

Le C — Tableaux

<https://tinyurl.com/44yddt4d>

Qu'est-ce que c'est ?

Un tableau est un ensemble de valeurs du même type.

Chaque emplacement d'un tableau a un numéro que l'on nomme "index". En C le premier élément est à l'index 0.

Déclaration

Pour déclarer un tableau, il faut préciser son type ainsi que le nombre d'éléments qu'il pourra contenir.

```
int main()
{
    type name[capacity];

    return 0;
}
```

Par exemple pour stocker 5 notes dans un tableau :

```
int main()
{
    int grades[5];

    return 0;
}
```

Liste d'initialisation

Par défaut, un tableau contient des valeurs inconnues. Il est cependant possible d'affecter des valeurs lors de la déclaration avec une liste d'initialisation.

```
int main()
{
    int grades[5] = {15, 20, 11};

    return 0;
}
```

Dans le cas ci-dessus, on remarque que sur les 5 emplacements disponibles, seules 3 valeurs sont affectées. Les 2 dernières valeurs auront donc une valeur inconnue.

Liste d'initialisation

Pour éviter les valeurs inconnues, on peut remplir tout le tableau de zéros :

```
int main()
{
    int grades[5] = {0};

    return 0;
}
```

Attention : cette méthode ne fonctionne qu'avec 0. Il est aussi impossible de mettre d'autres valeurs par défaut, seul le 0 doit être présent.

Liste d'initialisation

Avec une liste d'initialisation, on peut retirer la taille du tableau manuelle. La taille du tableau correspondra à la taille de la liste d'initialisation.

```
int main()
{
    // int grades[3] = {15, 20, 11};
    int grades[] = {15, 20, 11};

    return 0;
}
```

Accès aux éléments

Pour accéder aux éléments d'un tableau, il faut utiliser la syntaxe `tableau[index]` :

```
int main()
{
    int grades[5] = {15, 20, 11};

    int first = grades[0];
    int second = grades[1];

    return 0;
}
```

Attention : accéder à un élément en dehors du tableau donnera des résultats imprédictibles, incluant un risque d'arrêt du programme.

Accès aux éléments

```
int main()
{
    int grades[5] = {15, 20, 11};

    printf("The first grade is %d\n", grades[0]);
    printf("The second grade is %d\n", grades[1]);
    printf("The third grade is %d\n", grades[2]);

    return 0;
}
```

Sortie

```
The first grade is 15
The second grade is 20
The third grade is 11
```


Affectation d'éléments

Tout comme pour accéder aux éléments, pour affecter un élément d'un tableau il faut utiliser la syntaxe `tableau[index]`, suivi de `= valeur` comme pour une affectation de variable classique.

```
int main()
{
    int grades[5];
    grades[0] = 15;
    grades[1] = 20;
    grades[2] = 11;

    return 0;
}
```

Dernier élément

Le premier élément d'un tableau étant à l'index 0, le dernier élément d'un tableau est donc à l'index qui correspond à sa capacité - 1.



```
int main()
{
    int grades[5];

    int last = grades[4];

    return 0;
}
```



Parcours d'un tableau

Pour parcourir un tableau, nous pouvons utiliser une boucle for.

```
● ● ●  
#include <stdio.h>  
  
int main()  
{  
    int grades[5] = {15, 20, 11, 6, 18};  
  
    for (int i = 0; i < 5; i++)  
    {  
        printf("Grade %d: %d\n", i, grades[i]);  
    }  
  
    return 0;  
}
```

Sortie

```
● ● ●  
Grade 0: 15  
Grade 1: 20  
Grade 2: 11  
Grade 3: 6  
Grade 4: 18
```

Parcours d'un tableau

Attention à ne pas confondre la capacité d'un tableau avec le nombre d'éléments qu'il contient. Il est pratique d'avoir une variable "taille" pour gérer les tableaux n'ayant pas un nombre d'éléments fixe.



```
#include <stdio.h>

int main()
{
    int grades[5];
    int grades_count = 0;

    grades[0] = 15;
    grades[1] = 20;
    grades[2] = 11;
    grades_count = 3;

    for (int i = 0; i < grades_count; i++)
    {
        printf("Grade %d: %d\n", i, grades[i]);
    }

    return 0;
}
```

Exemple

Moyenne des notes

```
#include <stdio.h>

int main()
{
    int count = 5;
    int grades[count];

    for (int i = 0; i < count; i++)
    {
        printf("Enter grade %d: ", i + 1);
        scanf("%d", &grades[i]);
    }

    float average = 0;
    for (int i = 0; i < count; i++)
    {
        average += grades[i];
    }
    average /= count;

    printf("Average: %.2f\n", average);

    return 0;
}
```

Sortie

```
Enter grade 1: 15
Enter grade 2: 20
Enter grade 3: 11
Enter grade 4: 16
Enter grade 5: 14
Average: 15.20
```

Dimensions multiples

Jusqu'à maintenant nous avons créé et utilisé des tableaux à une seule dimension. En C il est possible d'avoir une infinité de dimensions pour un tableau.

```
int main()
{
    int grades[3][4];

    return 0;
}
```



grades[0][0]	grades[0][1]	grades[0][2]	grades[0][3]
grades[1][0]	grades[1][1]	grades[1][2]	grades[1][3]
grades[2][0]	grades[2][1]	grades[2][2]	grades[2][3]

Ici nous avons créé un tableau de 3 lignes et 4 colonnes.

Dimensions multiples

Tout comme pour les tableaux à une dimension, on peut donner une valeur par défaut aux tableaux à dimensions multiples

```
int main()
{
    int grades[3][4] = {
        {15, 20, 11, 4},
        {3, 13, 18, 5},
        {14, 7, 13, 12},
    };

    return 0;
}
```

Même chose avec {0}.

```
int main()
{
    int grades[3][4] = {0};
}
```

Dimensions multiples

La taille de la première dimension n'est pas nécessaire tant qu'il y a une liste d'initialisation.

```
int main()
{
    // int grades[3][4]
    int grades[][4] = {
        {15, 20, 11, 4},
        {3, 13, 18, 5},
        {14, 7, 13, 12},
    };

    return 0;
}
```


Dimensions multiples

L'accès et l'assignation d'éléments se fait de façon semblable aux tableaux à une dimension.

```
int main()
{
    int grades[3][4];

    grades[0][0] = 1;
    grades[0][1] = 2;
    grades[0][2] = 3;
    grades[0][3] = 4;

    grades[1][0] = 5;
    grades[1][1] = 6;
    grades[1][2] = 7;
    grades[1][3] = 8;

    return 0;
}
```

Exemple

Table de multiplication

```
#include <stdio.h>

int main()
{
    int multiplications[5][9];

    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 9; j++)
        {
            multiplications[i][j] = (i + 1) * (j + 1);
        }
    }

    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 9; j++)
        {
            printf("%2d ", multiplications[i][j]);
        }

        printf("\n");
    }
}
```

Sortie

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45

Exemple Multidimensions

Tips exercices

```
#include <stdio.h>

/*int grades[4][4] = {
    {5, 2, 1, 4},
    {3, 3, 8, 5},
    {4, 7, 3, 2},
    {1, 1, 1, 1},
};
*/

void print_array(int row, int column, int **grid)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < column; j++)
        {
            printf("%d ", grid[i][j]);
        }

        printf("\n");
    }
}
```

N'ayez pas peur des ** dans les exercices, ce sont des tableaux à deux dimensions.

Sortie

```
5 2 1 4
3 3 8 5
4 7 3 2
1 1 1 1
```

C fini !

