

Precedence and Order of Evaluation

Operators	Associativity
() [] -> .	left to right
! ~ ++ -- + - * (type) sizeof	right to left
* / %	left to right
+ -	left to right
<< >>	left to right
< <= > >=	left to right
== !=	left to right
&	left to right
^	left to right
	left to right
&&	left to right
	left to right
?:	right to left
= += -= *= /= %= &= ^= = <<= >>=	right to left
,	left to right

1

ASCII Table

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

Source: www.LookupTables.com

scanf Conversions

Character	Input Data; Argument type
d	decimal integer; int *
i	integer; int *. The integer may be in octal (leading 0) or hexadecimal (leading 0x or 0X).
o	octal integer (with or without leading zero); int *
u	unsigned decimal integer; unsigned int *
x	hexadecimal integer (with or without leading 0x or 0X); int *
c	characters; char *. The next input characters (default 1) are placed at the indicated spot. The normal skip-over white space is suppressed; to read the next non-white space character, use %ls
s	character string (not quoted); char *, pointing to an array of characters long enough for the string and a terminating '\0' that will be added.
e, f, g	floating-point number with optional sign, optional decimal point and optional exponent; float *
%	literal %; no assignment is made.

3

printf Conversions

Character	Argument type; Printed As
d, i	int; decimal number
o	int; unsigned octal number (without a leading zero)
x, X	int; unsigned hexadecimal number (without a leading 0x or 0X), using abcdef or ABCDEF for 10, ...,15.
u	int; unsigned decimal number
c	int; single character
s	char *: print characters from the string until a '\0' or the number of characters given by the precision.
f	double; [-] m.ddddd, where the number of d's is given by the precision (default 6).
e, E	double; [-] m.dddddE+/-xx or [-] m.dddddE+/-xx, where the number of d's is given by the precision (default 6).
g, G	double; use %e or %E if the exponent is less than -4 or greater than or equal to the precision; otherwise use %f. Trailing zeros and a trailing decimal point are not printed.
p	void *: pointer (implementation-dependent representation).
%	no argument is converted; print a %

4