SOLUTION de TD N°1 : LES SOUS PROGRAMMES
Fonction et procédures (Partie 02)

Exercice 15 : Remplir et Afficher une Matrice

Soit A une matrice contenant des valeurs entières avec n lignes et m colonnes.

1. Ecrire une procédure ***Lire_Mat*** qui permet de remplir – ou lire- la matrice A.
2. Ecrire une procédure ***Ecrire_Mat*** qui permet d'afficher – ou ecrire- le contenu de la matrice A.

Solution :

```
Type matrice : Tableau [1..100,1..100] de entier ;
```

1. Procédure Lire_Mat:

```
Procédure Lire_Mat (n, m: entier ; VAR A : matrice);
Var i, j: entier ;

Début
   //1. Remplir la matrice A
   Pour i de 1 à n Faire
      Pour j de 1 à m Faire
         Lire(A[i,j]);

Fin;
```

2. Procédure Ecrire_Mat:

```
Procédure Ecrire_Mat (A : matrice ; n, m: entier);
Var i, j: entier ;

Début
   //1. Afficher le contenu de la matrice A
   Pour i de 1 à n Faire
      Pour j de 1 à m Faire
         Ecrire(A[i,j]);

Fin;
```

Exercice 16 : Rotation de 90° d'une Matrice (Examen ASD-II 2020-2021)

Soit A une matrice carrée de dimension n* n contenant des valeurs entières.

3. Ecrire un algorithme qui effectue une rotation de 90° à droite. Cela est réalisé en 2 étapes :
   a. Calculer la transposée de la matrice A, ensuite
   b. Echanger les colonnes (première avec dernière, deuxième avec avant dernière, ...)

Exemple : A

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Calculer *transposé* →

Transposé de A

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

échanger les colonnes →

Résultat de rotation

13	9	5	1
14	10	6	2
15	11	7	3
16	12	8	4

Solution :

1. Procédure Transpose:

```
Procédure Transpose (A : matrice ; n: entier ; VAR B : matrice);
Var i, j: entier ;
Début
   Pour i de 1 à n Faire
      Pour j de 1 à n Faire
         B[i,j] ← A[j,i];

Fin;
```

2. Procédure Echange_Colonnes:

```
Procédure Echange_Colonnes (A : matrice ; n: entier ; VAR B : matrice);
Var i, j: entier ;
Début
    Pour i de 1 à n Faire
       Pour j de 1 à n Faire
          B[i,j] ← A[i,n-j+1];
Fin;
```

3. Algorithme Principal :

```
Algorithme rotation_90;
Type matrice : Tableau [1..100,1..100] de entier ;
Var A,B,C : matrice;
      i, j, n: entier ;
Procédures Lire_Mat, Ecrire_Mat, Transpose, Echange_Colonnes ;
Début
    Lire(n);
    //1. Remplir la matrice A
    Lire_Mat(n, n, A) ;

    //2. Calculer la transposé de A
    Transpose(A, n, B) ;

    //3. Echanger les colonnes de A
    Echange_Colonnes(B, n ,C) ;

    //4. Afficher la matrice Après rotation
    Ecrire_Mat(A, n, n) ;

Fin.
```

Exercice 17 : 1ère élément non répétitif

Soit **T** un tableau à une dimension de **n** valeurs entières (avec n <1000)

1. Ecrire une fonction ***Occurrences (T, n, Val)*** qui retourne le nombre d'occurrences (de répétitions) de la valeur **Val** dans le tableau **T**.
2. On utilisant la fonction précédente, Ecrire un algorithme qui permet de retourner le 1er élément non répétitif dans le tableau T.

Exemple : Le 1er élément non répétitif est : **8**

2	2	3	8	6	6	7	3	3	2	9	9

Solution :

1. Fonction Occurrences:

```
Fonction Occurances(T:tableau[1..1000] d'entier ; n, Val : entier) :entier ;
 Var i, cpt : entier ;

Début
    cpt ← 0;

    //Parcourir le tableau et compter les occurrences
    Pour i de 1 à n Faire
       Si (T[i] = Val) Alors
          cpt ← cpt + 1;
       Fsi
    Fpour

    Occurances ← cpt
Fin.
```

2. **Fonction Algorithme Principal :**

```
Algorithme non_repititif;
 Var   T : tableau[1..1000] d'entier ;
       n, i : entier ;
 Fonctions   Occurances ;

Début
   //Remplir le tableau
   Lire(n) ;
   Pour i de 1 à n Faire Lire(T[i]) ;

   //Parcourir le tableau et trouver l'élément avec « 1 » occurrence
   i ← 0;
   Répéter
      i ← i + 1;
   Jusqu'à (i > n ou Occurances(T, n, T[i]) = 1)

   Ecrire('le premier élément non répétitif est', T[i] ) ;
Fin.
```

Exercice 18 : (Examen PSD S2 2013/2014)

Soit T un tableau à une dimension de **N** valeurs entières (N <= 100)

1. Ecrire une fonction **Tab_Rech (T, N, val)** qui permet de rechercher une valeur entière **val** dans le tableau **T** et retourne sont indice dans le tableau **T** si elle existe (retourne l'indice de la première occurrence) sinon, retourne une valeur négative.
2. Ecrire une procédure **Tab_sans_double (T1, N, T2)**, en utilisant la fonction précédente, qui permet de construire un tableau **T2** qui ne contient qu'une seule occurrence de chaque nombre dans **T1** de **N** éléments.

Solution :

```
Type Tab = tableau [1..100] d'entier ;
```

3. **Fonction Tab_Rech**

```
Fonction Tab_Rech (T : Tab ; N, val : entier) : entier
    Var i: entier ;
        Trouve : booléen ;
Debut
    Trouve := faux ;
    TQ (i <= N) et (Trouve = faux) Faire
       DTQ
           Si (T[i] = val) Alors Trouve := vrai ;
           Sinon    i := i + 1 ;
       FTQ

    Si (Trouve = vrai) alors     Tab_Rech := i ;
    Sinon                        Tab_Rech := -1 ;
Fin.
```

4. **Procédure Tab_sans_double**

```
Procedure Tab_sans_double (T1 : Tab ; N: entier, VAR T2 : Tab)
    Var i, j: entier ;
Debut
    j:= 0 ;
    Pour i := 1 à N Faire
       DPour
           Si (Tab_Rech(T2, j, T1[i]) < 0 ) alors
              DSi
                     j:= j + 1 ;
                     T2 [j] := T1 [i] ;
              FSi
       FPour
Fin.
```