les caractères en C

les caractères

```
char c, d, e;
c = 'c';
d = '\0';   /* le caractère nul */
e = '\n';    /* le caractère  de nouvelle ligne */
```

`char` est un type entier.

`signed char` `char` `unsigned char`

La variable char peut-être utilisée comme une variable entière.

Est-ce que char est un entier signé (signed char) ou non signé (unsigned char)?
Cela dépend de l'implémentation ou de l'architecture de l'ordinateur.

En C toujours `sizeof(char) == 1` (char occupe un octet).

les caractères

01100100 en héxadécimal 0x64, en décimal 100

codes ASCII

Dec	Hex	char		Dec	Hex	char	Dec	Hex	char	Dec	Hex	char
0	00	NUL	*caract. null*	32	20	espace	64	40	@	96	60	'
1	01	SOH		33	21	!	65	41	A	97	61	a
2	02	STX		34	22	"	66	42	B	98	62	b
3	03	ETX		35	23	#	67	43	C	99	63	c
4	04	EOT		36	24	$	68	44	D	100	64	d
5	05	ENQ		37	25	%	69	45	E	101	65	e
6	06	ACK		38	26	&	70	46	F	102	66	f
7	07	BEL	*bell*	39	27	‘	71	47	G	103	67	g
8	08	BS	*backspace*	40	28	(	72	48	H	104	68	h
9	09	TAB	*tabul. horiz*	41	29	)	73	49	I	105	69	i
10	0A	LF	*line feed*	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	*Tabul. vertic*	43	2B	+	75	4B	K	107	6B	k
12	0C	FF		44	2C	,	76	4C	L	108	6C	l
13	0D	CR	*retour ligne*	45	2D	-	77	4D	M	109	6D	m
14	0E	SO		46	2E	.	78	4E	N	110	6E	n
15	0F	SI		47	2F	/	79	4F	O	111	6F	o
16	10	DLE		48	30	0	80	50	P	112	70	p
17	11	DC1		49	31	1	81	51	Q	113	71	q
18	12	DC2		50	32	2	82	52	R	114	72	r
19	13	DC3		51	33	3	83	53	S	115	73	s
20	14	DC4		52	34	4	84	54	T	116	74	t
21	15	NAK		53	35	5	85	55	U	117	75	u
22	16	SYN		54	36	6	86	56	V	118	76	v
23	17	ETB		55	37	7	87	57	W	119	77	w
24	18	CAN		56	38	8	88	58	X	120	78	x
25	19	EM		57	39	9	89	59	Y	121	79	y
26	1A	SUB		58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	*escape*	59	3B	;	91	5B	[	123	7B	{
28	1C	FS		60	3C	<	92	5C	\	124	7C	|
29	1D	GS		61	3D	=	93	5D	]	125	7D	}
30	1E	RS		62	3E	>	94	5E	^	126	7E	~
31	1F	US		63	3F	?	95	5F	_	127	7F	DEL

les caractères

```
char x, y;
x = 'c';
y = 99;
```

x et y contiennent la même valeur : le code ASCII du caractère 'c' .

```
x++; /* x contient maintenant 100, le code de 'd' */

y--; /* y contient 98 : le code du caractère 'b' */
```

```
printf( "x contient la lettre %c \n",  x);
printf( "x vaut %d \n", (int) x);
printf( "x en hexa %#2x \n", (int) x);
```

```
%c le format pour afficher un caractère
%d le format pour un int
%x  le format pour affichage en hexadécimal (# pour ajouter 0x au
début)
```

Sur le terminal les trois printf affichent :

```
x contient la lettre d
x vaut 100
x en hexa 0x64
```

les caractères : les opérations arithmétiques

```
char x='8';


/* traduire un caractère qui code un chiffre vers un
nombre */

unsigned int val_x = x - '0';  /* x vaut 8 */
```

```
/* une boucle sur toutes les lettres minuscules */

char c;
for( c = 'a'; c <= 'z', c++ ){


}
```

les caractères

Utilisation des propriétés de codage ASCII :

```
char x;

x = ... ;

/* vérifier si x contient (le code d') une lettre*/
if( 'a' <= x && x <= 'z' ){
   /* x est une lettre minuscule, les codes de
    * lettres minuscule entre 'a' et 'z' */

}
```

Cette condition est correct si l'ordinateur utilise le codage ASCII.

Dans certaines machines (très très rares) on peut avoir d'autres types de codage, comme EBCDIC.

tester la catégorie de caractères

les fonctions suivantes testent les propriétés de caractères et retournent une valeur int différente de 0 si le caractère passe le test et 0 sinon. Le paramètre est un int mais sera transformé en char.

```
#include <ctype.h>

int isalpha(c)    c est une lettre
int iscntrl(c)    caractère de contrôle
int isdigit(c)    un chiffre décimal
int isalnum(c)    équivalent à isalpha(c) || isdigit(c)
int isgraph(c)    le caractère imprimable sauf espace
int isprint(c)    caractère imprimable y compris l'espace
int islower(c)    lettre minuscule
int isupper(c)    lettre majuscule
int isspace(c)    caractère blanc (whitespace) : ' ' espace,
                  '\t' tabulation, '\n' newline, '\r' carriage return,
                  '\v' vertical tab.

int ispunct(c)    caractère imprimable différent de l'espace,
                  des lettres et des chiffres
int isxdigit(c)   un chiffre hexadécimal
int isblank(c)    un caractère séparant les mots : ' ' et  '\t' (et peut-
être d'autres)
```

caractères

De préférence ne pas assumer que les caractères sont codés en ASCII.

par exemple pour vérifier si `char x` contient un chiffre on préfère

```
isdigit(x)
```

au lieu de

```
'0' <= x &&  x <= '9'
```

conversion de lettres

```
#include <ctype.h>

int tolower(int c)  - convertit c en minuscule (si la
conversion impossible, c'est-à-dire c n'est pas une lettre
majuscule alors la fonction retourne c).

int toupper(int c)   - convertit c en majuscule (ou
retourne c si la conversion impossible)
```