INF155 – SÉANCE 4
LES TABLEAUX 1D

Anis Boubaker, Ph.D.
Maître d'enseignement
École de Technologie Supérieure

PLAN DE LA SÉANCE

- Tableaux: c'est quoi et pourquoi faire?
- Déclarer, affecter et obtenir les valeurs d'un tableau
- Passer un tableau en paramètre à un fonction

3 LES TABLEAUX À UNE DIMENSION

LES VARIABLES

- Les variables que nous avons vues, jusque là, nous permettent de stocker une valeur en mémoire (et une seule).

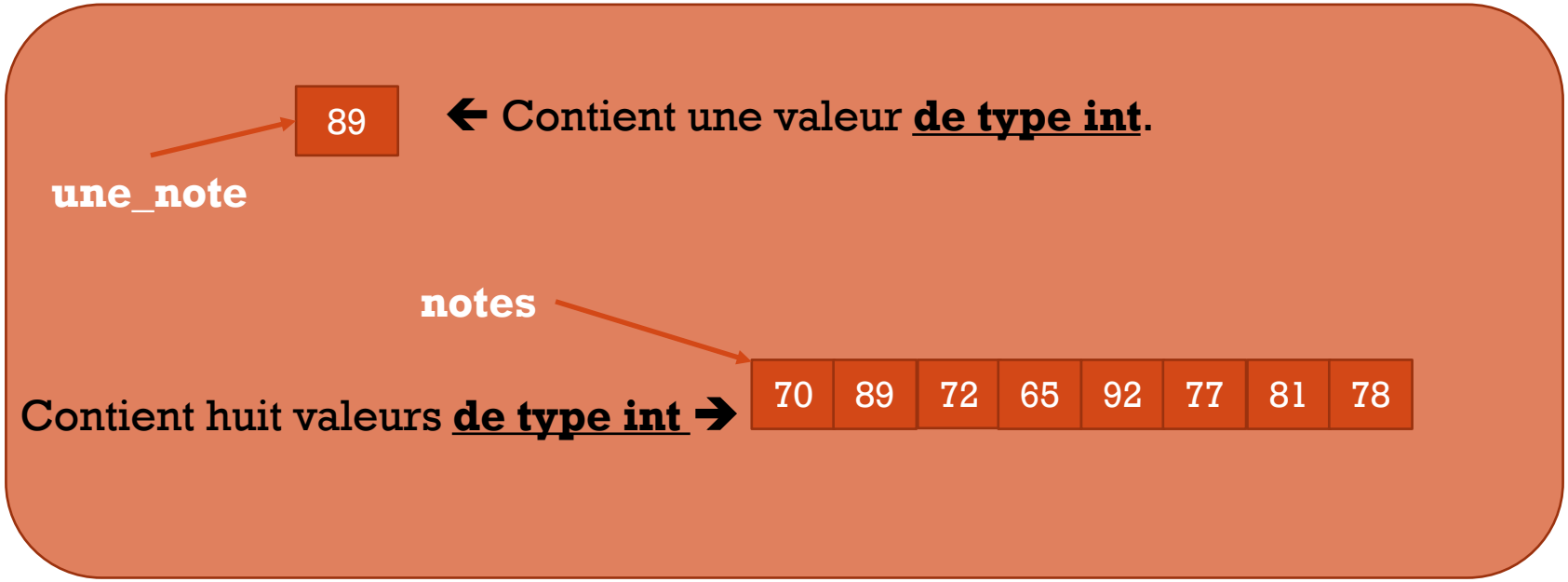
- Toutefois, il est fréquent de vouloir stocker plus d'une valeur au sein de la même variable.

- Exemple:
  - les notes d'une classe
  - Une liste de températures
  - Une liste de prix ...

Nous pouvons le faire
en utilisant les tableaux

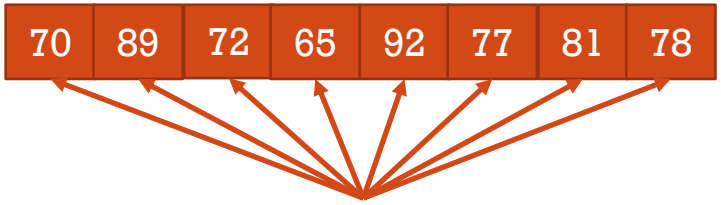
LES TABLEAUX

- Un **tableau**: est un suite de variables **de même type**, stockées en mémoire dans un espace contigu.



LES TABLEAUX

- Une variable qui permet de stocker plusieurs valeurs d'un même type. Nous appelons chaque valeur **une case du tableau**.



- L'espace mémoire réservé à un tableau permet de stocker ses valeurs successivement (l'une à la suite de l'autre)

LES TABLEAUX

- Avec un tableau, nous pouvons:
  - Obtenir la $i^{ème}$ valeur stockée dans le tableau
  - Affecter la valeur de la $i^{ème}$ case du tableau

- Les boucles nous seront très utiles dans la manipulation des tableaux (on itère sur toutes les cases du tableau)

DÉCLARATION D'UN TABLEAU

- Pour déclarer un tableau, nous devons spécifier:
  - L'identifiant du tableau
  - Le type de valeurs que le tableau va contenir
  - Le nombre de valeurs qu'il va contenir: **la taille du tableau**
  - ***Optionnel***: ses valeurs initiales, séparées par des virgules

Syntaxe:

```
type identifiant[taille]= {valeur1, valeur2, ... };
```

Exemple:

```
int notes[8]= {70, 89, 72, 65, 92, 77, 81, 78};
```

INITIALISATION D'UN TABLEAU

- Nous pouvons fournir les valeurs initiales d'un tableau (c'est optionnel!).
- Il **n'est pas** impératif de fournir toutes les valeurs :
  - On peut fournir seulement premières valeurs uniquement.
  - Le reste du tableau est alors rempli de 0.
- Exemple:

```
double temperatures[50] = {10.3, 20.0, 0.5, 25} ;
```

- Il n'est possible de fournir les valeurs d'un tableau sous cette forme **uniquement lors de la déclaration!**

EXERCICES

- Que contiendra le tableau mon_tab ?

```
double mon_tab[10] = {1};
```

- Comment feriez-vous pour initialiser toutes les valeurs d'un tableau à 0?

TAILLE D'UN TABLEAU

- Un tableau **a une taille fixe.** Il n'est **pas possible de modifier la taille d'un tableau** après l'avoir déclaré.

- À l'usage:
  - On crée des tableaux d'une taille suffisamment grande pour accommoder les divers cas d'utilisation de notre programme
  - On conserve en mémoire (dans une autre variable) le nombre de valeurs "effectives", car on n'utilisera que rarement le tableau en entier.

- Par exemple:
  - si on doit stocker des notes, nous savons qu'un groupe ne peut dépasser 70 étudiants ➔ On crée un tableau de taille 70
  - On crée une variable **nb_notes** qui contient le nombre de notes, et qu'on incrémente au fur et à mesure qu'on ajoute des notes.

TAILLE D'UN TABLEAU

- On ne doit **jamais** excéder la taille d'un tableau.

- Le compilateur ne nous empêchera pas d'écrire dans une case qui dépasse la taille du tableau ☹

- La conséquence, si ça se produit:
  - La valeur écrite risque d'écraser l'information dans une autre variable
  - La valeur écrite risque d'être écrasée car elle n'est pas réservée au tableau
  - Dépendamment du système, le système d'exploitation peut même interrompre l'exécution du programme

TAILLE D'UN TABLEAU

- La taille d'un tableau doit obligatoirement être un nombre entier;

```
int mon_tableu[2.5]; // Erreur!
```

- La taille ne peut-être le contenu d'une variable
(sous Visual, mais certains compilateurs conformes à C99 le permettent)

```
int taille = 5;
int mon_tableu[taille]; // Erreur!
```

- La taille peut-être une constante de précompilation (macro)

```
#define TAILLE 10
int mon_tableu[TAILLE]; // OK
```

TAILLE D'UN TABLEAU

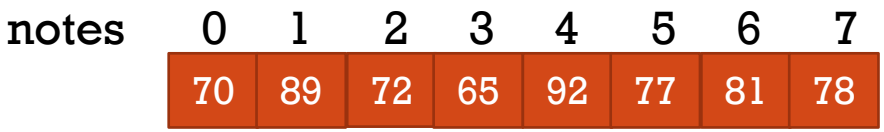
- Il est permis d'omettre de spécifier la taille d'un tableau, si on fournit la liste de valeurs initiales.
- La taille sera alors calculée en fonction du nombre de valeurs initiales fournies.
- Exemple:

```
double rabais = {0, 8.5, 12.5, 20}; //Tableau de 4 doubles

char nom = {'J','o','h','n',' ','S','n','o','w'}; //Tableau de 9 char
```

CASES D'UN TABLEAU

- Une fois initialisé, l'espace mémoire pour tout le tableau est réservé
- Les cases du tableau sont numérotées: partant **de l'indice 0** jusqu'à **taille-1**.



ACCÈS AUX CASES D'UN TABLEAU

- Nous accédons aux cases d'un tableau comme aux variables normales : en indiquant l'identifiant du tableau
- La différence est que nous devons également préciser l'indice de la case.

```
int mon_tableau[10] = {0};
mon_tableau[5] = 25;  //Assigne la valeur 25 à la case 5
//Accède à la case 5 du tableau et l'imprime à l'écran
printf("La case 5 contient la valeur: %d", mon_tableau[5]);
```

- **RAPPEL:** Les indices d'un tableau commencent à 0.

EXERCICES 1

- Soit un tableau de 100 éléments. Un collègue vous affirme que la case ayant pour indice 100 n'existe pas. A-t-il raison et pourquoi?

- Complétez le programme suivant afin d'afficher à l'écran le nombre total de bonnes notes. Une note est bonne si elle est supérieure à 80.

```
int main(void)
{
   int notes[8]= {70, 89, 72, 65, 92, 77, 81, 78};
   ...

   ...
}
```

EXERCICES 2

- Écrire un programme qui déclare un tableau de taille 100, qui y stocke les 100 premiers termes de la suite de Fibonacci puis qui les affiche.

Définition: la suite de Fibonacci se définit récursivement

- Fib(0) = 0
- Fib(1) = 1
- Pour n>1, Fib(n) = Fib(n-2)+Fib(n-1)

OBTENIR LA TAILLE D'UN TABLEAU

- Il est possible d'obtenir la taille, en octets, d'un tableau en utilisant l'opérateur **sizeof**.

```
int un_tableau[20];
int taille_en_octets = sizeof( un_tableau ); // = 80
```

- **IMPORTANT: sizeof** permet d'obtenir la taille en octets d'un tableau **uniquement dans la fonction où le tableau a été déclaré.**

- Comment feriez-vous pour obtenir le nombre d'éléments d'un tableau en utilisant **sizeof** ?

LES TABLEAUX ET LES FONCTIONS

ÉTS

TABLEAU EN PARAMÈTRE

- Comme pour une variable, il est possible de fournir un tableau en paramètre d'une fonction

- Pour qu'une fonction prenne un tableau en paramètre, nous devons le mentionner dans la liste des paramètres du prototype.

- Exemple:

```
int calculer_moyenne(int tableau[10], int nb_elements)
```

TABLEAU EN PARAMÈTRE

- Spécifier la taille du tableau, dans le prototype de la fonction, est optionnel (et d'ailleurs inutile...)

- Le prototype précédent est équivalent à:

```
int calculer_moyenne(int tableau[], int nb_elements)
```

TAILLE D'UN TABLEAU DANS UNE FONCTION

- **Il n'est pas possible** de déterminer la taille d'un tableau dans une fonction!
- Rappel: l'opérateur **sizeof** ne permet pas d'obtenir la taille d'un tableau passé en paramètre
- Nous devons donc passer un autre paramètre à la fonction, où l'on spécifie le nombre d'éléments effectifs du tableau

```
int calculer_somme(int tableau[], int nb_elements){
      int somme = 0;
      int i;

      for(i=0; i < nb_elements; i++){
            somme+=tableau[i];
      }

      return somme
}
```

TABLEAU EN PARAMÈTRE = PAR RÉFÉRENCE!

- Un tableau est **toujours** passé par référence en paramètre d'une fonction

- Un tableau en C est en fait un pointeur *masqué*…

- Si on modifie la case d'un tableau dans une fonction, le paramètre effectif (le tableau passé en paramètre) est modifié!

TABLEAU EN PARAMÈTRE = PAR RÉFÉRENCE!

```
void incrementer(int tableau[], int nb_elements){
        int i;
        for( i=0 ; i < nb_elements ; i++){
                tableau[i]++;
        }
}


int main(void){
        int mon_tableau[5]={10,20,30,40,50};
        int i;

        incrementer( mon_tableau, 5);

        for(int i=0; i<5; i++){
                printf("Valeur %d : %d", i, mon_tableau[i]);
        }
}
```


Fin