Développement logiciel (S4)

Initiation au langage C
5- Les chaînes de caractères

Objectifs : Être capable de développer des applications basiques de type console en C. Être capable d'utiliser les chaînes de caractères.

Avertissement

Tout au long de ce TP, vous répondrez au questions posées sur votre compte-rendu. Ce compte-rendu devra inclure les programmes réalisés. Ces programmes devront être commentés !

Les caractères

En C, un caractère est désigné par le type char. Un caractère est codé sur 8bits, qui correspondent au code ASCII du caractère.

b7 →				extra bit for the second set of characters							
b6 →				0	0	0	0	1	1	1	1
b5 →				0	0	1	1	0	0	1	1
b4 →				0	1	0	1	0	1	0	1
b3	b2	b1	b0	r\s	0	1	2	3	4	5	6	7
					column							
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	q	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	T	i	t
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

row

Tableau 1: Code ASCII

Manipulation n°1

Q.1) Analyser le programme fourni ci-après et déterminer ce qui se passe lorsqu'on l'exécute.

Q.2) Saisir et exécuter le programme. Vérifier votre réponse à Q1).

Q.3) Modifier le programme pour que s'affiche l'alphabet en majuscule.

Q.4) Modifier le programme pour que s'affiche l'alphabet en minuscule et à l'envers (z y x …a)

Q.5) Modifier le programme de la question 2 (alphabet en majuscule) en prenant comme variable de boucle un caractère (car) à la place d'un entier. On prendra par exemple le modèle suivant :

```
char car = 0;
for (car='A',  car  ?????; car++)
```

Programme

```
#include <stdio.h>
#include <stdlib.h>

int main()
{   int i = 0;
    for (i=65 ; i<123 ; i++)
    printf("%c ", i);
    return 0;
}
```

A noter un caractère est noté entre apostrophes (guillemets simples). Par exemple : 'A', 'Z', 'x' etc...

Chaînes de caractères

Définition

Une chaîne de caractères (string en anglais) est un tableau de caractères (char). En langage C, les chaînes de caractères sont systématiquement terminées par le caractère de contrôle nul dont le code ASCII est 0 que l'on représente ainsi : '\0'. Ce caractère n'est pas visible à l'écran. C'est le caractère de fin de chaîne
Ainsi la chaîne « PAPA » doit être stockée dans un tableau de 5 caractères ('P' ; 'A' ;'P' ;'A' ;'\0').

Déclaration

Pour déclarer une chaine de caractère, on déclare un tableau de caractères. Le format est le suivant :
`char chaine1 [N]`
où chaine1 est le nom de la variable et N est un entier représentant le nombre de caractères maximum -1 caractère (le '\0').
Exemple :
char toto[20] : toto est une chaîne de caractères de 19 caractères au maximum.

Initialisation

Comme pour un tableau, pour initialiser une chaîne de caractères, on peut initialiser chacun des éléments du tableau (ici chacun des caractères).
Exemple

```
toto[0] = 'P' ;
toto[1] = 'A' ;
toto[2] = 'P' ;
toto[3] = 'A' ;
toto[4] = '\0' ;
```

ou

```
char toto[20]={ 'P', 'A', 'P', 'A', '\0' };
```

Il est possible aussi d'écrire

```
char toto[20]= "PAPA";
```

mais seulement à l'initialisation !

Remarque :

```
char toto[]= "PAPA" ;
```

fonctionne aussi à l'initialisation mais réserve à la compilation en mémoire une chaîne de 5 caractères.

Pour modifier une chaîne dans un programme, il est nécessaire de modifier chacun des caractères un par un !

Variante : déclaration et initialisation à l'aide d'un pointeur

Il est possible de déclarer une **chaîne de caractère constante** sous la forme d'un pointeur :

Exemple :

```
char *chaine2 = "PAPA" ;
```

A la compilation cette méthode ne réserve pas de mémoire, elle se contente de stocker l'adresse d'une chaîne de caractères qui peut se trouver dans un endroit de la mémoire non modifiable par le programme, dans ce cas toute modification de chaine2 se conclura par un comportement indéterminé. Il est donc recommandé d'écrire :

```
const char *chaine2 = "PAPA" ;
```

Afficher une chaîne de caractères

Pour afficher une chaîne de caractère, on peut utiliser l'instruction **printf()** comme pour toute autre type. Le spécificateur de format pour une chaîne est %s (comme string).
Exemple d'utilisation : printf("%s \n", toto);
On peut aussi utiliser l'instruction **puts()** (put string). Avec cette instruction, il n'est pas nécessaire de spécifier le format (puisqu'elle ne s'applique qu'au chaînes).
puts(toto) est équivalent à printf("%s \n", toto);

Lire une chaîne de caractères saisie

Pour lire une chaîne de caractère, on peut utiliser **scanf()**. Comme pour printf() Le spécificateur de format est %s. format est
Exemple d'utilisation : scanf(%s, toto)
Attention, c'est bien scanf(%s, toto) et non pas scanf(%s, &toto) !!
En effet toto représente l'adresse de la chaîne de caractère toto[], il ne faut pas mettre le & !

Attention scanf ne prend pas en compte tous les caractères situés après une espace ou une tabulation. Scanf() ne peut donc pas gérer des phrases .

L'instruction **gets()** (get string) est idéale pour lire une ligne de texte (par exemple des phrases contenant des mots séparés par des espaces) terminées par un retour à la ligne. Le retour à la ligne final est remplacé par le caractère de fin de chaîne (\0) mais cela est transparent pour l'utilisateur.
Exemple d'utilisation : gets(toto)
Attention : gets() ne gère pas les débordements de taille ! Certains préfèrent utiliser l'instruction **fgets()** qui permet la lecture d'une chaine à partir d'un fichier mais qui limite la taille de la chaîne.
Exemple d'utilisation : fgets(toto, 100, stdin)
La chaîne toto sera limitée à 99 caractères (le 100eme sera le '\0') même si l'utilisateur saisit une chaîne plus longue ! Le mot stdin désigne le flux d'entrée standard : il correspond à la saisie au clavier.

Exercices sur les chaînes de caractères

Quelques opérations simples

Manipulation n°2

Q.1) Écrire un programme qui demande à l'utilisateur de saisir une chaîne de caractères (de longueur max =100 caractères). Puis qui compte le nombre de caractères de la chaîne.

Q.2) Compléter le programme et compter aussi le nombre d'espaces.

Q.3) Compléter le programme et compter le nombre de e (majuscules et minuscules) dans la chaîne de caractères saisie par l'utilisateur. Proposer ensuite à l'utilisateur de remplacer tous les e par une autre lettre saisie par l'utilisateur.

Le code de César decrypté !

Le code de César, (en toute rigueur on doit dire le chiffre de César), est une de plus simples et des plus anciennes méthode de chiffrement. Elle était utilisée par Jules César dans ces correspondances secrètes.
Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres (dans le cas d'un décalage à droite), on reprend au début. Par exemple avec un décalage de 3 vers la droite, A est remplacé par D, B devient E, et ainsi jusqu'à W qui devient Z, puis X devient A etc. Il s'agit d'une

permutation circulaire de l'alphabet. La longueur du décalage, 3 dans l'exemple évoqué, constitue la clé du chiffrement qu'il suffit de transmettre au destinataire — s'il sait déjà qu'il s'agit d'un chiffrement de César — pour que celui-ci puisse déchiffrer le message. Dans le cas de l'alphabet latin, le chiffre de César n'a que 26 clés possibles (y compris la clé nulle, qui ne modifie pas le texte).

La figure suivante montre une roue permettant de faire le codage ou décodage de César. Normalement les deux disques sont mobiles l'un par rapport à l'autre. Sur la roue extérieure figure l'alphabet du message en clair, sur la roue intérieure figure l'alphabet du message codé. Ici, la configuration des roues correspond à un décalage de 19 le A (1ere lettre de l'alphabet) est remplacé par le T (20eme lettre de l'alphabet).

Avec ce code, le mot « BTS » sera codé « UML » !
Remarque : le décodage se fait en décalant le message reçu de la quantité inverse de celle du codage. Par exemple, un message codé avec un décalage de +2 lettres sera décodé en effectuant un décalage de – 2 lettres.

Manipulation n°3

Q.1) Écrire un programme qui permet d'effectuer le codage ou le décodage d'une phrase (maximum 99 caractères) saisie au clavier. Le décalage (positif pour le codage, négatif pour le décodage) est rentré par l'utilisateur. **On choisit de ne coder que les majuscules** : les minuscules, la ponctuation etc, ne sont pas codés !

Q.2) Décoder le message suivant. Pour la clé, un seul indice : Avocat !

```
MOCKB ODKSD EX ZSQXYEP, XKZYVOYX ODKSD EX LYX !
```

La librairie string

Pour manipuler aisément des chaînes de caractères, il existe la bibliothèque string.h. Celle ci propose de nombreuses fonctions accessible à partir du moment où l'on a saisi la ligne :

```
#include <string.h>
```

Nous n'allons pas présenter ici toutes les fonctions de cette bibliothèque mais simplement quelques unes parmi les plus utiles.

La fonction **strcpy** permet de copier une chaîne dans l'autre.
La fonction **strcat** permet de concaténer deux chaînes de caractères, c'est-à-dire qu'elle les met bout à bout l'une à la suite de l'autre dans une même chaîne.
Le fonction **strcmp** permet de comparer deux chaînes pour l'ordre alphabétique.
La fonction **strlen** retourne la longueur d'une chaîne de caractères, c'est-à-dire son nombre de caractères (sans compter le '\0').

Bonus : manipulation n°4

Q.1) Écrire un programme qui lit un verbe régulier en "er" au clavier et qui en affiche la conjugaison au présent de l'indicatif de ce verbe. Voir exemple ci-après.
Cahier des charges : Contrôler s'il s'agit bien d'un verbe en "er" avant de conjuguer.

Conseils et astuces (ce n'est pas obligatoire mais cela peut simplifier la lecture de votre code) :

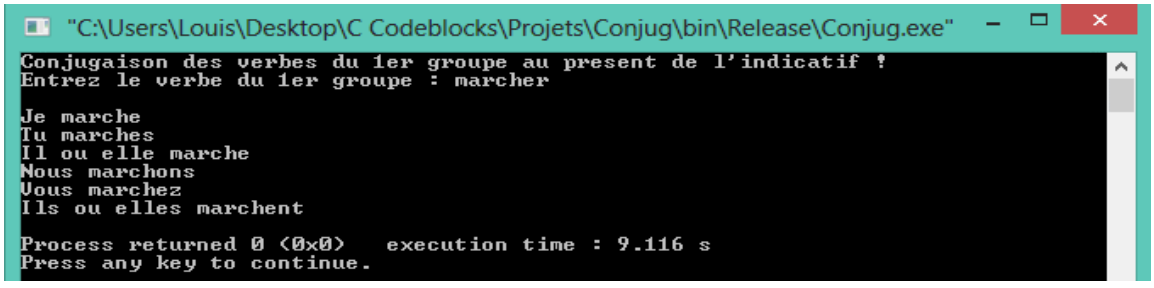
Vous pourrez par exemple utiliser les fonctions :
strlen() pour déterminer la longueur du verbe à conjuguer,
strcmp() pour vérifier que la fin du verbe est bien « er »,
strcpy() pour copier le radical du verbe (march pour le verbe marcher) dans une variable intermédiaire.

Vous pourrez aussi utiliser des tableaux de chaîne de caractères comme par exemple

```
char terminaison [6][4] = {"e", "es", "e", "ons", "ez", "ent" };
 char sujet[6][14] =
    { "Je ", "Tu ", "Il ou elle ", "Nous ", "Vous ", "Ils ou elles "   };
```

Super Bonus des champions : ajouter à votre programme les verbes du second groupe (en ir).

Exemple :



Retrouvez d'autres cours et documents sur :
http://www.louisreynier.com