provided they're in quotes. If not, they will be treated as tokens.

We are now at a point where we can begin to investigate how a BASIC program line is stored in memory. Computers differ in detail, but in general the first three or four bytes of a BASIC program line in the Text Area will contain the program line number and some information about the length of the line (see the panel). The line number that you attach to the line when you type it in is stored (although not as its ASCII equivalent — that would mean that line 61030 would require five bytes just to store its line number). Instead, the number is always stored in two-byte integer form. In this form, the numbers from 0 to 255 (which can be stored in one eight-bit byte, remember) are stored as a zero-byte followed by the byte containing the number. Numbers greater than 255 are stored exactly as paged addresses (see page 36): the value of the first byte is multiplied by 256, and added to the value of the second byte. 1000, for example, would be stored as 3,232 (3*256 + 232 = 1000). These two bytes are always in the same position in any line stored in the Text Area (although whether they are always the first two bytes or whatever depends upon the machine).

The information about the length of the line is placed in a single byte in the BBC and two bytes in the Spectrum. This represents simply the number of bytes in the line (including the two bytes for the line number and the line-length byte itself). If you know the address of the first byte of a BASIC program line in memory, and you add to it the contents of the line-length byte, then you will have the address of the first byte of the next program line. Since the biggest number expressible in one byte is 255, the maximum length of a BASIC program line on the BBC is, therefore, 255 characters. You might use the Mempeek program on page 59 to establish whether that is the limit of the number of *characters* you can type into a program line, or whether it is the limit of the length of the line as it is stored in the Text Area.

On the Commodore the line-length byte is replaced by two bytes called the *link address*. This is simply the actual address in two-byte form of the first byte of the next program line.

It's interesting to note that in the BBC and the Spectrum the next-line Start Address is calculated from the present address plus the line-length (which is slow but saves a byte); whereas on the Commodore the next address is stored as such

Conversion Chart

The American Standard Code for Information Interchange gives a standard character code value for the numbers 0-127. The codes 0-31 do not return printable characters, but are used to send control signals to peripherals such as the screen and the printer. The meaning of these codes, therefore, varies greatly from one machine to another — as the chart shows. Some machines, in particular the Commodore and the Spectrum in our chart, leave many codes unused (signified here by a ●). The codes 32-127 return the printable characters, and the standard ASCII codes in this range are common (with minor variations) to most computers. Your User Manual will give ASCII codes for your machine

ASCII CODE	ASCII	COMMODORE	SPECTRUM	BBC Micro
0	NUL — Does nothing	●	●	Null
1	SOH — Start heading	●	●	Next character to printer
2	STX — Start of text	●	●	Enable printer
3	ETX — End of text	●	●	Disable printer
4	EOT — End of transmission	●	●	Separate text/graphics cursors
5	ENQ — Enquire	White clr.key	●	Join text/graphics cursors
6	ACK — Acknowledge	●	PRINT	Enable VDU drivers
7	BEL — Ring bell	●	EDIT	Make short beep
8	BS — Backspace	Disables CBM key	Cursor left	Backspace cursor
9	HT — Horizontal tab	Enables CBM key	Cursor right	Forwardspace cursor
10	LF — Line feed	●	Cursor down	Cursor down
11	VT — Vertical tab	●	Cursor up	Cursor up
12	FF — Form feed	●	Delete key	Clear text area
13	CR — Carriage return	RETURN	ENTER	Return
14	SO — Shift out	L/case on	Number	Page mode on
15	SI — Shift in	●	●	Page mode off
16	DLE — Data link escape	●	INK	Clear graphics area
17	DC1 — Device control 1	Cursor down	PAPER	Def text colour
18	DC2 — Device control 2	Reverse on	FLASH	Def graphics colour
19	DC3 — Device control 3	Cursor home	BRIGHT	Def logical colour
20	DC4 — Device control 4	Delete key	INVERSE	Restore default log.clr
21	NAK — Negative acknowledge	●	OVER	Disable VDU drivers
22	SYN — Synchronous idle	●	AT	Select screen mode
23	ETB — End of transmission block	●	TAB	Reprogram display character
24	CAN — Cancel	●	●	Def graphics window
25	EM — End of medium	●	●	Plot m,x,y
26	SUB — Substitute	●	●	Restore default windows
27	ESC — Escape	●	●	Null
28	FS — File separator	Red clr.key	●	Def text window
29	GS — Group separator	Cursor right	●	Def graphics origin
30	RS — Record separator	Green clr.key	●	Move text cursor
31	US — Unit separator	Blue clr.key	●	Move text cursor to x,y

ASCII CODE	ASCII	ASCII CODE	ASCII
32	Space	80	P
33	!	81	Q
34	"	82	R
35	#	83	S
36	$	84	T
37	%	85	U
38	&	86	V
39	'	87	W
40	(	88	X
41	)	89	Y
42	*	90	Z
43	+	91	[
44	,	92	\
45	-	93	]
46	.	94	^
47	/	95	_
48	0	96	`
49	1	97	a
50	2	98	b
51	3	99	c
52	4	100	d
53	5	101	e
54	6	102	f
55	7	103	g
56	8	104	h
57	9	105	i
58	:	106	j
59	;	107	k
60	<	108	l
61	=	109	m
62	>	110	n
63	?	111	o
64	@	112	p
65	A	113	q
66	B	114	r
67	C	115	s
68	D	116	t
69	E	117	u
70	F	118	v
71	G	119	w
72	H	120	x
73	I	121	y
74	J	122	z
75	K	123	{
76	L	124	!
77	M	125	}
78	N	126	~
79	O	127	Delete