hexadecimal) that represent the letters, numbers and characters found on a standard keyboard, together with 32 control codes (that use codes 0 to 31 (denary) or 00 to 19 (hexadecimal)).

Table 1.2 shows part of the standard ASCII code table (only the control codes have been removed).

▼ **Table 1.2** Part of the ASCII code table

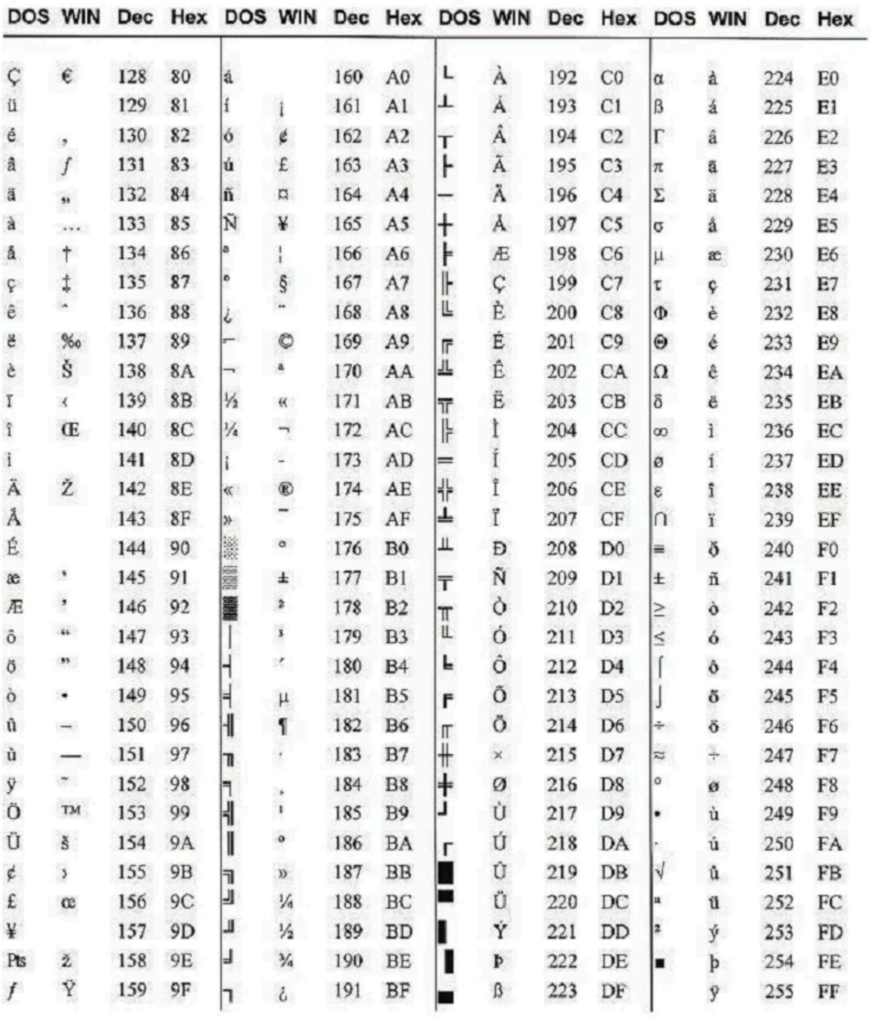
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20	<SPACE>	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	“	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	’	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	\	124	7C	|
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	^	126	7E	~
63	3F	?	95	5F	_	127	7F	<DELETE>

Consider the uppercase and lowercase codes in binary of characters. For example,

'a'	1	1	0	0	0	0	1	hex 61 (lower case)
'A'	1	0	0	0	0	0	1	hex 41 (upper case)
'y'	1	1	1	1	0	0	1	hex 79 (lower case)
'Y'	1	0	1	1	0	0	1	hex 59 (upper case)

The above examples show that the sixth bit changes from 1 to 0 when comparing the lowercase and uppercase of a character. This makes the conversion between the two an easy operation. It is also noticeable that the character sets (e.g. a to z, 0 to 9, etc.) are grouped together in sequence, which speeds up usability.

Extended ASCII uses 8-bit codes (0 to 255 in denary or 0 to FF in hexadecimal). This gives another 128 codes to allow for characters in non-English alphabets and for some graphical characters to be included:



DOS	WIN	Dec	Hex	DOS	WIN	Dec	Hex	DOS	WIN	Dec	Hex	DOS	WIN	Dec	Hex
Ç	€	128	80	á		160	A0	└	À	192	C0	α	à	224	E0
ü		129	81	í	¡	161	A1	┴	Á	193	C1	ß	á	225	E1
é	‚	130	82	ó	¢	162	A2	┬	Â	194	C2	Γ	â	226	E2
â	ƒ	131	83	ú	£	163	A3	├	Ã	195	C3	π	ã	227	E3
ä	„	132	84	ñ	¤	164	A4	─	Ä	196	C4	Σ	ä	228	E4
à	…	133	85	Ñ	¥	165	A5	┼	Å	197	C5	σ	å	229	E5
å	†	134	86	ª	¦	166	A6	╞	Æ	198	C6	µ	æ	230	E6
ç	‡	135	87	º	§	167	A7	╟	Ç	199	C7	τ	ç	231	E7
ê	ˆ	136	88	¿	¨	168	A8	╚	È	200	C8	Φ	è	232	E8
ë	‰	137	89	⌐	©	169	A9	╔	É	201	C9	Θ	é	233	E9
è	Š	138	8A	¬	ª	170	AA	╩	Ê	202	CA	Ω	ê	234	EA
ï	‹	139	8B	½	«	171	AB	╦	Ë	203	CB	δ	ë	235	EB
î	Œ	140	8C	¼	¬	172	AC	╠	Ì	204	CC	∞	ì	236	EC
ì		141	8D	¡	-	173	AD	═	Í	205	CD	ø	í	237	ED
Ä	Ž	142	8E	«	®	174	AE	╬	Î	206	CE	ε	î	238	EE
Å		143	8F	»	¯	175	AF	╧	Ï	207	CF	∩	ï	239	EF
É		144	90	░	°	176	B0	╨	Ð	208	D0	≡	ð	240	F0
æ	'	145	91	▒	±	177	B1	╤	Ñ	209	D1	±	ñ	241	F1
Æ	'	146	92	▓	²	178	B2	╥	Ò	210	D2	≥	ò	242	F2
ô	"	147	93	│	³	179	B3	╙	Ó	211	D3	≤	ó	243	F3
ö	"	148	94	┤	´	180	B4	╘	Ô	212	D4	⌠	ô	244	F4
ò	•	149	95	╡	µ	181	B5	╒	Õ	213	D5	⌡	õ	245	F5
û	–	150	96	╢	¶	182	B6	╓	Ö	214	D6	÷	ö	246	F6
ù	—	151	97	╖	·	183	B7	╫	×	215	D7	≈	÷	247	F7
ÿ	˜	152	98	╕	¸	184	B8	╪	Ø	216	D8	°	ø	248	F8
Ö	™	153	99	╣	¹	185	B9	┘	Ù	217	D9	•	ù	249	F9
Ü	š	154	9A	║	º	186	BA	┌	Ú	218	DA	·	ú	250	FA
¢	›	155	9B	╗	»	187	BB	█	Û	219	DB	√	û	251	FB
£	œ	156	9C	╝	¼	188	BC	▄	Ü	220	DC	ⁿ	ü	252	FC
¥		157	9D	╜	½	189	BD	▌	Ý	221	DD	²	ý	253	FD
Pts	ž	158	9E	╛	¾	190	BE	▐	Þ	222	DE	■	þ	254	FE
ƒ	Ÿ	159	9F	┐	¿	191	BF	▀	ß	223	DF		ÿ	255	FF

Figure 1.6 Extended ASCII code table

ASCII code has a number of disadvantages. The main disadvantage is that it does not represent characters in non-Western languages, for example Chinese characters. As can be seen in Figure 1.6 where DOS and Windows use different characters for some ASCII codes. For this reason, different methods of coding have been developed over the years. One coding system is called **Unicode**. Unicode can represent all languages of the world, thus supporting many operating systems, search engines and internet browsers used globally. There is overlap with standard ASCII code, since the first 128 (English) characters are the same, but Unicode can support several thousand different characters in total. As can be seen in Table 1.2 and Figure 1.6, ASCII uses one byte to represent a character, whereas Unicode will support up to four bytes per character.

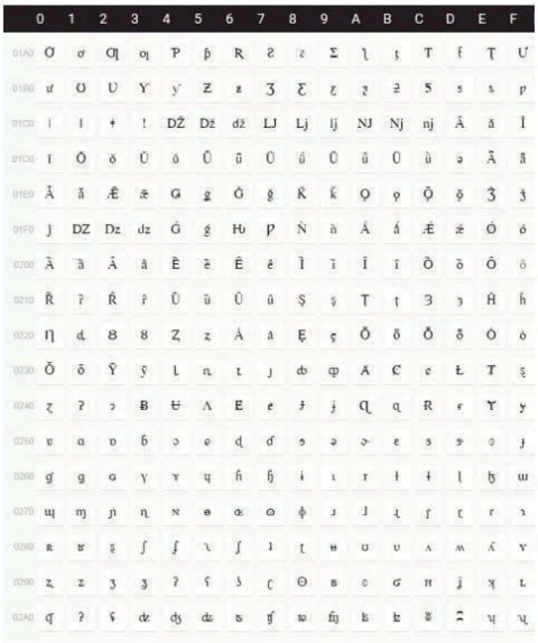
The Unicode consortium was set up in 1991. Version 1.0 was published with five goals; these were to:

- create a universal standard that covered all languages and all writing systems
- produce a more efficient coding system than ASCII
- adopt uniform encoding where each character is encoded as 16-bit or 32-bit code
- create unambiguous encoding where each 16-bit and 32-bit value always represents the same character
- reserve part of the code for private use to enable a user to assign codes for their own characters and symbols (useful for Chinese and Japanese character sets, for example).

Find out more

DOS appears in the ASCII extended code table. Find out what is meant by DOS and why it needs to have an ASCII code value.

A sample of Unicode characters are shown in Figure 1.7. As can be seen from the figure, characters used in languages such as Russian, Romanian and Croatian can now be represented in a computer).



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
01A0	Ơ	ơ	Ƣ	ƣ	Ƥ	ƥ	Ʀ	Ƨ	ƨ	Ʃ	ƪ	ƫ	Ƭ	ƭ	Ʈ	Ư
01B0	ư	Ʊ	Ʋ	Ƴ	ƴ	Ƶ	ƶ	Ʒ	Ƹ	ƹ	ƺ	ƻ	Ƽ	ƽ	ƾ	ƿ
01C0	ǀ	ǁ	ǂ	ǃ	Ǆ	ǅ	ǆ	Ǉ	ǈ	ǉ	Ǌ	ǋ	ǌ	Ǎ	ǎ	Ǐ
01D0	ǐ	Ǒ	ǒ	Ǔ	ǔ	Ǖ	ǖ	Ǘ	ǘ	Ǚ	ǚ	Ǜ	ǜ	ǝ	Ǟ	ǟ
01E0	Ǡ	ǡ	Ǣ	ǣ	Ǥ	ǥ	Ǧ	ǧ	Ǩ	ǩ	Ǫ	ǫ	Ǭ	ǭ	Ǯ	ǯ
01F0	ǰ	Ǳ	ǲ	ǳ	Ǵ	ǵ	Ƕ	Ƿ	Ǹ	ǹ	Ǻ	ǻ	Ǽ	ǽ	Ǿ	ǿ
0200	Ȁ	ȁ	Ȃ	ȃ	Ȅ	ȅ	Ȇ	ȇ	Ȉ	ȉ	Ȋ	ȋ	Ȍ	ȍ	Ȏ	ȏ
0210	Ȑ	ȑ	Ȓ	ȓ	Ȕ	ȕ	Ȗ	ȗ	Ș	ș	Ț	ț	Ȝ	ȝ	Ȟ	ȟ
0220	Ƞ	ȡ	Ȣ	ȣ	Ȥ	ȥ	Ȧ	ȧ	Ȩ	ȩ	Ȫ	ȫ	Ȭ	ȭ	Ȯ	ȯ
0230	Ȱ	ȱ	Ȳ	ȳ	ȴ	ȵ	ȶ	ȷ	ȸ	ȹ	Ⱥ	Ȼ	ȼ	Ƚ	Ⱦ	ȿ
0240	ɀ	Ɂ	ɂ	Ƀ	Ʉ	Ʌ	Ɇ	ɇ	Ɉ	ɉ	Ɋ	ɋ	Ɍ	ɍ	Ɏ	ɏ
0250	ɐ	ɑ	ɒ	ɓ	ɔ	ɕ	ɖ	ɗ	ɘ	ə	ɚ	ɛ	ɜ	ɝ	ɞ	ɟ
0260	ɠ	ɡ	ɢ	ɣ	ɤ	ɥ	ɦ	ɧ	ɨ	ɩ	ɪ	ɫ	ɬ	ɭ	ɮ	ɯ
0270	ɰ	ɱ	ɲ	ɳ	ɴ	ɵ	ɶ	ɷ	ɸ	ɹ	ɺ	ɻ	ɼ	ɽ	ɾ	ɿ
0280	ʀ	ʁ	ʂ	ʃ	ʄ	ʅ	ʆ	ʇ	ʈ	ʉ	ʊ	ʋ	ʌ	ʍ	ʎ	ʏ
0290	ʐ	ʑ	ʒ	ʓ	ʔ	ʕ	ʖ	ʗ	ʘ	ʙ	ʚ	ʛ	ʜ	ʝ	ʞ	ʟ
02A0	ʠ	ʡ	ʢ	ʣ	ʤ	ʥ	ʦ	ʧ	ʨ	ʩ	ʪ	ʫ	ʬ	ʭ	ʮ	ʯ

▲ **Figure 1.7** Sample of Unicode characters