number
dim uppercase
dim lowercase
dim special
dim password
dim length
dim character
dim holder
dim counter
dim check
dim msg1
dim msg2
dim msg3
dim msg4
```

```

'Set variables
password = InputBox ("What is the password")
number = 0
uppercase = 0
lowercase = 0
special = 0
counter = 0

'Determine password length
length = len(password)

Do While counter<length

    'Check password
    character = left(password,1)
    holder = asc(character)

    'Test for uppercase
    If holder>=65 and holder<= 90 then uppercase = 1 End If

    'Test for lowercase
    If holder>=97 and holder<= 122 then lowercase = 1 End If

    'Test for number
    If holder>=48 and holder<= 57 then number = 1 End If

    'Test for special character
    If holder=21 or holder=36 or holder=45 or holder=63 then special = 1 End If

    password = mid(password,2)
    counter = counter + 1
Loop

'Check the password
check = uppercase + lowercase + number + special
if length < 7 then msg1 = "Make the password at least 7 characters long. " End If
if uppercase = 0 then msg1 = "Make at least one character uppercase. " End If
if lowercase = 0 then msg2 = "Make at least one character lowercase. " End If
if number = 0 then msg3 = "Make at least one character a number. " End If
if special = 0 then msg4 = "Make at least one character a special character. " End If
if check>=3 and length>=7 then MsgBox "Password is strong" else MsgBox "Password
is not strong. " & msg1 & msg2 & msg3 & msg4 End If

'End the program

```

Saving the Program

Now that the program is finished, we need to double check our typing with the text in this manual and then save our program to our folder named “VBScripts”.

Make sure the Save in list box is displaying the VBScripts folder and the File name is “password checker.vbs” as shown in Figure 4.20.

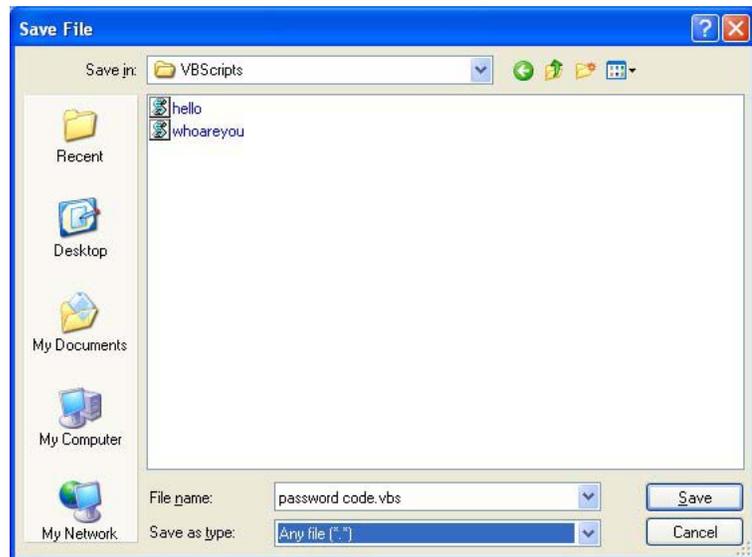


Figure 4.20 – Saving the Program

Running the Program

After noting that the program is saved, press the Start button and pick Run on the Start menu. The Run window will appear on the desktop as shown in Figure 4.21. Select the Browse command button, to open the Browse window.

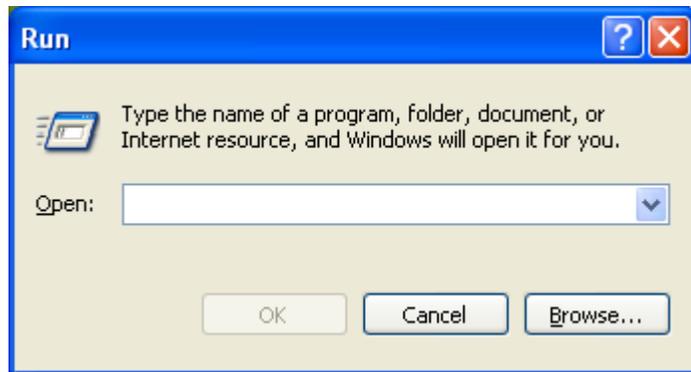


Figure 4.21 – Running the Program

In the Browse window, we need to open the VBScripts folder and select the hello script. If we cannot see the file listed, change the Files of type to “All Files” as shown in Figure 4.22.

Select the Open command button to return to the Run window as shown in Figure 4.23. Press the OK button to execute the script.

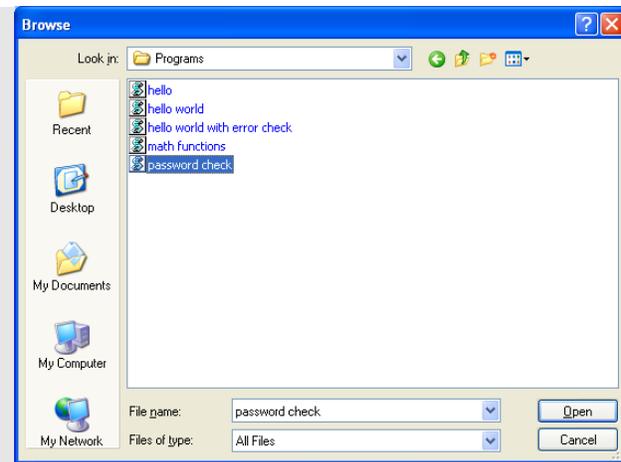


Figure 4.22 – Browse for the Program

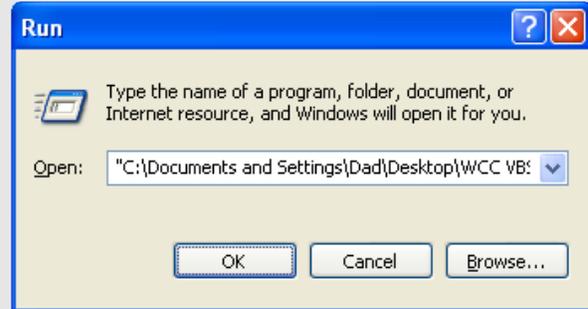


Figure 4.23 – Executing the Program

The first message box to appear will be the regular text message with the program name and the description of the program as shown in Figure 4.24. Press the OK command button to close the message box.

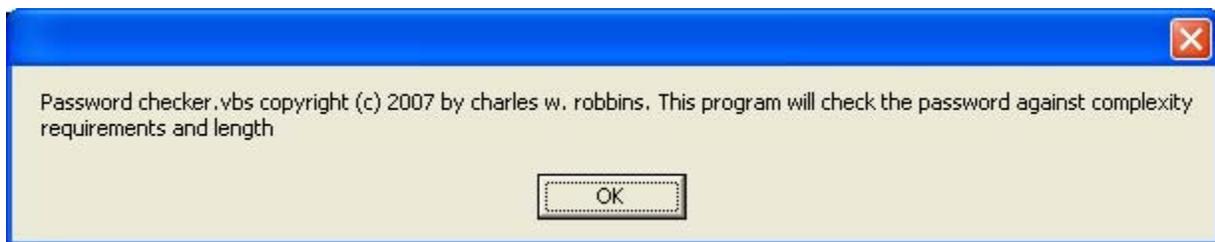


Figure 4.24 – The First Message Box in the Script

The second message will appear as shown in Figure 4.25 asking “What is the password”. Type a string of text for a password and press the OK command button and the script will execute.

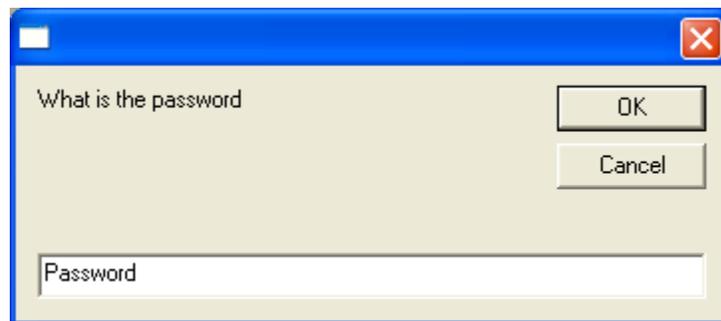


Figure 4.25 – The Second Message in the Script

There are many variations of this script we can practice and check text entered into the message box. While we are practicing with the message boxes and strings, we learn how to use if-then statements, variables, strings and comments. These are skills that we want to commit to memory.



Figure 4.26 – The Last Message Box in the Script

*** World Class CAD Challenge 9-8 * - Write a Script that displays a message box asking for a password. The program will determine whether the text is a string password and give the user suggestions on how to make a failing password into a strong one.**

Continue this drill four times using some other messages, each time completing the VBScript in less than 30 minutes to maintain your World Class ranking.