Séance 11 : Jeudi 14/03/2019

Exercice tri par comptage TP3

Tri par comptage : le tri par comptage consiste pour chaque élément du tableau à compter combien d'éléments sont plus petits que lui ; grâce à ce chiffre on connaît sa position dans le tableau résultat. Soit le tableau initial suivant :

Tableau initial	52	10	1	25	62	3	8	55
Tableau comptage	5	3	0	4	7	1	2	6
Tableau résultat	1	3	8	10	25	52	55	62

Écrire l'algorithme d'un sous-programme permettant de trier un tableau de 10 entiers **distincts** en utilisant la méthode décrite précédemment.
Le **tableau initial** est fourni en paramètre d'entrée, le **tableau de comptage** est calculé dans le sous-programme et permet de remplir et renvoyer le **tableau résultat**.

```
#include <iostream>
#include<math.h>
using namespace std;
const int MAX = 20 ;
void remplir_1D (int tab[MAX], int n)
{
    int i;
    for (i=0;i<n;i++)
    {
        cout<<"donnez la valeur"<<endl;
        cin>>tab[i];
    }
}
void afficher_1D (int tab[MAX], int n)
{
    int i;
    for (i=0;i<n;i++)
    {
        cout<<tab[i]<<" | ";
    }
    cout<<endl;
}

void tri_comptage (int tab_init[MAX], int tab_trie[MAX], int n)
{
    int tab_compt[MAX];
    int i,j;

    for (i=0;i<n;i++)
```

```
    {
        tab_compt[i] = 0;
        for (j=0;j<n;j++)
        {
            if (tab_init[j]<tab_init[i])
            {
                tab_compt[i]++;
            }
        }
    }
    //afficher_1D(tab_compt,n);
    for (i=0;i<n;i++)
    {
        tab_trie[tab_compt[i]] = tab_init[i];
    }
}
int main (void)
{
   int T[MAX]/*={52,10,1,25,62,3,8,55}*/,nb, Ttrie[MAX];
   cout <<"Combien de valeurs dans le tableau ?"<<endl;
   cin>>nb;
   remplir_1D(T,nb);
   afficher_1D(T,nb);
   tri_comptage(T,Ttrie,nb);
   afficher_1D(Ttrie,nb);

    return 0;
}
```

CM 5 : Tableaux 2D, nD

Exercices du TD6

1. Soit T un tableau 2D carré de taille 5*5 contenant des entiers. Écrire la déclaration et l'initialisation à 0 d'une telle structure de données.

```
Déclaration : T : tableau[5][5] d'entiers
Procédure InitTab(T : tableau[5][5] d'entiers)
Donnée / Résultat : T
Variable locales : i,j : entier
Début
     Pour i allant de 0 à 4 par pas de 1 faire
          Pour j allant de 0 à 4 par pas de 1 faire
               T[i][j]←0
          Fin Pour
     Fin Pour
Fin InitTab
```

2. Écrire un sous-programme `RemplirTab` qui propose à l'utilisateur de remplir un tableau T de taille 5*5.

5	1	8	6	0
6	9	7	4	2
1	1	0	9	7
4	5	7	3	0
0	2	5	0	9

```
Procédure RemplirTab(T : tableau[5][5] d'entiers)
Donnée / Résultat : T
Variable locales : i,j : entier
Début
     Pour i allant de 0 à 4 par pas de 1 faire
          Pour j allant de 0 à 4 par pas de 1 faire
               Afficher (« donnez la valeur »)
               Saisir(T[i][j])
          Fin Pour
     Fin Pour
Fin remplirTab
```

3. Écrire deux procédures d'affichage d'un tableau 2D de taille 5*5
   1. `Affichage_2D_ligne` : qui affichera le tableau ligne par ligne

```
Procédure Affiche_2D_ligne(T : tableau[5][5] d'entiers)
Donnée / Résultat : T
Variable locales : i,j : entier
Début
     Pour i allant de 0 à 4 par pas de 1 faire            / ligne
          Pour j allant de 0 à 4 par pas de 1 faire       // colonne
               Afficher (T[i][j])
          Fin Pour                              // fin colonne
          Afficher (saut de ligne)
     Fin Pour
Fin Affiche_2D_ligne
```

   2. `Affichage_2D_colonne` : qui affichera le tableau colonne par colonne

```
Procedure Affiche_2D_colonne(T : tableau[5][5] d'entiers)
Donnée / Résultat : T
Variable locales : i,j : entier
Debut
     Pour i allant de 0 à 4 par pas de 1 faire            // colonne
          Pour j allant de 0 à 4 par pas de 1 faire       // ligne
               Afficher (T[j][i])
          Fin Pour                                  // fin ligne
          Afficher (saut de ligne)
     Fin Pour
Fin Affiche_2D_colonne
```

4. Écrire trois fonctions permettant sur un tableau 2D de taille 5*5:
   1. de calculer la somme des éléments d'une ligne (le numéro de la ligne étant passé en paramètre)

```
Fonction SommeLigne(T : tableau[5][5] d'entiers, ligne : entier) :
entier
Donnée / Résultat: T
Donnée : ligne : numéro de la ligne dont on veut calculer la somme
Résultats : somme des éléments de la ligne "ligne"
Variables locales : i,som_lig : entier
Debut
som_lig ← 0
Pour i allant de 0 à 4 par pas de 1 faire
som_lig ← som_lig + T[ligne][i]
Fin Pour
Retourner (som_lig)
Fin SommeLigne
```

2. de calculer la somme des éléments d'une colonne (le numéro de la colonne étant passé en paramètre)

```
Fonction SommeColonne(T : tableau[5][5] d'entiers, colonne :
entier) : entier
Donnée / Résultat: T
Donnée : colonne : numéro de la colonne dont on veut calculer la
somme
Résultats : somme des éléments de la colonne "colonne"
Variables locales : i,som_col : entier
Debut
som_col ← 0
Pour i allant de 0 à 4 par pas de 1 faire
som_col ← som_col + T[i][colonne]
Fin Pour
Retourner (som_col)
Fin SommeColonne
```

3. de calculer les sommes des éléments de chaque diagonale (dans la mesure où le tableau est bien carré)

```
Fonction SommeDiagonale(T : tableau[5][5] d'entiers, som_diag2 :
entier) : entier
Précondition : le tableau T est carré
Donnée / Résultat: T, som_diag2
Résultats : somme des éléments de la première diagonale
Variable slocales : i,som_diag1 : entier
Debut
som_diag ← 0
som_diag2 ← 0
     Pour i allant de 0 à 4 par pas de 1 faire
som_diag1 ← som_diag1 + T[i][i]
som_diag2 ← som_diag2 + T[i][4-i]

     Fin Pour
     Retourner (som_diag1)
Fin
```

5. Écrire un sous-programme `RecherchePlusGrand` permettant de rechercher le plus grand élément de ce tableau et de retourner l'indice de ligne et l'indice de colonne correspondant à cet élément ainsi que l'élément lui-même.

```
Fonction RecherchePlusGrand(T : tableau[5][5] d'entiers, lig_max :
entier, col_max : entier) : entier
Donnée / Résultat: T, lig_max et col_max
Résultat : valeur du plus grand élément de T
Variable locales : i,j,maxi : entier
Debut
     maxi ← T[0][0]
lig_max  ← 0
col_max ← 0
     Pour i allant de 0 à 4 par pas de 1 faire
           Pour j allant de 0 à 4 par pas de 1 faire
                 Si (maxi< T[i][j]) alors
maxi←T[i][j]
lig_max ←i
col_max ← j
                 Fin Si
           Fin Pour
```

```
     Fin Pour
     Retourner maxi
Fin RecherchePlusGrand
```

Traduction en C/C++ des exercices du TD6

```
#include <iostream>
#include<math.h>
using namespace std;

void init_2D(int T [3][3]){
    int i,j;
    cout<<"initialisation du tableau 2D"<<endl;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            T[i][j]= 0;
        }
    }
}
void remplit_2D(int T [3][3]){
    int i,j;
    cout<<"remplissage du tableau 2D"<<endl;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            cout<<"donnez la valeur de la case "<<i<<" , "<<j<<endl;
            cin>>T[i][j];
        }
    }
    cout<<endl;
}
void affiche_ligne_2D(int T [3][3]){
    int i,j;
    cout<<"affichage ligne par ligne"<<endl;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            cout<<T[i][j]<<" | ";
        }
        cout<<endl;
    }
}
void affiche_colonne_2D(int T [3][3]){
    int i,j;
    cout<<"affichage colonne par colonne"<<endl;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            cout<<T[j][i]<<" | ";
        }
        cout<<endl;
    }
}
int somme_ligne (int T [3][3], int indl){
    int som = 0, i;
```

```
    for (i=0;i<3;i++)
    {
        som+=T[indl][i];
    }
    return som;
}
int somme_colonne (int T [3][3], int indc){
    int som = 0, i;
    for (i=0;i<3;i++)
    {
        som+=T[i][indc];
    }
    return som;
}
void somme_diagonales (int T[3][3], int &sd1, int &sd2){
    int i;
    sd1=0;
    sd2=0;
    for (i=0;i<=2;i++)
    {
        sd1 += T[i][i];
        sd2 += T[i][2-i];
    }
}
void recherche_plus_grand(int T[3][3], int & lig, int &col){
    int i,j;
    lig = 0;
    col = 0;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            if (T[i][j]>T[lig][col])
            {
                lig = i;
                col = j;
            }
        }
    }
}
int main (void)
{
    int tab[3][3]{7,2,9,7,8,4,4,5,6};
    int lig,col, d1,d2 ;
    affiche_ligne_2D(tab);
    recherche_plus_grand(tab, lig,col);
    cout<<"La plus grande valeur se trouve dans la case
["<<lig<<","<<col<<"] et vaut : "<<tab[lig][col]<<endl;
/*    do
    {
        cout<<"quel est l'indice de la ligne ?"<<endl;
        cin>>lig;
    } while (lig<0 || lig>=3);
    cout<<"la somme des valeurs de la ligne "<<lig<<" est :
"<<somme_ligne(tab,lig)<<endl;
    cout<<"la somme des valeurs de la colonne "<<lig<<" est :
"<<somme_colonne(tab,lig)<<endl;
    somme_diagonales(tab, d1,d2);
    cout<<"Diagonale 1 : "<<d1<<" et diagonale 2 : "<<d2<<endl;
    affiche_ligne_2D(tab);
    init_2D(tab);
    affiche_ligne_2D(tab);
    remplit_2D(tab);
```

```
    affiche_ligne_2D(tab);
    affiche_colonne_2D(tab);*/

    return 0;
}
```