Tableaux

1 Introduction : Po surveille sa ligne

Exercice 1. Po déclare des tableaux.

Maintenant que vous savez déclarer un tableau, répondez aux questions suivantes.

1) Rédigez l'instruction permettant de déclarer un tableau nommé `tabf`, et contenant 5 valeurs de type `float`. **float tabf[5];**
2) En supposant que la constante `N` vaille 5, rédigez la déclaration d'un tableau nommé `tab`, contenant `N` caractères (du type que vous préférez). **Pas de tableau dynamique /!**
3) Rédigez la déclaration d'un tableau `tabEntiers` de 100 cases dont les 5 premières contiennent les 5 premiers entiers naturels. **int tabEntiers[100] = {1,2,3,4,5};**
4) Rédigez la déclaration d'un tableau contenant les 5 premières lettres de l'alphabet.

Considérons maintenant l'instruction suivante. **char tabLettre[6] = {'a','b','c','d','e'};**

```
float tabFloat[5] = {3.14f};
```

5) Quel est le nom et le type de données du tableau déclaré précédemment ? **des floats**
6) Combien ce tableau comporte-t-il de cases ? **5 cases**
7) Combien d'octets ont été réservés lors de la déclaration de ce tableau ? **4 x 5 = 20 octets**
8) Quelle est la valeur de la case d'indice 0 ? D'indice 1 ? D'indice 5 ? **tab[0] = 3.14**
tab[1] = 0
tab[5] = 0

2 Application : Po se surveille

Exercice 2. Po modifie des tableaux.

1) Rédigez l'instruction permettant d'affecter la valeur -2 à la case d'indice 6 du tableau `tab`. **tab[6] = -2;**
2) En supposant que la variable entière `i` soit définie et initialisée, rédigez l'instruction permettant d'afficher la valeur contenue dans la case d'indice i du tableau `tab`.
3) Rédigez 3 instructions permettant d'afficher les cases 0, 1 et 2 du tableau `T` précédemment défini dans le cours.
4) L'exemple du cours montre comment lire le contenu de la case `i` du tableau `T`. Utilisez ce principe pour afficher l'ensemble des cases du tableau `T` à l'aide d'une instruction de boucle.

Exercice 3. Po manipule des tableaux.

1) Rédigez le prototype de la fonction `afficheTab()` ne retournant rien et prenant en paramètres un tableau d'entiers et sa taille. **void afficheTab(int tab[], int n);**
2) Complétez la fonction `afficheTab()` afin d'afficher l'ensemble des valeurs stockées dans un tableau, puis appelez-la sur le tableau `tab`.

```
void afficheTab(int tab[], int n) {
    for (int i = 0; i < n; i++) printf("%d ", tab[i]);
}
```

2) En supposant que la variable entière `i` soit définie et initialisée, rédigez l'instruction permettant d'afficher la valeur contenue dans la case d'indice i du tableau `tab`.

```
printf("%d\n", tab[i]);
```

3) Rédigez 3 instructions permettant d'afficher les cases 0, 1 et 2 du tableau `T` précédemment défini dans le cours.

```
for (int i = 0; i < 3; i++) printf("%d ", tab[i]);
```

3) Rédigez une fonction permettant de calculer la moyenne des éléments d'un tableau.

```
int moyenneTab(int tableau[], int n) {
    int moyenne = sommeTab(tableau, n) / n;
    return moyenne;
}
```

Exercice 4. Po remplit des tableaux.

Po a encore besoin d'un coup de main, mais ne veut pas se rendre ridicule devant Jo. L'ibijau lui a demandé de compter la quantité de nourriture ingérée chaque jour sur 15 jours. Farceur, Po a décidé de remplir ce tableau de valeurs aléatoires. Au cas où, il souhaite également en remplir un autre avec des vraies valeurs. Pouvez-vous l'aider en créant deux fonctions réalisant ce que Po souhaite ?

Les deux tableaux doivent être de type entier et contenir 15 cases. Le premier doit être remplit de valeurs aléatoires comprises entre 100 et 2500. Le second doit être remplit par le biais de l'utilisateur et d'appels à la fonction `scanf()`.

```
int fillRandomTab (int tab[], int n, int inf, int sup) {
    srand(time(NULL));
    for (int i = 0; i < n; i++) {
        tab[i] = (rand() % ((sup - inf +1) + inf));
    }
    return 0;
}
```

```
void remplTab(int tableau[], int n) {
    for (int i = 0; i < n; i++) scanf("%d", tab[i]);
    return;
}
```

3) Rédigez une fonction permettant de calculer la moyenne des éléments d'un tableau.

Exercice 4. Po remplit des tableaux.

Po a encore besoin d'un coup de main, mais ne veut pas se rendre ridicule devant Jo. L'ibijau lui a demandé de compter la quantité de nourriture ingérée chaque jour sur 15 jours. Farceur, Po a décidé de remplir ce tableau de valeurs aléatoires. Au cas où, il souhaite également en remplir un autre avec des vraies valeurs. Pouvez-vous l'aider en créant deux fonctions réalisant ce que Po souhaite ?

Les deux tableaux doivent être de type entier et contenir 15 cases. Le premier doit être remplit de valeurs aléatoires comprises entre 100 et 2500. Le second doit être remplit par le biais de l'utilisateur et d'appels à la fonction `scanf()`.

3 Tableaux 2D : Po a faim

Exercice 5. Po pratique les tableaux 2D.

1) Déclarez une matrice nommée `tab`, comportant 3 lignes et 7 colonnes.
2) Déclarez et initialisez une matrice nommée `mat`, comportant 2 lignes et 4 colonnes, et contenant sur la première ligne les valeurs 10, 20, 30 et 40, et sur la seconde ligne les valeurs -10, -20, -30 et -40.
3) Modifiez l'élément de la ligne 0 de la colonne 5 du tableau `tab` pour qu'il vaille -1.
4) Combien la matrice `tab` comporte-t-elle d'éléments ?

Exercice 6. Po pratique la sorcellerie.

Réalisez tout d'abord cet exercice d'entrainement avant de pratiquer la magie noire !

1) Réalisez la fonction `void AfficheMatrice (int mat[][10], int n)` permettant d'afficher l'ensemble des valeurs de la matrice `mat` comportant `n` lignes et 10 colonnes. Un retour à la ligne est nécessaire après l'affichage de chaque ligne de la matrice.
2) Créez une fonction permettant de remplir de façon aléatoire une matrice d'entiers de `n` lignes et 10 colonnes.

Vous êtes alors prêts à passer du côté obscur de la force... Vous devez aider Po dans sa tâche en réalisant la fonction demandée par Jo en vous servant de ses indication présentées dans les questions suivantes.

Rappelons que l'objectif est de préparer le menu de Po sur les 15 prochains jours. La matrice doit lui indiquer quels aliments manger dans la journée, sans dépasser un score de cent points. Vous pourrez vérifier votre code en affichant la matrice après chaque question.

3) Définissez le tableau `pointsAliments` permettant de stocker le nombre de points associé à chaque aliment présentés dans le cours.
4) Créez la fonction possédant le prototype suivant : `int creerMenu (int mat[][10], int n)`. Dans un premier temps, cette fonction doit remplir la matrice `mat` de `n` lignes avec la valeur 1 dans la première case, puis ajouter 1 à chaque case suivante. Rappelons qu'une matrice se parcourt ligne par ligne. Ne vous souciez pas du nombre de points total par jour.
5) Modifiez votre fonction de remplissage pour sélectionner au hasard un seul aliment à consommer dans la journée. Ajoutez pour cela les arguments suivants à votre fonction : `int pointsAliments[], int nbAlim` donnant respectivement le nombre de points de chaque aliment et la taille de ce tableau.

Exercice 5. Po pratique les tableaux 2D.

1) Déclarez une matrice nommée `tab`, comportant 3 lignes et 7 colonnes.
2) Déclarez et initialisez une matrice nommée `mat`, comportant 2 lignes et 4 colonnes, et contenant sur la première ligne les valeurs 10, 20, 30 et 40, et sur la seconde ligne les valeurs -10, -20, -30 et -40.
3) Modifiez l'élément de la ligne 0 de la colonne 5 du tableau `tab` pour qu'il vaille -1.
4) Combien la matrice `tab` comporte-t-elle d'éléments ?

6) Nous allons proposer à Po un menu aussi équilibré que possible. Modifiez maintenant votre fonction de la façon suivante :
   6.1) Ajoutez le paramètre suivant à votre fonction : `int maxPoints` donnant le nombre maximum de points à respecter dans la journée.
   6.2) Pour chaque journée, sélectionnez un aliment au hasard. Ajoutez-le à la liste des aliments du jour en incrémentant la case correspondant à l'aliment et recommencez tant que les `maxPoints` points ne sont pas dépassés.