**Les Tableaux**

1. **Définition et déclaration**

Un tableau est une collection d’objets de donnés de même type occupant des espaces mémoires contiguës (les uns à côté des autres) et repérés par des indices (le rang de l’élément dans le tableau).

Un tableau est défini par un nom, une taille et une dimension.

**Le nom** : est un identificateur comme celui des variables composé d’une suite de lettres, de chiffres et le caractère ‘\_’;

**La taille** : est le nombre maximum d’éléments que peut contenir un tableau. Si la taille d’un tableau est 10, ce tableau peut contenir de 0 jusqu’à 10 éléments.

**La dimension** : les tableaux peuvent être d’une seule dimension (tableaux vecteurs ou unidimensionnels) ou de deux dimensions (tableaux matrices ou bidimensionnels).

* Un tableau vecteur est composé d’une seule ligne et de plusieurs colonnes.
* Un tableau matrice est composé de plusieurs lignes et de plusieurs colonnes.

**Déclaration d’un tableau vecteur :**

* **Algo** : nomTableau : **tableau** [taille] **de** type
* **Langage C** : type nomTableau[taille] ;
* **Langage PHP** : $nomTableau = array ();

**Déclaration d’un tableau matrice :**

* **Algo** : nomTableau : **tableau** [nbLignes][nbcolonnes] **de** type
* **Langage C** : type nomTableau [nbLignes][nbcolonnes];
* **Langage PHP** : $nomTableau = array ();

L’accès à un élément se fait en spécifiant son rang ou sa position dans le tableau.

* Tableau Vecteur : nomTableau[indice\_élément]
* Tableau Matrice : nomTableau[indiceLigne][indiceColonne]

Le premier élément d’un tableau vecteur en Algo se trouve à la position 1 et en C & PHP, il se trouve à la position 0. Le dernier élément se trouve en algo à la position taille, et en C & PHP, il se trouve à la position (taille-1).

Le premier élément d’un tableau matrice en Algo se trouve à la position (1,1) et en C & PHP, il se trouve à la position (0,0). Le dernier élément se trouve en algo à la position (nbLignes, nbColonnes), et en C & PHP, il se trouve à la position (nbLignes-1, nbColonnes-1).

1. **Algorithmes élémentaires (tableaux vecteurs)**

**2.1. Stockage des éléments**

**Principe :** On parcourt le tableau case par case et pour chaque case visitée, on lui affecte la valeur saisie par l’utilisateur.

Exemple : remplir un tableau de 10 entiers.

**Algo** : remplissage

**Déclaration**

 tab : tableau [10] de entier

 i : entier

**Début**

 Pour i allant de 1 à 10 faire

 Afficher (“Donner un élément :”)

 Saisir ( tab[i] )

 Fin Pour

 **Fin** remplissage

 **2.2. Affichage des éléments**

**Principe** : on parcourt le tableau case et par case et à chaque case visitée, on affiche le contenu de la case.

Exemple : afficher les éléments d’un tableau de 10 entiers

**Algo** : Affichage

**Déclaration**

 tab : tableau [10] de entier

 i : entier

**Début**

 //on suppose que le tableau est rempli

 Pour i allant de 1 à 10 faire

 Afficher (“Elément :”, tab[i] )

 Fin Pour

 **Fin** Affichage

**2.3 La recherche séquentielle**

**Principe** : Comment rechercher une valeur dans un tableau : on parcourt le tableau case par case et on compare le contenu de la case avec la valeur à rechercher. S’il y a égalité, on arrête la recherche. Sinon on continue les comparaisons jusqu’à la fin du tableau.

Exemple : rechercher une valeur dans un tableau de 10 entiers.

**Algo** : Recherche

**Déclaration**

 tab : tableau [10] de entier

 i, valeur : entier

 trouve : booléen

**Début**

 // remplissage du tableau

 Pour i allant de 1 à 10 faire

 Afficher (“Donner un élément :”)

 Saisir ( tab[i] )

 Fin Pour

 //saisie de la valeur à rechercher

 Afficher (“Donner la valeur à rechercher :”)

 Saisir (valeur)

 //recherche séquentielle

 trouve <- faux

 i <- 1

 Tant que i <= 10 ET trouve = faux faire

Si tab[i] = valeur

Alors trouve <- vrai

Sinon i <- i + 1

Fin si

 Fin Tant que

 Si trouve = vrai

 Alors Afficher (“La valeur existe dans le tableau à la case num : “, i)

 Sinon Afficher (“La valeur n’existe pas dans le tableau “)

 Finsi

 **Fin** Recherche

**2.4. Les Algos de tri**

1. **Série d’exercices**

3.1. Écrire un algorithme et un programme C qui permet de stocker dans un tableau 10 prix et les affiche.

Algo : exo1

Déclaration

 tab : tableau[10] de réel

 i : entier

Début

 Pour i allant de 1 à 10 faire

 Afficher (“Donner un prix :”)

 Saisir (tab[i])

 Fin Pour

 Afficher (“Voici la liste des prix :”)

 Pour i allant de 1 à 10 faire

 Afficher (“Prix : “, tab[i] )

 Fin Pour

Fin Exo1

#include <stdio.h>

int main () {

 float tab[10];

 int i ;

 for (i = 0 ; i < 10 ; i++){

 printf("Donner un prix :");

 scanf("%f", &tab[i]);

 }

 printf("\n Voici la liste des prix :");

 for (i = 0; i < 10 ; i++){

 printf("\n Prix %d : %f", i, tab[i]);

 }

 return 0;

}

**Exercice 2** : Ecrire un algorithme et un programme C qui permet de saisir un tableau de 10 entiers, demande une valeur et affiche le nombre d’occurrences de la valeur dans le tableau.

Algo : Exo2

Déclaration

 tab : tableau[10] de entier

 i , valeur, occ : entier

Début

 Pour i allant de 1 à 10 faire

 Afficher (“Donner un entier :”)

 Saisir (tab[i])

 Fin Pour

 Afficher (“Donner la valeur à rechercher :”)

 Saisir (valeur)

occ <- 0

Pour i allant de 1 à 10 faire

 Si tab[i] = valeur

 Alors occ <- occ + 1

 Fin si

Fin Pour

Afficher (“La valeur apparait “, occ, “ fois”)

 Fin exo2

#include <stdio.h>

int main () {

 int tab[10];

 int i, valeur, occ ;

 for (i = 0 ; i < 10 ; i++){

 printf("Donner un entier :");

 scanf("%d", &tab[i]);

 }

 printf("Donner une valeur : ");

 scanf("%d", & valeur);

 occ = 0;

 for (i = 0 ; i < 10 ; i++){

 if (tab[i] == valeur){

 occ = occ + 1 ;

 }

 }

 printf("La valeur apparait %d fois", occ);

 return 0;

}

Exercice 3 : Ecrire un programme qui permet de saisir 10 notes, détermine la note moyenne, la note max et la note min du tableau et les affiche.

 Algo : Exo3

 Déclaration

 tab : tableau [10] de réel

 i : entier

 somme, moy, min , max : réel

 Début

 Pour i allant de 1 à 10 faire

 Afficher (“Donner une note :”)

 Saisir (tab[i])

 Fin Pour

somme <- 0

Pour i allant de 1 à 10 faire

 somme <- tab[ i ] + somme

Fin pour

moy <- somme / 10

Afficher (“La moyenne est de : “ , moy )

min <- 20

max <- 0

Pour i allant de 1 à 10 faire

Si tab[i] < min

Alors min <- tab[i]

Fin si

Si tab[i] > max

Alors max <- tab[i]

Fin si

Fin pour

Afficher (“ la note min est de :”, min)

Afficher (“ la note max est de :”, max)

Fin exo3

#include <stdio.h>

int main () {

 float tab[10];

 int i;

 float somme, moy, min, max ;

 for (i = 0 ; i < 10 ; i++){

 printf("Donner une note :");

 scanf("%f", &tab[i]);

 }

 somme = 0;

 for (i = 0 ; i < 10 ; i++){

 somme = tab[i] + somme ;

 }

 moy = somme / 10;

 printf("La moyenne est de : %f",moy );

 min = 20;

 max = 0;

 for (i = 0 ; i < 10 ; i++){

 if (tab[i] < min){

 min = tab[i];

 }

 if (tab[i] > max){

 max = tab[i];

 }

 }

 printf("La note min est de : %f", min);

 printf("La note max est de : %f", max);

 return 0;

}