Les tableaux et les chaînes de caractères

Un tableau est un ensemble de *cases mémoires contiguës identiques*, chacune pouvant contenir une donnée d'un certain type (*le même pour toutes les cases* d'un même tableau). On peut accéder directement à chaque case (ou « élément ») du tableau en utilisant un système de numérotation.

Déclaration Déclaration d'un tableau à 1 dimension :

***type_éléments nom_tableau*[*nb_lignes*] ;**

où ***type_éléments*** est le type de donnée que peut contenir chaque élément du tableau ; tous les éléments d'un même tableau ont le même ***type_élément*** ,
nom_tableau est le nom du tableau dans sa globalité ,
nb_lignes est le nombre d'éléments (« lignes ») du tableau.

Ex **int tab1[23];** déclare un tableau de 23 **int** nommé **tab1**
float toby[3]; déclare un tableau de 3 **float** nommé **toby**
char un_mot[10]; déclare un tableau de 10 **char** nommé **un_mot**

Déclaration d'un tableau à 2 dimensions :

***type_éléments nom_tableau*[*nb_lignes*][*nb_colonnes*] ;**

où ***type_éléments*** et ***nom_tableau*** ont des définitions identiques à celles ci-dessus,
nb_lignes est le nombre de « lignes » du tableau
nb_colonnes est le nombre de « colonnes » du tableau
(le nombre total d'éléments est donc le produit de ***nb_lignes*** par ***nb_colonnes***)

Ex **int tab4[5][83];** déclare un tableau de 415 **int** répartis en 5 lignes et 83 colonnes.

On peut de même déclarer des tableaux de plus de 2 dimensions .

Remarques En mémoire, le stockage des éléments d'un tableau multidimensionnel se fait uniquement « en ligne » , les indices les plus à droite dans la déclaration du tableau variant le plus vite (1ère ligne entière, puis 2ème ligne entière, ...). La représentation pluridimensionnelle du tableau n'est qu'une visualisation commode pour les êtres humains !

Il est impossible de déclarer des tableaux de « taille variable » (i.e. dont la taille serait contenue dans une variable, voire choisie par l'utilisateur !)

Accès aux éléments d'un tableau

Chaque élément d'un tableau est numéroté de la manière suivante :

si on a déclaré le tableau ***type_éléments nom_tableau*[*n1*][*n2*]...[*nk*];**

alors un élément de ce tableau s'écrira ***nom_tableau*[*i1*][*i2*]...[*ik*]**
où les indices i1∈[0;n1[**,** i2∈[0;n2[**, ... ,** ik∈[0;nk[

et se manipulera comme une variable de type ***type_éléments*** habituelle.

Attention La numérotation des indices ***i1, ... ,ik*** commence à **0** et se termine à ***(n1 – 1) , ... , (nk – 1)***

Ex **int toby[2][3];**

Les éléments de **toby** sont :

toby[0][0]	**toby[0][1]**	**toby[0][2]**
toby[1][0]	**toby[1][1]**	**toby[1][2]**

Adresse du 1er élément

Le nom global du tableau ***nom_tableau*** utilisé non suivi d'un crochet ouvrant **[** , représente l'adresse du 1er élément du tableau :

nom_tableau est identique à ***&nom_tableau*[0]**

Affectation

Chaque élément du tableau peut être initialisé indépendamment :

toby[1][2]=845;

Chaînes de caractères

Une « chaîne de caractères » est une suite de caractères contigus (par exemple, la présente phrase est une chaîne de caractères).

Une chaîne de caractères (*« string »*) constante s'écrit en C entre guillemets : "

Ex **"Ceci est une chaîne de caractères constante."**

Elle peut contenir des caractères spéciaux précédés par des \ (voir **printf**) :

"Ceci est une chaîne terminée par un retour à la ligne.\n"

(Rappel : **\n** représente un seul caractère de code ASCII décimal 10)

Une chaîne de caractères « variable » doit être stockée dans un tableau unidimensionnel de **char**

Elle doit alors toujours *être terminée* par le caractère « nul » : **<NULL>** noté **\0**
Le tableau de **char** doit donc être déclaré comme ayant *un élément de plus* que le nombre total de caractères de la chaîne que l'on désire y stocker.

Ex Si l'on désire stocker dans une variable tableau la chaîne :

"Chaine à stocker\n" (chaîne de 17 caractères, espaces compris)

il faut déclarer le tableau suivant :

char t[18];

et s'assurer que **t[17]** (donc le 18ème élément) contiendra bien **\0** , quitte à l'affecter soi-même en cas de doute :

t[17]='\0';

Après affectation, **t** contiendra donc :

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]	t[10]	t[11]	t[12]	t[13]	t[14]	t[15]	t[16]	t[17]
C	**h**	**a**	**i**	**n**	**e**		**à**		**s**	**t**	**o**	**c**	**k**	**e**	**r**	**\n**	**\0**

La plupart des fonctions du C gérant les chaînes de caractères, gèrent aussi le caractère nul.

Remarque Les chaînes constantes entre guillemets, sont elles-aussi stockées en interne par le programme dans des tableaux de **char**, mais l'utilisateur n'a pas à s'en préoccuper, ni à les déclarer.

Manipulation des chaînes de caractères

Il existe en C tout un ensemble de fonctions qui manipulent les chaînes de caractères.

► Sorties

- On peut afficher le contenu d'un tableau de **char**, jusqu'au **\0** non inclus, à l'aide de **printf** et du format **%s** :

  printf("%s",*nom_tableau*);

- Il existe également la fonction **puts**

  puts(*nom_tableau*);

  Attention : **puts** effectue un saut de la ligne après avoir affiché la chaîne

► Entrées

- On peut lire une chaîne entrée au clavier (elle ne doit pas alors être entourée de **"**) et l'affecter globalement à un tableau de **char** préalablement déclaré, à l'aide de **scanf** et du format **%s** :

  scanf("%s", *nom_tableau*);

  Attention : ***nom_tableau*** ne doit pas être précédé du **&** d'adressage, ce qui est normal puisque ***nom_tableau*** contient déjà l'adresse de ***nom_tableau*****[0]**
  Par ailleurs, **scanf** ajoute spontanément le **\0** final dans le tableau.

- On peut aussi utiliser **gets**

  gets(*nom_tableau*);

  gets remplace le **\n** de fin de saisie au clavier par un **\0**

► Affectation

On peut affecter la chaîne au tableau, caractère par caractère, mais c'est peu commode !

Il existe donc des fonctions permettant d'affecter globalement des chaînes entières (d'où l'utilité du **\0** pour que le programme sache où la chaîne se termine). Ces fonctions sont en général disponibles dans la bibliothèque **string.h** (L'inclure au début du programme **#include <string.h>**)

- **strcpy** permet de copier une chaîne constante ou le contenu d'une chaîne variable (i.e. d'un tableau de **char**) dans un tableau de **char** à partir de son 1er élément (le caractère nul est également copié). Les caractères déjà présents sont écrasés.

  strcpy(*nom_tableau_réceptacle*,"*chaîne_à_copier*");
  ou
  strcpy(*nom_tableau_réceptacle*, *nom_tableau_à_copier*);

  Il est clair que ***nom_tableau_réceptacle*** doit avoir au moins autant d'éléments que la chaîne copiée (**\0** inclus).

  Ex **strcpy(t,"Coucou");**

  où **t** est le tableau défini et affecté plus haut.
  t contiendra alors :

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]	t[10]	t[11]	t[12]	t[13]	t[14]	t[15]	t[16]	t[17]
C	**o**	**u**	**c**	**o**	**u**	**\0**	**à**		**s**	**t**	**o**	**c**	**k**	**e**	**r**	**\n**	**\0**

  Les caractères présents initialement dans **t[0]** à **t[6]** ont donc été remplacés par les nouveaux caractères suivis du **\0**. Ceux qui suivent sont alors inaccessibles (du moins à l'aide de moyens aisés)

- **strcat** permet de concaténer (d' »accrocher ») une chaîne de caractères à la suite d'une autre ; le **\0** de la 1ère est supprimé et reporté à la fin de la nouvelle chaîne obtenue.

  strcat(*nom_tableau1*, *nom_tableau_à_accrocher*);

  Ex Si **chili** contient la chaîne **" c'est nous"** (1 espace en début de chaîne)
  strcat(t,chili); conduira au contenu de **t** suivant :

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]	t[10]	t[11]	t[12]	t[13]	t[14]	t[15]	t[16]	t[17]
C	**o**	**u**	**c**	**o**	**u**		**c**	**'**	**e**	**s**	**t**		**n**	**o**	**u**	**s**	**\0**

► Divers

- **strlen** renvoie la longueur de d'une chaîne, jusqu'au premier **\0** non compris.

  int longueur;
  longeur=strlen(*nom_tableau*);

  Ex Si **t** contient :

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]	t[10]	t[11]	t[12]	t[13]	t[14]	t[15]	t[16]	t[17]
C	**o**	**u**	**c**	**o**	**u**	**\0**	**à**		**s**	**t**	**o**	**c**	**k**	**e**	**r**	**\n**	**\0**

  strlen(t) fournira comme résultat **6**

- **stcmp** compare 2 chaînes caractère par caractère, jusqu'à rencontrer une différence ou jusqu'à atteindre **\0** . La comparaison s'effectue dans l'ordre lexicographique (i.e. celui de la table ASCII)

  int result;
  result=strcmp(*nom_tableau1*,*nom_tableau2*);

  strcmp renoie **0** si les 2 chaînes contenues dans ***nom_tableau1*** et ***nom_tableau2*** sont identiques ;

  elle renvoie un nombre positif si les 1ers caractères différents entre les 2 chaînes sont tels que celui de ***nom_tableau1*** a un code ASCII plus grand que celui de ***nom_tableau2*** ;

  elle renvoie un nombre négatif dans le cas contraire.

  Ex Si **t** contient

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]	t[10]	t[11]	t[12]	t[13]	t[14]	t[15]	t[16]	t[17]
C	**o**	**u**	**c**	**o**	**u**		**c**	**'**	**e**	**s**	**t**		**n**	**o**	**u**	**s**	**\0**

  strcmp(t,"Coucou!");
  renvoie un nombre négatif, car dans **t** il y a un espace à la place du **!** et le code ASCII de l'espace (32) est plus petit que celui du **!** (33)

Il existe encore d'autres fonctions (commençant quasi-toutes par **str**) que l'on peut découvrir dans la bibliothèque **string.h**

► Sous-chaîne

Une sous-chaîne est une partie d'une chaîne. Elle doit être terminée par un **\0**. Pour manipuler une sous-chaîne, il faut l'adresse du 1er caractère : pour cela on utilise l'opérateur **&**

***&nom_tableau*[*indice_début*]**

représente l'adresse du caractère d'indice ***indice_début*** dans la chaîne initiale contenue dans ***nom_tableau***.

Ex **strcpy(vovo, &t[9]);**
copie dans le tableau **vovo** la chaîne **"est nous"** qui est une sous-chaîne de la chaîne contenue dans le tableau **t** défini ci-dessus.