Algorithmique et programmation

Chapitre 9:Chaînes de caractères

1. Déclaration, saisie et affichage
2. Fonctions de traitement
3. Fonctions de conversion

Définition

- Une **chaîne de caractères** (String en anglais) est un tableau d'éléments de type char, dont le dernier élément est le caractère nul \0.
- Le caractère \0 indique la fin de la chaîne de caractères.
- Une chaîne composée de *n* éléments sera en fait un tableau de *n+1* éléments de type char.

M	I	P	C		S	3	\0

Déclaration (1)

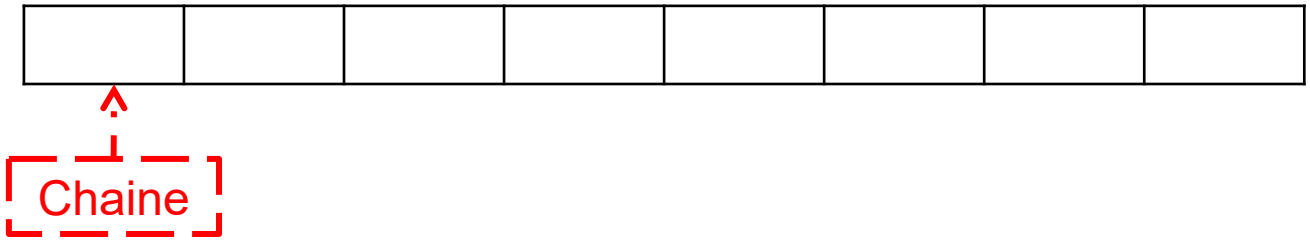
```
char Nom_chaîne[taille];
```

Le nom de la chaîne est un pointeur sur le premier caractère

taille = nombre d'éléments + 1

Exemple:

```
char Chaine[7+1];
```



Déclaration (2)

```
char *Nom_chaîne;
 Nom_chaîne=(char*)malloc(taille*sizeof(char));
 …/*Ajouter un traitement en cas d'échec*/
```

- **Exemple**:

```
char *Ch;
int taille;
taille=20;
Ch=(char*)malloc(taille*sizeof(char));
 … /*Ajouter un traitement en cas d'échec*/
free(Ch);
```

- **Remarque**: à la fin du traitement, il faut libérer la zone mémoire allouée à l'aide de la fonction **free()**.

Initialisation

```
char Nom_chaîne[taille] ={'caract1','caract2',…'\0'};
```

Ou bien

```
char Nom_chaîne[taille] ="caract1caract2…";
```

Ou bien

```
char *Nom_chaîne ="caract1caract2…";
```

Exemple:

```
char Chaine[7+1] = {'M','I','P','C',' ','S','3','\0'};
```

```
char Chaine[7+1] = "MIPC S3";
```

```
char *Chaine = "MIPC S3";
```

Saisie

• La lecture d'une chaîne de caractères peut être effectuée à l'aide des deux fonctions suivantes: **scanf** (le code format à utiliser est %s) et **gets**.

• **Exemple:**

```
char Chaine[20];

scanf(" %s",Chaine);
```

```
char Chaine[20];

gets(Chaine);
```

Affichage

- L'affichage d'une chaîne de caractères peut être effectué à l'aide des deux fonctions: **printf** (le code format à utiliser est %s) et **puts**.
- **Exemple:**

```
char Chaine[20]="MIPC S3";
printf(" %s",Chaine);
```

```
char Chaine[20]="MIPC S3";
puts(Chaine);
```

Traitement (1)

- **Calcul de la longueur d'une chaîne de caractères**

```
char ch[20];
int i=0;
printf(" Entrer une chaine de caractères:\n");
gets(ch);
while(ch[i]!='\0')   /*     *(ch+i)!='\0'      */
  { i++; }
printf ("La longueur de la chaine est : %d  \n",i);
```

Traitement (2)

- **Calcul du nombre d'apparition d'un caractère dans une chaîne de caractères**

```
char ch[20],caract;
int app=0,i=0;
printf(" Entrer une chaine de caractères:\n");
gets(ch);
printf(" Entrer la caractère à chercher:\n");
scanf("%c",&caract);
for(i=0;ch[i]!='\0';i++)
  {if(ch[i]==caract)  app++; }
printf ("Le nombre d'apparition de %c dans la chaine %s est : %d \n",caract,ch,app);
```

Fonctions de traitement

- **string.h** : une bibliothèque contenant les fonctions de manipulation des chaînes de caractères
- **strlen**: une fonction qui retourne la longueur d'une chaîne de caractères passée en argument (passage par adresse).

```
strlen(Chaine);
```

```
char *ch = "MIPC S3";
int long;
long = strlen(ch);            /*long = 7  */
printf(" La longueur de la chaine de caractères est: %d\n",long);
```

Fonctions de traitement (suite)

•**strcat**: une fonction qui recopie une chaîne de caractères à la suite d'une autre chaîne après l'écrasement du caractère nul « \0 ».

strcat(Chaine1,Chaine2);

```
char ch1[50] = "MIPC";
char *ch2 = " S3";
printf(" Avant la concaténation  ch1: %s\n",ch1);          /*  MIPC  */
strcat(ch1,ch2);
printf(" Après la concaténation  ch1: %s\n",ch1);          /*  MIPC S3  */
```

Fonctions de traitement (suite)

- **strncat**: une fonction qui recopie les *n* premiers caractères d'une chaîne de caractères à la suite d'une autre chaîne après l'écrasement du caractère nul « \0 ».

strncat(Chaine1,Chaine2,n);

```
char ch1[50] = "MIPC";
char *ch2 = " S3";
printf(" Avant la concaténation  ch1: %s\n",ch1);          /*  MIPC  */
strncat(ch1,ch2,2);
printf(" Après la concaténation  ch1: %s\n",ch1);          /*  MIPC S  */
```

Fonctions de traitement (suite)

- **strcmp**: une fonction qui compare deux chaînes de caractères (chaîne 1 et chaîne 2) selon l'ordre lexicographique du code ASCII et retourne une valeur entière:

strcmp(Chaine1,Chaine2);

> 0 si chaine 1 est après chaine 2

= 0 si les deux chaines sont identiques

< 0 si chaine 1 est avant chaine 2

```
char  ch1[20],ch2[20];
gets(ch1); gets(ch2);
if(strcmp(ch1,ch2) > 0) printf(" %s après %s \n",ch1,ch2);
if(strcmp(ch1,ch2) < 0) printf(" %s avant %s \n",ch1,ch2);
if(strcmp(ch1,ch2) == 0) printf(" %s et %s sont identiques\n",ch1,ch2);
```

Fonctions de traitement (suite)

Code ASCII

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01		SOH
^B	2	02		STX
^C	3	03		ETX
^D	4	04		EOT
^E	5	05		ENQ
^F	6	06		ACK
^G	7	07		BEL
^H	8	08		BS
^I	9	09		HT
^J	10	0A		LF
^K	11	0B		VT
^L	12	0C		FF
^M	13	0D		CR
^N	14	0E		SO
^O	15	0F		SI
^P	16	10		DLE
^Q	17	11		DC1
^R	18	12		DC2
^S	19	13		DC3
^T	20	14		DC4
^U	21	15		NAK
^V	22	16		SYN
^W	23	17		ETB
^X	24	18		CAN
^Y	25	19		EM
^Z	26	1A		SUB
^[	27	1B		ESC
^\	28	1C		FS
^]	29	1D		GS
^^	30	1E	▲	RS
^-	31	1F	▼	US

Dec	Hex	Char
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂

Fonctions de traitement (suite)

- **strncmp**: une fonction qui compare les n premiers caractères de deux chaînes et retourne une valeur entière (positive, nulle et négative)

```
strncmp(Chaine1,Chaine2,n);
```

- **stricmp et strnicmp**: deux fonctions fonctionnent de la même façon que **strcmp** et **strncmp** sans tenir compte de la différence entre les majuscules et les minuscules.

```
stricmp(Chaine1,Chaine2);
```

```
strnicmp(Chaine1,Chaine2,n);
```

Fonctions de traitement (suite)

•**strcpy**: une fonction qui recopie une chaîne de caractères dans l'emplacement d'une autre chaîne après l'écrasement de la chaîne destinataire.

strcpy(Chaine1,Chaine2);

```
char ch1[50] = "MIPC";
char *ch2 = " S3";
printf(" Avant: ch1= %s\n",ch1);          /*   ch1="MIPC"   */
strcpy(ch1,ch2);
printf(" Après: ch1= %s\n",ch1);          /*   ch1="  S3"   */
```

Fonctions de traitement (suite)

•**strncpy**: une fonction qui recopie les *n* premiers caractères d'une chaîne de caractères dans l'emplacement d'une autre chaîne après l'écrasement de leurs équivalents dans la chaîne destinataire.

strncpy(Chaine1,Chaine2,n);

```
char ch1[50] = "MIPC";
char *ch2 = " S3";
printf(" Avant: ch1= %s\n",ch1);          /*  ch1="MIPC"  */
strncpy(ch1,ch2,2);
printf(" Après: ch1= %s\n",ch1);          /*  ch1="  SPC"  */
```

Fonctions de traitement (suite)

•**strchr**: une fonction qui recherche le premier occurrence d'un caractère dans une chaîne de caractères et retourne le pointeur équivalent.

```
strchr(Chaine,caractère);
```

•**strstr**: une fonction qui recherche le premier occurrence d'une suite complète de caractères dans une chaîne de caractères et retourne le pointeur équivalent.

```
strstr(Chaine,suite_caractères);
```

Fonctions de conversion

- **atoi**: une fonction qui convertit une chaîne de caractères en une valeur numérique (valeur de retour) de type **int**.

```
atoi(Chaine);
```

- **atol**: une fonction qui convertit une chaîne de caractères en une valeur numérique (valeur de retour) de type **double**.

```
atol(Chaine);
```

- **atof**: une fonction qui convertit une chaîne de caractères en une valeur numérique (valeur de retour) de type **float**.

```
atof(Chaine);
```

Remarque: Le premier caractère invalide arrête l'exploration et si aucun caractère n'est exploitable, ces fonctions fournissent un résultat nul.

Exercice1

Ecrire un programme qui lit, en donnée, un verbe du premier groupe et qui affiche la conjugaison au présent de l'indicatif, sous la forme:

je chante
tu chantes
il chante
nous chantons
vous chantez
ils chantent

On s'assurera que le mot fourni se termine bien par « er ». On supposera qu'il s'agit d'un verbe régulier.

Les fonctions à utiliser sont:

- void **lecture** (char chaine[max])
- int **verbe_er** (char chaine[max])
- void **conjuguer** (char chaine[max])
- void **main**()