BASICALLY BASIC

Graham Hall, B.Sc.

BASIC Strings

A BASIC string is a sequence of one or more letters or symbols in any combination, enclosed within quotation marks. It is treated as a single collection of alphanumeric data which can be manipulated by means of string functions and assigned to string variables. The use of string variables, string assignment and string arrays has been described in previous parts of 'BASICALLY BASIC'. To understand how the computer represents alphanumeric data and can perform operations with strings, we need to look at the ASCII code (American Standard Code for Information Interchange).

ASCII Code

The ASCII code is a standard developed by the computer industry in which each symbol used in BASIC is assigned a unique binary digit pattern (bit pattern). When a symbol is typed on the keyboard, the terminal converts it to its binary code. For example, if you type the letter A on the keyboard, the terminal converts it to 01000001, which the computer recognises as A. Table 1 lists the set of ASCII character codes. The binary numbers have been converted to decimal numbers to make the assignments easier to understand.

The ASCII code is used to determine alphabetic precedence when alphanumeric data is compared with relational operators. This is described in the next section.

String Operators

A relational operator is a symbol used to compare the value of one variable or expression to another variable or expression within a BASIC program. The use of relational operators to determine numeric relationships has been described previously. Relational operators can also be used to compare alphanumeric data. The comparison is made in terms of the ASCII value of characters to establish alphabetical sequence. Consider the following program:

```
10 LET A$ = "BAS"
20 LET B$ = "SIC"
30 IF A$ < B$ THEN PRINT A$:GOTO 50
40 PRINT B$
50 END
```

Line 10 assigns the string BAS to the string variable called A$.
Line 20 assigns the string SIC to the string variable called B$.
Line 30 is a relational string expression. It compares string A$ with B$ to determine if A$ occurs first in alphabetic sequence. The comparison is made character by character using the ASCII character code. In this case the first letter of the string A$ is 'B' which precedes 'S', the first letter of the string B$, in the ASCII table. Therefore string A$ precedes B$ in alphabetic sequence so the condition is true and the string A$ is printed on the terminal. If the first two characters are the same the comparison proceeds to the second two characters, until a difference is found. For example, if A$ was assigned to the string BAY and B$ was assigned to the string BAT, the first character in each string match. The next two characters also match. Finally the last character of the string A$ is compared with the last character of the string B$. The ASCII code of Y (89) is greater than the ASCII code of T (84), hence the result of the comparison is false and line 40 prints the string B$ on the terminal.

Table 2 lists the string relational operators available in BASIC and their meaning.

Note: it is not permissible to compare a numeric or integer expression to a string expression using a relational operator. If this is attempted an error message will be output by the computer.

String Functions

BASIC provides a set of string functions (similar to the math and print functions described last month) to enable certain operations to be performed on strings. The following descriptions are intended to be general since the functions may perform differently for different computer systems. You should refer to your systems Language Reference Manual for a complete list of functions available in your version of BASIC. The string function names ending with a dollar sign ($) return a string value whereas function names not ending with a dollar sign return a numeric value.

ASCII Function

The ASCII function returns a numeric value that is the equivalent ASCII code for the first character in the string given as the argument to the function. The general format of the ASCII function is: ASC (string), where string is either a string constant or a string variable. For example, the command PRINT ASC ("P") will output 80, the decimal ASCII value of the character P, on the terminal. The following program uses the ASC function with a string variable as an argument:

```
10 A$ = "BASICALLY"
20 PRINT ASC (A$)
30 END
```

When the program is run the ASCII function returns the decimal ASCII value of the first character in the string assigned to the string variable A$, hence the decimal 66 will be output to the terminal.

CHR$ Function

The CHR$ (Character) function is the inverse of the ASCII function. It returns a single character string having an ASCII value of the numeric value specified as the argument to the function. The range of the ASCII codes is 0 to 127. If the value specified to the CHR$ function is outside this range it is treated as modulo 127. This means that 128 is treated as 0, 129 as 1 and so on. A non-integer argument to the CHR$ function will be truncated and the character returned will be that represented by the truncated number.

The following program demonstrates the use of the CHR$ function with different arguments.

```
10 PRINT CHR$ (70)
20 PRINT CHR$ (198)
30 PRINT CHR$ (10)
40 PRINT CHR$ (60.1)
50 END
RUN
F
F

<
```

The output from the program is as shown
Line 10 prints the character represented by the ASCII code 70 (an upper-case F).
Line 20 specifies an argument of 198. This is treated as modulo 127 (198 - 128 = 70) which is the ASCII code for an upper-case F.
Line 30 outputs a line feed character on the terminal causing a blank line of output. The decimal number specified as an argument to the CHR$ function on line 40 is truncated to an integer. The truncated number (60) represents the character <. (Characters may be different with your micro.)

The remaining BASIC string functions will be described in the next 'BASICALLY BASIC'. ■

Operator	Example	Meaning
=	X$=Y$	Strings X$ and Y$ are equivalent in characters after removing trailing blanks and nulls
<	X$<Y$	String X$ occurs before string Y$ in alphabetical sequence
< =or=<	X$< =Y$	String X$ is equivalent to or occurs before Y$ in alphabetical sequence
>	X$>Y$	String X$ occurs after string Y$ in alphabetical sequence
> =or=>	X$>=Y$	String X$ is equivalent to or occurs after Y$ in alphabetical sequence
<>	X$<>Y$	String X$ is not equal to string Y$
==	X$==Y$	Strings X$ and Y$ are identical in characters and length.

Table 2. String relational operators.

▼ **Table 1. ASCII character code.**

Decimal	Character	Meaning
000	NUL	Null
001	SO11	Start of heading
002	STX	Start of text
003	ETX	End of text
004	EOT	End of transmission
005	ENQ	Enquiry
006	ACK	Acknowledge
007	BEL	Bell
008	BS	Backspace
009	HT	Horizontal tab
010	LF	Line feed
011	VT	Vertical tab
012	FF	Form feed
013	CR	Carriage return
014	SO	Shift out
015	SI	Shift in
016	DLE	Data link escape
017	DC1	Device control 1
018	DC2	Device control 2
019	DC3	Device control 3
020	DC4	Device control 4
021	NAK	Negative acknowledge
022	SYN	Synchronous idle
023	ETB	End of transmission block
024	CAN	Cancel
025	EM	End of medium
026	SUB	Substitute
027	ESC	Escape
028	FS	File separator
029	GS	Group separator
030	RS	Record separator
031	US	Unit separator
032	SP	Space or blank
033	!	Exclamation mark
034	"	
035	#	Number, sign
036	$	
037	%	
038	&	
039	'	
040	(	
041	)	
042	*	
043	+	
044	,	
045	—	
046	.	
047	/	
048	0	
049	1	
050	2	
051	3	
052	4	
053	5	
054	6	
055	7	
056	8	
057	9	
058	:	
059	;	
060	<	
061	=	
062	>	
063	?	
064	@	
065	A	
066	B	
067	C	
068	D	
069	E	
070	F	
071	G	
072	H	
073	I	
074	J	
075	K	
076	L	
077	M	
078	N	
079	O	
080	P	
081	Q	
082	R	
083	S	
084	T	
085	U	
086	V	
087	W	
088	X	
089	Y	
090	Z	
091	[	
092	\	
093	]	
094	↑	
095	←	
096	`	
097	a	
098	b	
099	c	
100	d	
101	e	
102	f	
103	g	
104	h	
105	i	
106	j	
107	k	
108	l	
109	m	
110	n	
111	o	
112	p	
113	q	
114	r	
115	s	
116	t	
117	u	
118	v	
119	w	
120	x	
121	y	
122	z	
123		
124	|	
125		
126	~	
127	DEL	