

Algorithmique & Langage C

IUT GEII S1

Les caractères

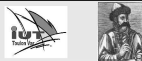
Les chaînes de caractères

Notes de cours

(cinquième partie)

5

cours_algo_lgc5.10.odp



Licence



COMMONS DEED



**• Paternité - Pas d'Utilisation Commerciale -
• Partage des Conditions Initiales à l'Identique 2.0 France**

- Vous êtes libres :
 - * de reproduire, distribuer et communiquer cette création au public
 - * de modifier cette création, selon les conditions suivantes :
 -
- **Paternité.** Vous devez citer le nom de l'auteur original.
-
- **Pas d'Utilisation Commerciale.**
 - Vous n'avez pas le droit d'utiliser cette création à des fins commerciales.
 -
- **Partage des Conditions Initiales à l'Identique.**
 - Si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.
 - * A chaque réutilisation ou distribution, vous devez faire apparaître clairement aux autres les conditions contractuelles de mise à disposition de cette création.
 - * Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droits.
- Ce qui précède n'affecte en rien vos droits en tant qu'utilisateur (exceptions au droit d'auteur : copies réservées à l'usage privé du copiste, courtes citations, parodie...)
- voir le contrat complet sous : <http://fr.creativecommons.org/contrats.htm>



Les caractères

Un **caractère** peut être :

une *lettre* A,B,.....,Z a,b,.....,z

un *chiffre* 0,1,2,.....,9

une *ponctuation* ou un *symbole* +,/,?!,&,\$,%,...

un *caractère de contrôle* : CR,LF,ESC,.....

Les caractères sont codés au moyen de nombres.

Il y a une correspondance entre chaque caractère et un nombre entier.

Le code le plus employé est le code **ASCII**

American Standard for Computer Information Interchange



code ASCII

Le code ASCII est un code à 7 bits à l'origine.

$2^7 = 128$ caractères possibles de 0 à 127 ou 0x00 à 0x7F

Il fut étendu plus tard à 8 bits soit un octet.

$2^8 = 256$ caractères possibles de 0 à 256 ou 0x00 à 0xFF

0x00 – 0x7F : code ascii de base (appelé aussi ASCII-US)

0x80 – 0xFF : code ascii étendu

Plusieurs correspondances existent pour la partie étendue.

En France on utilise souvent les jeux de caractères ISO-8859-1 (dit ISO-LATIN-1) ou ISO-8859-15 (dit alphabet latin n°9)



code ASCII de base (7 bits)

	0	1	2	3	4	5	6	7	
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	
0x20	SP	!	"	#	\$	%	&	'	
0x30	0	1	2	3	4	5	6	7	
0x40	@	A	B	C	D	E	F	G	
0x50	P	Q	R	S	T	U	V	W	
0x60	`	a	b	c	d	e	f	g	
0x70	p	q	r	s	t	u	v	w	

128 caractères de 0x00 à 0x7F

	8	9	A	B	C	D	E	F
0x00	BS	HT	LF	VT	NP	CR	SO	SI
0x10	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	()	*	+	,	-	.	/
0x30	8	9	:	;	<	=	>	?
0x40	H	I	J	K	L	M	N	O
0x50	X	Y	Z	[\]	^	_
0x60	h	i	j	k	l	m	n	o
0x70	x	y	z	{		}	~	DEL

↑ poids fort

Lecture de la table des codes ASCII

	0	1	2	3	4	5	6	7
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB
0x20	SP	!	"	#	\$	%	&	'
0x30	0	1	2	3	4	5	6	7
0x40	@	A	B	C	D	E	F	G
0x50	P	Q	R	S	T	U	V	W
0x60	`	a	b	c	d	e	f	g
0x70	p	q	r	s	t	u	v	w

↖ poids faible

↗ poids fort

Le code ascii du caractère C est donc : 0x43



Caractères lettres

Majuscules : de **A:0x41** à **Z:0x5A** (uppercase)

Minuscules : de **a:0x61** à **z :0x7A** (lowercase)

Des lettres consécutives dans l'alphabet ont des codes ascii consécutifs :

code de B = (code de A) + 1

code de C = (code de B) + 1 = (code de A) + 2

...

L'écart entre une lettre en minuscule et la même lettre en majuscules est constant et vaut 0x20 :

code de x = code de X + 0x20



Caractères chiffres

De **0:0x30** à **9:0x39**

Un caractère chiffre a un code différent de la valeur numérique qu'il représente.

Code ascii d'un caractère chiffre=valeur du chiffre + 0x30

Conséquences :

Des chiffres consécutifs ont des codes consécutifs
On peut retrouver la valeur à partir du caractère en retranchant 0x30 (48 en décimal)

Le code du caractère 0 n'est pas 0 ! (c'est 0x30)



Les caractères de contrôle

Les codes compris entre 0x00 et 0x1F ne sont pas des caractères destinés à être affichés.
Ce sont des caractères associés à une fonction.

On les appelle *caractères de contrôle*.

Seuls quelques uns sont encore utilisés aujourd'hui :

0x0D : CR carriage return

0x0A : LF line feed

0x08 : BS back space

0x07 : BELL (cloche)

0x09 : HT tabulation

0x1B : ESC (escape)

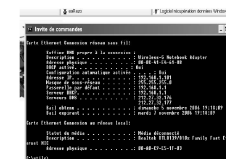
.....



CR / LF



CR retour chariot
LF



L'association CR+LF produit le passage à la ligne ($\backslash n$).

Souvent le terminal est configuré pour que la réception de CR seul soit interprétée comme CR+LF



Ascii étendu (8bits)

Le code ascii de base ne comporte aucun caractère accentué ni de caractère spécifique qui ne sont pas utilisés en anglais. (pas de é, è, ê, à, ç, Ç, ÿ, ß, œ, €...)

En utilisant le 8^{ème} bit de l'octet, on dispose de 128 codes supplémentaires qui permettent de coder des caractères spécifiques aux langues latines, cyrilliques, etc...

Malheureusement il existent plusieurs codages différents !
On parle de **jeux de caractères**.

On utilise souvent le jeu ISO-8859-1 (dit ISO latin 1)
ou sa révision récente ISO-8859-15 qui inclue l'euro € et œ.



ISO 8859-15 Latin alphabet n°9

				<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>																0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1																																																																										
0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1																																																																										
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1																																																																										
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1																																																																										
				<table border="1"> <tr><td>00</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td><td>08</td><td>09</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td></td></tr> </table>																00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15																																																								
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15																																																																												
0	1	2	3													7	8	9	A	B	C	D	E	F																																																																			
00000000				SP	0	@	P	`	p			NBSP	°	À	Ð	à	ð	0																																																																									
00001001				!	1	A	Q	a	q			i	±	Á	Ñ	á	ñ	1																																																																									
00101002				"	2	B	R	b	r			ç	²	Â	Ò	â	ò	2																																																																									
00111003				#	3	C	S	c	s			£	³	Ã	Ó	ã	ó	3																																																																									
01101004				¢	4	D	T	d	t			€	ˆ	Ž	Ä	ä	ö	4																																																																									
01101005				§	5	E	U	e	u			¥	µ	Å	Ö	å	ö	5																																																																									
01111006				€	6	F	V	f	v			Š	¶	Æ	Ö	æ	ö	6																																																																									
01111007				'	7	G	W	g	w			Š	·	Ç	×	ç	÷	7																																																																									
10001008				(8	H	X	h	x			Š	˜	È	Ø	è	ø	8																																																																									
10001009)	9	I	Y	i	y			©	¹	É	Ú	é	ù	9																																																																									
10101010				*	:	J	Z	j	z			ª	º	Ê	Û	ê	ú	A																																																																									
10111011				+	:	K	[k	{			«	»	Ë	Ü	ë	ü	B																																																																									
11101012				,	<	L	\	l				¬	ƒ	Ï	Û	ï	ü	C																																																																									
11101013				-	=	M]	m	}			SHY	œ	Í	Ý	í	ý	D																																																																									
1111014				.	>	N	^	n	~			®	ÿ	Î	Þ	î	þ	E																																																																									
11111015				/	?	O	_	o				-	ı	Ï	ß	ï	ÿ	F																																																																									
				0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	hex.																																																																							



ISO 8859-14 Latin alphabet n°8 (Celtic)

				b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀															
00				0000000011111111															
01				0000001111110000															
02				0000010110011100															
03				0000011010101101															
04				0000100101010101															
05				0000101010101010															
06				0000110101010101															
07				0000111010101010															
08				0000111101010101															
09				0000111110101010															
10				0001000000000000															
11				0001000000000001															
12				0001000000000010															
13				0001000000000011															
14				0001000000000100															
15				0001000000000101															
16				0001000000000110															
17				0001000000000111															
18				0001000000001000															
19				0001000000001001															
20				0001000000001010															
21				0001000000001011															
22				0001000000001100															
23				0001000000001101															
24				0001000000001110															
25				0001000000001111															
26				0001000000010000															
27				0001000000010001															
28				0001000000010010															
29				0001000000010011															
30				0001000000010100															
31				0001000000010101															
32				0001000000010110															
33				0001000000010111															
34				0001000000011000															
35				0001000000011001															
36				0001000000011010															
37				0001000000011011															
38				0001000000011100															
39				0001000000011101															
40				0001000000011110															
41				0001000000011111															
42				0001000000100000															
43				0001000000100001															
44				0001000000100010															
45				0001000000100011															
46				0001000000100100															
47				0001000000100101															
48				0001000000100110															
49				0001000000100111															
50				0001000000101000															
51				0001000000101001															
52				0001000000101010															
53				0001000000101011															
54				0001000000101100															
55				0001000000101101															
56				0001000000101110															
57				0001000000101111															
58				0001000000110000															
59				0001000000110001															
60				0001000000110010															
61				0001000000110011															
62				0001000000110100															
63				0001000000110101															
64				0001000000110110															
65				0001000000110111															
66				0001000000111000															
67				0001000000111001															
68				0001000000111010															
69				0001000000111011															
70				0001000000111100															
71				0001000000111101															
72				0001000000111110															
73				0001000000111111															
74				0001000001000000															
75				0001000001000001															
76				0001000001000010															
77				0001000001000011															
78				0001000001000100															
79				0001000001000101															
80				0001000001000110															
81				0001000001000111															
82				0001000001001000															
83				0001000001001001															
84				0001000001001010															
85				0001000001001011															
86				0001000001001100															
87				0001000001001101															
88				0001000001001110															
89				0001000001001111															
90				0001000001010000															
91				0001000001010001															
92				0001000001010010															
93				0001000001010011															
94				0001000001010100															
95				0001000001010101															
96				0001000001010110															
97				0001000001010111															
98				0001000001011000															
99				0001000001011001															
100				0001000001011010															
101				0001000001011011															
102				0001000001011100															
103				0001000001011101															
104				0001000001011110															
105				0001000001011111															
106				0001000001100000															
107				0001000001100001															
108				0001000001100010															
109				0001000001100011															
110				0001000001100100															
111				0001000001100101															
112				0001000001100110															
113				0001000001100111															
114				0001000001101000															
115				0001000001101001															
116				0001000001101010															
117				0001000001101011															
118				0001000001101100															
119				0001000001101101															
120				0001000001101110															
121				0001000001101111															
122				0001000001110000															
123				0001000001110001															
124				0001000001110010															
125				0001000001110011															
126				0001000001110100															
127				0001000001110101															
128				0001000001110110															
129				0001000001110111															
130				0001000001111000															
131				0001000001111001															
132				0001000001111010															
133				0001000001111011															
134				0001000001111100															
135				0001000001111101															
136				0001000001111110															
137				0001000001111111															
138				0001000001111111															
139				0001000001111111															
140				0001000001111111															
141				0001000001111111															
142				0001000001111111															
143				0001000001111111															
144				0001000001111111															
145				0001000001111111															
146				0001000001111111															
147				0001000001111111															
148				0001000001111111															
149				0001000001111111															
150				0001000001111111															
151				0001000001111111															
152				0001000001111111															
153				0001000001111111															
154				0001000001111111															
155				0001000001111111															
156				0001000001111111															
157				0001000001111111															
158				0001000001111111															
159				0001000001111111															
160				0001000001111111															
161				0001000001111111															
162				0001000001111111															
163				0001000001111111															
164				0001000001111111															
165				0001000001111111															
166				0001000001111111															
167				0001000001111111															
168				0001000001111111															
169				0001000001111111															
170				0001000001111111															
171				0001000001111111															
172				0001000001111111															
173				0001000001111111															
174				0001000001111111															
175				0001000001111111															
176				0001000001111111															
177				0001000001111111															
178				0001000001111111															
179				0001000001111111															
180				0001000001111111															
181				0001000001111111															
182				0001000001111111															
183				0001000001111111															
184				0001000001111111															
185				0001000001111111															
186				0001000001111111															
187				0001000001111111															
188				0001000001111111															
189				0001000001111111															
190				0001000001111111															
191				0001000001111111															
192				0001000001111111															
193				0001000001111111															
194				0001000001111111															
195				0001000001111111															
196				0001000001111111															
197				0001000001111111															
198				0001000001111111															
199				0001000001111111															
200				0001000001111111															



Unicode

En fait même avec plusieurs jeux de caractères, on ne parvient pas à coder tous les caractères utilisés dans toutes les langues. La norme Unicode permet de coder de manière unique tous les caractères utilisés par toutes les langues du monde :

Dans la version 3.1 de Unicode, près de 245000 caractères sont définis.

Il y a 17 "plans" de 16 bits.

Soit $17 \times 65536 = 1\ 114\ 112$ codes possibles.

Un caractère Unicode est noté U+yyxxxx

Avec yy numéro hexa du plan (0 non significatifs omis)
xxxx code hexa du caractère dans le plan yy



Unicode

On retrouve nos jeux courants au début (plan 0).
0000-007F : le code ascii, 0080-FF : latin-1, etc...

Ex : A est
Codé 0041
ascii étendu
à 16bits

Sous-ensemble	Réservés		Décimal		Sous-ensemble	Réservés		Décimal	
	Début	Fin	Début	Fin		Début	Fin	Début	Fin
latin basique	U+0000	U+007F	0	127	(mandaique)	U+0840	U+085F	2 112	2 143
supplément latin-1	U+0080	U+00FF	128	255	—	U+0860	U+087F	2 144	2 175
latin étendu - A	U+0100	U+017F	256	383	arabe étendu-A?	U+0880	U+08BF	2 176	2 239
latin étendu - B	U+0180	U+024F	384	591	—	U+08C0	U+08FF	2 240	2 303
extensions API	U+0250	U+02AF	592	685	dévanagari	U+0900	U+097F	2 304	2 431
lettres modificatives	U+02B0	U+02FF	688	767	bengali	U+0980	U+09FF	2 432	2 559
diacritiques combinants	U+0300	U+036F	768	879	gourmoukhi	U+0A00	U+0A7F	2 560	2 687
grec et copte	U+0370	U+03FF	880	1 023	goudjarati (gujrati)	U+0A80	U+0AFF	2 688	2 815
cyrillique	U+0400	U+04FF	1 024	1 279	oriya	U+0B00	U+0B7F	2 816	2 943
supplément cyrillique	U+0500	U+052F	1 280	1 327	tamoul	U+0B80	U+0BFF	2 944	3 071
arménien	U+0530	U+058F	1 328	1 423	télougou	U+0C00	U+0C7F	3 072	3 199
hébreu	U+0590	U+05FF	1 424	1 535	kannara	U+0C80	U+0CFF	3 200	3 327
arabe	U+0600	U+06FF	1 536	1 791	malayalam	U+0D00	U+0D7F	3 328	3 455
syriaque	U+0700	U+074F	1 792	1 871	singhalais	U+0D80	U+0DFF	3 456	3 583
supplément arabe	U+0750	U+077F	1 872	1 919	thaï	U+0E00	U+0E7F	3 584	3 711
thâna	U+0780	U+07BF	1 920	1 983	laotien	U+0E80	U+0EFF	3 712	3 839
n'ko	U+07C0	U+07FF	1 984	2 047	tibétain	U+0F00	U+0FFF	3 840	4 095
(samaritain)	U+0800	U+083F	2 048	2 111					

Unicode

珞 a pour code F917 : plan 0 code F917

voir PDF : fr en 自	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F90	豈	更	車	賈	滑	串	句	龜	龜	契	金	喇	奈	懶	癩	羅
F91	蘿	螺	裸	邏	樂	洛	烙	珞	落	酪	駱	亂	卵	欄	爛	蘭
F92	鸞	嵐	濫	藍	檻	拉	臘	蠟	廊	朗	浪	狼	郎	來	冷	勞
F93	擄	櫓	爐	盧	老	蘆	虜	路	露	魯	鷲	碌	祿	綠	蓀	錄
F94	鹿	論	壘	弄	籠	壘	牢	磊	路	雷	壘	屢	樓	淚	漏	累
F95	縷	陋	勒	肋	凜	凌	稜	綾	菱	陵	讀	擘	樂	諾	丹	寧



Unicode

Exemple : 𐎠 a pour code 103A2 (plan 1 code 03A2)

Vieux perse [modifier]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
103A	𐎠	𐎡	𐎢	𐎣	𐎤	𐎥	𐎦	𐎧	𐎨	𐎩	𐎪	𐎫	𐎬	𐎭	𐎮	𐎯
103B	𐎰	𐎱	𐎲	𐎳	𐎴	𐎵	𐎶	𐎷	𐎸	𐎹	𐎺	𐎻	𐎼	𐎽	𐎾	𐎿
103C	𐏀	𐏁	𐏂	𐏃	—				𐏄	𐏅	𐏆	𐏇	𐏈	𐏉	𐏊	𐏋
103D	𐏌	𐏍	𐏎	𐏏	𐏐	—										



UTF-8

En pratique Unicode est peu utilisé directement.

Dans un texte la majorité des caractères font partis du code ascii de base (0-127).

L'utilisation directe d'Unicode doublerait donc la taille alors que la plupart du temps le poids fort reste à 00.

En UTF-8 :

Le numéro de chaque caractère est donné par le standard Unicode.

Les caractères de numéro 0 à 127 sont codés sur un octet dont le bit de poids fort est toujours nul.

Les caractères de numéro supérieur à 127 sont codés sur plusieurs octets. Dans ce cas, les bits de poids fort du premier octet forment une suite de 1 de longueur égale au nombre d'octets utilisés pour coder le caractère, les octets suivants ayant 10 comme bits de poids fort.



UTF-8

Représentation binaire UTF-8	Signification
0 xxx xxxx	1 octet , caractères de 0 à 127 (7 bits) ASCII-US
110 x xxxx 10 xx xxxx	2 octets , codant 8 à 11 bits
1110 xxxx 10 xx xxxx 10 xx xxxx	3 octets, codant 12 à 16 bits
1111 0xxx 10 xx xxxx 10 xx xxxx 10 xx xxxx	4 octets, codant 17 à 21 bits

Avantages :

- Efficace pour les langues utilisant une majorité de caractères ascii-us.
- Permet de représenter une grande partie des caractères Unicode.
- Compatible avec les codes C existants pour la fin de chaîne (caractère 0)

Inconvénients :

- Plus difficile de compter les caractères.
- Peu efficace pour les langues utilisant une majorité de caractères non ascii-us (chinois, arabe, ...)



UTF-8 exemples

Pour coder le caractère **A** pas de difficulté :

A : décimal 65 <127 ou hexa 0x41 <0x7F donc

Unicode: U+0041 et UTF-8: un octet 0x41 (**0100 0001**)

Pour coder le caractère **é** :

En ascii étendu il faut connaître le jeux utilisé (iso-latin1 =E9).

En Unicode U+00E9

En UTF-8 : >127 donc au moins deux octets

U+00E9 -> 0000 0000 1110 1001

motif **110**x xxxx **10**xx xxxx
UTF-8 **1100 0011 1010 1001**

Donc **é** est codé **0xC3A9** en UTF-8



Définition du codage utilisé dans les documents

Par exemple dans l'en-tête d'un document html :

```
<head>
<title>Accueil | Ubuntu-fr</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
....
</head>
```

Dans un navigateur on peut changer le jeux de caractères utilisés si la détection automatique n'a pas fonctionné :

Firefox : Affichage->Encodage des caractères

Avec Ubuntu l'outil :

Applications->Accessoires->Table de caractères vous permet d'insérer n'importe quel caractère dans vos documents et donne le code d'un caractère dans les différents codages.



Les caractères en C

On utilise des variables de type **char** pour stocker des caractères. **char** représente des variables sur un octet.

Déclaration :

char c ;

Affectation :

c = 'A' ; // notation 'x' : code ascii de x

c = 65 ; // notation décimale

c = 0x41 ; // notation hexadécimale



Affichage d'un caractère

```
char a ;
```

```
a = 'F' ;
```

on peut utiliser **printf** avec le format **%c** :

```
printf( "%c" , a ) ;
```

ou bien la fonction **putchar**

Ecran

```
F
```

```
putchar( a ) ;
```



Affichage du code d'un caractère

On peut utiliser **printf** avec le format **%d** ou **%x** :

%d : code en décimal

%x : code en hexadécimal

```
char h ;
```

```
h = 'A' ;
```

```
printf( "%c %x %d" , h , h , h ) ;
```

Ecran

```
A 41 65
```



Saisie d'un caractère

On peut utiliser **scanf** avec le format **%c** :

```
char x ;
```

```
scanf("%c",&x);
```

ou bien la fonction **getchar** :

```
x = getchar() ;
```



Saisie d'un caractère

Attention lorsqu'on saisie un caractère par **scanf**, le caractère CR qui est tapé pour poursuivre l'exécution n'est pas lu et reste dans le buffer associé au clavier; La prochaine lecture sur le clavier risque alors d'être perturbée. Pour éviter ce problème il suffit de lire le caractère CR après la saisie d'un caractère.

```
char c ;
```

```
printf("Entrez un caractere ");
```

```
scanf("%c",&c);
```

```
getchar();
```

Une lecture supplémentaire
pour vider le buffer
du caractère CR

```
printf("Merci pour %c !\n",c);
```



Les chaînes de caractères

Une **chaîne de caractères** est une suite de caractères dont la **fin est marquée par le caractère '\0'**.

Une chaîne de caractères permet de stocker et de manipuler des mots, des phrases, etc....

On stocke les chaînes dans des *tableaux de char*.

Attention du fait de la nécessité de marquer la fin, un tableau de n char pourra contenir au plus une chaîne composée de n-1 caractères et du caractère '\0'

La déclaration : `char ch[12]` permettra de stocker un chaîne composée de 11 caractères et du '\0'.



Chaîne de caractères

`char ch[12];` marque la fin de la chaîne

ch	'b'	'o'	'n'	'j'	'o'	'u'	'r'	'\0'	'S'	'@'	0x1B	'z'
	0	1	2	3	4	5	6	7	8	9	10	11

ch contient la chaîne "bonjour"

Valeurs présentes dans le tableau mais ne faisant pas partie de la chaîne



Chaîne de caractères

La chaîne commence à la première case du tableau et se termine à la première valeur '\0' rencontrée.

Les caractères suivants dans le tableau ne font pas partie de la chaîne. (On peut éventuellement y trouver d'autres valeurs '\0', c'est la première qui marque la fin de la chaîne).

Si la valeur '\0' n'est pas présente dans le tableau, la chaîne n'est pas valide : Les fonctions opérants sur les chaînes vont continuer de lire les octets en mémoire jusqu'à trouver un '\0' (ou à obtenir une erreur mémoire si on va trop loin).



'\0' 0 0x00 et '0' ???

'\0' est le caractère ayant tous les bits à zéro.

'\0' :

	b7								b0
	0	0	0	0	0	0	0	0	0

On peut l'écrire 0x00 ou simplement 0 **mais pas '0' !**.

La notation '\0' permet de distinguer la valeur zéro lorsqu'elle représente une fin de chaîne de la valeur zéro utilisée comme simple valeur numérique.

'\0' égale 0x00 égale 0 (marque de fin de chaîne et valeur numérique 0)

'0' égale 0x30 égale 48 (caractère zéro)



Saisie d'une chaîne (1)

On peut utiliser **scanf** avec le format **%s** :

```
char ch[30] ;
```

```
printf("Entrer un mot : ");
```

```
scanf("%30s", ch);
```

Nom du tableau dans lequel
la chaîne va être saisie

Attention pas de & dans ce cas

Nombre max de caractères
pouvant être saisis - 1

Avec **scanf** la saisie de la chaîne s'arrête dès le premier
caractères espace (' ') rencontré. Les caractères tapés
après ne sont pas lus (mais restent dans le buffer...)



Saisie d'une chaîne (2)

On peut utiliser la fonction **fgets** :

```
char ch[30] ;
```

```
printf("Entrer un mot ou une phrase : ");
```

```
fgets(ch, 30, stdin);
```

Indique une saisie à partir
du clavier (entrée standard)

Nom du tableau dans lequel
la chaîne va être saisie

Nombre max de caractères
pouvant être saisis - 1

Avec **fgets** le tableau va contenir tous les caractères tapés
jusqu'à CR compris.
(ou jusqu'à 29 caractères si pas de CR rencontré avant)



Affichage d'une chaîne

On peut utiliser **printf** avec le format **%s** :

```
char ch[30];
```

```
printf("Entrez votre nom : ");
```

```
fgets(ch,30,stdin);
```

```
printf("Bonjour monsieur %s !", ch );
```

ou bien la fonction puts :

```
puts( ch ) ;
```



Traitement des chaînes de caractères

On va utiliser les mêmes méthodes que pour les tableaux quelconques mais au lieu de poursuivre le traitement jusqu'au dernier élément, on va s'arrêter lorsqu'on rencontre la fin de chaîne '\0'.

```
char ch[80];
```

```
int i = 0 ;
```

```
while ( ch[i] != '\0')
```

```
{
```

```
    Traiter le caractère n° i ;
```

```
    i=i+1 ;
```

```
}
```



Algorithmes à connaître

Transposer les algorithmes vus sur les tableaux aux chaînes.

Recherche de la lettre dont le rang dans l'alphabet est la plus élevée <=> Recherche du maximum

Copie de chaînes <=> Copie de tableaux

Compter le nombre d'apparition d'une lettre donnée <=> Compter le nombre d'éléments égaux à une valeur

Etc....

La principale différence est qu'on ne va pas jusqu'à la fin du tableau mais jusqu'à la première occurrence de la marque de fin de chaîne '\0'.



Exemple

```
char s[80] ;  
int i =0 ;  
int nbe = 0 ;
```

```
printf("Entrez votre nom (en lettres minuscules) :") ;  
scanf( "%80s" , s ) ;  
while(s[i]!='\0')  
{  
    if (s[i]=='e'){  
        nbe = nbe + 1 ; }  
    i = i + 1 ;  
}  
printf("%s comporte %d lettres\n", i ) ;
```

```
s[0] = s[0] - 0x20 ;
```

```
printf("Bonjour monsieur %s\n, il y a %d e dans votre nom",nbe) ;
```

