Chapitre 5

Les tableaux et les chaînes de caractères

Un tableau permet de mémoriser plusieurs données du même type. Contrairement aux variables simples, les tableaux permettent de stocker des données nombreuses en mémoire centrale. Les données du tableau ne doivent pas toutes être déclarées à la main dans le code ; la déclaration du tableau en une ligne suffit pour déclarer toutes les données. En revanche, toutes les données du tableau doivent être de même type. Les éléments d'un tableau sont rangés selon un ou plusieurs axes appelés dimensions du tableau.

I. Les tableaux à une dimension (les vecteurs)

Dans les tableaux à une dimension, chaque élément est repéré par un seul indice (valeur entière). Un tableau à une dimension est parfois appelé vecteur. Il peut être représenté sous la forme suivante :

	0	**1**	**2**	**…**	**..**	**…**	**n-1**
T	e(0)	e(1)	e(2)	…	…	…	e(n-1)

- Ce tableau est représenté par la variable T tels que e(0), e(1), e(2) …, e(n-1) représentent les éléments du tableau.
- Les positions {0, 1, 2, ..., (n-1)} représentent les indices ou indexes du tableau. Elles donnent la position d'un élément du tableau. 0 représente l'indice du premier élément et (n-1) l'indice du dernier élément.
- Taille (Dimension) du tableau : représente le nombre de ses éléments (cases). Ici la taille est n.
- Les e(i), 0,1, 2, …, (n-1) doivent être de même type.

1. Déclaration d'un tableau à une dimension

On déclare un tableau en utilisant la syntaxe suivante :

<u>En algorithmique :</u> On utilise deux façons :

Première façon :

Type id_type = **tableau** [premier_indice..dernier_indice] **de** type_des_éléments ;
Var id_tab : id_type ;

Deuxième façon :
Var id_tab : **tableau** [premier_indice..dernier_indice] **de** type_des_éléments ;

<u>En langage C</u>

Type_elements id_tab [nombre_elements] ;

Exemple :

<u>En algorithmique</u>

Type Tab = **tableau** [0..99] **de** réel ; //Tab est un nouveau type qui est un tableau de 100 réels
Var T : Tab ; // T est une variable de type Tab.

Ou simplement :
Var T **= tableau** [0..99] **de** réel ;

<u>En langage C</u>

int T[100];

2. Accès aux éléments du tableau

L'accès aux éléments d'un tableau se fait en indiquant le nom du tableau suivi de l'indice de l'élément dans le tableau comme suit :

Nom_du_tableau [indice_de_element] ;

Exemple :

Soit le tableau V contenant les 5 éléments suivants :

	0	**1**	**2**	**3**	**4**
V	18	15	12	16	14

On a : V [0] = 18 ; V [1] = 15 ; V [2] = 12 ; V [3] = 16 ; V [4] = 14 ;

3. Représentation en mémoire

Lors de la déclaration d'un tableau, une zone mémoire lui sera réservée. Elle sera utilisée pour le stockage de ses données. La taille de cette zone en octets est la multiplication de la taille du tableau par la taille du type de ses éléments (un tableau de trois entiers sera représenté par six octets : chaque entier est codé sur deux octets).

Le tableau V précédent correspond à l'adresse mémoire de son premier élément (V=&V[0]). Il s'agit de la première cellule de la zone mémoire qui lui est réservé.

Supposant que V correspond à l'adresse mémoire 1000, le contenu des cellules mémoires sera alors comme suit :

adresse 0000			
adresse 0001			
adresse 0002			
….			
….			
….			
adresse 1000	→	18	V=&V[0]
adresse 1001	→	15	&V[1]
adresse 1002	→	12	&V[2]
adresse 1003	→	16	&V[3]
adresse 1004	→	14	&V[4]
adresse 1005			
adresse 1000			

V= & V[0] =1000
V[0] = '18' &V[0] = 1000
V[1] = '15' &V[1] = 1001
V[2] = '12' &V[2] = 1002
V[3] = '16' &V[3] = 1003
V[4] = '14' &V[4] = 1004

4. L'affectation

Pour affecter une valeur à un élément i d'un tableau nommé par exemple A, on écrira : A[i] ← valeur.

Par exemple, l'instruction : A [0] ← 20 ; affecte au premier élément du tableau A la valeur 20.

Pour affecter la même valeur à tous les éléments d'un tableau A de type entier et de dimension 100, on utilise une boucle.

Exemple :

En algorithmique

```
Pour i allant de 0 jusqua 99 faire
    A[i] ←0 ;
FinPour ;
```

En langage C

```
For (i =0 ; i < = 99 ; i++)
    A[i] : =0 ;
```

5. La lecture

Comme les variables simples, il est possible aussi d'assigner des valeurs aux éléments d'un tableau lors de l'exécution c.-à-d. les valeurs sont saisies par l'utilisateur à la demande du programme.

Remarque :

Un tableau peut-être totalement ou partiellement lu. Ce n'est pas obligatoire d'utiliser toutes les composantes du tableau, pour cela on déclare une variable entière N qui représente la taille du tableau à utiliser. La valeur de cette variable sera introduite au cours de l'exécution (par lecture).

Exemple : Lecture d'un tableau de N éléments.

En algorithmique

```
algorithme lire_tableau ;
Type Tab = tableau [1..100] de entier ;
Var T : Tab ;
    N, i : entier ;
Debut
   Repeter
      Ecrire ('introduire le nombre des éléments du tableau') ;
      Lire (N) ;
   Jusqua ((N>=1) et (N<=100)) ;
   Ecrire ('introduire les éléments du tableau T') ;
   Pour i allant de 1 jusqua N faire
      Lire (T[i]) ;
   FinPour ;
Fin.
```

En langage C

```
For (i =0 ; i < = (N-1) ; i++)
    scanf ("%d", &T[i]) ;
```

6. L'écriture (Affichage)

De façon analogue à la lecture, l'écriture de la valeur d'un élément donné d'un tableau s'écrira comme suit : Ecrire (T[i]). Cette instruction permet d'afficher la valeur de l'élément i du tableau T.

Exemple : Affichage d'un tableau T de N éléments.

En algorithmique

```
Pour i allant de 1 jusqua N faire
   Ecrire (T[i]) ;
FinPour ;
```

En lngage C

```
For (i = 0 ; i < = (N-1) ; i++)
        printf ("%d", &T[i]) ;
```

Exercice :

Soit T un tableau de vingt éléments de types entiers. Ecrire le programme qui permet de calculer la somme des éléments de ce tableau.

Solution :

```
algorithme somme,
Type Tab = tableau [1 ..100] de entier ;
Var T : tab ;
     somme : entier ;
Debut
Pour i allant de 1 jusqua 20 faire
    Lire (T[i]) ;
FinPour ;
Pour i allant de 1 jusqua 20 faire
    Somme ← somme +T[i];
FinPour ;
Ecrire ('la somme des éléments du tableau est', somme) ;
Fin.
```

7. Manipulation de tableaux à une seule dimension

a) Somme de deux vecteurs T1 et T2

La somme de deux vecteurs T1 et T2 de dimension N donne un vecteur T de même dimension.

```
algorithme somme ;
Var T1, T2, T : tableau [1..100] de réel ;
      N, i : entier ;
Debut
  Repeter
    Ecrire ('introduire le nombre des éléments du tableau') ;
    Lire (N) ;
  Jusqua ((N>=1) et (N<=100)) ;
  Pour i allant de 1 jusqua N faire
       Lire (T1[i]) ;
  FinPour ;
  Pour i allant de 1 jusqua N faire
    Lire (T2[i]) ;
  FinPour ;
  Pour i allant de 1 jusqua N faire
    T[i] ← T1[i] + T2[i] ;
  FinPour
  Pour i allant de 1 jusqua N faire
    Ecrire (T[i]) ;
  FinPour
Fin.
```

b) Produit scalaire de deux vecteurs T1 et T2

Le produit de deux vecteurs T1 et T2 de même dimension donne un scalaire (nombre).

```
algorithme produit_scalaire ;
Var T1, T2 : tableau [1..100] de réel ;
      N, i : entier ;
      ps : réel ;
Debut
    Repeter
      Ecrire ('introduire le nombre des éléments du tableau') ;
      Lire (N) ;
    Jusqua ((N>=1) et (N<=100)) ;
    Pour i allant de 1 jusqua N faire
       Lire (T1[i]) ;
    FinPour ;
    Pour i allant de 1 jusqua N faire
       Lire (T2[i]) ;
    FinPour ;
    ps ← 0 ;
    Pour i allant de 1 jusqua N faire
         ps ←ps + T1[i] * T2[i] ;
    FinPour
      Ecrire ('Le produit scalaire est :', ps) ;
Fin .
```

c) Recherche d'un élément dans un vecteur

La recherche d'une valeur dans un tableau consiste à récupérer sa position si cette valeur existe dans le tableau. Dans le cas où cette valeur existe plusieurs fois, la position de la première valeur est récupérée.

```
algorithme recherche_valeur ;
Var T : Tableau [1..100] de entier  ;
        N, Val,i, pos : entier ;
       Trouve : booleen ;
Debut
    Repeter
     Lire (N) ;
    Jusqua ((N>=1) et (N<=100)) ;
 Pour i allant de 1 jusqua N faire
    Lire (T[i]) ;
 FinPour
 Lire (Val) ;
 i ← 1 ;
Trouve ← Faux ;
 Tantque ((i <= N) et (Trouve=Faux)) faire
      Si (T[i] = Val) alors
           Trouve ← Vrai ;
           pos← i ;
      FinSi ;
     i ← i + 1 ;
  FinTantque ;
  Si Trouve = Vrai alors
    Ecrire ('La valeur ', Val, 'existe') ;
    Ecrire ('Sa position est', pos) ;
  Sinon
    Ecrire ('La valeur', Val, 'n"existe pas') ;
  FinSi ;
Fin.
```

d) Recherche de l'élément maximum dans un vecteur

```
Algorithme element_maximum ;
Var T : Tableau [1..100] de reel ;
      N, i : entier ;
      Max : reel ;
 Debut
       Repeter
          Lire (N) ;
       Jusqua ((N >=1) et (N<=100)) ;
       Pour i allant de 1 jusqua N faire
            Lire (T[i]) ;
       FinPour ;
       Max ← T[1] ;
       Pour i allant de 2 jusqua N faire
             Si (T[i] > Max) alors
                  Max ← T[i] ;
             FinSi ;
       FinPour ;
       Ecrire ('Le maximum est :', Max) ;
 Fin.
```

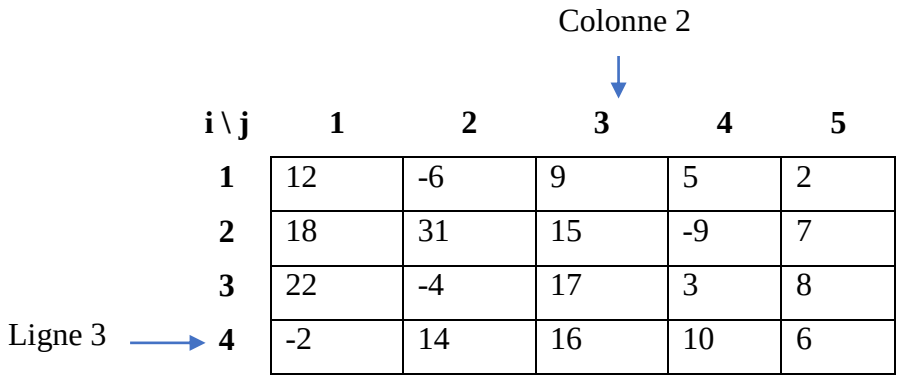
e) Algorithme de tri

Trier un tableau c'est ranger les éléments d'un tableau en ordre croissant ou décroissant. Dans ce qui suit un algorithme de tri permettent de trier un tableau T dans un ordre croissant.

```
Algorithme trie_croissant;
Var T : Tableau [1..100] de entier ;
      N, i, j, x : entier ;
Debut
    Repeter
      Lire (N) ;
    Jusqua ((N>= 1) et (N<=100)) ;
    Pour i allant de 1 à N faire
      Lire (T[i]) ;
    FinPour
   Pour i allant de 1 à (N-1) faire
    Pour j allant de (i+1) à N faire
        Si (T[i] > T[j]) alors
            x ← T[i];
            T[i] ← T[j] ;
            T[j] ← x ;
        Finsi ;
    FinPour;
  FinPour;
  Pour i allant de 1 à N faire
     Ecrire (T[i]) ;
  FinPour
Fin.
```

II. Tableaux à deux dimensions (les matrices)

L'algorithmique nous offre la possibilité de déclarer et d'utiliser des tableaux dans lesquels les valeurs ne sont pas repérées par un seul indice comme les tableaux à une seule dimension, mais par deux indices, le premier indice sert à représenter les lignes et le second indice pour les colonnes. Ce type de tableau est appelé Matrice. Le schéma ci-dessous représente un tableau T à deux dimensions.



T

1. Déclaration d'un tableau à deux dimensions

Pour déclarer un tableau à deux dimensions, on utilise la syntaxe suivante :

a. En algorithmique : On utilise deux façons :

Première façon :

Type id_type **= tableau** [premier_indice_ligne..dernier_indice_ligne, premier_indice_colonne .. dernier_indice_colonne] **de** type_des_éléments ;

Var id_mat **:** id_type ;

Deuxième façon :

Var id_mat: **tableau** [premier_indice_ligne..dernier_indice_ligne, premier_indice_colonne .. dernier_indice_colonne] **de** type_des_éléments ;

b. En langage C :

Type_elements id_mat [nombre_lignes][nombre_colonnes] ;

Exemple :

En algorithmique

La déclaration de la matrice T précédente est comme suit :
Type Mat**= tableau** [1..4, 1 .. 5] **de** réel;
Var T **:** Mat ;
Ou
Var T **: tableau** [1..4, 1 .. 5] **de** réel;
En langage C :

float T [4][5] ;

2. Accès aux éléments d'un tableau à deux dimensions

On accède à la $i^{ème}, j^{ème}$ valeur d'un tableau à deux dimensions en utilisant la syntaxe suivante :
Nom_du_matrice [i][j] ;
Exemple : dans la matrice T précédente :

- T [2][3] ← 3 signifie mettre la valeur 3 dans la case de numéro de ligne 2 et de numéro de colonne 3.
- En considérant le cas où x est une variable de type réel, x ← T [2][3] signifie mette 3 dans x.

3. Lecture et affichage d'une matrice

L'algorithme suivant permet de lire et d'écrire une matrice de n lignes et m colonnes :

```
algorithme lire_afficher_matrice;
Type Mat = tableau [1..50, 1..50] de entier ;
Var A : Mat ;
    N, M, i : entier ;
Debut
   Repeter
   Ecrire ('introduire le nombre de ligne de la matrice') ;
   Lire (N) ;
   Jusqua ((N>=1) et (N<=50)) ;
```

```
    Repeter
    Ecrire ('introduire le nombre de colonnes de la matrice') ;
    Lire (M) ;
    Jusqua ((M>=1) et (M<=50)) ;
    Pour i allant de 1 jusqua N faire
        Pour j allant de 1 jusqua M faire
            Lire (A[i,j]) ;
        FinPour ;
    FinPour ;
    Pour i allant de 1 jusqua N faire
        Pour j allant de 1 jusqua M faire
            Ecrire(A[i,j]) ;
        FinPour ;
    FinPour ;
Fin.
```

Exercice :

Ecrire un algorithme permettant de calculer le nombre des valeurs positives, le nombre des valeurs négatives et le nombre des valeurs nuls dans une matrice lue au clavier.

Solution :

```
algorithme NombrePosNegNul ;
Var Mat : tableau [1..50, 1..50] de reel ;
    N, M, i ,j : entier ;
    NbrP, NbrN, NbrNul : entier ;
Debut
  Repeter
    Ecrire ('introduire le nombre de lignes de la matrice') ;
    Lire (N) ;
  Jusqua ((N>=1) et (N<=50)) ;
  Repeter
    Ecrire ('introduire le nombre de colonnes de la matrice') ;
    Lire (M) ;
  Jusqua ((M>=1) et (M<=50)) ;
  Pour i allant de 1 jusqua N faire
     Pour j allant de 1 jusqua M faire
          Lire (Mat [i, j]) ;
     FinPour ;
  FinPour ;
  NbrP ← 0, NbrN ← 0, NbrNul ← 0 ;
  Pour i allant de 1 jusqua N faire
      Pour j allant de 1 jusqua M faire
          Si (Mat [i, j] > 0) alors
              NbrP ← NbrP +1 ;
          Sinon
              Si (Mat [i, j] < 0) alors
                  NbrN ← NbrN + 1 ;
              Sinon
```

```
                    NbrNul ← NbrNul + 1 ;
                Finsi ;
            Finsi ;
        FinPour ;
    FinPour ;
  Ecrire (NbrP, NbrN, NbrNul) ;
Fin.
```

III. Chaînes de caractères

1. Définition

Les chaînes de caractères sont des séquences finies de caractères. Le nombre de caractères composant une chaîne de caractères est la *longueur* de cette chaîne.

Une chaîne de longueur nulle ne comprend aucun caractère : c'est la chaîne vide.

En algorithmique, une chaîne de caractères est désignée entre deux apostrophes.

Exemple

• La chaîne de caractères 'Baobab' est constituée des caractères 'B', 'a', 'o', 'b', 'a' et 'b'.

• La chaîne de caractères '123' n'est constituée que de chiffres (à ne pas confondre avec la grandeur numérique 123).

• La chaîne de caractères '' représente une chaîne de caractères vide.

• La chaîne de caractères ' ' est une chaîne de caractères ne contenant qu'un seul caractère qui est ici le caractère espace (à ne pas confondre avec la chaîne de caractères vide).

2. Déclaration

On déclare une chaîne de caractères en utilisant la syntaxe suivante :

a. En algorithmique : On utilise deux façons en indiquant ou non le nombre de caractères :
Première façon : **var** nom_variable **:** chaine [longueur] ;
Deuxième façon : **var** nom_variable **:** chaine ;

b. En langage C :

Il n'existe pas de type spécial pour les chaînes de caractères en langage C. Une chaîne de caractères est déclarée comme un tableau à une dimension de char (vecteur de caractères) dont la fin est indiquée par le caractère '\0'. La taille de la chaîne est égale à la longueur maximale de la chaine plus un pour qu'on puisse stocker le caractère '\0' dénotant la fin.

char nom_variable [longueur] ;

Exemple :

En algorithmique

var ch : chaîne [10] ; ou **var** ch : chaîne ;

En langage c

```
char ch [10]; // la longueur maximale de la chaîne est 9
```

3. Initialisation d'une chaîne de caractères

Pour initialiser une chaîne de caractères, nous déclarons la variable chaîne et affectons sa valeur initiale.

***Exemple* :**

<u>En algorithmique :</u>

Var TXT : chaine [15] ;

TXT ← 'Bonjour' ; // initialisation de la variable *TXT* à la chaîne *'Bonjour'*.

<u>En langage C :</u>

char TXT[] = { 'b','o','n','j','o','u','r', '\0'}; /* juste, mais à éviter !! */

char TXT[] = 'bonjour' ; // l'ordinateur réserve automatiquement le nombre d'octets nécessaires pour la chaîne, c.-à-d.: le nombre de caractères + 1 (ici: 8 octets).

char TXT[8] = 'bonjour' ; // indication explicite du nombre d'octets à réserver, le nombre d'octets doit être supérieur ou égale à la longueur de la chaîne.

4. Accès aux éléments d'une chaîne de caractères

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.

Exemple : Pour la chaîne TXT précédente, nous avons :

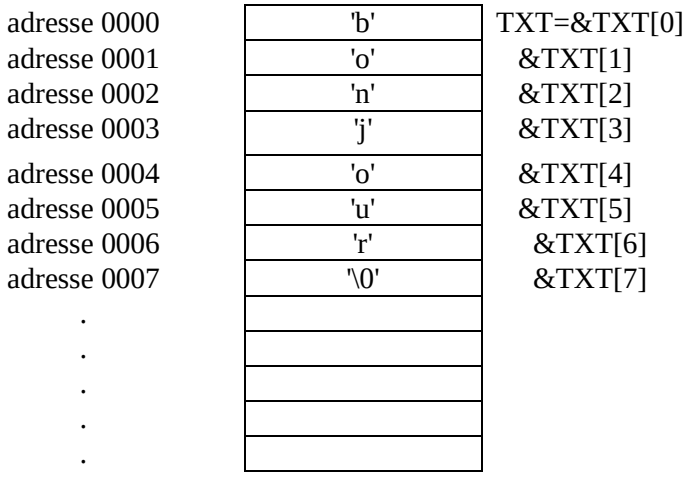
TXT[0] = 'b' ; TXT[1] = 'o' ; TXT[2] = 'n' ; TXT[3] = 'j' ; TXT[4] = 'o' ; TXT[5] = 'u' ;

TXT[6] = 'r' ;

5. Représentation en mémoire

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 de la chaîne TXT.



6. Lecture et affichage

En algorithmique, une chaîne de caractères est lue (affichée) globalement (d'un seul coup) et non pas caractère par caractère.

Exemple :

a. En algorithmique :

Lire (TXT) ; Ecrire (TXT) ;

b. En langage C : on peut lire (afficher) une chaîne de caractères de deux façons :

Première façon : **scanf** ("%s", TXT) ; **printf** ("%s", TXT) ;

Deuxième façon : **gets** (TXT); **puts** (TXT);

La différence entre scanf et gets est que scanf amène uniquement le texte introduit avant le premier blanc (espace) dans la variable à lire. gets amène tout le texte introduit jusqu'au retour chariot (retour à la ligne) dans la variable à lire.

7. Comparaison de chaînes de caractères

Dans un système informatique, à chaque caractère est associé une valeur numérique : son code ASCII (American Standard Code for Information Interchange). L'ensemble des codes est recensé dans une table nommée "table des codes ASCII". Quand on stocke un caractère en mémoire (dans une variable), on mémorise en réalité son code ASCII.

Pour comparer deux chaînes de caractères, on compare les caractères de même rang dans les deux chaînes en commençant par le premier caractère de chaque chaîne (le premier caractère de la première chaîne est comparé au premier caractère de la seconde chaîne, le deuxième caractère de la première chaîne est comparé au deuxième caractère de la seconde chaîne, et ainsi de suite…).

Exemples : Comparaison de deux chaînes

• 'baobab' < 'sport' car le code ASCII de b (98 en base 10) est inférieur au code ASCII de 's' (115 en base 10).

• "baobab" > "banania" car le code ASCII de 'o' (111) est supérieur au code ASCII de 'n'(110). La comparaison ne peut pas se faire sur les deux premiers caractères car ils sont identiques.

• '333’ > '1230’ car le code ASCII de '3' (51) est supérieur au code ASCII de '1' (49). Attention, ici ce ne sont pas des valeurs numériques qui sont comparées, mais bien des caractères.

• '333' < '3330' car la seconde chaîne a une longueur supérieure à celle de la première (la comparaison ne peut pas se faire sur les trois premiers caractères car ils sont identiques).

• 'Baobab' < "baobab" car le code ASCII de 'b' (98) est supérieur au code ASCII de 'B' (66).

8. Fonctions et procédures sur les chaînes de caractères

Les fonctions et les procédures standards qui manipulent les chaînes de caractères sont nombreuses. Ici nous allons donner quelques d'eux.

a. En algorithmique

Les fonctions standards relatives aux chaînes de caractères				
Syntaxe	***Rôle***	***Type***		***Exemples***
		Paramètres de la fonction	***Résultat***	

long (ch)	Retourne un entier représentant la longueur de ch (nombre de caractères de ch)	Chaîne	Entier	L ← long ('Bonne chance') ; // L=12
				L ← long ('') ; // L=0 ;
				L ← long (' ') ; // L= 1 ;

pos (chl, ch2)	Retourne la première position de la chaîne ch1 dans la chaîne ch2.	Chaînes	Entier	P ← pos ('a', 'Programmation') ;// P=6
				P ← pos ('gra', 'Programmation') ; // P= 4
				P ← pos ('s', 'Programmation') ;// P= 0

copier (ch, p, n)	Retourne une sous chaîne de n caractères à partir de la position p de la chaîne ch.	Chaîne, Entier, Entier	Chaîne	CH ← Copier ('Bonjour', 4, 3) ; // CH = 'jour'
concat (chl, ch2, ...)	Retourne la concaténation (l'enchainement) de plusieurs chaînes en une seule. C'est l'équivalent de ch1+ch2+…	Chaînes	Chaîne	CH1 ← 'Juillet' ; CH2 ← Concat ('5', '/', CH1, '/', '1962') ; CH2 ←5/juillet/1962

Les procédures standards relatives aux chaînes de caractères				
Syntaxe	***Rôle***	***Type***		***Exemples***
		Paramètres de la procédure	***Résultat***	
efface (ch, p, n)	Supprime n caractères de la chaîne ch à partir de la position p.	Chaîne, Entier, Entier	Chaîne	CH← 'Bon chance' ; efface (CH, 1, 4) ; // CH = chance
insere (ch1, ch2, p)	Insère la chaîne ch1 dans la chaîne ch2 à la position p.	Chaîne/Caractère, Chaîne, Entier	Chaîne	CH1 ← 'rit' ; CH2 ← 'algomique' ; insere (CH1, CH2, 5) ; // CH2 devient 'algorithmique'
convch (n, ch)	Convertit une valeur numérique n (entier, réel) en une chaîne ch.	Numérique, chaîne	Chaîne	convch (2019, CH) ;// CH = '2019'
valeur (**ch, n, err)**	Convertit une chaîne de caractères ch en une valeur numérique n. De plus, elle fournit un code d'erreur err qui indique si l'opération s'est déroulée correctement.	Chaîne/Caractère, Numérique, Entier	Numérique, Entier	valeur ('2019', n, err) ; n = 2019 et err=0 valeur ('25H64', n, err) ; n = 0 et err = 3

b. En langage C

	Les fonctions standards relatives aux chaînes de caractères			
Syntaxe	***Bibliothèque***	***Rôle***	***Type***	
			Paramètres de la fonction	***Résultat***
Strlen (ch)	String.h	Permet de connaître la longueur de la chaîne ch.	ch : Chaîne	Entier
Strcpy (ch1, ch2)	string.h	Copie le contenu de la chaîne ch2 dans la chaîne ch1.	ch1 : Chaîne ch2 : Chaine	Chaîne
Strcat (ch1, ch2)	string.h	Permet de concaténer les contenus de deux chaînes de caractères (ajouter le contenu de la chaîne ch2 à celui de la chaîne ch1).	ch1 : Chaîne ch2 : Chaine	Chaîne
Strcmp (ch1, ch2)	string.h	Permet de comparer deux chaînes de caractères.	ch1 : Chaîne ch2 : Chaine	Valeur positive si ch1>ch2 Valeur nulle si ch1=ch2 Valeur négative si ch1<ch2
Strchr (ch1, ch)	string.h	Recherche la présence du caractère ch dans la chaîne ch1.	ch1 : Chaîne ch : Caractère	Adresse de la première occurrence du caractère s'il existe sinon NUL
Strpbrk (ch1, ch2)	string.h	Recherche la présence d'un des caractères de la chaîne ch2 dans la chaîne ch1.	ch1 : Chaîne ch2 : Chaine	Adresse du premier caractère rencontré sinon NUL
Strstr (ch1, ch2)	string.h	Recherche la présence de la chaîne ch2 dans la chaîne ch1.	ch1 : Chaîne ch2 : Chaine	Adresse de la première occurrence de la chaîne si elle existe sinon NUL
Atoi (ch)	stdlib.h	Convertit ch en entier	ch : Chaine	Entier
Atol (ch)	stdlib.h	Convertit ch en entier long	ch : Chaine	Entier long
Atof (ch)	stdlib.h	Convertit ch en double (réel)	ch : Chaine	Double
strdel (ch,p,n)	publib.h	Supprime n caractères de la chaîne ch à partir de la position p et stocke le résultat dans la chaîne ch.	ch :chaine p,n :entier	Chaine