GRANDE ÉCOLE D'**INGÉNIEUR GÉNÉRALISTE**

Programmation Embarquée

2021-2022

www.hei.fr

Sommaire

HEI
INGÉNIEURS
POUR LE MONDE

yncréa

V. Tableaux et Structures

1. Tableaux

Définition:

Une tableau est un regroupement de variables ***de même type***, il est identifié par un nom. Chacune des variables du tableau est numérotée, ce numéro s'appelle un indice.

Chaque variable du tableau est donc caractérisée par le nom du tableau et son indice. Si par exemple, T est un tableau de 10 variables, alors chacune d'elles sera numérotée et il sera possible de la retrouver en utilisant simultanément le **nom du tableau** et **l'indice de la variable**.
Les différentes variables de T porteront des numéros de 0 a 9, et nous appellerons chacune de ces variables un élément de T.

Déclaration:

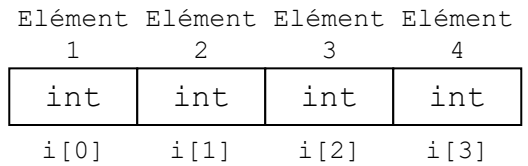
Comme les variables d'un tableau doivent être de **même type**, il convient de préciser ce type au moment de la déclaration du tableau. De même, on précise lors de la déclaration du tableau le nombre de variables qu'il contient. La syntaxe est :

```
<type> <nomdutableau>[<taille>];
```

Exemple:

```
int i [4] ;
```

*Tableau de nom « **i** » pouvant stocker 4 données de type **int**.*



Le nombre de dimensions d'un tableau n'est pas limité, il est fixé par le nombre de valeurs entre crochets : taille1, taille2, etc...

*Leur produit taille1*taille2*...*tailleN fournit le nombre d'éléments du tableau*

***Exemple de déclaration** : int k [3][4];*
*Crée un tableau nommé **k** et possédant 12 élements de type **int***

V. Tableaux et Structures

1. Tableaux

Initialisation:

affectation multiple :

```
int v[5] ;
v[0]= v[1]= v[2]= v[3]= v[4]= 0 ;
```

boucle :

```
int k, v[5] ;
for (k=0; k<5; k++)
   v[k]= 0 ;
```

initialisation à la définition :

```
          int v[10]={0,1,2,3,4,5,6,7,8,9} ;
Ou        int v[]={0,1,2,3,4,5,6,7,8,9} ;

          int v[10]={0,1,2,3,4} ; /*les 5 derniers éléments reçoivent 0 */

          int v[10]={0,0,0,0,0,0,0,0,0,0} ;
Ou        int v[10]={0} ;
```

En initialisant les tableaux, faites attention au **débordement d'indice** ! c.à.d pour un tableau k[i] de taille 10, i doit prendre des valeurs inférieurs ou égales à 9.

Rappel:
Les éléments d'un tableau à n éléments sont indices de 0 à n -1.

V. Tableaux et Structures

1. Tableaux

Accès aux éléments:

Les éléments d'un tableau à n éléments sont indicés de 0 à n-1.
On note T[i] l'élément d'indice i du tableau T.
Pour un tableau *int T [5] = {1 , 3 , 5 , 7 , 9 } ;*
Les cinq éléments du tableau de l'exemple ci-avant sont donc notés:
T[0] , T[1] , T[2] , T[3] et T[4] .

Saisie et affichage:

La déclaration à droite est celle d'un tableau de *4 int appelé A*. Il convient ensuite d'effectuer les saisies des 4 valeurs. On peut par exemple procéder de la sorte :

```
printf("Saisissez quatre valeurs:\n");
printf("1") ;
scanf(%d",&A[0]);
printf("2");
scanf(%d",&A[1]);
printf("3");
scanf(%d",&A[2]);
printf("4");
```

V. Tableaux et Structures

1. Tableaux

Saisie et affichage:

Faire des copier/coller pour rédiger un tel code est lourd! Nous procéderons plus élégamment en faisant une boucle :

```
printf("Saisissez trois valeurs:\n");
for(i=0;i<4;i++){
    scanf("%d",&A[i]);
}
```

Pour afficher un tableau via une boucle:

```
for(i=0;i<4;i++){
    printf("%d ",A[i]);
}
```

V. Tableaux et Structures: TD 3

1. Ecrire un programme plaçant dans un tableau int T [10]; les valeurs 1; 2; ….; 10, puis affichant ce tableau. Vous initialiserez le tableau à la déclaration.

2. Ecrire un programme en C qui:
a) Fait le même exercice que 1 en initialisant le tableau avec une boucle.
b) Affichez la somme des n éléments du tableau T.
c) Demande à l'utilisateur de saisir un int et dites-lui si ce nombre se trouve dans T et combien de fois.

3. Ecrire un programme qui demande à l'utilisateur de remplir un tableau int T[10] avec des valeurs paires. Utilisez la boucle do … while pour le contrôle de saisie.

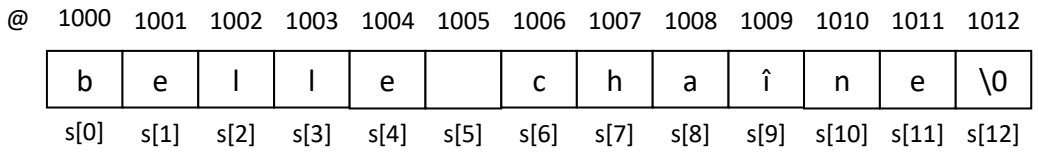
V. Tableaux et Structures

2. Chaînes de caractères (string)

Définition:

En langage C, les chaînes de caractères, qu'elles soient constantes ou variables, sont des tableaux à une dimension, ayant des éléments de type **char.**
Les chaînes de caractères se terminent **toujours** par un **caractère nul** qui est le **\0**. La fonction du caractère "nul" est très simple : il indique où se termine la chaîne de caractères!

Exemple: un tableau qui stocke la chaîne ***belle chaîne***



V. Tableaux et Structures

2. Chaînes de caractères (string)

Déclaration:

Comme une chaîne de caractères est un tableau de char, on la déclare :

```
char <nom chaine>[<taille_chaine >];
```

Par exemple, on déclare une chaine c de 200 caractères de la sorte :

```
char c[200];
```

Attention! Le nombre maximal de lettres qu'il sera possible de placer dans c ne sera certainement pas 200 mais 199, car il faut placer après le dernier caractère de la chaîne un caractère nul, le \0!

V. Tableaux et Structures

2. Chaînes de caractères (string)

Initialisation:
On initialise une chaîne à la déclaration de la sorte :

```
char <nom chaine>[<taille_chaine>]=<valeur_initialisation>;
```

Où la valeur d'initialisation contient la juxtaposition de caractères formant la chaîne entourée de guillemets.
e.g. `char c[50]= "Toto";`

Cette instruction déclare une chaîne de caractères *c* initialisée à "Toto". Les 5 premiers éléments du tableau seront occupés par les 4 caractères de la chaîne ainsi que par le caractère nul, les autres contiendront des valeurs non significatives.

Observez bien l'exemple suivant : `char c[4]= "Toto";`
Cette déclaration engendrera un warning à la compilation et probablement une erreur à l'exécution car l'affectation du caractère nul à la 5-eme position du tableau donnera lieu à un débordement d'indice.

V. Tableaux et Structures

2. Chaînes de caractères (string)

Opérations d'entrées-sorties:

1ère méthode : « lourde ! »

Une *string* est une suite de caractères individuels gérés en mémoire sous forme de tableau **char.** Un procédé élémentaire pour saisir une chaîne de caractères consiste à lire, via une boucle, tous les caractères successivement, et à les ranger dans les éléments du tableau.

```
#include <stdio.h>

int main(){
char name [40];
int i=0;
printf("Votre nom: ");
while ((name[i]= getchar())!='\n')
{
i++;
}
name[i]='\0';
printf("\nVotre nom est: ");
i=0;
while (name[i] != '\0')
{
putchar(name[i]);
i++;
}
return 0;
}
```

V. Tableaux et Structures

2. Chaînes de caractères (string)

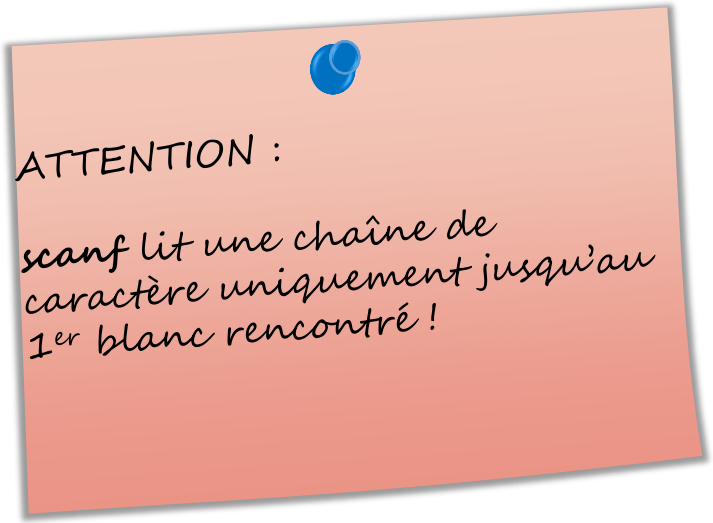
2ème méthode : Saisie et affichage avec *scanf* et *printf*

Les fonctions ***scanf*** et ***printf*** possèdent une spécification de format ***%s*** qui permet de manipuler des *string*. Exemple:

```
char a[41] ; /*tableau de 40 caractères au total */
scanf("%s",a) ;  /*pas d'opérateur d'adressage & devant a */
```

L'opérateur d'adressage & habituel avec **scanf** manque devant la variable tableau **a.** La raison en est que, pour le compilateur, le nom d'un tableau (quel que soit le type des éléments) équivaut à l'adresse du début du tableau en mémoire, c'est-à-dire à l'adresse du premier élément du tableau. En d'autres termes **a** équivaut à **&a[0].**

Pour afficher la chaîne de caractères saisie, on passe par la fonction **printf** avec le format %s : `printf("%s",a);`



V. Tableaux et Structures

2. Chaînes de caractères (string)

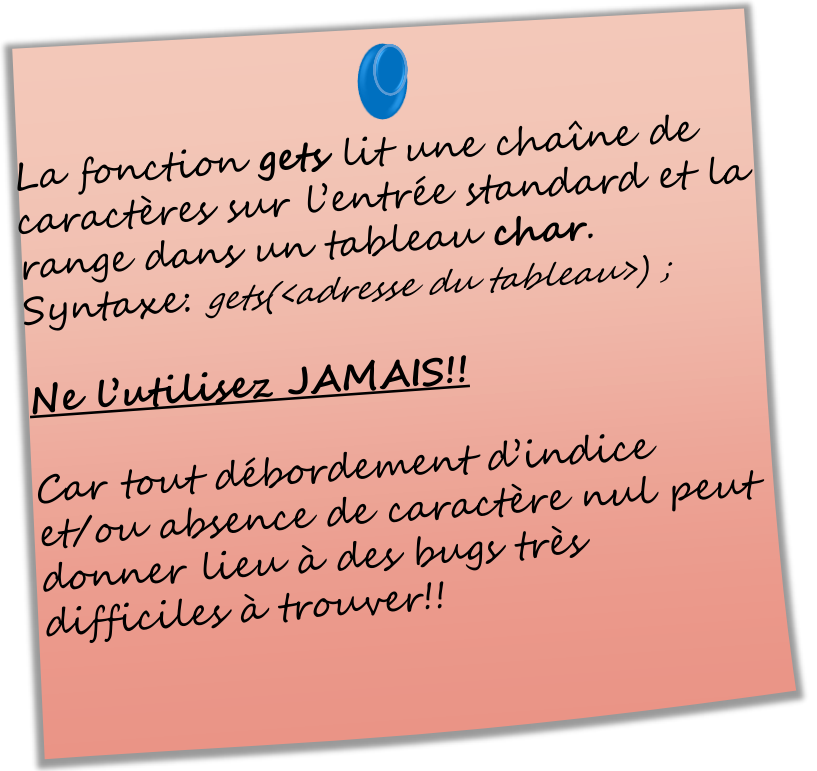
3ème méthode : Saisie et affichage avec fgets et puts.

fgets lit une chaîne de caractères sur l'entrée standard et la range dans un tableau **char**. La fonction f**gets** nécessite comme argument le nom de la chaîne, sa taille et stdin qui signifie simplement que les données vont venir du clavier.
La syntaxe est :

```
fgets(<chaine>, <taille>, stdin );
```

puts affiche une chaîne de caractères sur la sortie standard. La syntaxe est :

```
puts(<adresse du tableau>) ;
```



V. Tableaux et Structures

2. Chaînes de caractères (string)

La bibliothèque ***string.h***:

Cette bibliothèque propose des fonctions de manipulation de chaînes de caractères,
à savoir :

- strcpy : copier une chaîne
- strcat : concaténer deux chaînes
- strlen : longueur d'une chaîne de caractère
- strcmp: comparer deux chaînes

V. Tableaux et Structures

2. Chaînes de caractères (string)

La bibliothèque ***string.h***:

strcpy : recopie une chaîne dans un tableau **char**

```
char s[13] ;
strcpy(s, "Belle chaine") ;
```

Faites attention:
strcpy (<destination>, <source>);
Et pas l'inverse!!

strcat : concaténation de 2 chaînes de caractères

```
char nom[13]= "toto" ;
```


```
char ext[5]= ".c" ;
```


```
strcat(nom, ext) ; /* l'extension est ajoutée dans le nom */
```

résultat dans nom:

V. Tableaux et Structures

2. Chaînes de caractères (string)

La bibliothèque ***string.h***:

strlen : longueur de chaîne de caractères

```
int l ;
char s[] = "Bonjour!";
l = strlen(s) ;                          affiche:
printf("il a %d caractères\n",l) ;       il a 8 caractères.
```

strcmp: La fonction ***strcmp()*** compare deux chaînes de caractères caractère par caractère. Si le premier caractère de deux chaînes de caractères est égal, le caractère suivant de deux chaînes de caractères est comparé. Ceci continue jusqu'à ce que les caractères correspondants de deux chaînes de caractères soient différents ou qu'un caractère nul '\0' soit atteint.

- 0 si les deux chaînes sont identiques (égales)
- négatif si la valeur ASCII du premier caractère non identique est inférieure à la seconde.
- entier positif si la valeur ASCII du premier caractère non identique est supérieure à la seconde.

```
#include<stdio.h>
#include<string.h>
int main (){

char a[26];
int s1=0;

fgets(a,26,stdin);

s1=strlen(a);

printf("La taille de la
chaine a est %d \n",s1);

return 0 ;
}
```

V. Tableaux et Structures: TD 4

1. Créer une chaîne de caractères contenant la valeur « Je n'aime pas le chocolat" et affichez-la avec %s. Vous donnerez au tableau la plus petite taille possible.

2. Même exercice que 1 mais sans utiliser %s.

3. Ecrire un programme qui:
   a) Demande à l'utilisateur de saisir proprement une chaîne de caractère (sans débordement d'indice) avec le caractère nul.
   b) calculer sans *strlen* la taille de la chaîne (nombre de caractères sans compter le caractère nul).

4. Ecrire un programme qui demande à l'utilisateur de saisir deux chaînes de caractères, déterminez la plus longue des deux.

5. Ecrire un programme qui demande à l'utilisateur de saisir une chaîne de caractères, copiez-là dans une deuxième chaîne.

6. Ecrire un programme qui effectue la concaténation de deux chaînes de caractères, affichez la concaténation de la première à la suite de la deuxième.

V. Tableaux et Structures

3. Structures

Définition:

Nous avons utilisé des tableaux pour désigner, avec un nom unique, un ensemble de variables. Par exemple, un tableau T à n éléments est un ensemble n de variables désignées par la lettre T.

Dans un tableau, les variables doivent être de type homogène, cela signifiant qu'il n'est pas possible de juxtaposer des *char* et des *int* dans un tableau.
Lorsque l'on souhaite faire cohabiter dans une variable des types hétérogènes, on utilise des structures.

Une structure, appelé enregistrement dans d'autres langages, est une variable contenant plusieurs variables, appelées **champs**.

V. Tableaux et Structures

3. Structures

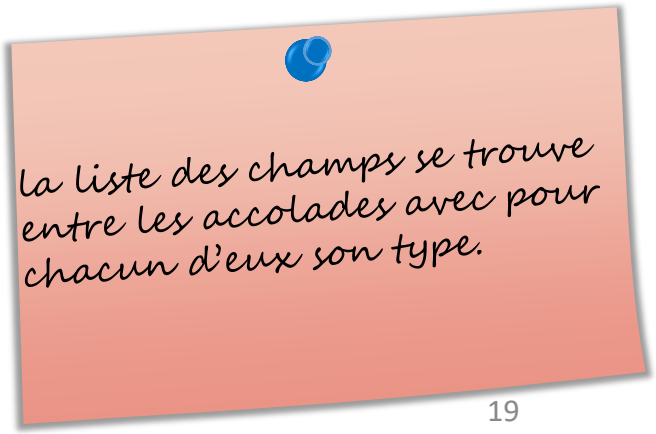
Déclaration:

Avant de pouvoir définir une variable structurée, il faut fournir au compilateur une définition de structure décrivant l'aspect de la structure à créer. On indique, pour cela, les champs de la structure. Chaque champ est introduit avec son type et son nom. La syntaxe d'une définition de structure est :

```
struct <Nom_Structure>
                {
                        <Type_Champ1> <Nom_Champ1> ;
                        <Type_Champ2> <Nom_Champ2> ;
                        ...
                        <Type_ChampN> <Nom_ChampN> ;
                } ;
```

```c
#include<stdio.h>
int main ( )
{
struct client
      {
         char nom[20] ;
         char prenom[20] ;
         unsigned int codepostal ;
         char ville[20] ;
      } ;

return 0 ;
}
```



V. Tableaux et Structures

3. Structures

Définition de variables structurées:

Après avoir déclaré au compilateur la structure susceptible d'être utilisée dans le programme, on peut définir des variables possédant ce type.
Exemple :

```
struct livre
       {
              char auteur [12] ;
              char titre [20] ;
              short an;
       } ;
struct livre a1, a2, a3 ;
```

La déclaration d'une structure et la définition d'une variable du type concerné peuvent être condensées dans une même instruction. Exemple:

```
struct livre
              {
char auteur [12] ;
char titre [20] ;
short an;
                    } a1, a2, a3 ;
```

V. Tableaux et Structures

3. Structures

Opérations sur les variables structurées:

On accède aux champs d'une variable structurée à l'aide de la notation pointée:

```
nomvariable.nomduchamp
```

```
#include<stdio.h>
int main ( )
{
struct client
      {
         char nom[20] ;
         char prenom[20] ;
         unsigned int codepostal ;
         char ville[20] ;
      } ;

return 0 ;
}
```

V. Tableaux et Structures

3. Structures

Opérations sur les variables structurées:

On accède aux champs d'une variable structurée à l'aide de la notation pointée:

`nomvariable.nomduchamp`

```
#include<stdio.h>
int main ( )
{
struct point
      {
         double abs ;
         double ord ;
         char nom[15];
      } ;

int main (){

struct point p;
p.ord=2;
p.abs=p.ord + 1;
strcpy (p.nom, "point A");
printf("p =(%s,%f,%f)\n",p.nom,
p.abs,p.ord);
return 0 ;
}
```

Affiche:
p = (point A, 3.000000, 2.000000)

V. Tableaux et Structures: TD 5

1. Ecrire une structure client qui contient: nom, prénom, code postal, ville.
   a) Initialisez une variable CL1 lors de la création (strcpy…) . Affichez CL1
   b) Demandez à l'utilisateur de saisir les valeurs d'une deuxième variable client CL2. Affichez CL2.