SUP'MANAGEMENT

ECOLE SUPERIEURE DE MANAGEMENT
DE COMMERCE ET D'INFORMATIQUE
Reconnue par l'Etat

Les chaînes de caractères

Cours Algorithmique et Programmation C Avancées

Module Informatique I I

Plan

- Déclaration et mémorisation
- Les chaînes de caractères constantes
- Initialisation de chaînes de caractères
- Accès aux éléments d'une chaîne
- Précédence alphabétique et lexicographique
- Travailler avec des chaînes de caractères
- Tableaux de chaînes de caractères
- Exercices d'application

Déclaration et mémorisation

- ***Déclaration de chaînes de caractères en C***

  char <NomVariable> [<Longueur>];

- ***Exemples***

  char NOM [20];

  char PRENOM [20];

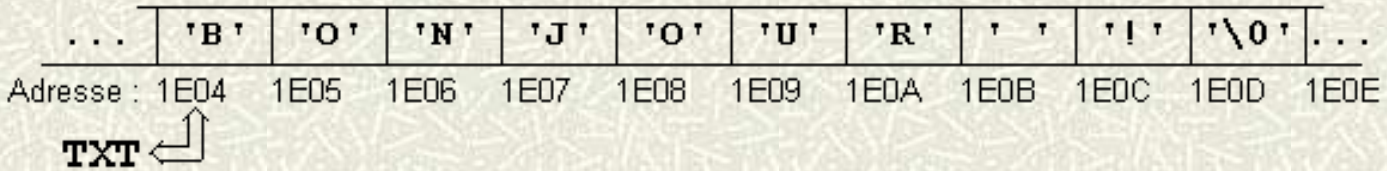
  char PHRASE [300];

Mémorisation

Le nom d'une chaîne est le représentant de ***l'adresse du premier caractère*** de la chaîne. Pour mémoriser une variable qui doit être capable de contenir un texte de N caractères, nous avons besoin de N+1 octets en mémoire.

- ***Exemple: Mémorisation d'un tableau***

char TXT[10] = "BONJOUR !";



La représentation interne d'une chaîne de caractères est terminée par le symbole **'\0'** (NUL).

Les chaînes de caractères constantes

- La chaîne de caractères vide est : **""**

- Dans les chaînes de caractères, nous pouvons utiliser toutes les séquences d'échappement définies comme caractères constants:

"Ce \ntexte \nsera réparti sur 3 lignes."

- Le symbole " peut être représenté à l'intérieur d'une chaîne par la séquence d'échappement \": **"Affichage de \"guillemets\" \n"**

- Le symbole ' peut être représenté à l'intérieur d'une liste de caractères par la séquence d'échappement \' : **{'L','\'','a','s','t','u','c','e','\0'}**

Initialisation de chaînes de caractères

char CHAINE[] = {'H','e','l','l','o','\0'};

Ou

char CHAINE[] = "Hello";

Jamais char CHAINE = "Hello";

Les chaînes de caractères constantes

```
char TXT[] = "Hello";
```

TXT:	'H'	'e'	'l'	'l'	'o'	'\0'

```
char TXT[8] = "Hello";
```

TXT:	'H'	'e'	'l'	'l'	'o'	'\0'	0	0

```
char TXT[6] = "Hello";
```

TXT:	'H'	'e'	'l'	'l'	'o'	'\0'

Les chaînes de caractères constantes

```
char TXT[5] = "Hello";
```

TXT:	'H'	'e'	'l'	'l'	'o'	

ERREUR pendant l'exécution

```
char TXT[4] = "Hello";
```

ERREUR pendant la compilation

Exercice

Lesquelles des chaînes suivantes sont initialisées correctement ? Corrigez les déclarations fausses et indiquez pour chaque chaîne de caractères le nombre d'octets qui sera réservé en mémoire.

- **a) char a[] = "un\ndeux\ntrois\n";**
- **b) char b[12] = "un deux trois";**
- **c) char c[] = 'abcdefg';**
- **d) char d[10] = 'x';**
- **e) char e[5] = "cinq";**
- **f) char f[] = "Cette " "phrase" "est coupée";**
- **g) char g[2] = {'a', '\0'};**
- **h) char h[4] = {'a', 'b', 'c'};**
- **i) char i[4] = "'o'";**

Solution

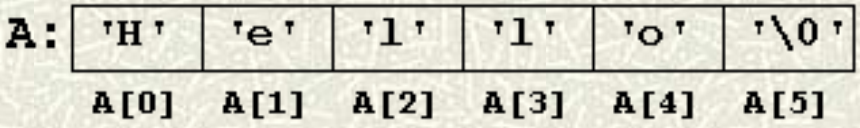
- **a) char a[] = "un\ndeux\ntrois\n";**
- **b) char b[12] = "un deux trois";**
- **c) char c[] = 'abcdefg';**
- **d) char d[10] = 'x';**
- **e) char e[5] = "cinq";**
- **f) char f[] = "Cette " "phrase" "est coupée";**
- **g) char g[2] = {'a', '\0'};**
- **h) char h[4] = {'a', 'b', 'c'};**
- **i) char i[4] = "'o'";**

Accès aux éléments d'une chaîne

- L'accès à un élément d'une chaîne de caractères peut se faire de la même façon que l'accès à un élément d'un tableau. En déclarant une chaîne par:
- **char A[6];**nous avons défini un tableau A avec six éléments, auxquels on peut accéder par:

  A[0], A[1], ... , A[5]
- ***Exemple***

```
char A[6] = "Hello";
```



Précédence alphabétique et lexicographique

- La précédence des caractères dans l'alphabet d'une machine est dépendante du code de caractères utilisé. Pour le code ASCII, nous pouvons constater l'ordre suivant:

  . . . ,0,1,2, ... ,9, . . . ,A,B,C, ... ,Z, . . . ,a,b,c, ... ,z, . . .

- Les symboles spéciaux (' ,+ ,- ,/ ,{ ,] , ...) et les lettres accentuées (é ,è ,à ,û , ...) se trouvent répartis autour des trois grands groupes de caractères (chiffres, majuscules, minuscules). Leur précédence ne correspond à aucune règle d'ordre spécifique.

Relation de précédence

- De la précédence alphabétique des caractères, on peut déduire une ***relation de précédence*** *'est inférieur à'* sur l'ensemble des caractères. Ainsi, on peut dire que

  '0' *est inférieur à* **'Z'** et noter **'0' < 'Z'**

- car dans l'alphabet de la machine, le code du caractère '0' (ASCII: 48) est inférieur au code du caractère 'Z' (ASCII: 90).

Précédence lexicographique des chaînes de caractères

- **a)** La chaîne vide "" précède lexicographiquement toutes les autres chaînes.
- **b)** La chaîne A = "a_1a_2a ... a_p" (p caractères) précède lexicographiquement la chaîne B = "b_1b_2 ... b_m" (m caractères) si l'une des deux conditions suivantes est remplie:

  1) 'a_1' < 'b_1'

  2) 'a_1' = 'b_1' et "a_2a_3 ... a_p" précède lexicographiquement "b_2b_3 ... b_m"

Précédence lexicographique des chaînes de caractères

- ***Exemples***

  - "**ABC**" précède "**BCD**" car '**A**'<'**B**'
  - "**ABC**" précède "**B**" car '**A**'<'**B**'
  - "**Abc**" précède "**abc**" car '**A**'<'**a**'
  - "**ab**" précède "**abcd**" car "" précède "**cd**"
  - " **ab**" précède "**ab**" car ' '<'**a**'

  (le code ASCII de ' ' est 32, et le code ASCII de '**a**' est 97)

Précédence lexicographique des chaînes de caractères

- ***Conversions et tests***

  En tenant compte de l'ordre alphabétique des caractères, on peut contrôler le type du caractère (chiffre, majuscule, minuscule).

- ***Exemples***

```
if (C>='0' && C<='9') printf("Chiffre\n");
if (C>='A' && C<='Z') printf("Majuscule\n");
if (C>='a' && C<='z') printf("Minuscule\n");
```

Il est facile, de convertir des lettres majuscules dans des minuscules:

```
if (C>='A' && C<='Z') C = C-'A'+'a';
```

ou vice-versa:

```
if (C>='a' && C<='z') C = C-'a'+'A';
```

Travailler avec des chaînes de caractères

- Les bibliothèques de fonctions de C contiennent une série de fonctions spéciales pour le traitement de chaînes de caractères.
  - Les fonctions de <stdio.h>
  - Les fonctions de <string>
  - Les fonctions de <stdlib>
  - Les fonctions de <ctype>

Les fonctions de <stdio.h>

- **Affichage de chaînes de caractères**

printf avec le spécificateur de format **%s** permet d'intégrer une chaîne de caractères dans une phrase.

Exemples

```
char NOM[] = "hello, world";

printf(":%s:", NOM);          -> :hello, world:
printf(":%5s:", NOM);         -> :hello, world:
printf(":%15s:", NOM);        -> :   hello, world:
printf(":%-15s:", NOM);       -> :hello, world   :
printf(":%.5s:", NOM);        -> :hello:
```

Les fonctions de <stdio.h>

- **Affichage de chaînes de caractères**

  puts est idéale pour écrire une chaîne constante ou le contenu d'une variable dans une ligne isolée.

Syntaxe:

puts(<Chaîne>)

Effet:

puts écrit la chaîne de caractères désignée par <Chaîne> sur *stdout* et provoque un retour à la ligne. En pratique,

puts(TXT); est équivalent à **printf("%s\n",TXT);**

Les fonctions de <stdio.h>

- **Affichage de chaînes de caractères**

Exemples:

```
char TEXTE[] = "Voici une première ligne.";
puts(TEXTE);
puts("Voici une deuxième ligne.");
```

Les fonctions de <stdio.h>

- **Lecture de chaînes de caractères**

  scanf avec le spécificateur **%s** permet de lire un mot isolé à l'intérieur d'une suite de données du même ou d'un autre type.

Effet:

scanf avec le spécificateur **%s** lit un ***mot*** du fichier d'entrée standard *stdin* et le mémorise à l'adresse qui est associée à **%s**.

Les fonctions de <stdio.h>

- **Lecture de chaînes de caractères**

Exemple

```
char LIEU[25];
int JOUR, MOIS, ANNEE;
printf("Entrez lieu et date de naissance : \n");
scanf("%s %d %d %d", LIEU, &JOUR, &MOIS, &ANNEE);
```

Les fonctions de <stdio.h>

- ***Remarques importantes***

- La fonction **scanf** a besoin des ***adresses de ses arguments:***

* Les noms des variables numériques (**int**, **char**, **long**, **float**, ...) doivent être marqués par le symbole **'&'**

***** *Comme le nom d'une chaîne de caractères est le représentant de l'adresse du premier caractère de la chaîne,* ***il ne doit pas être précédé de l'opérateur adresse '&' !***

Les fonctions de <stdio.h>

- **gets** est idéal pour lire une ou plusieurs lignes de texte (p.ex. des phrases) terminées par un retour à la ligne.
  Syntaxe:

  gets(<Chaîne>)

  Effet:
  gets lit une ***ligne*** de de caractères de *stdin* et la copie à l'adresse indiquée par <Chaîne>. Le retour à la ligne final est remplacé par le symbole de fin de chaîne **'\0'**.

Exemple

```
int MAXI = 1000;
char LIGNE[MAXI];
gets(LIGNE);
```

Exercices

Exercice1 :

Ecrivez un programme qui prend une chaîne en majuscules et qui la convertit en minuscules.

le code ASCII des caractères alphabétiques en minuscules est compris entre 97 (a) et 122 (z)

le code ASCII des caractères alphabétiques en majuscules est compris entre 65 (A) et 90 (Z)

NB : la fonction non ANSI strlwr réalise cette conversion.

Exercice2 :

Ecrivez un programme qui prend une chaîne en minuscules et qui la convertit en majuscules.

NB : la fonction non ANSI strupr réalise cette conversion.

Les fonctions de <string>

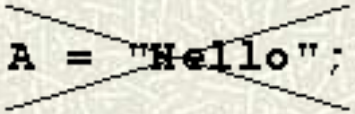
- **Fonctions pour le traitement de chaînes de caractères**

Fonctions	Description
strlen(<s>)	fournit la longueur de la chaîne ***sans*** compter le '\0' final
strcpy(<s>, <t>)	copie <t> vers <s>
strcat(<s>, <t>)	ajoute <t> à la fin de <s>
strcmp(<s>, <t>)	compare <s> et <t> lexicographiquement et fournit un résultat négatif si <s> précède <t> , zéro si <s> est égal à <t> positif si <s> suit <t>
strncpy(<s>, <t>, <n>)	copie au plus <n> caractères de <t> vers <s>
strncat(<s>, <t>, <n>)	ajoute au plus <n> caractères de <t> à la fin de <s>

Les fonctions de <string>

- ***Remarques***

  - Comme le nom d'une chaîne de caractères représente une adresse fixe en mémoire, on ne peut pas 'affecter' une autre chaîne au nom d'un tableau:



  il faut bien copier la chaîne caractère par caractère ou utiliser la fonction **strcpy** respectivement **strncpy**:

  strcpy(A, "Hello");

Les fonctions de <string>

- ***Remarques***
  - La concaténation de chaînes de caractères en C ne se fait pas par le symbole `'+'`
  - Il faut ou bien copier la deuxième chaîne caractère par caractère ou bien utiliser la fonction **strcat** ou **strncat**.

Les fonctions de <stdlib>

La bibliothèque <*stdlib*> contient des déclarations de fonctions pour la conversion de nombres en chaînes de caractères et vice-versa.

- **Conversion de chaînes de caractères en nombres**

atoi(<s>) retourne la valeur numérique représentée par <s> comme **int**

atol(<s>) retourne la valeur numérique représentée par <s> comme **long**

atof(<s>) retourne la valeur numérique représentée par <s> comme **double** (!)

Les fonctions de <stdlib>

- **Règles générales pour la conversion:**
  - Les espaces au début d'une chaîne sont ignorés
  - Il n'y a pas de contrôle du domaine de la cible
  - La conversion s'arrête au premier caractère non convertible
  - Pour une chaîne non convertible, les fonctions retournent zéro

Les fonctions de <ctype>

Les fonctions de <*ctype*> servent à classifier et à convertir des caractères.

Fonctions de classification et de conversion

Les fonctions de ***classification*** suivantes fournissent un résultat du type **int** différent de zéro, si la condition respective est remplie, sinon zéro.

Les fonctions de <ctype>

Fonctions de classification

La fonction:	Retourne une valeur différente de zéro
isupper(<c>)	si <c> est une majuscule **('A'...'Z')**
islower(<c>)	si <c> est une minuscule **('a'...'z')**
isdigit(<c>)	si <c> est un chiffre décimal **('0'...'9')**
isalpha(<c>)	si **islower(<c>)** ou **isupper(<c>)**
isalnum(<c>)	si **isalpha(<c>)** ou **isdigit(<c>)**
isxdigit(<c>)	si <c> est un chiffre hexadécimal (**'0'...'9'** ou **'A'...'F'** ou **'a'...'f'**)
isspace(<c>)	si <c> est un signe d'espacement (' ', **'\t'**, **'\n'**, **'\r'**, **'\f'**)

<c> représente une valeur du type **int** qui peut être représentée comme caractère.

Les fonctions de <ctype>

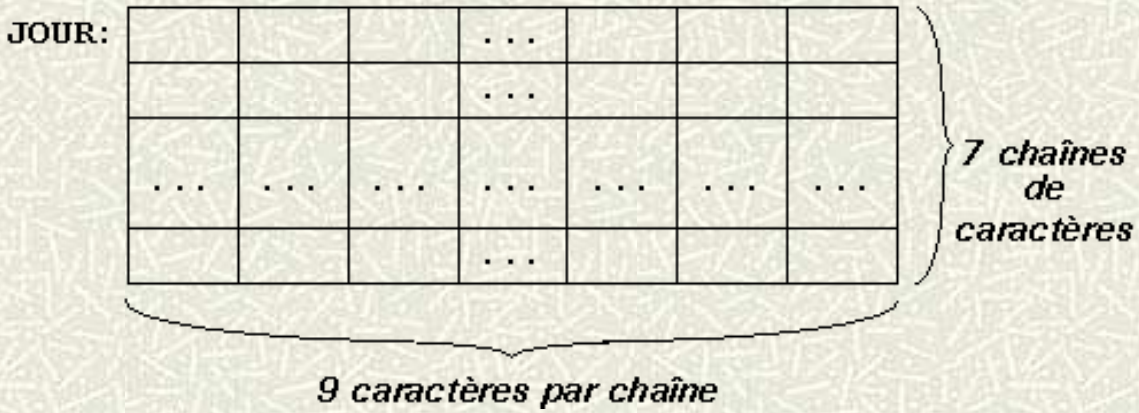
- Les fonctions de ***conversion*** suivantes fournissent une valeur du type **int** qui peut être représentée comme caractère; la valeur originale de <c> reste inchangée:
- **tolower(<c>)** retourne <c> converti en minuscule si <c> est une majuscule
- **toupper(<c>)** retourne <c> converti en majuscule si <c> est une minuscule

Tableaux de chaînes de caractères

- **Déclaration, initialisation et mémorisation**

Un tableau de chaînes de caractères correspond à un tableau à deux dimensions du type **char**, où ***chaque ligne contient une chaîne de caractères***.

La déclaration **char JOUR[7][9];**



Tableaux de chaînes de caractères

La déclaration et initialisation:

char JOUR[7][9]= {"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "dimanche"};

JOUR:									
	'l'	'u'	'n'	'd'	'i'	'\0'			
	'm'	'a'	'r'	'd'	'i'	'\0'			
	'm'	'e'	'r'	'c'	'r'	'e'	'd'	'i'	'\0'
	...	...	...	...	...	...	...	...	...
	'd'	'i'	'm'	'a'	'n'	'c'	'h'	'e'	'\0'

Accès aux différentes composantes

- ***Accès aux chaînes***

  Il est possible d'accéder aux différentes ***chaînes de caractères*** d'un tableau, en indiquant simplement la ligne correspondante.

- ***Exemple***

```
char JOUR[7][9]= {"lundi", "mardi", "mercredi", "jeudi", "vendredi",
    "samedi", "dimanche"};

int I = 2;

printf("Aujourd'hui, c'est %s !\n", JOUR[I]);
```

affichera la phrase:

Aujourd'hui, c'est mercredi !

Accès aux différentes composantes

- ***Affectation***

Des expressions comme JOUR[I] représentent *l'adresse* du premier élément d'une chaîne de caractères. N'essayez donc pas de 'modifier' une telle adresse par une affectation directe !

L'attribution d'une chaîne de caractères à une composante d'un tableau de chaînes se fait en général à l'aide de la fonction **strcpy**:

Accès aux différentes composantes

- ***Affectation***

Exemple

La commande

strcpy(JOUR[4], "Friday");

changera le contenu de la 5^e composante du tableau JOUR de "vendredi" en "Friday".

Accès aux différentes composantes

- ***Accès aux caractères***

  Evidemment, il existe toujours la possibilité d'accéder directement aux différents ***caractères*** qui composent les mots du tableau.

Exemple

L'instruction

for(I=0; I<7; I++) printf("%c ", JOUR[I][0]);

va afficher les premières lettres des jours de la semaine:

l m m j v s d