Chapter 17: Working with Strings.

We have used strings to store non-numeric information, build output, and capture input. We have also seen, in Chapter 11, using the Unicode values of single characters to build strings.

This chapter shows several new functions that will allow you to manipulate string values.

The String Functions:

BASIC-256 includes eight common functions for the manipulation of strings. Table 8 includes a summary of them.

Function	Description
`string(expression)`	Convert expression (string, integer, or decimal value) to a string value.
`length(string)`	Returns the length of a string.
`left(string, length)`	Returns a string of length characters starting from the left.
`right(string, length)`	Returns a string of length characters starting from the right.
`mid(string, start, length)`	Returns a string of length characters starting from the middle of a string.
`upper(expression)`	Returns an upper case string.
`lower(expression)`	Returns a lower case string.
`instr(haystack, needle)`	Searches the string "haystack" for the "needle" and returns it's location.

Table 8: Summary of String Functions

String() Function:

The **string**() function will take an expression of any format and will return a string. This function is a convenient way to convert an integer or floating-point number into characters so that it may be manipulated as a string.

```
# string.kbs
# convert a number to a string

a = string(10 + 13)
print a
b = string(2 * pi)
print b
```

Program 106: The String Function

```
23
6.283185
```

Sample Output 106: The String Function

New Concept

string(*expression*)

Convert expression (string, integer, or decimal value) to a string value.

Length() Function:

The *length()* function will take a string expression and return it's length in characters (or letters).

```
# length.kbs
# find length of a string

# should print 6, 0, and 17
print length("Hello.")
print length("")
print length("Programming Rulz!")
```

Program 107: The Length Function

```
6
0
17
```

Sample Output 107: The Length Function

New Concept

`length(`***`expression`***`)`

Returns the length of the string expression. Will return zero (0) for the empty string "".

Left(), Right() and Mid() Functions:

The **left**(), **right**(), and **mid**() functions will extract sub-strings (or parts of a string) from a larger string.

```
# leftrightmid.kbs
# show right, left, and mid string functions

a = "abcdefghijklm"

print left(a,4)     # prints first 4 letters

print right(a,2)    # prints last 2 letters

print mid(a,4,3)    # prints 4th-7th letters
print mid(a,10,9)   # prints 10th and 11th letters
```

Program 108: The Left, Right, and Mid Functions

```
abcd
kl
def
jklm
```

Sample Output 108: The Left, Right, and Mid Functions

New Concept

`left(string, length)`

Return a sub-string from the left end of a string. If length is equal or greater then the actual length of the string the entire string will be returned.

New Concept

`right(`*`string, length`*`)`

Return a sub-string from the right end of a string. If length is equal or greater then the actual length of the string the entire string will be returned.

New Concept

`mid(`*`string, start, length`*`)`

Return a sub-string of specified length from somewhere on the middle of a string. The start parameter specifies where the sub-string begins (1 = beginning of string).

Upper() and Lower() Functions:

The **upper**() and **lower**() functions simply will return a string of upper case or lower case letters. These functions are especially helpful when you are trying to perform a comparison of two strings and you do not care what case they actually are.

```
# upperlower.kbs

a = "Hello."

print lower(a)    # prints all lowercase

print upper(a)    # prints all UPPERCASE
```

Program 109: The Upper and Lower Functions

```
hello.
HELLO.
```

Sample Output 109: The Upper and Lower Functions

New Concept

```
lower(string)
upper(string)
```

Returns an all upper case or lower case copy of the string expression. Non-alphabetic characters will not be modified.

Instr() Function:

The **instr**() function searches a string for the first occurrence of another string. The return value is the location in the big string of the smaller string. If the substring is not found then the function will return a zero (0).

```
# instr.kbs
# is one string inside another

a = "abcdefghijklm"
print 'the location of "hi" is ';
print instr(a,"hi")
print 'the location of "bye" is ';
print instr(a,"bye")
```

Program 110: The Instr Function

```
the location of "hi" is 8
the location of "bye" is 0
```

Sample Output 110: The Instr Function

New Concept

`instr(haystack, needle)`

Find the sub-string (*needle)* in another string expression (*haystack*). Return the character position of the start. If sub-string is not found return a zero (0).

Big Program

The decimal (base 10) numbering system that is most commonly used uses 10 different digits (0-9) to represent numbers.

Imagine if you will what would have happened if there were only 5 digits (0-4) – the number 23 ($2*10^1+3*10^0$) would become 43 ($4*5^1+3*5^0$) to represent the same number of items. This type of transformation is called radix (or base) conversion.

The computer internally does not understand base 10 numbers but converts everything to base 2 (binary) numbers to be stored in memory.

The "Big Program" this chapter will convert a positive integer from any base 2 to 36 (where letters are used for the 11th - 26th digits) to any other base.

```
# radix.kbs
# convert a number from one base (2-36) to another

digits = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"

frombase = getbase("from base")
inputstring "number in base " + frombase + " >", number
number = upper(number)

# convert number to base 10 and store in n
n = 0
for i = 1 to length(number)
   n = n * frombase
   n = n + instr(digits, mid(number, i, 1)) - 1
next i

tobase = getbase("to base")

# now build string in tobase
result = ""
while n <> 0
   result = mid(digits, n % tobase + 1, 1) + result
   n = n \ tobase
end while

print "in base " + tobase + " that number is " + result
end

function getbase(message)
   # get a base from 2 to 36
   do
      inputinteger message+"> ", base
   until base >= 2 and base <= 36
   return base
end function
```

Program 111: Big Program - Radix Conversion

```
from base> 10
number in base 10 >999
to base> 16
in base 16 that number is 3E7
```

Sample Output 111: Big Program - Radix Conversion

Exercises:

Word Search

```
u r h t g n e l
p g i r a g k f
p r n l c f l r
e q i i e f e t
r d r g r f x s
v i i r h t t n
p m m x o t s i
r e w o l f w i
```

instr, left, length, lower, mid, right, string, upper

Problems

1. Have the user enter a string and display the string backwards.

2. Modify problem 1 to create a palindrome testing program. Remove all characters from the string that are not letters before reversing it. Compare the results and print a message that the text entered is the same backwards as forwards.

```
enter a string >never odd or even
neveroddoreven
neveroddoreven
```

```
is a palindrome
```

3. You work for a small retail store that hides the original cost of an item on the price tag using an alphabetic code. The code is "roygbivace" where the letter 'r' is used for a 0, 'o' for a 1, … and 'e' is used for a 9. Write a program that will convert a numeric cost to the code and a code to a cost.

```
cost or code >9.84
ecb
```

```
cost or code >big
4.53
```

4: You and your friend want to communicate in a way that your friends can't easily read. The Cesar cipher (http://en.wikipedia.org/wiki/Caesar_cipher) is an easy but not very secure way to encode a message. If you and your friend agree to shift the same number of letters then you can easily share a secret message. Decoding a message is accomplished by applying a shift of 26 minus the original shift.

A sample of some of the shifts for the letters A-D are shown below. Notice that the letters wrap around.

Shift	A	B	C	D
1	B	C	D	E
13	M	N	O	P
25	Z	A	B	C

Write a program that asks for the shift and for a string and displays the text with the cipher applied.

```
shift >4
message >i could really go for
```

	some pizza M GSYPH VIEPPC KS JSV WSQI TMDDE shift >22 message >M GSYPH VIEPPC KS JSV WSQI TMDDE I COULD REALLY GO FOR SOME PIZZA