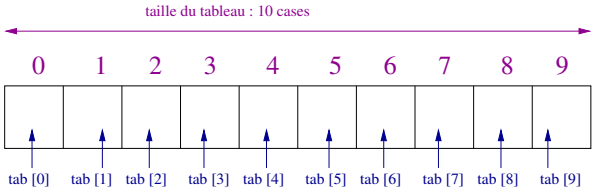
Chapitre 7

Tableaux à une dimension

Un tableau en C se déclare à l'aide de 3 informations :

1. Le type des éléments du tableau
2. Le nom du tableau
3. La taille du tableau (le nombre d'éléments)



Par exemple, pour déclarer la variable `tab` comme étant un tableau de 10 entiers, on écrira :

```
int tab [10] ;
```

La taille doit être une expression constante (ça ne peut pas être une variable du programme). Les indices vont obligatoirement de 0 à $taille - 1$.

Note : Les éléments du tableau ne sont absolument pas initialisés : ce serait une erreur de croire qu'ils sont tous à zéro juste après la création du tableau !

Pour accéder à la case d'indice i, on utilisera `tab [i]`.

Exemple d'utilisation :

- `tab [3] = 12;` met la valeur 12 dans la case numéro 3 [1] du tableau `tab`.
- `printf ("%d", tab [3])` affiche la valeur contenue dans la case numéro 3 du tableau `tab`.
- `tab [3] = tab [3] + 2 ;` ajoute 2 à la valeur contenue dans la case numéro 3 du tableau. Puisqu'elle contenait auparavant la valeur 12, elle contiendra à présent la valeur 14.

L'exemple suivant :

1. donc la 4^{ieme} case du tableau. Rappelez-vous, on commence à numéroter à partir de 0 !

```
/* Exemple pour tester l'utilisation des tableaux
*/
#include <stdio.h>

int main () {
    int tab [10] ; /* un tableau de 10 entiers est initialisee */
    int i ;

    /* On demande a l'utilisateur de remplir les 10 cases du tableau */
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("Quelle valeur pour la case %d ?\n", i) ;
        scanf ("%d", &tab [i]) ;
    }

    /* On affiche a present le tableau complet */
    printf ("Voici le tableau que vous avez rempli :\n") ;
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("%d ", tab [i]) ;
    }
    printf ("\n") ;

    return 0 ;
}
```

demande à l'utilisateur de remplir chacune des 10 cases du tableau. Puis, affiche le tableau complet à l'écran.

1 Exercices

Question 7-1 Vérification des notions de base → exercice de cours

1. Reprendre l'exemple du cours (section 7) et le tester. Programme à écrire et compiler :

```
/* Exemple pour tester l'utilisation des tableaux
*/
#include <stdio.h>

int main () {
    int tab [10] ; /* un tableau de 10 entiers est initialisee */
    int i ;

    /* On demande a l'utilisateur de remplir les 10 cases du tableau */
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("Quelle valeur pour la case %d ?\n", i) ;
        scanf ("%d", &tab [i]) ;
    }

    /* On affiche a present le tableau complet */
    printf ("Voici le tableau que vous avez rempli :\n") ;
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("%d ", tab [i]) ;
    }
    printf ("\n") ;

    return 0 ;
}
```

2. Modifiez le programme afin l'affichage du tableau soit réalisé par une procédure. Programme à écrire et compiler :

```
/* Exemple pour tester l'utilisation des tableaux
*/
#include <stdio.h>

void affichage (int tab []) {
    int i ;
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("%d ", tab [i]) ;
    }
    printf ("\n") ;
}

int main () {
    int tab [10] ; /* un tableau de 10 entiers est initialisee */
    int i ;

    /* On demande a l'utilisateur de remplir les 10 cases du tableau */
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("Quelle valeur pour la case %d ?\n", i) ;
        scanf ("%d", &tab [i]) ;
    }

    affichage (tab) ;

    return 0 ;
}
```

3. Ecrivez une procédure qui double chacune des valeurs saisies dans le tableau. Programme modifié :

```c
/* Exemple pour tester l'utilisation des tableaux
*/
#include <stdio.h>

void affichage (int tab [10]) {
    int i ;
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("%d ", tab [i]) ;
    }
    printf ("\n") ;
}

int main () {
    int tab [10] ; /* un tableau de 10 entiers est initialisee */
    int i ;

    /* On demande a l'utilisateur de remplir les 10 cases du tableau */
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("Quelle valeur pour la case %d ?\n", i) ;
        scanf ("%d", &tab [i]) ;
    }

    printf ("Tableau avant\n") ;
    affichage (tab) ;

    /* On double chaque valeur du tableau */
    for (i = 0 ; i < 10 ; i = i + 1) {
        tab [i] = tab [i] * 2 ;
    }

    printf ("Tableau apres\n") ;
    affichage (tab) ;

    return 0 ;
}
```

Question 7-2 Valeurs extrêmes

→ exercice d'assimilation

Ecrivez un programme qui :

1. initialise un tableau de 10 valeurs entières
2. recherche la valeur minimale et l'indice de sa place dans le tableau
3. recherche la valeur maximale et l'indice de sa place dans le tableau
4. affiche ces valeurs extrêmes et leur indice respectif.

Programme attendu :

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void affichage (int tab [10]) {
    int i ;
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("%d ", tab [i]) ;
    }
    printf ("\n") ;
}

int main () {
    int i ;
    int tab [10] ;
    int min ;
    int indmin ;
    int max ;
    int indmax ;

    /* On demande a l'utilisateur de remplir les 10 cases du tableau */
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("Quelle valeur pour la case %d ?\n", i) ;
        scanf ("%d", &tab [i]) ;
    }

    printf ("Tableau avant\n") ;
    affichage (tab) ;

    /* recherche minimum */
    min = tab [0] ;
    indmin = 0 ;
    for (i = 1 ; i < 10 ; i ++) {
        if (tab [i] < min) {
            min = tab [i] ;
            indmin = i ;
        }
    }
    printf ("Minimum : tab [%d] = %d\n", indmin, min) ;

    /* recherche maximum */
    max = tab [0] ;
    indmax = 0 ;
    for (i = 1 ; i < 10 ; i ++) {
        if (tab [i] > max) {
            max = tab [i] ;
            indmax = i ;
        }
    }
    printf ("Maximum : tab [%d] = %d\n", indmax, max) ;

    return 0 ;
}
```

Question 7-3 Amplitude et moyenne → exercice d'entrainement

Ecrivez un programme qui :

1. initialise un tableau de 10 valeurs réelles
2. affiche l'amplitude du tableau (écart entre le min et le max)
3. affiche la moyenne de ses valeurs.

Programme attendu :

```
#include <stdio.h>
#include <stdlib.h>

void affichage (int tab [10]) {
    int i ;
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("%d ", tab [i]) ;
    }
    printf ("\n") ;
}

void doubleImpair (int tab [10]) {
    int i ;
    for (i = 0 ; i < 10 ; i = i + 1) {
        if ((tab [i] % 2) != 0) {
        tab [i] = 2 * tab [i] ;
        }
    }
}

int main () {
    int i ;
    int tab [10] ;
    int min ;
    int max ;
    int somme ;

    /* On demande a l'utilisateur de remplir les 10 cases du tableau */
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("Quelle valeur pour la case %d ?\n", i) ;
        scanf ("%d", &tab [i]) ;
    }

    printf ("Tableau avant\n") ;
    affichage (tab) ;

    /* recherche minimum */
    min = tab [0] ;
    for (i = 1 ; i < 100 ; i ++) {
        if (tab [i] < min) {
            min = tab [i] ;
        }
    }
    printf ("Minimum : %d\n", min) ;

    /* recherche maximum */
    max = tab [0] ;
    for (i = 1 ; i < 100 ; i ++) {
        if (tab [i] > max) {
        }
    }
    printf ("Maximum : %d\n", max) ;

    printf ("Amplitude : %d\n", max - min) ;

    /* calcul moyenne */
    somme = 0 ;
    for (i = 0 ; i < 100 ; i ++) {
        somme = somme + tab [i] ;
    }
    printf ("Moyenne : %d\n", somme / 100) ;
    return 0 ;
}
```

Écrire une procédure `doubleLesImpairs` qui prend 2 paramètres : un tableau d'entiers et sa taille puis double la valeur de chaque élément **impair** du tableau. Vous appellerez ensuite cette procédure et afficherez le tableau ainsi modifié. Programme attendu :

```
/* Exemple pour tester l'utilisation des tableaux
*/
#include <stdio.h>

void affichage (int tab [10]) {
    int i ;
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("%d ", tab [i]) ;
    }
    printf ("\n") ;
}

void doubleImpair (int tab [10]) {
    int i ;
    for (i = 0 ; i < 10 ; i = i + 1) {
    if ((tab [i] % 2) != 0) {
        tab [i] = 2 * tab [i] ;
    }
    }
}

int main () {
    int tab [10] ; /* un tableau de 10 entiers est initialisee */
    int i ;

    /* On demande a l'utilisateur de remplir les 10 cases du tableau */
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("Quelle valeur pour la case %d ?\n", i) ;
        scanf ("%d", &tab [i]) ;
    }

    printf ("Tableau avant\n") ;
    affichage (tab) ;

    doubleImpair (tab) ;

    printf ("Tableau apres\n") ;
    affichage (tab) ;

    return 0 ;
}
```

Question 7-5 Fonction nbOccurrences

→ exercice d'entrainement

Écrire une fonction `nbOccurrences` qui prend 3 paramètres

1. Un tableau d'entiers
2. Sa taille
3. Une valeur entière quelconque

puis calcule et renvoie le nombre de fois où cette valeur est présente dans le tableau. Programme attendu :

```c
#include <stdio.h>

int nbOccurrences (int table [10], int val) ;
void affichageTableau (int table [10]) ;

int nbOccurrences (int tab [10], int val) {
    int i ;
    int nb = 0 ;

    for (i = 0 ; i < 10 ; i ++) {
        if (tab [i] == val) {
            nb ++ ;
        }
    }
    return nb ;
}

void affichageTableau (int table [10]) {
    int i ;

    /* affichage du tableau */
    for (i = 0 ; i < 10 ; i ++) {
        printf ("table [%d] = %d\n", i, table [i]) ;
    }
}


int main () {
    int i ;
    int casei ;
    int val ;
    int tab [10] ;

    printf ("Valeur a chercher ? ") ;
    scanf ("%d", &val) ;

    for (i = 0 ; i < 10 ; i ++) {
        printf ("%d ieme case du tableau ?", i) ;
        scanf ("%d", &casei) ;
        tab [i] = casei ;
    }

    printf ("Tableau \n") ;
    affichageTableau (tab) ;
    printf ("Nb d'occurrence de %d dans le tableau : %d\n", val, nbOccurrences (tab, val)) ;

    return 0 ;
}
```

Question 7-6 Déplacement d'un personnage → en vue du projet

Soit un système de coordonnées x et y pour lequel vous définirez un tableau à 2 cases. Imaginons que ces coordonnées représentent la position d'un personnage évoluant sur une grille où x varie de 0 à 19 et y de 0 à 19. Ecrivez un programme qui :

1. initialise le personnage aux coordonnées (0, 0).
2. demande à l'utilisateur de saisir un caractère permettant de déplacer le personnage (4=gauche, 6=droite, 8=haut et 2=bas). Reprendre pour cela votre code fait en 3-12

3. modifie les coordonnées du personnage en fonction des saisies de l'utilisateur et affiche les nouvelles coordonnées. Attention, le personnage ne doit pas sortir de la carte. Ainsi, si on lui demande d'aller à droite alors qu'il est déja sur le bord droit de la carte (donc en x = 0), il ne doit pas bouger, et un message d'alerte doit être affiché.
4. répéter ces 2 dernières étapes tant que l'utilisateur n'a pas saisi le caractère 'q' (quitter).

Question 7-7 Unicité dans un tableau

→ pour aller plus loin

Ecrivez un programme qui :

1. initialise un tableau de 10 valeurs entières
2. vérifie qu'ils sont tous différents. On affichera soit la première valeur doublon trouvée, soit un message de succès.

Programme attendu :

```
#include <stdio.h>
#include <stdlib.h>

void affichage (int tab [10]) {
    int i ;
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("%d ", tab [i]) ;
    }
    printf ("\n") ;
}

int main () {
    int tab [10] ; /* un tableau de 10 entiers est initialisee */
    int mem [10] ;
    int idxmem ;
    int i ;
    int j ;

    /* On demande a l'utilisateur de remplir les 10 cases du tableau */
    for (i = 0 ; i < 10 ; i = i + 1) {
        printf ("Quelle valeur pour la case %d ?\n", i) ;
        scanf ("%d", &tab [i]) ;
    }

    affichage (tab) ;

    idxmem = 0 ;
    for (i = 0 ; i < 10 ; i ++) {
        for (j = 0 ; j < idxmem ; j ++) {
            if (tab [i] == mem [j]) {
                printf ("doublon trouve : %d\n", tab [i]) ;
                return 0 ;
            }
        }
        mem [idxmem] = tab [i] ;
        idxmem ++ ;
    }
    printf ("pas de doublons \n") ;
    return 0 ;
}
```

Question 7-8 Prédicat estSymetrique

→ pour aller plus loin

Écrire une fonction `estSymetrique` qui prend 2 paramètres un tableau d'entiers et sa taille, puis teste si le tableau est

un "palindrome numérique", c'est-à-dire si la suite d'éléments du premier au dernier est identique à la suite d'éléments du dernier au premier. Par exemple, { 1, 2, 4, 2, 1 } est un tableau symétrique, mais { 1, 2, 4, 1 } ne l'est pas. La fonction doit renvoyer vrai (1 par exemple) si le tableau est symétrique et faux (0) dans le cas contraire. Programme attendu :

```
#include <stdio.h>

int estSymetrique (int table [10]) ;
void affichageTableau (int table [10]) ;

int estSymetrique (int tab [10]) {
    int i ;

    for (i = 0 ; i < 10 / 2; i ++) {
        if (tab [i] != tab [10 -1 - i]) {
            return 0 ;
        }
    }
    return 1 ;
}

void affichageTableau (int table [10]) {
    int i ;

    /* affichage du tableau */
    for (i = 0 ; i < 10 ; i ++) {
        printf ("table [%d] = %d\n", i, table [i]) ;
    }
}


int main () {
    int i ;
    int tab [10] ;
    int casei ;


    for (i = 0 ; i < 10 ; i ++) {
        printf ("%d ieme case du tableau ?", i) ;
        scanf ("%d", &casei) ;
        tab [i] = casei ;
    }

    printf ("Tableau avant \n") ;
    affichageTableau (tab) ;

    printf ("Le tableau est-il symetrique ? %d\n", estSymetrique (tab)) ;

    return 0 ;
}
```

Question 7-9 Fonction plusGrandeSuite → pour aller plus loin

Attention, cet exercice est sensiblement plus difficile que les précédents.

Écrire une fonction `plusGrandeSuite` qui prend 2 paramètres un tableau d'entiers et sa taille, puis calcule et renvoie la valeur maximale d'une suite d'éléments du tableau. Par exemple, si on considère le tableau { 31 ; -41 ; 59 ; 26 ; -53 ; 58 ; 97 ; -93 ; -23 ; 84}, la suite maximale est { 59 ; 26 ; -53 ; 58 ; 97} et la valeur retournée par la fonction `plusGrandeSuite` sera donc

187, la somme des éléments de ce sous-tableau maximal. Programme attendu :

```
#include <stdio.h>

int plusGrandeSuite (int table [10]) ;
void affichageTableau (int table [10]) ;

int plusGrandeSuite (int tab [10]) {
    int i ;
    int j ;
    int k ;
    int somme ;
    int val_max ;

    somme = tab [0] ;

    for (i = 0 ; i < 10 ; i ++) {
        for (j = i ; j < 10 ; j ++) {
            somme = 0 ;
            for (k = j ; k < 10 ; k ++) {
                somme = somme + tab [k] ;
                if (val_max < somme)
                    val_max = somme ;
            }
        }
    }
    return val_max ;
}

void affichageTableau (int table [10]) {
    int i ;

    /* affichage du tableau */
    for (i = 0 ; i < 10 ; i ++) {
        printf ("table [%d] = %d\n", i, table [i]) ;
    }
}

int main () {
    int i ;
    int casei ;
    int tab [10] ;

    for (i = 0 ; i < 10 ; i ++) {
        printf ("%d ieme case du tableau ?", i) ;
        scanf ("%d", &casei) ;
        tab [i] = casei ;
    }

    printf ("Tableau \n") ;
    affichageTableau (tab) ;
    printf ("Valeur maximale : %d\n", plusGrandeSuite (tab)) ;

    return 0 ;
}
```

2 Validation des compétences acquises à l'issue de cette séance

Je maitrise les compétences demandées à l'issue de cette séance si **je suis capable** de :

- ☐ déclarer un tableau de n cases exactement
- ☐ accéder à la i^{eme} case du tableau pour en afficher le contenu ou modifier sa valeur
- ☐ parcourir successivement toutes les cases du tableau pour y réaliser des actions (en afficher les valeurs ou y placer des valeurs précises)