ASCII 码表

ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符
0	NUL	32	(space)	64	@	96	、
1	SOH	33	!	65	A	97	a
2	STX	34	”	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	,	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DCI	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	X	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	—	127	DEL

NUL	VT 垂直制表	SYN 空转同步
SOH 标题开始	FF 走纸控制	ETB 信息组传送结束
STX 正文开始	CR 回车	CAN 作废
ETX 正文结束	SO 移位输出	EM 纸尽

EOY	传输结束	SI	移位输入	SUB	换置
ENQ	询问字符	DLE	空格	ESC	换码
ACK	承认	DC1	设备控制 1	FS	文字分隔符
BEL	报警	DC2	设备控制 2	GS	组分隔符
BS	退一格	DC3	设备控制 3	RS	记录分隔符
HT	横向列表	DC4	设备控制 4	US	单元分隔符
LF	换行	NAK	否定	DEL	删除

键盘常用 ASCII 码

ESC 键 VK_ESCAPE (27)
回车键： VK_RETURN (13)
TAB 键： VK_TAB (9)
Caps Lock 键： VK_CAPITAL (20)
Shift 键： VK_SHIFT ($10)
Ctrl 键： VK_CONTROL (17)
Alt 键： VK_MENU (18)
空格键： VK_SPACE ($20/32)
退格键： VK_BACK (8)
左徽标键： VK_LWIN (91)
右徽标键： VK_LWIN (92)
鼠标右键快捷键：VK_APPS (93)
Insert 键： VK_INSERT (45)
Home 键： VK_HOME (36)
Page Up： VK_PRIOR (33)
PageDown： VK_NEXT (34)
End 键： VK_END (35)
Delete 键： VK_DELETE (46)

方向键(←)： VK_LEFT (37)
方向键(↑)： VK_UP (38)
方向键(→)： VK_RIGHT (39)
方向键(↓)： VK_DOWN (40)

F1 键： VK_F1 (112)
F2 键： VK_F2 (113)
F3 键： VK_F3 (114)
F4 键： VK_F4 (115)
F5 键： VK_F5 (116)
F6 键： VK_F6 (117)
F7 键： VK_F7 (118)
F8 键： VK_F8 (119)
F9 键： VK_F9 (120)
F10 键： VK_F10 (121)
F11 键： VK_F11 (122)
F12 键： VK_F12 (123)

Num Lock 键： VK_NUMLOCK (144)

小键盘 0： VK_NUMPAD0 (96)
小键盘 1： VK_NUMPAD0 (97)
小键盘 2： VK_NUMPAD0 (98)
小键盘 3： VK_NUMPAD0 (99)
小键盘 4： VK_NUMPAD0 (100)
小键盘 5： VK_NUMPAD0 (101)
小键盘 6： VK_NUMPAD0 (102)
小键盘 7： VK_NUMPAD0 (103)
小键盘 8： VK_NUMPAD0 (104)
小键盘 9： VK_NUMPAD0 (105)
小键盘.： VK_DECIMAL (110)
小键盘*： VK_MULTIPLY (106)
小键盘+： VK_MULTIPLY (107)
小键盘-： VK_SUBTRACT (109)
小键盘/： VK_DIVIDE (111)

Pause Break 键： VK_PAUSE (19)
Scroll Lock 键： VK_SCROLL (145)

ASCII 非打印控制字符表

ASCII 表上的数字 0–31 分配给了控制字符，用于控制像打印机等一些外围设备。例如，12 代表换页/新页功能。此命令指示打印机跳到下一页的开头

十进制	十六进制	字符		十进制	十六进制	字符
0	00	空		16	10	数据链路转意
1	01	头标开始		17	11	设备控制 1
2	02	正文开始		18	12	设备控制 2
3	03	正文结束		19	13	设备控制 3
4	04	传输结束		20	14	设备控制 4
5	05	查询		21	15	反确认
6	06	确认		22	16	同步空闲
7	07	震铃		23	17	传输块结束
8	08	backspace		24	18	取消
9	09	水平制表符		25	19	媒体结束
10	0A	换行/新行		26	1A	替换
11	0B	竖直制表符		27	1B	转意
12	0C	换页/新页		28	1C	文件分隔符
13	0D	回车		29	1D	组分隔符
14	0E	移出		30	1E	记录分隔符
15	0F	移入		31	1F	单元分隔符

ASCII 打印字符

数字 32–126 分配给了能在键盘上找到的字符，当您查看或打印文档时就会出现。数字 127 代表 DELETE 命令。

ASCII 打印字符表

十进制	十六进制	字符		十进制	十六进制	字符
32	20	space		80	50	P
33	21	!		81	51	Q
34	22	"		82	52	R
35	23	#		83	53	S
36	24	$		84	54	T
37	25	%		85	55	U
38	26	&		86	56	V
39	27	'		87	57	w
40	28	(		88	58	X
41	29	)		89	59	Y
42	2A	*		90	5A	Z
43	2B	+		91	5B	[
44	2C	,		92	5C	\
45	2D	-		93	5D	]
46	2E	.		94	5E	^
47	2F	/		95	5F	_
48	30	0		96	60	`
49	31	1		97	61	a
50	32	2		98	62	b
51	33	3		99	63	c
52	34	4		100	64	d
53	35	5		101	65	e
54	36	6		102	66	f
55	37	7		103	67	g
56	38	8		104	68	h
57	39	9		105	69	i
58	3A	:		106	6A	j
59	3B	;		107	6B	k
60	3C	<		108	6C	l

61	3D	=		109	6D	m
62	3E	>		110	6E	n
63	3F	?		111	6F	o
64	40	@		112	70	p
65	41	A		113	71	q
66	42	B		114	72	r
67	43	C		115	73	s
68	44	D		116	74	t
69	45	E		117	75	u
70	46	F		118	76	v
71	47	G		119	77	w
72	48	H		120	78	x
73	49	I		121	79	y
74	4A	J		122	7A	z
75	4B	K		123	7B	{
76	4C	L		124	7C	|
77	4D	M		125	7D	}
78	4E	N		126	7E	~
79	4F	O		127	7F	DEL

扩展 ASCII 打印字符

扩展的 ASCII 字符满足了对更多字符的需求。扩展的 ASCII 包含 ASCII 中已有的 128 个字符（数字 0–32 显示在下图中），又增加了 128 个字符，总共是 256 个。即使有了这些更多的字符，许多语言还是包含无法压缩到 256 个字符中的符号。因此，出现了一些 ASCII 的变体来囊括地区性字符和符号。例如，许多软件程序把 ASCII 表（又称作 ISO 8859-1）用于北美、西欧、澳大利亚和非洲的语言。

扩展的 ASCII 打印字符表

190

十进制	十六进制	字符		十进制	十六进制	字符
128	80	Ç		192	C0	└
129	81	ü		193	C1	┴
130	82	é		194	C2	┬
131	83	â		195	C3	├
132	84	ä		196	C4	─
133	85	à		197	C5	┼
134	86	å		198	C6	╞
135	87	ç		199	C7	╟
136	88	ê		200	C8	╚

137	89	ë		201	C9	╔
138	8A	è		202	CA	╩
139	8B	ï		203	CB	╦
140	8C	î		204	CC	╠
141	8D	ì		205	CD	═
142	8E	Ä		206	CE	╬
143	8F	Å		207	CF	╧
144	90	É		208	D0	╨
145	91	æ		209	D1	╤
146	92	Æ		210	D2	╥
147	93	ô		211	D3	╙
148	94	ö		212	D4	Ô
149	95	ò		213	D5	╒
150	96	û		214	D6	╓
151	97	ù		215	D7	╫
152	98	ÿ		216	D8	╪
153	99	Ö		217	D9	┘
154	9A	Ü		218	DA	┌
155	9B	¢		219	DB	█
156	9C	£		220	DC	▄
157	9D	¥		221	DD	▌
158	9E	₧		222	DE	▐
159	9F	*ƒ*		223	DF	▀
160	A0	á		224	E0	α
161	A1	í		225	E1	ß
162	A2	ó		226	E2	Γ
163	A3	ú		227	E3	π
164	A4	ñ		228	E4	Σ
165	A5	Ñ		229	E5	σ
166	A6	ª		230	E6	µ
167	A7	º		231	E7	τ
168	A8	¿		232	E8	Φ
169	A9	⌐		233	E9	Θ
170	AA	¬		234	EA	Ω
171	AB	½		235	EB	δ
172	AC	¼		236	EC	∞
173	AD	¡		237	ED	φ

174	AE	«		238	EE	ε
175	AF	»		239	EF	∩
176	B0	░		240	F0	≡
177	B1	▒		241	F1	±
178	B2	▓		242	F2	≥
179	B3	│		243	F3	≤
180	B4	┤		244	F4	⌠
181	B5	╡		245	F5	⌡
182	B6	╢		246	F6	÷
183	B7	╖		247	F7	≈
184	B8	╕		248	F8	≈
185	B9	╣		249	F9	·
186	BA	║		250	FA	·
187	BB	╗		251	FB	√
188	BC	╝		252	FC	ⁿ
189	BD	╜		253	FD	²
190	BE	╛	254	FE	■	
191	BF	┐		255	FF	ÿ

ASCII 码对照表

下表列出了字符集中的 0 - 127。

代码	字符	代码	字符	代码	字符	代码	字符
0		32	[空格]	64	@	96	`
1		33	!	65	A	97	a
2		34	″	66	B	98	b
3		35	#	67	C	99	c
4		36	$	68	D	100	d
5		37	%	69	E	101	e
6		38	&	70	F	102	f
7		39	’	71	G	103	g
8	**	40	(	72	H	104	h
9	**	41	)	73	I	105	i
10	**	42	*	74	J	106	j
11		43	+	75	K	107	k
12		44	,	76	L	108	l
13	**	45	-	77	M	109	m

14		46	.	78	N	110	n
15	□	47	/	79	O	111	o
16	□	48	0	80	P	112	p
17	□	49	1	81	Q	113	q
18	□	50	2	82	R	114	r
19	‼	51	3	83	S	115	s
20	¶	52	4	84	T	116	t
21	⊥	53	5	85	U	117	u
22	□	54	6	86	V	118	v
23	□	55	7	87	W	119	w
24	□	56	8	88	X	120	x
25	□	57	9	89	Y	121	y
26	□	58	:	90	Z	122	z
27	←	59	;	91	[	123	{
28	□	60	<	92	\	124	|
29	□	61	=	93	]	125	}
30	-	62	>	94	^	126	~
31		63	?	95	_	127	□

下表列出了字符集中的 128 - 255。

代码	字符	代码	字符	代码	字符	代码	字符
128	€	160	[空格]	192	À	224	à
129	□	161	¡	193	Á	225	á
130	,	162	¢	194	Â	226	â
131	*f*	163	£	195	Ã	227	ã
132	„	164	¤	196	Ä	228	ä
133	…	165	¥	197	Å	229	å
134	†	166	¦	198	Æ	230	æ
135	‡	167	§	199	Ç	231	ç
136	ˆ	168	¨	200	È	231	ç
137	‰	169	©	201	É	232	è
138	Š	170	ª	202	Ê	233	é
139	‹	171	«	203	Ë	234	ê
140	Œ	172	¬	204	Ì	235	ë
141	□	173		205	Í	236	ì
142	Ž	174	®	206	Î	237	í
143	□	175	¯	207	Ï	238	î
144	□	176	°	208	Ð	239	ï
145	‘	177	±	209	Ñ	240	ð
146	’	178	²	210	Ò	241	ñ
147	“	179	³	211	Ó	242	ò
148	”	180	´	212	Ô	243	ó
149	•	181	µ	213	Õ	244	ô
150	-	182	¶	214	Ö	245	õ

151	—	183	•	215	×	246	ö
152	˜	184	¸	216	Ø	247	÷
153	™	185	¹	217	Ù	248	ø
154	š	186	º	218	Ú	249	ù
155	›	187	»	219	Û	250	ú
156	œ	188	¼	220	Ü	251	û
157	□	189	½	221	Ý	252	ü
158	ž	190	¾	222	Þ	253	ý
159	Ÿ	191	¿	223	ß	254	þ

** 数值 8、9、10 和 13 可以分别转换为退格符、制表符、换行符和回车符。这些字符都没有图形表示，但是对于不同的应用程序，这些字符可能会影响文本的显示效果。

"空"表示在当前平台上不支持的字符。

ASCII

百科名片

ASCII（American Standard Code for Information Interchange，美国信息互换标准代码）是基于拉丁字母的一套电脑编码系统。它主要用于显示现代英语和其他西欧语言。它是现今最通用的单字节编码系统，并等同于国际标准 ISO/IEC 646。

目录

名称
产生
简介
标准 ASCII 表
常见 ASCII 码的大小规则
查询 ASCII 技巧
字符集简史
ASCII 国际问题

- 扩展 ASCII
- 双字节字符集
- 虚拟键盘按键的 ASCII 值
- ASCII 码的算法
- 汉字编码

展开

编辑本段名称

美国信息交换标准代码
(American Standard Code for Information Interchange, ASCII)

编辑本段产生

在计算机中，所有的数据在存储和运算时都要使用二进制数表示（因为计算机用高电平和低电平分别表示 1 和 0），例如，象 a、b、c、d 这样的 52 个字母（包括大写）、以及 0、1 等数字还有一些常用的符号（例如*、#、@等）在计算机中存储时也要使用二进制数来表示，而具体用哪些二进制数字表示哪个符号，当然每个人都可以约定自己的一套（这就叫编码），而大家如果要想互相通信而不造成混乱，那么大家就必须使用相同的编码规则，于是美国有关的标准化组织就出台了所谓的ASCII 编码，统一规定了上述常用符号用哪些二进制数来表示。

美国标准信息交换代码是由美国国家标准学会(American National Standard Institute ，ANSI)制定的，标准的单字节字符编码方案，用于基于文本的数据。起始于 50 年代后期，在 1967 年定案。它最初是美国国家标准，供不同计算机在相互通信时用作共同遵守的西文字符编码标准，它已被国际标准化组织（International Organization for Standardization, ISO）定为国际标准，称为 ISO 646 标准。适用于所有拉丁文字字母。

编辑本段简介

ASCII 码使用指定的 7 位或 8 位二进制数组合来表示 128 或 256 种可能的字符。标准 ASCII 码也叫基础 ASCII 码，使用 7 位二进制数来表示所有的大写和小写字母，数字 0 到 9、标点符号， 以及在美式英语中使用的特殊控制字符。其中：

0～31 及 127(共 33 个)是控制字符或通信专用字符（其余为可显示字符），如控制符：LF（换行）、CR（回车）、FF（换页）、DEL（删除）、BS（退格）、BEL（振铃）等；通信专用字符：SOH（文头）、EOT（文尾）、ACK（确认）等；ASCII 值为 8、9、10 和 13 分别转换为退格、制表、换行和回车字符。它们并没有特定的图形显示，但会依不同的应用程序，而对文本显示有不同的影响。

32～126(共 95 个)是字符(32sp 是空格），其中 48～57 为 0 到 9 十个阿拉伯数字；

65～90 为 26 个大写英文字母，97～122 号为 26 个小写英文字母，其余为一些标点符号、运算符号等。

同时还要注意，在标准 ASCII 中，其最高位(b7)用作奇偶校验位。所谓奇偶校验，是指在代码传送过程中用来检验是否出现错误的一种方法，一般分奇校验和偶校验两种。奇校验规定：正确的代码一个字节中 1 的个数必须是奇数，若非奇数，则在最高位 b7 添 1；偶校验规定：正确的代码一个字节中 1 的个数必须是偶数，若非偶数，则在最高位 b7 添 1。

后 128 个称为扩展 ASCII码，目前许多基于x86的系统都支持使用扩展（或“高”）ASCII。扩展 ASCII 码允许将每个字符的第 8 位用于确定附加的 128 个特殊符号字符、外来语字母和图形符号。

编辑本段标准 ASCII 表

Bin	Dec	Hex	缩写/字符	解释
00000000	0	00	NUL(null)	空字符
00000001	1	01	SOH(start of headling)	标题开始
00000010	2	02	STX (start of text)	正文开始
00000011	3	03	ETX (end of text)	正文结束
00000100	4	04	EOT (end of transmission)	传输结束
00000101	5	05	ENQ (enquiry)	请求
00000110	6	06	ACK (acknowledge)	收到通知
00000111	7	07	BEL (bell)	响铃
00001000	8	08	BS (backspace)	退格
00001001	9	09	HT (horizontal tab)	水平制表符
00001010	10	0A	LF (NL line feed, new line)	换行键
00001011	11	0B	VT (vertical tab)	垂直制表符
00001100	12	0C	FF (NP form feed, new page)	换页键
00001101	13	0D	CR (carriage return)	回车键
00001110	14	0E	SO (shift out)	不用切换
00001111	15	0F	SI (shift in)	启用切换
00010000	16	10	DLE (data link escape)	数据链路转义

00010001	17	11	DC1 (device control 1)	设备控制 1
00010010	18	12	DC2 (device control 2)	设备控制 2
00010011	19	13	DC3 (device control 3)	设备控制 3
00010100	20	14	DC4 (device control 4)	设备控制 4
00010101	21	15	NAK (negative acknowledge)	拒绝接收
00010110	22	16	SYN (synchronous idle)	同步空闲
00010111	23	17	ETB (end of trans. block)	传输块结束
00011000	24	18	CAN (cancel)	取消
00011001	25	19	EM (end of medium)	介质中断
00011010	26	1A	SUB (substitute)	替补
00011011	27	1B	ESC (escape)	溢出
00011100	28	1C	FS (file separator)	文件分割符
00011101	29	1D	GS (group separator)	分组符
00011110	30	1E	RS (record separator)	记录分离符
00011111	31	1F	US (unit separator)	单元分隔符
00100000	32	20	(space)	空格
00100001	33	21	!	
00100010	34	22	″	
00100011	35	23	#	
00100100	36	24	$	
00100101	37	25	%	
00100110	38	26	&	

00100111	39	27	’	
00101000	40	28	(	
00101001	41	29	)	
00101010	42	2A	*	
00101011	43	2B	+	
00101100	44	2C	,	
00101101	45	2D	-	
00101110	46	2E	.	
00101111	47	2F	/	
00110000	48	30	0	

续表

00110001	49	31	1	
00110010	50	32	2	
00110011	51	33	3	
00110100	52	34	4	
00110101	53	35	5	
00110110	54	36	6	
00110111	55	37	7	
00111000	56	38	8	
00111001	57	39	9	
00111010	58	3A	:	
00111011	59	3B	;	

00111100	60	3C	<	
00111101	61	3D	=	
00111110	62	3E	>	
00111111	63	3F	?	
01000000	64	40	@	
01000001	65	41	A	
01000010	66	42	B	
01000011	67	43	C	
01000100	68	44	D	
01000101	69	45	E	
01000110	70	46	F	
01000111	71	47	G	
01001000	72	48	H	
01001001	73	49	I	
01001010	74	4A	J	
01001011	75	4B	K	
01001100	76	4C	L	
01001101	77	4D	M	
01001110	78	4E	N	
01001111	79	4F	O	
01010000	80	50	P	
01010001	81	51	Q	

01010010	82	52	R	
01010011	83	53	S	
01010100	84	54	T	
01010101	85	55	U	
01010110	86	56	V	
01010111	87	57	W	
01011000	88	58	X	
01011001	89	59	Y	
01011010	90	5A	Z	
01011011	91	5B	[	
01011100	92	5C	\	
01011101	93	5D	]	
01011110	94	5E	^	
01011111	95	5F	_	
01100000	96	60	`	
01100001	97	61	a	
01100010	98	62	b	

续表

01100011	99	63	c	
01100100	100	64	d	
01100101	101	65	e	
01100110	102	66	f	

01100111	103	67	g	
01101000	104	68	h	
01101001	105	69	i	
01101010	106	6A	j	
01101011	107	6B	k	
01101100	108	6C	l	
01101101	109	6D	m	
01101110	110	6E	n	
01101111	111	6F	o	
01110000	112	70	p	
01110001	113	71	q	
01110010	114	72	r	
01110011	115	73	s	
01110100	116	74	t	
01110101	117	75	u	
01110110	118	76	v	
01110111	119	77	w	
01111000	120	78	x	
01111001	121	79	y	
01111010	122	7A	z	
01111011	123	7B	{	
01111100	124	7C	|	

01111101	125	7D	}	
01111110	126	7E	~	
01111111	127	7F	DEL (delete)	删除

八进制	十六进制	十进制	字符	八进制	十六进制	十进制	**字符**
0	0	0	nul	100	40	64	@
1	1	1	soh	101	41	65	A
2	2	2	stx	102	42	66	B
3	3	3	etx	103	43	67	C
4	4	4	eot	104	44	68	D
5	5	5	enq	105	45	69	E
6	6	6	ack	106	46	70	F
7	7	7	bel	107	47	71	G
10	8	8	bs	110	48	72	H
11	9	9	ht	111	49	73	I
12	0a	10	nl	112	4a	74	J
13	0b	11	vt	113	4b	75	K
14	0c	12	ff	114	4c	76	L
15	0d	13	er	115	4d	77	M
16	0e	14	so	116	4e	78	N
17	0f	15	si	117	4f	79	O

20	10	16	dle	120	50	80	P
21	11	17	dc1	121	51	81	Q
22	12	18	dc2	122	52	82	R
23	13	19	dc3	123	53	83	S
24	14	20	dc4	124	54	84	T
25	15	21	nak	125	55	85	U
26	16	22	syn	126	56	86	V
27	17	23	etb	127	57	87	W
30	18	24	can	130	58	88	X
31	19	25	em	131	59	89	Y
32	1a	26	sub	132	5a	90	Z
33	1b	27	esc	133	5b	91	[
34	1c	28	fs	134	5c	92	\
35	1d	29	gs	135	5d	93	]
36	1e	30	re	136	5e	94	^
37	1f	31	us	137	5f	95	_
40	20	32	sp	140	60	96	’
41	21	33	!	141	61	97	a
42	22	34	″	142	62	98	b
43	23	35	#	143	63	99	c
44	24	36	$	144	64	100	d
45	25	37	%	145	65	101	e

46	26	38	&	146	66	102	f
47	27	39	`	147	67	103	g
50	28	40	(	150	68	104	h
51	29	41	)	151	69	105	i
52	2a	42	*	152	6a	106	j
53	2b	43	+	153	6b	107	k
54	2c	44	,	154	6c	108	l
55	2d	45	-	155	6d	109	m
56	2e	46	.	156	6e	110	n
57	2f	47	/	157	6f	111	o
60	30	48	0	160	70	112	p
61	31	49	1	161	71	113	q
62	32	50	2	162	72	114	r
63	33	51	3	163	73	115	s
64	34	52	4	164	74	116	t
65	35	53	5	165	75	117	u
66	36	54	6	166	76	118	v
67	37	55	7	167	77	119	w
70	38	56	8	170	78	120	x
71	39	57	9	171	79	121	y
72	3a	58	:	172	7a	122	z
73	3b	59	;	173	7b	123	{

74	3c	60	<	174	7c	124	|
75	3d	61	=	175	7d	125	}
76	3e	62	>	176	7e	126	~
77	3f	63	?	177	7f	127	del

编辑本段常见 ASCII 码的大小规则

0~9<A~Z<a~z

1）数字比字母要小。如 “7”<“F”；

2）数字 0 比数字 9 要小，并按 0 到 9 顺序递增。如 “3”<“8” ；

3）字母 A 比字母 Z 要小，并按 A 到 Z 顺序递增。如“A”<“Z” ；

4）同个字母的大写字母比小写字母要小 32。如“A”<“a” 。

记住几个常见字母的 ASCII 码大小：

换行 LF 为 0x0A；回车 CR 为 0x0D；空格为 0x20；“0”为 0x30； “A”为 0x41；“a”为 0x61。

另外还有 128-255 的 ASCII 字符

编辑本段查询 ASCII 技巧

方便查询 ACSII 码对应的字符：新建一个文本文档，按住 ALT+要查询的码值（注意，这里是十进制）松开即可显示出对应字符。例如：按住 ALT+97,则会显示出‘a’。

编辑本段字符集简史

6000 年前 象形文字

3000 年前 字母表

1838 年到 1854 年 Samuel F. B. Morse 发明了电报,字母表中的每个字符对应于一系列短的和长的脉冲

1821 年到 1824 年 Louis Braille 发明盲文，6 位代码，它把字符、常用字母组合、常用单字和标点进行编码。

一个特殊的 escape 代码表示后续的字符代码应解释为大写。一个特殊的 shift 代码允许后续代码被解释为数字。

1931 年 CCITT 标准化 Telex 代码，包括 Baudot #2 的代码，都是包括字符和数字的 5 位代码。

1890 年 早期计算机的字符码是从 Hollerith 卡片,6 位字符码系统 BCDIC（Binary-Coded Decimal Interchange Code：二进制编码十进制交换编码）

60 年代 扩展为 8 位 EBCDIC,IBM 大型主机的标准

1967 年 美国信息交换标准码（ASCII：American Standard Code for Information Interchange）

在字符长度是 6 位、7 位还是 8 位的问题上产生了很大的争议。从可靠性的观点来看不应使用替换字符，

因此 ASCII 不能是 6 位编码，但由于费用的原因也排除了 8 位版本的方案（当时每位的储存空间成本仍很昂贵）。

这样，最终的字符码就有 26 个小写字母、26 个大写字母、10 个数字、32 个符号、33 个句柄和一个空格，总共 128 个字符码。

ASCII 现在记录在 ANSI X3.4-1986字符集－用于信息交换的 7 位美国国家标准码（7-Bit ASCII：7-Bit American National

Standard Code for Information Interchange），由美国国家标准协会（American National Standards Institute）发布。

图 2-1 中所示的 ASCII 字符码与 ANSI 文件中的格式相似。

编辑本段ASCII 国际问题

ASCII 是美国标准，所以它不能良好满足其它讲英语国家的需要。例如英国的英镑符号（£）在哪里？

拉丁语字母表重音符号

使用斯拉夫字母表的希腊语、希伯来语、阿拉伯语和俄语。

汉字系统的中国象形汉字，日本和朝鲜。

1967 年，国际标准化组织（ISO：International Standards Organization）推荐一个 ASCII 的变种，

代码 0x40、0x5B、0x5C、0x5D、0x7B、0x7C 和 0x7D“为国家使用保留”，而代码 0x5E、0x60 和 0x7E 标为

“当国内要求的特殊字符需要 8、9 或 10 个空间位置时，可用于其它图形符号”。这显然不是一个最佳的国际解决方案，

因为这并不能保证一致性。但这却显示了人们如何想尽办法为不同的语言来编码的。

编辑本段扩展 ASCII

1981 年 IBM PC ROM256 个字符的字符集，即 IBM 扩展字符集

1985 年 11 Windows 字符集被称作“ANSI 字符集”,遵循了 ANSI 草案和 ISO 标准(ANSI/ISO 8859-1-1987,简“Latin 1”。

ASCII 扩展字符表

[1]

ANSI 字符集的最初版本:

1987 年 4 月代码页 437,字符的映像代码,出现在MS-DOS 3.3

扩展 ASCII 字符是从 128 到 255（0x7f-0xff）的字符。

编辑本段双字节字符集

双字节字符集（DBCS：double-byte character set）,解决中国、日本和韩国的象形文字符和 ASCII 的某种兼容性。

DBCS 从 256 代码开始，就像 ASCII 一样。与任何行为良好的代码页一样，最初的 128 个代码是 ASCII。

然而，较高的 128 个代码中的某些总是跟随着第二个字节。

这两个字节一起（称作首字节和跟随字节）定义一个字符，通常是一个复杂的象形文字。

编辑本段虚拟键盘按键的 ASCII 值

ESC 键 VK_ESCAPE (27)
回车键： VK_RETURN (13)
TAB 键： VK_TAB (9)
Caps Lock 键： VK_CAPITAL (20)
Shift 键： VK_SHIFT (16)
Ctrl 键： VK_CONTROL (17)
Alt 键： VK_MENU (18)
空格键： VK_SPACE (32)
退格键： VK_BACK (8)
左徽标键： VK_LWIN (91)
右徽标键： VK_LWIN (92)
鼠标右键快捷键：VK_APPS (93)
Insert 键： VK_INSERT (45)
Home 键： VK_HOME (36)
Page Up： VK_PRIOR (33)
PageDown： VK_NEXT (34)
End 键： VK_END (35)
Delete 键： VK_DELETE (46)
方向键(←)： VK_LEFT (37)
方向键(↑)： VK_UP (38)
方向键(→)： VK_RIGHT (39)
方向键(↓)： VK_DOWN (40)
F1 键： VK_F1 (112)
F2 键： VK_F2 (113)
F3 键： VK_F3 (114)
F4 键： VK_F4 (115)
F5 键： VK_F5 (116)
F6 键： VK_F6 (117)
F7 键： VK_F7 (118)
F8 键： VK_F8 (119)
F9 键： VK_F9 (120)
F10 键： VK_F10 (121)
F11 键： VK_F11 (122)
F12 键： VK_F12 (123)
Num Lock 键： VK_NUMLOCK (144)
小键盘 0： VK_NUMPAD0 (96)
小键盘 1： VK_NUMPAD1 (97)
小键盘 2： VK_NUMPAD2 (98)
小键盘 3： VK_NUMPAD3 (99)
小键盘 4： VK_NUMPAD4 (100)
小键盘 5： VK_NUMPAD5 (101)
小键盘 6： VK_NUMPAD6 (102)
小键盘 7： VK_NUMPAD7 (103)

小键盘 8： VK_NUMPAD8 (104)
小键盘 9： VK_NUMPAD9 (105)
小键盘.： VK_DECIMAL (110)
小键盘*： VK_MULTIPLY (106)
小键盘+： VK_ADD (107)
小键盘-： VK_SUBTRACT (109)
小键盘/： VK_DIVIDE (111)
Pause Break 键： VK_PAUSE (19)
Scroll Lock 键： VK_SCROLL (145)

编辑本段ASCII 码的算法

A 在 ascii 中定义为 01000001，也就是十进制 65，有了这个标准后，当我们输入 A 时，计算机就可以通过 ascii 码知道输入的字符的二进制编码是 01000001。而没有这样的标准，我们就必须自己想办法告诉计算机我们输入了一个 A；没有这样的标准，我们在别的机器上就需要重新编码以告诉计算机我们要输入 A。ascii 码指的不是十进制，是二进制。只是用十进制表示习惯一点罢了，比如在 ascii 码中，A 的二进制编码为 01000001，如果用十进制表示是 65，用十六进制表示就是 41H。

在 ascii 码表中，只包括了一些字符、数字、标点符号的信息表示，这主要是因为计算机是美国发明的，在英文下面，我们使用 ascii 表示就足够了！但是在汉字输入下面，用 ascii 码就不能表示了，而汉字只是中国的通用表示，所以如果我们要在计算机中输入汉字，就必须有一个像 ascii 码的标准来表示每一个汉字，这就是中国的汉字国标码，它定义了汉字在计算机中的一个表示标准。通过这个标准，但我们输入汉字的时候，我们的输入码就转换为区位码，通过唯一的区位码得到这个汉字的字形码并显示出来。当然汉字的区位码在计算机中也是用二进制表示的！

二进制数转换为十进制数

二进制数第 0 位的权值是 2 的 0 次方，第 1 位的权值是 2 的 1 次方……
所以，设有一个二进制数：0110 0100，转换为 10 进制为：
下面是竖式：
0110 0100 换算成 十进制
第 0 位 0 * 2^0 = 0
第 1 位 0 * 2^1 = 0
第 2 位 1 * 2^2 = 4
第 3 位 0 * 2^3 = 0
第 4 位 0 * 2^4 = 0
第 5 位 1 * 2^5 = 32
第 6 位 1 * 2^6 = 64
第 7 位 0 * 2^7 = 0

100
用横式计算为：
0 * 2^0 + 0 * 2^1 + 1 * 2^2 +0*2^3+ 0 * 2^4 + 1 * 2^5 + 1 * 2^6 + 0 * 2^7 = 100
0 乘以多少都是 0，所以我们也可以直接跳过值为 0 的位：
1 * 2^2 + 1 * 2^5 + 1 * 2^6 = 100

6.2.2 八进制数转换为十进制数

八进制就是逢 8 进 1。

八进制数采用 0～7 这八数来表达一个数。

八进制数第 0 位的权值为 8 的 0 次方，第 1 位权值为 8 的 1 次方，第 2 位权值为 8 的 2 次方……

所以，设有一个八进制数：1507，转换为十进制为：

用竖式表示：

1507 换算成十进制。

第 0 位 7 * 8^0 = 7

第 1 位 0 * 8^1 = 0

第 2 位 5 * 8^2 = 320

第 3 位 1 * 8^3 = 512

839

同样，我们也可以用横式直接计算：

7 * 8^0 + 0 * 8^1 + 5 * 8^2 + 1 * 8^3 = 839

结果是，八进制数 1507 转换成十进制数为 839

6.2.3 八进制数的表达方法

C,C++语言中，如何表达一个八进制数呢？如果这个数是 876,我们可以断定它不是八进制数，因为八进制数中不可能出 7 以上的阿拉伯数字。但如果这个数是 123、是 567，或 12345670，那么它是八进制数还是 10 进制数，都有可能。

所以,C,C++规定，一个数如果要指明它采用八进制，必须在它前面加上一个 0，如：123 是十进制，但 0123 则表示采用八进制。这就是八进制数在 C、C++中的表达方法。

由于 C 和 C++都没有提供二进制数的表达方法，所以，这里所学的八进制是我们学习的，CtC++语言的数值表达的第二种进制法。

现在，对于同样一个数，比如是 100，我们在代码中可以用平常的 10 进制表达，例如在变量初始化时：

```
int a = 100;
```

我们也可以这样写：

```
int a = 0144; //0144 是八进制的 100；一个 10 进制数如何转成 8 进制，我们后面会学到。
```

千万记住，用八进制表达时，你不能少了最前的那个 0。否则计算机会通通当成 10 进制。不过，有一个地方使用八进制数时，却不能使用加 0，那就是我们前面学的用于表达字符的“转义符”表达法。

6.2.4 八进制数在转义符中的使用

我们学过用一个转义符'\'加上一个特殊字母来表示某个字符的方法，如：'\n'表示换行(line)，而'\t'表示 Tab 字符，'\''则表示单引号。今天我们又学习了一种使用转义符的方法：转义符'\'后面接一个八进制数，用于表示 ASCII 码等于该值的字符。

比如，查一下第 5 章中的 ASCII 码表，我们找到问号字符（?)的 ASCII 值是 63，那么我们可以把它转换为八进值：77，然后用 '\77'来表示'?'。由于是八进制，所以本应写成 '\077'，但因为 C,C++规定不允许使用斜杠加 10 进制数来表示字符，所以这里的 0 可以不写。

事实上我们很少在实际编程中非要用转义符加八进制数来表示一个字符，所以，6.2.4 小节的内容，大家仅仅了解就行。

6.2.5 十六进制数转换成十进制数

2 进制，用两个阿拉伯数字：0、1；

8 进制，用八个阿拉伯数字：0、1、2、3、4、5、6、7；

10 进制，用十个阿拉伯数字：0 到 9；

16 进制，用十六个阿拉伯数字……等等，阿拉伯人或说是印度人，只发明了 10 个数字啊？

16 进制就是逢 16 进 1，但我们只有 0~9 这十个数字，所以我们用 A，B，C，D，E，F 这五个字母来分别表示 10，11，12，13，14，15。字母不区分大小写。

十六进制数的第 0 位的权值为 16 的 0 次方,第 1 位的权值为 16 的 1 次方,第 2 位的权值为 16 的 2 次方……

所以，在第 N（N 从 0 开始）位上，如果是是数 X （X 大于等于 0，并且 X 小于等于 15，即：F）表示的大小为 X * 16 的 N 次方。

假设有一个十六进数 2AF5, 那么如何换算成 10 进制呢？

用竖式计算：

2AF5 换算成 10 进制:

第 0 位： 5 * 16^0 = 5

第 1 位： F * 16^1 = 240

第 2 位： A * 16^2 = 2560

第 3 位： 2 * 16^3 = 8192

10997

直接计算就是：

5 * 16^0 + F * 16^1 + A * 16^2 + 2 * 16^3 = 10997

(别忘了，在上面的计算中，A 表示 10，而 F 表示 15)

现在可以看出，所有进制换算成 10 进制，关键在于各自的权值不同。

假设有人问你，十进数 1234 为什么是 一千二百三十四？你尽可以给他这么一个算式：

1234 = 1 * 10^3 + 2 * 10^2 + 3 * 10^1 + 4 * 10^0

6.2.6 十六进制数的表达方法

如果不使用特殊的书写形式，16 进制数也会和 10 进制相混。随便一个数：9876，就看不出它是 16 进制或 10 进制。

C，C++规定，16 进制数必须以 0x 开头。比如 0x1 表示一个 16 进制数。而 1 则表示一个十进制。另外如：0xff,0xFF,0X102A,等等。其中的 x 也也不区分大小写。(注意：0x 中的 0 是数字 0，而不是字母 O)

以下是一些用法示例：

```
int a = 0x100F;
int b = 0x70 + a;
```

至此，我们学完了所有进制：10 进制，8 进制，16 进制数的表达方式。最后一点很重要，C/C++中，10 进制数有正负之分，比如 12 表示正 12，而-12 表示负 12，；但 8 进制和 16 进制只能用达无符号的正整数，如果你在代码中里：-078，或者写：-0xF2,C,C++并不把它当成一个负数。

6.2.7 十六进制数在转义符中的使用

转义符也可以接一个 16 进制数来表示一个字符。如在 6.2.4 小节中说的 '?' 字符,可以有以下表达方式：

```
'?' //直接输入字符
'\77' //用八进制，此时可以省略开头的 0
'\0x3F' //用十六进制
```

同样，这一小节只用于了解。除了空字符用八进制数 ’\0’ 表示以外，我们很少用后两种方法表示一个字符。

6.3 十进制数转换到二、八、十六进制数

6.3.1 10 进制数转换为 2 进制数

给你一个十进制，比如：6，如果将它转换成二进制数呢？

10 进制数转换成二进制数，这是一个连续除 2 的过程：

把要转换的数，除以 2，得到商和余数，

将商继续除以 2，直到商为 0。最后将所有余数倒序排列，得到数就是转换结果。

听起来有些糊涂？我们结合例子来说明。比如要转换 6 为二进制数。

“把要转换的数，除以 2，得到商和余数”。

那么：

要转换的数是 6， 6 ÷ 2，得到商是 3，余数是 0。 （不要告诉我你不会计算 6÷3！）

“将商继续除以 2,直到商为 0……”

现在商是 3，还不是 0，所以继续除以 2。

那就： 3 ÷ 2, 得到商是 1,余数是 1。

“将商继续除以 2，直到商为 0……”

现在商是 1，还不是 0，所以继续除以 2。

那就： 1 ÷ 2, 得到商是 0，余数是 1 （拿笔纸算一下，1÷2 是不是商 0 余 1!）

“将商继续除以 2，直到商为 0……最后将所有余数倒序排列”

好极！现在商已经是 0。

我们三次计算依次得到余数分别是：0、1、1，将所有余数倒序排列，那就是：110 了！

6 转换成二进制，结果是 110。

把上面的一段改成用表格来表示，则为：

被除数 计算过程 商 余数

6 6/2 3 0

3 3/2 1 1

1 1/2 0 1

（在计算机中，÷用 / 来表示）

如果是在考试时，我们要画这样表还是有点费时间，所更常见的换算过程是使用下图的连除：

（图：1）

请大家对照图，表，及文字说明，并且自己拿笔计算一遍如何将 6 转换为二进制数。

说了半天，我们的转换结果对吗？二进制数 110 是 6 吗？你已经学会如何将二进制数转换成 10 进制数了，所以请现在就计算一下 110 换成 10 进制是否就是 6。

6.3.2 10 进制数转换为 8、16 进制数

10 进制数转换成 8 进制的方法，和转换为 2 进制的方法类似，唯一变化：除数由 2 变成 8。

来看一个例子，如何将十进制数 120 转换成八进制数。

用表格表示：

被除数 计算过程 商 余数

120 120/8 15 0

15 15/8 1 7

1 1/8 0 1

120 转换为 8 进制，结果为：170。

10 进制数转换成 16 进制的方法，和转换为 2 进制的方法类似，唯一变化：除数由 2 变成 16。

同样是 120，转换成 16 进制则为：

被除数 计算过程 商 余数

120 120/16 7 8

7 7/16 0 7

120 转换为 16 进制，结果为：78。

请拿笔纸，采用（图：1）的形式，演算上面两个表的过程。

6.4 二、十六进制数互相转换

二进制和十六进制的互相转换比较重要。不过这二者的转换却不用计算，每个 C，C++程序员都能做到看见二进制数，直接就能转换为十六进制数，反之亦然。

我们也一样，只要学完这一小节，就能做到。

首先我们来看一个二进制数：1111，它是多少呢？

你可能还要这样计算：1 * 20 + 1 * 21 + 1 * 22 + 1 * 23 = 1 * 1 + 1 * 2 + 1 * 4 + 1 * 8 = 15。

然而，由于 1111 才 4 位，所以我们必须直接记住它每一位的权值，并且是从高位往低位记，：8、4、2、1。即，最高位的权值为 23 ＝ 8，然后依次是 22 ＝ 4，21＝2， 20 ＝ 1。

记住 8421，对于任意一个 4 位的二进制数，我们都可以很快算出它对应的 10 进制值。

下面列出四位二进制数 xxxx 所有可能的值（中间略过部分）

仅 4 位的 2 进制数 快速计算方法 十进制值 十六进值

1111 = 8 + 4 + 2 + 1 = 15 F

1110 = 8 + 4 + 2 + 0 = 14 E

1101 = 8 + 4 + 0 + 1 = 13 D

1100 = 8 + 4 + 0 + 0 = 12 C

1011 = 8 + 0 + 2 + 1 = 11 B

1010 = 8 + 0 + 2 + 0 = 10 A

1001 = 8 + 0 + 0 + 1 = 9 9

....

0001 = 0 + 0 + 0 + 1 = 1 1

0000 = 0 + 0 + 0 + 0 = 0 0

二进制数要转换为十六进制，就是以 4 位一段，分别转换为十六进制。

如(上行为二制数，下面为对应的十六进制)：

1111 1101 ， 1010 0101 ， 1001 1011

F D ， A 5 ， 9 B

反过来，当我们看到 FD 时，如何迅速将它转换为二进制数呢？

先转换 F：

看到 F，我们需知道它是 15（可能你还不熟悉 A～F 这五个数），然后 15 如何用 8421 凑呢？应该是 8 + 4 + 2 + 1，所以四位全为 1 ：1111。

接着转换 D：

看到 D，知道它是 13，13 如何用 8421 凑呢？应该是：8 + 4 + 1,即：1101。

所以,FD 转换为二进制数，为： 1111 1101

由于十六进制转换成二进制相当直接，所以，我们需要将一个十进制数转换成 2 进制数时，也可以先转换成 16 进制，然后再转换成 2 进制。

比如，十进制数 1234 转换成二制数，如果要一直除以 2，直接得到 2 进制数，需要计算较多次数。所以我们可以先除以 16，得到 16 进制数：

被除数 计算过程 商 余数

1234 1234/16 77 2

77 77/16 4 13 (D)

4 4/16 0 4

结果 16 进制为： 0x4D2

然后我们可直接写出 0x4D2 的二进制形式： 0100 1101 0010。

其中对映关系为：

0100 -- 4

1101 -- D

0010 -- 2

同样，如果一个二进制数很长，我们需要将它转换成 10 进制数时，除了前面学过的方法是，我们还可以先将这个二进制转换成 16 进制，然后再转换为 10 进制。

下面举例一个 int 类型的二进制数：

01101101 11100101 10101111 00011011

我们按四位一组转换为 16 进制： 6D E5 AF 1B

在 PASCAL 中的编程：

```
var a:integer;
begin
write('Input a number:');
readln(a);
if a>255 then writeln('Bu zai fan wei');
if a<=255 then writeln(chr(a));
readln;
end.
```

编辑本段汉字编码

0-127 是 7 位 ASCII 码的范围，是国际标准。

至于汉字，不同的字符集用的 ascii 码的范围也不一样，常用的汉字字符集有 GB2312-80,GBK, Big5,unicode 等。下面我重点说一说最常用的 GB_2312 的字符集。

GB_2312 字符集是目前最常用的汉字编码标准，windows 95/98/2000 中使用的 GBK 字符集 就包含了 GB2312，或者说和 GB2312 兼容，GB_2312 字符集包含了 6763 个的 简体汉字，和 682 个标准中文符号。在这个标准中，每个汉字用 2 个字节来表示，每个字节的 ascii 码为 161-254 (16 进制 A1 - FE)，第一个字节对应于 区码的 1-94 区,第二个字节 对应于位码的 1-94 位。

161-254 其实很好记忆，大家知道英文字符的中，可打印的字符范围为 33-126。将 这对 数加上 128（或者说最高位置 1），就得到汉字使用的字符的范围。