Algorithmique & Langage C
IUT GEII S1

Les caractères
Les chaînes de caractères

Notes de cours
(cinquième partie)

5

cours_algo_lgc5.11.odp

Licence

•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 identiqueà 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

poids fort →

	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

Lecture de la table des codes ASCII

poids faible →

	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 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***.

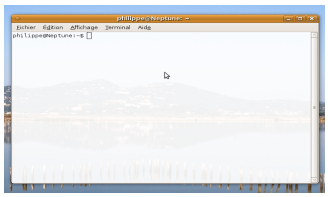
Seuls quelques uns sont encore utilisés aujourd'hui :

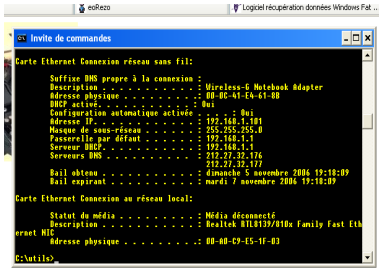
0x0D : CR carriage return
0x0A : LF line feed
0x08 : BS back space
0x07 : BELL (cloche)
0x09 : HT tabulation
0x1B : ESC (escape)
.....


CR / LF

CR retour chariot

LF





L'association CR+LF produit le passage à la ligne (\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 8ème 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 jeux 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

b4 b3 b2 b1	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
0 0 0 0	00		SP	0	@	P	`	p			NBSP	°	À	Ð	à	ð	0
0 0 0 1	01		!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ	1
0 0 1 0	02		"	2	B	R	b	r			¢	²	Â	Ò	â	ò	2
0 0 1 1	03		#	3	C	S	c	s			£	³	Ã	Ó	ã	ó	3
0 1 0 0	04		$	4	D	T	d	t			€	Ž	Ä	Ô	ä	ô	4
0 1 0 1	05		%	5	E	U	e	u			¥	µ	Å	Õ	å	õ	5
0 1 1 0	06		&	6	F	V	f	v			Š	¶	Æ	Ö	æ	ö	6
0 1 1 1	07		'	7	G	W	g	w			§	·	Ç	×	ç	÷	7
1 0 0 0	08		(	8	H	X	h	x			š	ž	È	Ø	è	ø	8
1 0 0 1	09		)	9	I	Y	i	y			©	¹	É	Ù	é	ù	9
1 0 1 0	10		*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú	A
1 0 1 1	11		+	;	K	[	k	{			«	»	Ë	Û	ë	û	B
1 1 0 0	12		,	<	L	\	l	|			¬	Œ	Ì	Ü	ì	ü	C
1 1 0 1	13		-	=	M	]	m	}			SHY	œ	Í	Ý	í	ý	D
1 1 1 0	14		.	>	N	^	n	~			®	Ÿ	Î	Þ	î	þ	E
1 1 1 1	15		/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ	F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

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

b4 b3 b2 b1	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
0 0 0 0	00		SP	0	@	P	`	p			NBSP	Ḟ	À	Ŵ	à	ŵ	0
0 0 0 1	01		!	1	A	Q	a	q			Ḃ	ḟ	Á	Ñ	á	ñ	1
0 0 1 0	02		"	2	B	R	b	r			ḃ	Ġ	Â	Ò	â	ò	2
0 0 1 1	03		#	3	C	S	c	s			£	ġ	Ã	Ó	ã	ó	3
0 1 0 0	04		$	4	D	T	d	t			Ċ	Ṁ	Ä	Ô	ä	ô	4
0 1 0 1	05		%	5	E	U	e	u			ċ	ṁ	Å	Õ	å	õ	5
0 1 1 0	06		&	6	F	V	f	v			Ḋ	¶	Æ	Ö	æ	ö	6
0 1 1 1	07		'	7	G	W	g	w			§	Ṗ	Ç	Ṫ	ç	ṫ	7
1 0 0 0	08		(	8	H	X	h	x			Ẁ	ẁ	È	Ø	è	ø	8
1 0 0 1	09		)	9	I	Y	i	y			©	ṗ	É	Ù	é	ù	9
1 0 1 0	10		*	:	J	Z	j	z			Ẃ	ẃ	Ê	Ú	ê	ú	A
1 0 1 1	11		+	;	K	[	k	{			ḋ	Ṡ	Ë	Û	ë	û	B
1 1 0 0	12		,	<	L	\	l	|			Ỳ	ỳ	Ì	Ü	ì	ü	C
1 1 0 1	13		-	=	M	]	m	}			SHY	Ẅ	Í	Ý	í	ý	D
1 1 1 0	14		.	>	N	^	n	~			®	ẅ	Î	Ŷ	î	ŷ	E
1 1 1 1	15		/	?	O	_	o				Ÿ	ṡ	Ï	ß	ï	ÿ	F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	hex

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 x 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	
	Début	Fin	Début	Fin
latin basique	U+0000	U+007F	0	127
supplément latin-1	U+0080	U+00FF	128	255
latin étendu - A	U+0100	U+017F	256	383
latin étendu - B	U+0180	U+024F	384	591
extensions API	U+0250	U+02AF	592	685
lettres modificatives	U+02B0	U+02FF	688	767
diacritiques combinants	U+0300	U+036F	768	879
grec et copte	U+0370	U+03FF	880	1 023
cyrillique	U+0400	U+04FF	1 024	1 279
supplément cyrillique	U+0500	U+052F	1 280	1 327
arménien	U+0530	U+058F	1 328	1 423
hébreu	U+0590	U+05FF	1 424	1 535
arabe	U+0600	U+06FF	1 536	1 791
syriaque	U+0700	U+074F	1 792	1 871
supplément arabe	U+0750	U+077F	1 872	1 919
thâna	U+0780	U+07BF	1 920	1 983
n'ko	U+07C0	U+07FF	1 984	2 047
(samaritain)	U+0800	U+083F	2 048	2 111

Sous-ensemble	Réservés		Décimal	
	Début	Fin	Début	Fin
(mandaïque)	U+0840	U+085F	2 112	2 143
—	U+0860	U+087F	2 144	2 175
¿arabe étendu-A?	U+0880	U+08BF	2 176	2 239
—	U+08C0	U+08FF	2 240	2 303
dévanâgarî	U+0900	U+097F	2 304	2 431
bengalî	U+0980	U+09FF	2 432	2 559
gourmoukhî	U+0A00	U+0A7F	2 560	2 687
goudjarâtî (gujrâtî)	U+0A80	U+0AFF	2 688	2 815
oriyâ	U+0B00	U+0B7F	2 816	2 943
tamoul	U+0B80	U+0BFF	2 944	3 071
télougou	U+0C00	U+0C7F	3 072	3 199
kannara	U+0C80	U+0CFF	3 200	3 327
malayalam	U+0D00	U+0D7F	3 328	3 455
singhalais	U+0D80	U+0DFF	3 456	3 583
thaï	U+0E00	U+0E7F	3 584	3 711
laotien	U+0E80	U+0EFF	3 712	3 839
tibétain	U+0F00	U+0FFF	3 840	4 095

Unicode

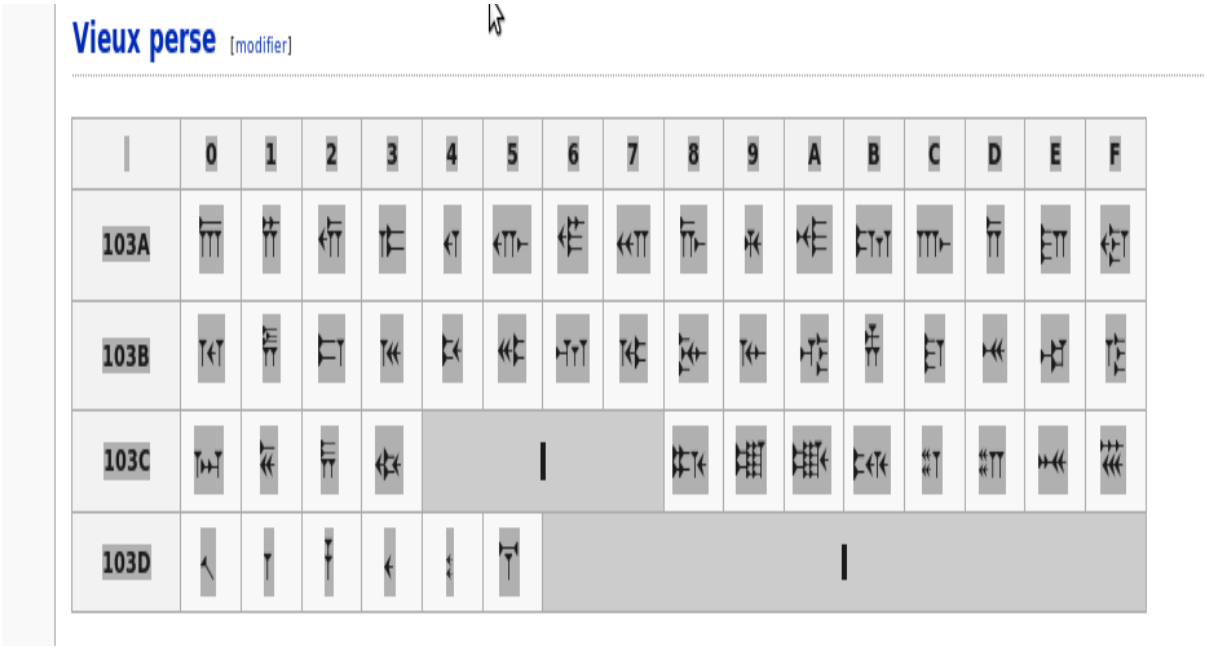
珞 a pour code F917 : plan 0 code F917

voir PDF : fr en	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	襤	拉	臘	蠟	廊	朗	浪	狼	郎	來	冷	勞
F93	擄	櫓	爐	盧	老	蘆	虜	路	露	魯	鷺	碌	祿	綠	菉	錄
F94	鹿	論	壟	弄	籠	聾	牢	磊	賂	雷	壘	屢	樓	淚	漏	累
F95	縷	陋	勒	肋	凜	凌	稜	綾	菱	陵	讀	拏	樂	諾	丹	寧

© Copyright 2005, Philippe Arlotto http://arlotto.univ-tln.fr
Creative Commons Attribution-ShareAlike 2.0 license

Unicode

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



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
0xxx xxxx	1octet , caractères de 0 à 127 (7 bits) ASCII-US
110x 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 :

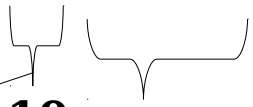
Plus difficile de compter les caractères.
Peu efficace pour les langues utilisant une majorité de caractères non ascii-us (chinois, arabe, ...)


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 (**0**100 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 **110**0 0011 **10**10 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.


En résumé

Les caractères sont représentés par des motifs binaires conventionnels => encodage. Encodages courants :

code ASCII : chaque caractère est représenté sur un octet donc le poids faible est à 0 (0x00 à 0x7F).
128 caractères : majuscules, minuscules , chiffres, ponctuations, contrôles. **Mais pas d'accent !**

code ASCII étendu à un octet : ASCII + 128 caractères supplémentaires (0x80 à 0xFF) → certains caractères accentués, symboles monétaires, ... Plusieurs codages existent : jeux de caractères Ex : latin1, ISO-8859-15, ...

UTF-8 : nombre d'octets variable : un octet identique à l'ASCII si représentable par l'Ascii (0x00 à 0x7F) et 2, 3 ou 4 octets sinon. De très nombreux caractères sont représentables sans ambiguïté. De plus en plus utilisé.


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();

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

Une lecture supplémentaire pour vider le buffer du caractère CR

Les chaînes de caractères

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

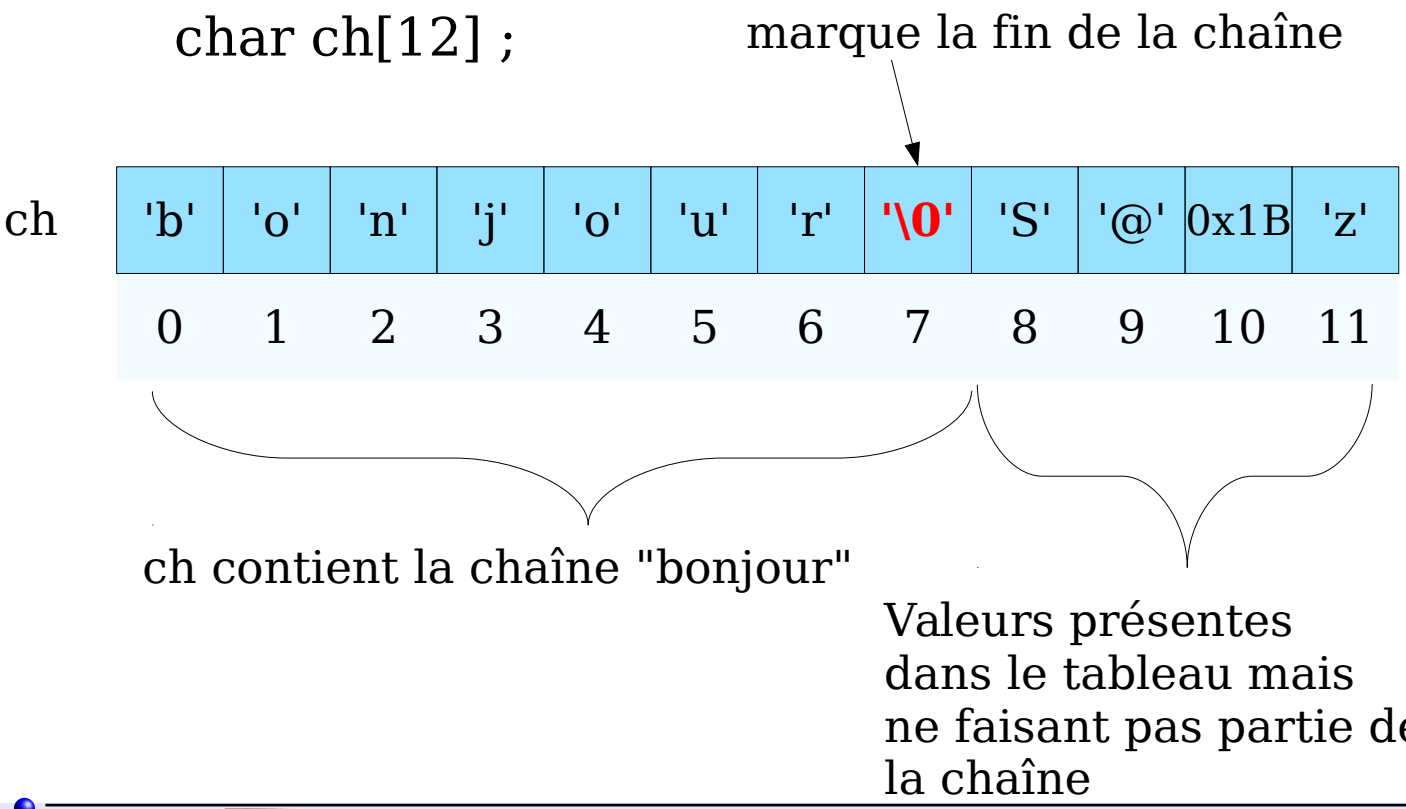
Une chaîne de caractères permet de stocker et de manipuler des mots, des phrases, etc....

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



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.

	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 ;
}
```

```
char ch[80] ;
int i ;

for (i=0 ; ch[i]!='\0' ; i++)
{
  Traiter le caractère n° i ;

}
```


Fonctions ayant des tableaux en paramètres

Fonction retournant la moyenne des valeurs d'un tableau de 10 éléments de type float :

Prototype :

```
float moyenne_tab ( float [] )  ;
```

ou

```
float moyenne_tab ( float * )  ;
```

Attention il n'y a pas d'indication ni de contrôle de la taille du tableau. En fait c'est seulement l'adresse du premier élément qui est passée à la fonction.


Fonctions ayant des tableaux en paramètres

Définition de moyenne_tab :

```
float moyenne_tab ( float tab [ ]  ) {
   int i  ;
   float s = 0 ;
   for ( i = 0 ; i < 10 ; i = i + 1 ) {
     s = s + tab[i] ; }
   s = s / 10 ;
   return s ;
}
```

Appel :

```
float t[10] ;
float moy ;
moy =   moyenne_tab( t  ) ;
```


Fonctions ayant des tableaux en paramètres

La fonction précédente ne peut traiter que des tableaux de 10 éléments. En passant la taille en paramètre, on peut rendre la fonction plus générale :

```
float moyenne_tab ( float * tab , int taille  )  ;

float moyenne_tab ( float * tab  , int taille  ) {
   int i  ;
   float s = 0 ;
   for ( i = 0 ; i < taille ; i = i + 1 ) {
     s = s + tab[i] ; }
   s = s / taille ;
   return s ;
}

moy =  moyenne_tab ( t , 10  ) ;
```


Fonctions sur les chaînes de caractères

Une fonction qui retourne le nombre de caractères 'e' dans une chaîne :

Prototype :

```
int nbe ( char * ) ;
```

une chaîne est un tableau de char

Définition :

```
int nbe ( char * ch ) {
        int ne = 0  , i = 0 ;
        while ( ch[i]!='\0' ) {
                if ( ch[i] == 'e' ) {
                        ne = ne + 1 ; }
              i = i + 1 ;
              }
        return ne ; }
```

Appel :

```
int n ;
n = nbe("ecologiquement" );
```

La fonction peut modifier les éléments d'un tableau

Comme on passe l'adresse du premier élément, une fonction peut agir directement sur les éléments d'un tableau.
Fonction qui remplace les a par des b dans une chaîne :

Prototype : `void Remplace_a_par_b ( char *  ) ;`

Définition :

```
void Remplace_a_par_b ( char * s ) {
        int i ;
        for ( i = 0 ; s[i] != '\0' ; i++ ) {
                if ( s[i] == 'a' ) {
                        s[i] = 'b' ;  }
        }
```

Appel :

```
char t[]="abracadabra!";
Remplace_a_par_b(t);
printf("%s",t) ;   // bbrbcbdbbrb!
```

Prototypes : const , noms de paramètres

On peut ajouter le mot clé const devant le type d'un paramètre tableau pour indiquer que la fonction ne modifie pas la valeur des éléments du tableau.

```
float moyenne_tab ( const char * , int  ) ;
```

On peut nommer les paramètres dans le prototype d'une fonction pour se rappeler leur signification :

```
float moyenne_tab ( const char * tableau , int taille ) ;
```

(les noms sont alors vus comme des commentaires par le compilateur)


Fonctions sur les chaînes de la librairie standard

La librairie standard du C fournie de nombreuses fonctions bien utiles pour traiter les chaînes de caractères.
Les prototypes se trouvent dans le fichier string.h.

Exemples :

Longueur d'une chaîne :

```
int strlen ( char * ch ) ;
```

Recopie de ch2 dans ch1 :

```
void strcpy ( char * ch1 ,  const char * ch2 ) ;
```

Ajout de ch2 dans ch1 après la fin de ch1 :

```
void strcat ( char * ch1 , const char * ch2 ) ;
```

Comparaison de chaînes :

```
int strcmp ( const char * ch1 , const char * ch2 ) ;
```

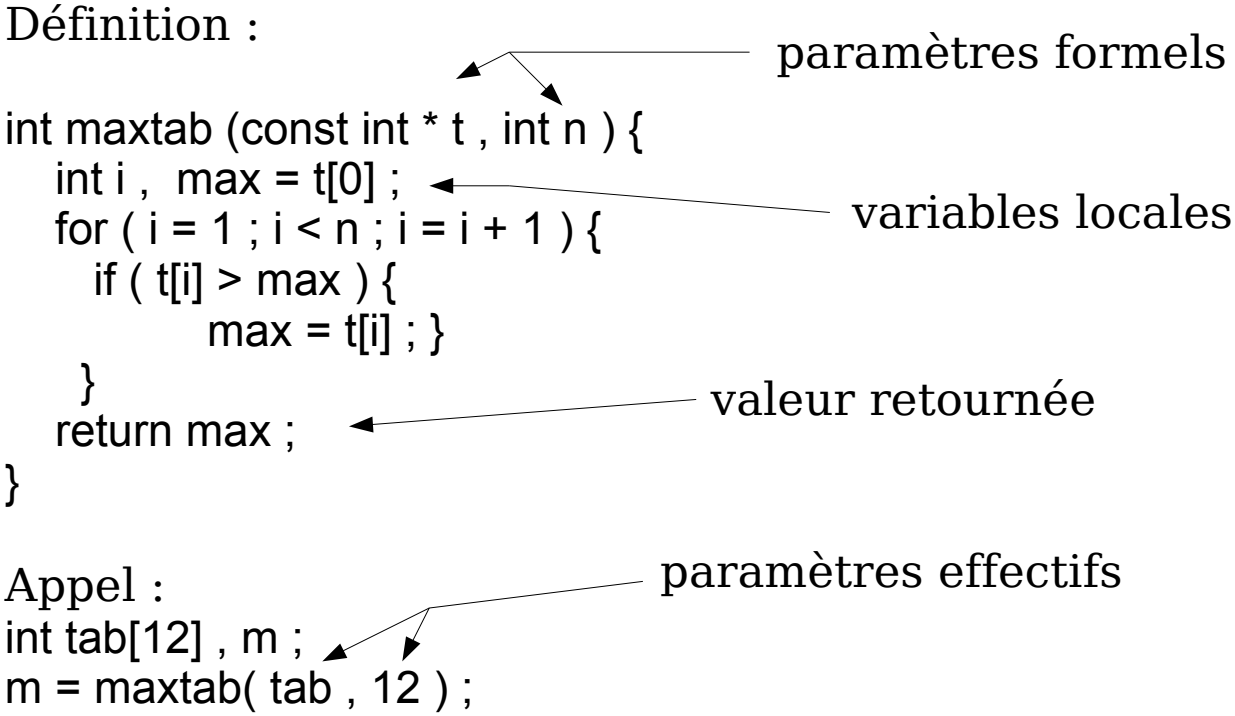
retourne 0 si ch1 et ch2 sont identiques.

voir le fichier d'exemple : fct_chaines.c


Vocabulaire

Prototype : int maxtab (const int * t , int n) ;

Définition :



Appel :

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) ;
```