REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de M'sila
Faculté des Mathématiques et de l'Informatique
Département d'informatique

جامعة المسيلة
كلية الرياضيات والإعلام الآلي
قسم الإعلام الآلي

Chapitre 5:
Les tableaux et les chaînes de caractères

Algorithmique et structure de données

Présenté par: Dr. Benazi Makhlouf
Année universitaire: 2022/2023

Contenu du chapitre 02:

1. Introduction
2. Le type tableau
3. Les tableaux multidimensionnels
4. Les chaînes de caractères

1. Introduction

- En programmation, les données sont organisées sous forme de constantes et de variables
- Il existe différents types de données, qui peuvent être divisés en deux parties :
    - Types simples. Tels que : entiers, réels, caractères et booléens.
    - Types composés : **les tableaux**, et **les structures** ou enregistrements.
- Supposons que nous voulions entrer les notes de 1000 étudiants il serait déraisonnable d'utiliser 1000 variables pour stocker des notes, puis d'écrire 1000 instructions d'entrées (lire) dans le programme.
- Il est préférable d'utiliser une seule variable, qui peut contenir toutes les valeurs des notes, et d'utiliser une boucle pour les saisir.
- La structure qui peut stocker plusieurs valeurs en même temps s'appelle un tableau.

2. Les tableaus

Tableau (en anglais, array) : structure de données complexe, constituée d'un ensemble fini d'éléments homogènes (de même type), accessible par des indexes indiquant leur emplacement.

Un tableau peut être vu comme un groupe de variables, de même type, portant le même nom.

indice: Lorsque les données sont stockées dans un tableau, l'élément est identifié par un indice qui, en C, est un entier non négatif (≥ 0). L’ indice commence de 0 à N - 1 (où N est la taille du tableau).

Tableau unidimensionnel (vecteur)

On utilise un seul indice pour accéder à ses éléments

Le tableau est représenté en mémoire par une séquence de cellules adjacentes. Une nouvelle cellule ne peut pas être supprimée ou ajoutée au tableau après sa création (statique).

indice	0	1	2	3	4	5	6	7
valeur	15	7	-3	0	9	2	0	-3

Déclaration

Algorithme

```
nomTableau [Taille] : Tableau de typeElements
```

En langage C

```
typeElements nomTableau [Taille] ;
```

Exemple

Algorithme

```
const N=100
note[N] : tableau de réel
tab1[50],tab2[20] : tableau d'entier
```

En langage C

```
const int N=100 ;
float note[N] ;
int tab1[50],tab2[20];
```

L’initialisation

En C, il est possible de spécifier des valeurs initiales pour tous les éléments du tableau dans des accolades { et } lors de la déclaration de du tableau. Les valeurs doivent être du même type et séparées par des virgules « , ».

Vous pouvez spécifier le nombre d'éléments entre les deux Crochets « [ ] » ou les laisser vides pour qu'ils soient calculés

Exemple

```
int tab[] = {15, 7, -2, 0, 9, 2, 0, -3};
```

indice	0	1	2	3	4	5	6	7
valeur	15	7	2-	0	9	2	0	3-

Utilisation

Le tableau ne peut pas être traité comme un seule bloc tel que tab*10, mais chaque élément doit être traité séparément. Pour accéder à un seul élément du tableau, nous utilisons le nom du tableau avec un indice à l'intérieur de deux crochets [et]

Remarque : la tentative d'accès à un élément qui n'existe pas (si l'indice est supérieur ou égal à la taille du tableau ou négatif) provoque l'arrêt du programme.

Exemple

`tab[5-3]←tab[tab[3]+1]` ⇔ `tab[2]←tab[0+1]` ⇔ `tab[2]←7`

Indice	0	1	2	3	4	5	6	7
valeur	15	7	7	0	9	2	0	-3

La lecture d'un tableau

```
lire(tab[0])
lire(tab[1])
…
lire(tab[N-1])
```

Algorithme	C
pour i←0 **à** N-1 **faire** écrire("nb", i, "⇒") lire(tab[i]) **finPour**	**for**(i = 0; i < N; i++)**{** printf("nb %d ⇒", i); scanf("%d", &tab[i]); **}**

L’affichage d’un tableau

Algorithme	C

pour i←0 **à** N-1 **faire**
écrire(tab[i])
finPour
	for(i = 0; i < N; i++){
	printf("%d\t", tab[i]);
	}

Exemple

Écrire un programme qui reçoit les moyennes de N étudiant, où N est déterminé par l'utilisateur, puis calcule le nombre d'étudiants qui n'ont pas pris la matière. (Moyenne inférieure à 10).

```
Algorithme nb_ajourne
Const MAX=200
var  note[MAX] :tableau de réel
 i, aj, N :entier
début
Faire
 écrire("entrer le nbr des étudiants (<", MAX, ")")
 lire(N)
TQ N>MAX
pour i←0 à N-1 faire
  écrire("note ", i, "⇒")
  lire(note[i])
finPour
aj ←0
pour i←0 à N-1 faire
  si note[i]<10 alors
    aj←aj+1
  finSi
finPour
écrire("le nbr des ajournés est ", aj)
fin
```

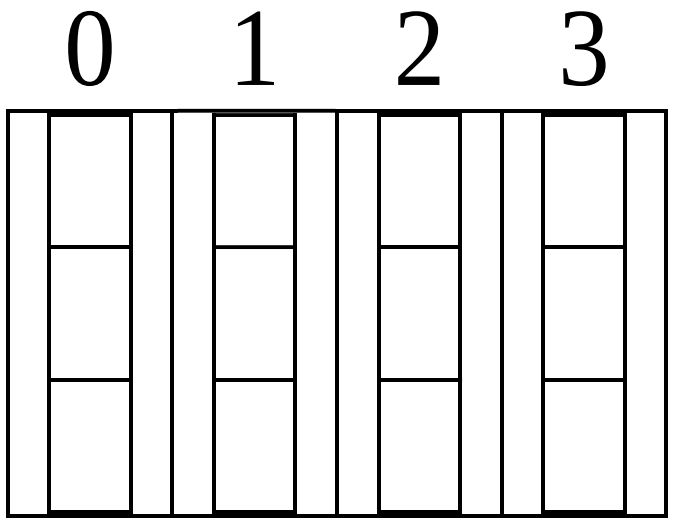
C

```c
#include<stdio.h>
#define MAX 200
int main(){
 float note[MAX] ;
 int i, N, aj=0; // aj càd nb ajournés
 // récupérer le nbr des étudiants
 do{
   printf("entrer le nbr des étudiants (<%d)",MAX) ;
   scanf("%d",N) ;
 }while (N>MAX) ;
 // Remplir le tableau
 for(i = 0; i < N; i++){
   printf("note %d ⇒", i);
   scanf("%d", &note[i]);
 }
 // calculer le nbr des ajournés
 for(i = 0; i < N; i++)
   if(note[i]<10) aj++ ;
 // affichage du résultat.
 printf("le nbr des ajournés est %d", aj);
}
```

3. Les tableaux multidimensionnels

Un tableau bidimensionnel (également appelé matrice): est un tableau des tableaux

Les éléments sont accessibles via deux indices, le premier spécifiant le numéro de ligne et le second spécifiant le numéro de l'élément dans cette ligne (colonne).



Numéro de colonne

N ligne	0	1	2	3	4
0	15	7	-3	0	9
1	6	12	4	33	85
2	2	-8	17	28	-52
3	14	42	36	49	-12

Déclaration

Algorithme

nomMatrice [Lignes][Colonnes] : **Tableau de** typeElements

C

typeElements nomMatrice [Lignes][Colonnes] ;

Exemple

Algorithme

const L=100

const C=100

M[L][C] : tableau de réel

mat1[50][30],mat2[30][20] : tableau d’entier

C

const int L=100 , C=200 ;

float M[L][C] ;

int mat1[50][30],mat2[30][20];

Initialisation

int mat[][] = **{{**15, 7, -3 ,0 ,9**},{**6, 12, 4,33,85**},{**2, -8, 17 ,28,-52**},{**14, 42, 36, 49, -12**}}**;

n ligne \ numéro de colonne	0	1	2	3	4
0	15	7	3-	0	9
1	6	12	4	33	85
2	2	8-	17	28	52-
3	14	42	36	49	12-

Utilisation

Pour accéder à un seul élément de la matrice, nous utilisons le nom de la matrice avec un indice à l'intérieur de deux crochets [et] spécifiant le numéro de ligne, et un autre indice à l'intérieur de deux crochets [et] spécifiant le numéro de colonne.

syntaxe `mat[ligne][ colonne]`

exemple `mat[1][3]←mat[1][3]+2`

N ligne \ Numéro de colonne	0	1	2	3	4
0	15	7	3-	0	9
1	6	12	4	35	85
2	2	8-	17	28	52-
3	14	42	36	49	12-

La lecture d'une matrice

```
pour   j←0   à   C-1
faire
lire(M[0][j])
fpour
pour   j←0   à   C-1
faire
lire(M[1][j])
fpour
…
pour   j←0   à   C-1
faire
lire(M[L-1][j])
fpour
```

```
pour i←0 à L-1 faire
 pour j←0 à C-1 faire
  écrire("M[", i, ",", j, "]⇒")
  lire(M[i][j])
 finPour
 finPour
```

```
for(i = 0; i < L; i++)
 for(j = 0; j < C; j++){
  printf("M[%d, %d] ⇒", i,j);
  scanf("%d", &M[i][j]);
 }
```

L’affichage d’une matrice

Algorithme	C

pour i←0 **à** L-1 **faire**
  pour j←0 **à** C-1 **faire**
  écrire(M[i][j])
  finPour
finPour
	for(i = 0; i < L; i++)**{**
	  for(j = 0; j < C; j++)
	  printf("%d\t", M[i][j]);
	  printf("\n") ;
	}

Exemple

Écrivez un programme qui lit les températures par heure pour 30 jours, sous la forme d'une matrice (30 par 24), puis les affiche à l'écran, après quoi il affiche la température la plus élevée et quand elle a été enregistrée.

```
Algorithme nb_ajourne
Const Jr =30   Hr=24
var  T[Jr][Hr] :tableau de réel
  maxT :réel
  i, j,maxJr,maxHr :entier
début
pour i←0 à Jr-1 faire
 pour j←0 à Hr-1 faire
  écrire("T[", i+1, ",", j, "]⇒")
  lire(T[i][j])
 finPour
finPour
pour i←0 à Jr-1 faire
 pour j←0 à Hr-1 faire
  écrire(M[i][j])
 finPour
finPour
maxT←T[0][0]
maxJr←0
maxHr←0
```

```
pour i←0 à Jr-1 faire
 pour j←0 à Hr-1 faire
  si (T[i][j]>maxT) alors
   maxT←T[i][j]
   maxJr←i
   maxHr←j
  finSi
 finPour
finPour
écrire("la température maximale est ", maxT," et a été
 enregistrée le ", maxJr+1, " à ", maxHr )
fin
```

```c
#include<stdio.h>
#define Jr 30 // nb lignes
#define Hr 24 // nb colonnes
int main(){
float   T[Jr][Hr]   ,maxT;   //   température
int i, j,maxJr,maxHr;
// Remplir les températures
for(i = 0; i < Jr; i++)
 for(j = 0; j < Hr; j++){
  printf("T[%d, %d] ⇒", i+1,j);
  scanf("%d", &T[i][j]);
 }
// afficher toutes les températures
for(i = 0; i < Jr; i++){
 for(j = 0; j < Hr; j++)
   printf("%d\t", M[i][j]);
 printf("\n") ;
}
//   recherche   de   la   température   maximale
maxT=T[0][0];
maxJr=0 ;
maxHr=0 ;
maxfor(i = 0; i < Jr; i++)
 for(j = 0; j < Hr; j++)
  if (T[i][j]>maxT){
   maxT=T[i][j];
   maxJr=i;
   maxHr=j;
  }
//affichage des résultats
printf("la  température  maximale  est
%d et a été enregistrée le %d à %d",
maxT, maxJr+1, maxHr) ;
}
```

4. Les chaînes de caractères

- La chaîne de caractères est un ensemble ordonné de caractères
- Ils sont toujours entre guillemets double « " »
- tels que "informatique", "Bonne chance\n", "1", "3.14".
- En **C**, les tables de type **char** sont utilisées pour créer des chaînes.
- Lorsque vous lisez une chaîne à partir du clavier, chaque caractère est placé dans une zone
- le caractère '\0' est ajouté à la fin du texte pour indiquer sa fin.
- Le caractère '\0' est appelé « **null** » qui porte le code 0.
- Il existe une constante déclarée dans la bibliothèque **stdio.h** appelée **NULL** en majuscules.

```
#define NULL 0
```

NULL ⇔ **'\0'** ⇔0

Déclaration & Initialisation

Algorithmes

```
var str :chaine de caractères
var str[30] :caractères
```

en C

```
char str[30] ;
```

Initialisation en C

```
char slt[] = {'W', 'e', 'l', 'c', 'o', 'm', 'e', '\0'};
```

Cette instruction crée un tableau à 8 caractères (7 cases pour le mot « Welcome » et une zone contenant le caractère '\0'. Toutefois, il existe un moyen plus simple et plus rapide de créer et d'initialiser une chaîne:

	0	1	2	3	4	5	6	7
slt	W	e	l	c	o	m	e	\0

```
char slt[] = "Welcome";
```

Cela conduit au même résultat, qui est la création d'un tableau à 8 caractères, se terminant par le caractère '\0'.

La taille de la table peut également être spécifiée :

	0	1	2	3	4	5	6	7	8	…	28	29
slt	W	e	l	c	o	m	e	\0		…		

```
char slt[30] = "Welcome";
```

Affectation

Les chaînes étant des tableaux, une chaîne ne peut pas être affectée à une variable directement après sa déclaration. Et l'opération suivante est faux.

```
char slt[30];
slt = "Welcome";// erreur : affectation a un tableau.
```

Pour affecter une chaîne à une variable ou copier une variable à une autre variable, nous utilisons la fonction strcpy().

```
strcpy(slt , "Welcome" );
```

Affichage & Lecture

Affichage

```
écrire(str)
```

En C

```
printf("%s",str) ;
Puts(str);
```

Lecture

```
lire(str)
```

En C

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

ou

```
#include<string.h>
…
gets(slt) ; // pour du texte contenant des espaces
```

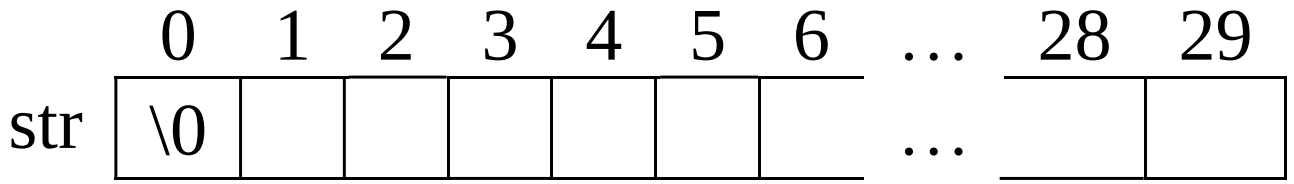
Quelques fonctions spécifiques aux chaînes en C

`strcpy(s1, s2);`	Copie la chaîne s2 dans la chaîne s1.
`strcat(s1, s2);`	pour ajouter une chaîne S2 à la fin de la chaîne S1.
`strlen(s1);`	Cette fonction renvoie la longueur de la chaîne s1.
`strcmp(s1, s2);`	Renvoie 0 si s1 et s2 sont identiques ; inférieur à 0 si s1<s2 ; supérieur à 0 si s1>s2.

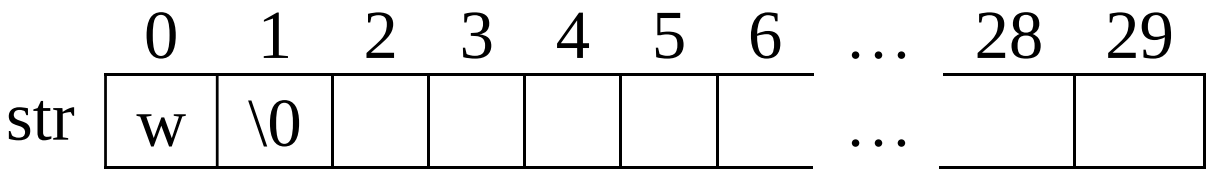
Exemples

Exemple 1 : Chaîne vide str="" ; qui est de longueur 0

Le contenu des cases n'a pas d'importance après '\0', la chaîne se termine au premier '\0'. Ainsi, n'importe quelle chaîne peut être convertie en chaîne vide en plaçant str[0]='\0' ;



Exemple 2 : chaîne contenant un seul caractère, elle est différente du type caractère. Ainsi, "w"≠'w' parce que "w" est un tableau.



Exemples

Exemple 3 : écrire le programme qui entre du texte, puis convertit les lettres majuscules en minuscules et les minuscules en majuscules.

```
algorithme inverse
var txt :chaine[200]
    i :entier
début
 écrire(("entrer un texte")
 lire(txt)
 i←0
 TQ txt[i]≠'\0' faire
   si txt[i]>='A' et txt[i]<='Z'
alors
   txt[i]=txt[i]+'a'-'A'
  sinon
   si (txt[i]>='a' et
txt[i]<='z') alors
     txt[i]=txt[i]-('a'-'A');
   finSi
  finSi
 finTQ
 écrire(txt)
fin.
```

Exemples

```
#include<stdio.h>
#include<string.h>
int main(){
 char txt[200] ;
 int i ;
 printf("entrer un texte\n") ;
 gets(txt)
 for(i=0 ;txt[i] !='\0' ;i++)
  if
(txt[i]>='A'&&txt[i]<='Z')
   txt[i]+='a'-'A' ;
  else
   if
(txt[i]>='a'&&txt[i]<='z')
    txt[i]-='a'-'A' ;

 printf("%s",txt) ;
 return 0 ;
}
```

Fin Chapitre 05