Topic: 2.3.5 Built-in functions

Location

Many programming languages allow the searching for a short string of characters within a longer string. This would be similar to searching this book for the word 'computer'.

Common Key words for finding the location of one string within another are LOCATE, FIND and POSITION

Function	Purpose	Result
LOCATE	Returns the starting position of the first string, within the second string (returns '0' if the string is not found):	
	`x$=Locate("math", "mathematics")`	1
	`x$=Locate("the", "mathematics")`	3
	`x$=Locate("tea", "mathematics")`	0

Extraction

At times, a programmer will only want part of a string. Any part of a string can be obtained using the RIGHT, LEFT and MID functions.

The following string manipulation functions are called by typing:

Result = FunctionName(String [,x,y])

Function	Purpose	Result
RIGHT	Returns the number of specified characters from the right of the string:	
	`x$=Right("Computing", 3)`	"ing"
LEFT	Returns the number of specified characters from the left of the string:	
	`x$=Left("Computing", 3)`	"Com"
MID	Returns the number of specified characters from the middle of the string:	
	`x$=Mid("Computing", 5, 2)`	"ut"

Topic: 2.3.5 Built-in functions

Concatenation

Concatenation is where two or more strings are joined together to make a single string.

Note that when two strings are added together the second string is added to the end of the first:

```
"Peter" + "Jones" = "PeterJones"
"Peter" + " " + "Jones" = "Peter Jones"
"36.85" + "47" = "36.8547"
"47" + "36.85" ="4736.85"
"3/10/03" + "15" = "3/10/0315"
```

Length

Sometimes a programmer will need to know how many characters there are in a string.

Function	Purpose	Result
LEN	Returns the number of characters in the specified string: `x=Len("Computing")`	9

Conversion

Strings and numbers

Strings are a sequence of ASCII characters, even if they contain only numbers such as "241", and so they cannot be used within an arithmetic calculation – they need to be 'evaluated' first.

Likewise a number such as 27.3 is not stored as ASCII characters and so cannot be used within any of the string functions (such as Left, Right, Mid).

The function STR converts a number into a string and VAL converts a string into a number:

Function	Purpose	Result
STR	Returns the string form of the value: `x$=Str(31)`	"31"
VAL	Returns the numeric form of the string: `x=Val("42Chig")`	42

Topic: 2.3.5 Built-in functions

Characters and ASCII codes

The following two functions are used to either find the ASCII code for a specific character or to find the character from the ASCII code:

Function	Purpose	Result
ASC	Returns the ASCII code for the specified character: `x=Asc("A")`	65
CHR	Returns the character whose ASCII code is specified: `x$=Chr(68)`	"D"

ASCII character codes

Below is a table showing the most common characters and their ASCII codes:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Topic: 2.3.5 Built-in functions

Note the following:

- the codes less than 32 are 'control' codes that are not used in strings;
- 'space' (code 32) has the lowest code;
- next comes most of the punctuation symbols;
- then digits 0–9
- then uppercase letters
- then lowercase letters
- all other characters (e.g. é, å, π, etc.) have codes higher than these.

Comparing strings

When comparing strings, the codes of each string are compared, character by character, to decide which string is greater.

Because it is the ASCII codes that are compared the following applies:

```
"Computing" <> "computing"
```

in fact:

```
"Computing" < "computing"
"10" < "2"
"02" < "1"
"1 120" < "1,120"
" computing" < "Computing"
```

Topic: 2.3.5 Built-in functions

Sorting filenames

The following table shows a list of inconsistently names music tracks and the order in which they would be sorted:

Required order	Sorted order
Track 1.mp3	Track 5.mp3
Track 2.mp3	track 4.mp3
Track 3.mp3	Track 06.mp3
track 4.mp3	Track 1.mp3
Track 5.mp3	Track 11.mp3
Track 06.mp3	Track 2.mp3
track 7.mp3	Track 22.mp3
Track8.mp3	Track 3.mp3
Track 11.mp3	Track12.mp3
Track12.mp3	Track21.mp3
track 13.mp3	Track8.mp3
Track21.mp3	track 7.mp3
Track 22.mp3	track 13.mp3

In order to get them sorted into the required order, they must be renamed with consistent use of uppercase letters, spaces and leading zeros.

Recommended naming would be:

```
Track 01.mp3, Track 02.mp3, …, Track 10.mp3, Track 11.mp, …
```

Output data onto screen/file/printer, formatting the data for output as necessary.

Output will be either to the screen, a file or the printer.

Topic: 2.3.5 Built-in functions

Output controls

Output to the screen could be via a dialogue/message box, a text box, a list box or simply direct on the form.

Custom string formatting can be accomplished using specific characters recognized by the Format$ function, shown in the table below:

< Force lowercase	Display all characters in lowercase format.
> Force uppercase	Display all characters in uppercase format.
@ Character placeholder	Display a character or a space. If the string has a character in the position where the @ appears in the format string, display it; otherwise, display a space in that position. Placeholders are filled from right to left unless there is an ! character in the format string.
& Character placeholder	Display a character or nothing. If the string has a character in the position where the & appears, display it; otherwise, display nothing. Placeholders are filled from right to left unless there is an ! character in the format string.
! Force left to right fill of placeholders	The default is to fill from right to left.
\ (Display the next character in the format string)	Many characters in the format expression have a special meaning and can't be displayed as literal characters unless they are preceded by a backslash. The backslash itself isn't displayed. Using a backslash is the same as enclosing the next character in double quotation marks. To display a backslash, use two backslashes (\\). Examples of characters that can't be displayed as literal characters are the date- and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y, and /:), the numeric-formatting characters (#, 0, %, E, e, comma, and period), and the string-formatting characters (@, &, <, >, and !).
"ABC" (Display the string inside the double quotation marks)	To include a string in format from within code, you must enclose the text in quotation marks (to embed a quotation mark within a quoted string, use to consecutive quotation marks for the embedded quotation mark).

Topic: 2.3.5 Built-in functions

To demonstrate custom string formats using combinations of the characters listed above, set up another "Try It" project, and place the following code in the cmdTryIt_Click event of cmdTryIt button:

```
Private Sub cmdTryIt_Click()
Dim strName As String
strName = InputBox("Please enter your name:")

Print "Using '>':"; Tab(25); Format$(strName, ">")
Print "Using '<':"; Tab(25); Format$(strName, "<")
Print "Using '@':"; Tab(25); "Hello there, "; Format$(strName, "@@@@@@@@@@"); ". How are you?"
Print "Using '&':"; Tab(25); "Hello there, "; Format$(strName, "&&&&&&&&&&"); ". How are you?"
Print "Using '!@':"; Tab(25); "Hello there, "; Format$(strName, "!@@@@@@@@@@"); ". How are you?"
Print "Using '!&':"; Tab(25); "Hello there, "; Format$(strName, "!&&&&&&&&&&"); ". How are you?"
Print "Using '\':"; Tab(25); Format$(strName, "\H\e\l\l\o\,\ &&&&&&&&&&\.")
Print "Using embedded quotes:"; Tab(25); Format$(strName, """Hello, ""&&&&&&&&&&""."""")

End Sub
```

Run the project and click the "Try It" button. When the input box appears, enter a name in mixed case (in this example, **Bruce** was entered). The strings will be displayed as follows: