**2.CHAINES DE CARACTERES**

Une chaîne de caractères est un tableau de caractères. Elle représente un cas particulier des tableaux qui bénéficie de certains traitements particuliers en plus de ceux réservés aux tableaux en général. Une chaîne de caractères peut également être déclarée comme pointeur sur char (voir le chapitre pointeurs).

**2.1 INITIALISATION**

Syntaxe : char Identificateur [Taille constante] = "Texte\0" ;

 • Le caractère **'\0'** indique la fin de la chaîne de caractères. Il est conseillé de ne pas l'omettre.

**Exemples**:

 #define taille1 3 /\*taille1: constante de valeur 3\*/

#define taille2 4 /\*taille2: constante de valeur 4\*/

 main( )

 {

char t [taille1]="ali"; /\*t chaîne de caractères initialisée au mot ali\*/

char a [taille2]="ali\0"; /\*a chaîne de caractères initialisée au mot ali et terminée par le caractère '\0'\*/

 }

**Remarque** : Une affectation du genre t="texte\0" est impossible si t est déclaré comme un tableau de caractères

**2.2 LECTURE ET AFFICHAGE**

 Une variable de type chaîne de caractères peut être lue et affichée caractère par caractère au moyen de **scanf** et **printf** utilisant le format **%c**. Elle peut également être lue (affichée) globalement (d'un seul coup) au moyen de la fonction **scanf** (printf) utilisant cette fois-ci le format %s ou au moyen de la fonction gets (puts).

Syntaxe :

scanf("%s", Chaîne de caractères);

printf("%s", Chaîne de caractères);

gets(Chaîne de caractères);

puts(Chaîne de caractères);

 • 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.

Exemple:

 /\*Soit Turbo C la chaîne de caractères que l'on introduit lors de l'exécution de l'exemple suivant\*/

#include <stdio.h>

#define taille 20 /\*taille : constante de valeur 20\*/

 main( )

{

char t [taille]; /\*t : chaîne de caractères de taille 20\*/

scanf ("%s",t); /\*lit t (on ne met pas d'adresse &)\*/

 printf ("%s",t); /\*affiche Turbo\*/

 gets (t); /\*lit t (on ne met pas d'adresse &)\*/

 puts (t); /\*affiche Turbo C\*/

 }

**2.3 FONCTIONS SUR LES CHAINES DE CARACTERES**

 Des fonctions prédéfinies appliquées aux chaînes de caractères sont définies dans le fichier "string.h". Nous en citons :

|  |  |  |  |
| --- | --- | --- | --- |
| fonction | Appliquée à | retourne | rôle |
| strlen | Une chaîne de caractères  | Un entier  | Retourne la longueur d'une chaîne de caractères |
| strcmp | Deux chaînes de caractères | Un entier | Compare deux chaînes et retourne 0 si elles sont égales, une valeur différente sinon |
| strcpy | Deux chaînes de caractères | Rien (void) | Copie une chaîne en deuxième argument dans une autre en premier argument.  |

Exemples: Fonctions sur les chaînes de caractères

#include <stdio.h>

#define taille 20 /\*taille : constante de valeur 20\*/

main( )

 {

 char a [taille], b[taille]="ali\0"; /\*a et b chaînes de caractères; b initialisée à "ali\0"\*/

 int n; /\*n entier\*/

 n=strlen(b); /\*n reçoit 3 (longueur du mot "ali")\*/

 strcpy(a,b); /\*a reçoit la chaîne "ali\0"(copie de b)\*/

 n=strcmp(a,b); /\*n reçoit 0 (résultat de la comparaison de a et b)\*/

}

**Remarque**:

 Les fonctions du fichier "string.h" ne peuvent être appliquées à des caractères de type char. Elles sont appliquées à des chaînes de caractères (tableaux de caractères) qui se terminent par '\0' (On ne peut alors comparer, par exemple, deux caractères a='x' et b='y' au moyen de **strcmp**).

**Remarque :**

Le type char est en fait prévu pour stocker une lettre ! Attention, j'ai bien dit **UNE** lettre.

Comme la mémoire ne peut stocker que des nombres, on a inventé une table qui fait la conversion entre les nombres et les lettres. Ainsi, cette table indique par exemple que le nombre 65 équivaut à la lettre A.

Le langage C permet de faire très facilement la traduction lettre <=> nombre correspondant. Pour obtenir le nombre associé à une lettre, il suffit d'écrire cette lettre entre apostrophes, comme ceci : 'A'  . À la compilation, 'A' sera remplacé par la valeur correspondante

Test :

int main(int argc, char \*argv[])

{

 char lettre = 'A';

 printf("%d\n", lettre);

 return 0;

}

On sait donc que la lettre A majuscule est représentée par le nombre 65. B vaut 66, C vaut 67, etc.

testez avec des minuscules, et vous verrez que les valeurs sont différentes. 'a' n'est pas identique à 'A'  : l'ordinateur fait la différence entre les majuscules et les minuscules (on dit qu'il "respecte la casse").

La plupart des caractères « de base » sont codés entre les nombres 0 et 127. Une table fait la conversion entre les nombres et les lettres : la table ASCII (prononcez « Aski »). Le site [AsciiTable.com](http://www.asciitable.com) propose cette table mais ce n'est pas le seul, on peut aussi la retrouver sur Wikipédia et bien d'autres sites encore.

**1) Conversion en minuscule**

Ecrire un programme C qui demande à l'utilisateur un caractère, et teste s'il s'agit d'une lettre majuscule, si oui il renvoie cette lettre en minuscule, sinon il lui affiche un message l’informant que la lettre  saisie n’est pas majuscule.

/Programme : Conversion en Minuscule

#include <stdio.h>

int main()

{

char c, m;

printf("Entrer une lettre Majuscule: ");

scanf("%c", &c);

if(c >= 'A' && c <= 'Z'){ // ou if(c >= 65 && c <= 90)

m = c + 'a' - 'A'; // ou m = c + 32;

printf("\n--> Conversion en Minuscule: %c \n", m);

}

Else

{

printf("\nLe caratere '%c' n'est pas une lettre Majuscule\n", c);

}

}

## 2) Conversion en minuscule ou majuscule

Modifier le programme pour convertir une en majuscule s'il est minuscule et vice versa. Le programme doit afficher un message d'erreur si le caractère saisi n'est pas une lettre.

//Programme : Conversion en Majuscule/Minuscule

#include <stdio.h>

int main(){

char c, m;

printf("Entrer une lettre: ");

scanf("%c", &c);

if(c >= 'A' && c <= 'Z')

{

m = c + 'a' - 'A';

printf("\n--> Conversion en Minuscule: %c \n", m);

}

else if(c >= 'a' && c <= 'z')

{

m = c - ('a' - 'A');

printf("\n--> Conversion en Majuscule: %c \n", m);

}

else

{

printf("\nLe caratere '%c' n'est pas une lettre\n", c);

}

}

**Exercice :**

Ecrire un programme qui lit une chaîne de caractères CH et qui convertit toutes les majuscules dans des minuscules et vice-versa.

Le résultat sera mémorisé dans la même variable CH et affiché après la conversion.

#include <stdio.h>

main()

{

 /\* Déclarations \*/

 char CH[100]; /\* chaîne à convertir \*/

 int I; /\* indice courant \*/

 /\* Saisie de la chaîne \*/

 printf("Entrez la chaîne à convertir : ");

 gets(CH);

 /\* Conversion de la chaîne \*/

 for (I=0; CH[I]; I++)

 {

 if (CH[I]>='A' && CH[I]<='Z')

 CH[I] = CH[I]-'A'+'a';

 else if (CH[I]>='a' && CH[I]<='z')

 CH[I] = CH[I]-'a'+'A';

 }

 /\* Affichage de la chaîne convertie \*/

 printf("Chaîne convertie : %s\n", CH);

 return 0;

**3 TABLEAUX A PLUSIEURS DIMENSIONS**

 **3.1 DECLARATION**

Syntaxe : Type Identificateur [Taille1] [Taille2] … [Taille n] ;

• Taillei est la taille de la dimension i. Elle doit être une constante définie avant ou au moment de la déclaration.

• Un élément d'un tableau t à n dimensions est repéré par ses indices, sous forme t[i1][i2]…[in].

Exemples:

/\*Ce programme lit le nombre de buts marqués par chacun des 11 joueurs de 8 équipes\*/

 #include<stdio.h>

#define taille1 8 /\*taille1: constante de valeur 8\*/

#define taille2 11 /\*taille2: constante de valeur 11\*/

main( )

 {

 int t [taille1][taille2]; /\*t : matrice de 8 lignes, 11 colonnes\*/

int i, j;

for(i=0;i<taille1 ;i++)

for(j=0;j<taille2 ;j++)

scanf ("%d",&t[i][j]);

}

**Déclaration et lecture d'un tableau à trois dimensions :**

/\*Ce programme lit les notes en Informatique et Mathématiques de 5 étudiants. Chaque étudiant a quatre notes par matière :

|  |  |  |
| --- | --- | --- |
| Etudiants | Informatique | Mathématiques |
| Note1 | Note2 | Note3 | Note4 | Note1 | Note2 | Note3 | Note4 |
| 1 | 19 | 17 | 15 | 18 | 14 | 15.5 | 16 | 17 |
| 2 | 13 | 7 | 9.5 | 12 | 13 | 9 | 11.5 | 10 |
| 3 | 14 | 14.5 | 16 | 16.5 | 16 | 17 | 17.5 | 16.5 |
| 4 | 8.5 | 10.5 | 10 | 12 | 12 | 13 | 13.5 | 6.5 |
| 5 | 15 | 19 | 15 | 16.5 | 15 | 13 | 12.5 | 14.5 |

#include <stdio.h>

#define taille1 5 /\*taille1: constante de valeur 5 (5 étudiants)\*/

#define taille2 2 /\*taille2: constante de valeur 2 (2 matières)\*/

 #define taille3 4 /\*taille2: constante de valeur 4 (4 notes)\*/

 main( )

{

float t [taille1][taille2][taille3];

int i,j,k;

for(i=0;i<taille1<i++) /\*lit les quatre notes des étudiants étudiant par étudiant, commençant par les

 notes d'informatique puis celles des mathématiques\*/

for (j=0; j<taille2;j++)

for (k=0; k<taille2;k++)

scanf ("%f",&t[i][j][k]);

}

**3.2 INITIALISATION**

Syntaxe :

 Type Identificateur [m] … [p] = { Liste0, … , Listem-1} ;

 • Listei est l'initialisation de l'élément d'indice i et qui fut un sous tableau de dimension n-1 si n est la dimension du tableau en question.

**Exemple**:

Le programme ci-dessous initialise la matrice M (3 lignes, 2 colonnes) :

 M

|  |  |
| --- | --- |
| 0 | 1 |
| 2 | 3 |
| 4 | 5 |

Initialisation d'un tableau à plusieurs dimensions

#define taille1 3 /\*taille1: constante de valeur 3\*/

 #define taille2 2 /\*taille2: constante de valeur 2\*/

 main( )

 {

 int M [taille1][taille2]={{0,1},{2,3},{4,5}}; /\*initialisation ligne par ligne\*/

} /\*M[0]={0,1}, M[1]={2,3} et M[2]={4,5}\*/