Exercice 1

Ecrire les actions paramétrées (procédure ou fonction) permettant de résoudre les problèmes suivants :

1- Calcul de la factorielle de N (N ! = 1x2x3x…xN, avec 0 !=1).
2- Calcul de la somme S=1+2+... +N
3- Calcul du Maximum entre deux entier A et B.
4- Calcul de la puissance nième (n>=0) d’un nombre réel X positif non nul.
5- Calcul du nombre de chiffres pairs dans un entier N.
6- Calcul du quotient et du reste de la division entière d’un entier A par un entier B.

1- Calcul de la factorielle de N (N ! = 1x2x3x…xN, avec 0 !=1).

```
Fonction      Factoriel(N:Entier):Entier
Var
   i, fact: Entier;
Debut
   fact <-- 1;
   Pour i <--1 a N
   Faire
       fact <-- fact * i;
   Fait
   Retourner fact;
Fin;
```

2- Calcul de la somme S=1+2+... +N

```
Fonction      SommeN(N:Entier):Entier
Var
   i, S: Entier;
Debut
   S <-- 0;
   Pour i <--1 a N
   Faire
       S <-- S + i;
   Fait

   Retourner S;
Fin;
```

3- Calcul du Maximum entre deux entiers A et B.

```
Fonction      Max2Ent(A, B:Entier):Entier
Var
   max: Entier;
Debut
   Si (A > B) alors
      max <-- A;
   Sinon
      max <-- B;
   Fsi

   Retourner max;
Fin;
```

4- Calcul de la puissance nième (n>=0) d’un nombre réel X positif non nul.

```
Fonction    Puissance(X:Reel, N:Entier):Reel
Var
   i: Entier;
   p : Reel;
Debut
   p <-- 1;
   Pour i <--1 a N
   Faire
```

```
      p <-- p * X;
   Fait

   Retourner p;
Fin;
```

5- Calcul du nombre de chiffres pairs dans un entier N.

```
Fonction     NbrChiffresPairs(X: Entier) : Entier
Var
   nbPair : Entier
Debut
   nbPair <-- 0;
   Tant Que (X <> 0)
   Faire
      Si ((X mod 10) mod 2 = 0) alors
         nbPair <-- nbPair + 1;
      Fsi
      X <--- X Div 10;
   Fait

   Retourner nbPair;
Fin;
```

6- Calcul du quotient et du reste de la division entière d'un entier A par un entier B.

```
Procedure    CalculRestQuot(E/ X, Y: Entier, S/ Rest, Quot: Entier)
Debut
   Rest <--- X  ;
   Quot <--- 0;
   Tant Que (Rest >= Y)
   Faire
      Quot <--- Quot + 1;
      Rest <--- Rest - Y;
   Fait
Fin;
```

Exercice 2

1- Ecrire une AP Permute permettant de permuter deux caractères.
2- Soit CH une chaine de caractère. En utilisant l'action précédente, écrire un algorithme permettant d'inverser la chaine CH.

```
Procedure    Permuter2Cars(E/S C1, C2: Caractere)
Var
   Temp: Caractere;
Debut
   Temp <-- C1;
   C1 <-- C2;
   C2 <-- Temp;
Fin;

Algorithme     InverserChaine;
Var
  ch: Chaine;
  i, J: Entier;
Procedure    Permuter2Cars(E/S C1, C2: Caractere)
  ...
Fin;
Debut
  Ecrire("Donner une chaine de caracteres");
  Lire(ch);
  J <-- Taille(ch);
```

```
  i <-- 1;
  Tant Que (i < J)  Faire
      Permuter2Cars(ch[i], ch[J]);
      i <-- i + 1;
      J <-- J - 1;
  Fait
  Ecrire("Chaine ch Inversee:", ch);
Fin.
```

Exercice 3

1- Ecrire deux fonctions permettant de calculer respectivement le PGCD et le PPCM de deux entiers naturels non nuls.
2- Soit T un tableau de N entiers naturels non nuls, (2<=N<=50). En utilisant les fonctions précédentes, écrire un algorithme permettant de :
 - Afficher le PGCD et le PPCM des éléments de T.
 - Afficher tous les couples premiers entre eux de T.

```
Fonction    PGCD(A, B:Entier):Entier
Var
   Temp: Entier;
Debut
   // pgcd(a, b) = pgcd(b, a mod b)
   // pgcd(a, 0) = a
   Tant Que (B <> 0)
   Faire
      Temp <--- B;
      B <--- A mod B;
      A <--- Temp;
    Fait

    Retourner  A;
Fin;

Fonction     PPCM(A, B: Entier):Entier
Debut
     // On utilise la propiete: ppcm(a,b) * pgcd(a,b) = a * b
     Retourner ((A * B) Div PGCD(A, B));

Fin;

Algorithme    TabPremiersEntre;
Var
   T: Tableau[1..50] de Entier;
   i, J, pgcdT, ppcmT: Entier;

Fonction PGCD(A, B: Entier):Entier
   ...
Fin;
Fonction PPCM(A, B: Entier):Entier
   ...
Fin;
Debut
   Repeter
      Ecrire("Donner N compris entre 2 et 50");
      Lire(N);
   Jusqu'a ((N >= 2) et (N <= 50));
   Pour i <-- 1 a N Faire
       Ecrire("Donner ", i, " eme valeur de T");
       Lire(T[i]);
   Fait
```

```
  pgcdT <-- T[1];
  ppcmT <-- T[1];
  Pour i <-- 2 a N Faire
      pgcdT <-- PGCD(pgcdT, T[i]);
      ppcmT <-- PPCM(ppcmT, T[i]);
  Fait
  Ecrire("PGCD des elements de T est: ", pgcdT);
  Ecrire("PPCM des elements de T est: ", ppcmT);

  Pour i <-- 1 a (N-1) Faire
    Pour J <-- (i + 1) a N Faire
      Si (PGCD(T[i], T[J]) = 1) alors
        Ecrire(T[i], " et ", T[J], " sont Premiers entre eux");
      Fsi
    Fait
   Fait
Fin.
```

Exercice 4

1- Ecrire une fonction Miroir permettant de renvoyer le miroir d’un entier naturel. (exemple : Miroir(23568)=86532)
2- Ecrire une procédure IntFrac permettant de calculer la partie entière et la partie fractionnaire d’un nombre réel. (exemple : pour X=235.2601, partie entière = 235, partie fractionnaire=0.2601)
3- Ecrire une procédure Fexpo permettant de transformer une partie fractionnaire sous forme exponentiel (M x10n, avec M>=0). (exemple : pour F=0.2601, M=2601 et n=4).
4- Soit T un tableau de N nombres reels strictement positifs, (N<=50). En utilisant les actions précédentes, écrire un algorithme permettant d’afficher les éléments dont la partie entière est le miroir de la partie fractionnaire. (exemple : X=23658,85632)

```
Fonction    Miroir(A: Entier): Entier
Var
  mir : Entier;
Debut
  mir <--- 0;

  /* On obtient chacun des chiffres de X en faisant des divisions successives par 10
    jusqu'a avoir un quotient = 0. Les chiffres sont les restes apres chaque division par 10.
   */

  Tant Que (A <> 0)
  Faire
     mir <--- (mir * 10) + (A mod 10);
     A <--- A Div 10;
  Fait

  Retourner mir;
Fin;

/* Dans le cas ou X est > 0 on obtient la partie entiere en debutant pE de 0 et en ajoutant 1
chaque fois a pE jusqu'a ce que la condition (pE < X) devienne fausse c'est a dire pE aura la
valeur la partie entiere qui est juste superieur au reel X donne. Donc en sortant de la boucle
ne pas oublier de soustraire 1 de pE. Par exemple si X=8.23 on sortira de la boucle lorsque
pE est egale a 9 (car 9 est le premier entier > 8.23), donc ne pas oublier de soustraire 1 de
pE pour avoir pE=8. La partie fractionnaire pF est egale a X - pE. Dans le cas ou X < 0 il suffit
de soustraire 1 de pE chaque fois.
*/
Procedure   IntFrac(E/ X:Reel, S/ pE: Entier, S/ pF:Reel)
Debut
```

```
   pE <-- 0;
   Si (X > 0) alors
         Tant Que (pE < X) Faire
              pE <-- pE + 1;
         Fait
         pE <-- pE - 1;
   Sinon
         Tant Que (pE > X) Faire
              pE <-- pE - 1;
         Fait
         pE <-- pE + 1;
   Fsi
   pF <-- X - pE;
Fin;

/* Exemple: X=0.2356. Si on fait X <-- X * 10 on a X=2.356, si on fait encore X <-- X * 10 on a
X=23.56,  encore une fois X <-- X * 10 on a X=235.6, encore une fois X <-- X * 10 on a
X=2356.0
Donc pour avoir 2356 il faut chaque fois multiplier X par 10 jusqu'a avoir la partie fractionnaire
egale a 0 et dans ce cas N est combien de fois on a multiplie par 10.
*/
Procedure  Fexpo(E/ X:Reel, S/ M: Entier, S/ N:Entier)
Var
   pF: Reel;
   pE: Entier;
Debut
   N <-- 0;
   IntFrac(X, pE, pF);   // obtenir la 1ere partie fractionnaire de X (qui est au debut X).
   Tant Que (pF <> 0)  Faire
      N <-- N + 1;
      X <-- X * 10;
      IntFrac(X, pE, pF);
   Fait
   M <-- pE;
Fin;


Algorithme    TabMiroirs;
Var
   T: Tableau[1..50] de Reel;
   i, pE, pE1, nb, N: Entier;
   pF: Reel;
Fonction Miroir(A: Entier): Entier
   ...
Fin;
Procedure   IntFrac(E/ X:Reel, S/ pE: Entier, S/ pF:Reel)
   ...
Fin;
Procedure  Fexpo(E/ X:Reel, S/ M: Entier, S/ N:Entier)
   ...
Fin;
Debut
   Repeter
      Ecrire("Donner N compris entre 1 et 50");
      Lire(N);
   Jusqu'a ((N >= 1) et (N <= 50));
   Pour i <-- 1 a N Faire
       Ecrire("Donner ", i, " eme valeur de T");
       Lire(T[i]);
   Fait
```

```
  Pour i <-- 1 a N Faire
    IntFrac(T[i], pE, pF);
    Fexpo(pF, pE1, nb);
    Si (pE = Miroir(pE1)) alors
      Ecrire(T[i], " a la partie entiere", pE, " miroir de la partie fractionnaire", pE1);
    Fsi
  Fait
Fin.
```

Exercice 5

Ecrire une action paramétrée SYM permettant de vérifier si une matrice carrée d'ordre N, est symétrique (N<=20).

Soit A une matrice de NxN entiers avec N ≤ 20. Ecrire un algorithme qui lit (remplit) cette matrice et vérifie si elle est symétrique en utilisant l’action paramétrée SYM, et, dans ce cas, affiche les valeurs non dupliquées ainsi que leurs positions respectives.

1	3	7	5	2
3	**-1**	2	1	-2
7	2	**2**	6	0
5	1	6	**8**	-5
2	-2	0	-5	**-2**

```
Fonction  SYM(A:Tableau[1..20, 1..20] de Entier, N:Entier):Booleen
Var
   i, J: Entier;
   symM:Booleen;
Debut
  symM <--- Vrai;
  i <--- 2;
  Tant Que ((i <= N) ET symM)
  Faire
     J <--- 1;
     /* Les positions des elements symetriques sont [i,J] et [J,i].
        On compare les elements situes sous la diagonale qui sont
        ([i, J] avec J < i) avec leurs correspondants qui sont au dessus de la
        diagonale qui sont [J, i]
     */
     Tant Que ((J < i) ET symM)
     Faire
        Si (A[i, J] <> A[J, i])  alors
           symM <--- Faux;
        Sinon
           J <--- J + 1;
        Fsi
     Fait
     i <--- i + 1;
  Fait

  Retourner symM;
Fin;


Algorithme    MatriceSymetrique;
Var
  A: Tableau[1..20, 1..20] de Entier;
  i, J: Entier;

Fonction  SYM(A:Tableau[1..20, 1..20] de Entier, N:Entier):Booleen
  ...
Fin;
```

```
Debut
  Repeter
      Ecrire("Donner N compris entre 1 et 20");
      Lire(N);
   Jusqu'a ((N >= 1) et (N <= 20));
   Pour i <-- 1 a N Faire
     Pour J <-- 1 a N Faire
       Ecrire("Donner ", i, " eme et ", J, " eme valeur de A");
       Lire(A[i, J]);
     Fait
   Fait

   Si (SYM(A, N)) alors
        Ecrire("La matrice A est Symetrique");
   Sinon
        Ecrire("La matrice A N'est PAS Symetrique");
        Ecrire("Liste des Valeurs Non Dupliquees:");
        Pour i <-- 2 a N Faire
               Pour J <-- 1 a N Faire
                    Si (A[i, J] <> A[J, i]) alors
                        Ecrire("A[", i, ",", J, "]", A[i, J]);
                        Ecrire("A[", J, ",", i, "]", A[J, i]);
                    Fsi
               Fait
       Fait
   Fsi
Fin.
```

Série Complémentaire

Exercice 6

Ecrire une action paramétrée ANAGRAMME qui vérifie si deux mots sont anagrammes. Sachant qu'un mot est dit anagramme d'un autre mot s'ils sont formés des mêmes lettres.
Exemples :
CHIEN anagramme de CHINE, NICHE,
GELER n'est pas anagramme d' ALGER

```
Fonction    ANAGRAMME(mot1, mot2: Chaine):Booleen;
Var
   T1, T2, i, J, nb1, nb2: Entier;
   anagram: Booleen;
Debut
   T1 <-- Taille(mot1);
   T2 <-- Taille(mot2);
   anagram <-- Faux ;

   // Les 2 chaines mot1 et mot2 sont anagrammes s'ils ont la meme taille et si tous les
caracteres de mot1
   // ont le meme nombre d'occurrence dans mot1 et dans mot2 (les 2 chaines ont les meme
caracteres
   // mais dans un ordre different).
   Si (T1 = T2)   alors
      anagram <-- Vrai;
      i <-- 1 ;
      Tant Que ((i <= T1) et anagram)  Faire
         car <-- mot1[i];
         nb1 <-- 0;
         Pour J <-- 1 a T1 Faire
           Si (mot1[J] = car) alors
              nb1 <-- nb1 + 1 ;
```

```
        Fsi
      Fait

      nb2 <-- 0;
      Pour J <-- 1 a T1 Faire
        Si (mot2[J] = car) alors
           nb2 <-- nb2 + 1 ;
        Fsi
      Fait

      Si (nb1 <> nb2) alors
        anagram <-- Faux;
      Fsi

      i <-- i + 1;
    Fait
  Fsi

  Retourner anagram;
Fin;
```

Exercice 7

1- Ecrire une Procédure DecToBin qui permet de convertir un entier positif en une chaine de caractères binaire (‘0’ ou ‘1’) représentant son code Binaire.
2- Ecrire une Procédure BinToDec qui permet de convertir une chaine de caractères binaire (‘0’ ou ‘1’) représentant un code Binaire en un entier.
3- Ecrire une Fonction XOR qui permet de calculer le ou exclusif (XOR) entre deux caractères binaire, on rappelle que : AB '0'$SiA = Bet$'1'$Sinon$
4- Ecrire une Procédure BinToGray qui permet de convertir une une chaine représentant un code Binaire en une chaine représentant le code de Gray équivalant.
5- Ecrire une Procédure GrayToBin qui permet de convertir une chaine représentant un code de Gray en une chaine représentant le code Binaire équivalant.
6- En utilisant les actions paramétrées précédentes, écrire un algorithme de transcodage qui, suivant un choix donnée en entrée (Décimale, Binaire, Gray), affiche les deux autres codes équivalents.

On va considerer une chaine binaire comme composee de 32 bits avec le bit[1] etant le bit de plus fort poids (MSB: Most Significant Bit).

```
Procedure  DecToBin(E/ N: Entier, S/ ch: Chaine[32])
Var
  i : Entier;
Debut
  Pour i <--- 1 a 32
  Faire
     ch[i] <--- '0';
  Fait

  i <--- 32;
  Tant Que (N <> 0)
  Faire
     Si ( (N mod 2) = 0) alors
        ch[i] <--- '0';
     Sinon
        ch[i] <--- '1';
     Fsi
     N <--- N Div 2;
     i <--- i - 1;
  Fait
```

```
Fin;

Fonction  BinToDec(ch: Chaine[32]): Entier
Var
   i, N, P : Entier;
Debut

   N <--- 0;
   i <--- 32;
   P <--- 1;
   Tant Que (i >= 1)
   Faire
       Si ( ch[i] = '1') alors
           N <--- N + P;
       Fsi
       P <--- P * 2;
       i <--- i - 1;
   Fait
   Retourner N;
Fin;

Fonction  XOR(c1, c2: Caractere) : Caractere
Var
   c : Caractere;
Debut
   Si (((c1 = '0') et (c2 = '1'))
           ou
           ((c1 = '1') et (c2 = '0'))) alors
         c <--- '1';
   Sinon
         c <--- '0';
   Fsi

   Retourner c;
Fin;
```

4- Ecrire une Procédure BinToGray qui permet de convertir une une chaine représentant un code Binaire en une chaine représentant le code de Gray équivalant.

Rappel sur le code gray.
Le code gray est un code dans lequel le passage d'une valeur a la valeur suivante se fait en changeant uniquement un seul bit.
Exemple sur 3 bits ci-dessous la valeur decimale, binaire et le code gray:

Decimale	Binaire	Code Gray
0	0-0-0	0-0-0
1	0-0-1	0-0-1
2	0-1-0	0-1-1
3	0-1-1	0-1-0
4	1-0-0	1-1-0
5	1-0-1	1-1-1
6	1-1-0	1-0-1
7	1-1-0	1-0-0

Par exemple les valeurs decimale 3 et 4 sont codes en binaire respectivement:
0-1-1 (b[1]=0, b[2]=1 et b[3]=1)
1-0-0 (b[1]=1, b[2]=0 et b[3]=0)
donc on a les 3 bits b[1], b[2] et b[3] qui changent pour passer de 3 a 4.
En code gray les valeurs decimales 3 et 4 sont codes respectivement:
0-1-0 (b[1]=0, b[2]=1 et b[3]=0)
1-1-0 (b[1]=1, b[2]=1, b[3]=0)

donc pour passer de 3 a 4 en code gray on a b[1] uniquement qui change (b[2] et b[3] ne changent pas). Il en est de meme pour passer de 0 a 1, de 1 a 2, de 2 a 3, ... on a un seul bit qui change.

Conversion du Binaire vers Code Gray
Methode: Recopier le bit de poids fort (MSB: Most Significant Bit) du binaire vers le gray, ensuite faire XOR entre le bit courant binaire avec le precedent bit binaire pour avoir le bit gray courant.
Exemple:
Binaire: 1-1-1-0
b[1]=1, b[2]=1, b[3]=1 et b[4]=0
MSB: b[1] dans g[1]
Ensuite a partir de i=2 chaque fois on a g[i] <--- b[i-1] XOR b[i]
g[2] <--- b[1] XOR b[2] = 1 XOR 1 = 0
g[3] <--- b[2] XOR b[3] = 1 XOR 1 = 0
g[4] <--- b[3] XOR b[4] = 1 XOR 0 = 1
code gray = 1001

```
Procedure  BinToGray(E/ chBin[32], S/ chGray: Chaine[32])
Var
   i : Entier;
Debut
   chGray[1] <--- chBin[1];
   Pour i <--- 2 a 32
   Faire
       chGray[i] <--- XOR(chBin[i - 1], chBin[i]);
   Fait

Fin;
```

5- Ecrire une Procédure GrayToBin qui permet de convertir une chaine représentant un code de Gray en une chaine représentant le code Binaire équivalant.

Conversion Code Gray vers Binaire
Methode: Recopier le bit de poids fort (MSB: Most Significant Bit) du gray code vers le binaire, ensuite faire XOR entre le bit courant binaire avec le precedent bit gray pour avoir le bit binaire courant.
Exemple:
Gray: 1-1-1-0
g[1]=1, g[2]=1, g[3]=1 et g[4]=0
MSB: g[1] dans b[1]
Ensuite a partir de i=2 chaque fois on a b[i] <--- b[i-1] XOR g[i]
b[2] <--- b[1] XOR g[2] = 1 XOR 1 = 0
b[3] <--- b[2] XOR g[3] = 0 XOR 1 = 1
b[4] <--- b[3] XOR g[4] = 1 XOR 0 = 1
code gray = 1001

```
Procedure  GrayToBin(E/ chGray[32], S/ chBin: Chaine[32])
Var
   i : Entier;
Debut
   chBin[1] <--- chGray[1];
   Pour i <--- 2 a 32
   Faire
       chBin[i] <--- XOR(chBin[i - 1], chGray[i]);
   Fait

Fin;
```

```
Algorithme    DecBinGrayTranscodage;
Var
  ch1, ch2: Chaine[32];
  choix, ent : Entier;
  fini: Booleen;

Procedure  DecToBin(E/ N: Entier, S/ ch: Chaine[32])
...
Fin;
Fonction  BinToDec(ch: Chaine[32]): Entier
...
Fin;
Fonction  XOR(c1, c2: Caractere) : Caractere
...
Fin;
Procedure  BinToGray(E/ chBin[32], S/ chGray: Chaine[32])
...
Fin;
Procedure  GrayToBin(E/ chGray[32], S/ chBin: Chaine[32])
...
Fin;

Debut
   fini <--- Faux;
   Tant Que (fini = Faux)
   Faire
      Ecrire("Entrer choix du code en Entree:");
      Ecrire("1: Decimale");
      Ecrire("2: Binaire");
      Ecrire("2: Gray");
      Ecrire("0: Sortir du Programme");
      Lire(choix);
      Cas choix Vaut
            1:
               Ecrire("Entrer un entier naturel");
               Lire(ent);
               DecToBin(ent, ch1);
               Ecrire("Le nombre ", ent, " en binaire = ", ch1);
               BinToGray(ch1, ch2);
               Ecrire("Le nombre ", ent, " en code gray = ", ch2);
            2:
               Ecrire("Entrer la chaine en binaire");
               Lire(ch1);
               ent <--- BinToDec(ch1);
               Ecrire("La chaine binaire ", ch1, " en decimale ", ent);
               BinToGray(ch1, ch2);
               Ecrire("La chaine binaire ", ch1, " en code gray = ", ch2);
             3:
               Ecrire("Entrer la chaine en code gray");
               Lire(ch1);
               GrayToBin(ch1, ch2);
               Ecrire("La chaine code gray ", ch1, " en binaire = ", ch2);
               ent <--- BinToDec(ch2);
               Ecrire("La chaine code gray ", ch1, " en decimale ", ent);
             0:
               fini <--- Vrai;
             Sinon
               Ecrire('Mauvais choix - Entrer un autre choix");
      FCas
```

```
    Fait
Fin.
```

Exercice 8

Un nombre est appelé prodigieux s’il est divisible par le produit de ses chiffres non nuls.
Exemple : A=2016, 2x1x6=12 et 2016 est divisible par 12.
1- Ecrire une action paramétrée PRODIGIEUX qui vérifie si un entier A est prodigieux.
2- Soit M une matrice carrée NxN entiers (N<=50). Ecrire un algorithme qui remplace les éléments prodigieux de la diagonale par la somme des éléments de la ligne correspondante, puis affiche la matrice si elle a subit des modifications.

```
Fonction   PRODIGIEUX(A: Entier) : Booleen
Var
   P, chfr, tmp : Entier;
   prodG: Booleen;
Debut
   prodG <-- Vrai;
   P <-- 1;
   tmp <-- A;
   Tant Que (A <> 0)
   Faire
      chfr <-- A mod 10;
      Si (chfr <> 0) alors
         P <-- P * chfr;
      Fsi
      A <--- A Div 10;
   Fait

   Si (tmp mod P <> 0) alors
      prodG <-- Faux;
   Fsi

   Retourner prodG;
Fin;


Algorithme   MatriceProdigieux;
Var
   M: Tableau[1..50, 1..50] de Entier;
   i, J, S, N: Entier;
   MChange: Booleen;

Fonction    PRODIGIEUX(A: Entier) : Booleen
   ...
Fin;
Debut
Repeter
      Ecrire("Donner N compris entre 1 et 50");
      Lire(N);
   Jusqu'a ((N >= 1) et (N <= 50));
   Pour i <-- 1 a N Faire
     Pour J <-- 1 a N Faire
       Ecrire("Donner ", i, " eme et ", J, " eme valeur de A");
       Lire(A[i, J]);
     Fait
   Fait

   MChange <-- Faux;
   Pour i <-- 1 a N Faire
     Si (PRODIGIEUX(M[i, i])) alors
```

```
        // calculer la somme de la ligne i
        S <-- 0;
        Pour J <-- 1 a N Faire
           S <-- S + M[i, J];
        Fait
        Si (M[i, i] <> S) alors
           MChange <-- Vrai;
           M[i, i] <-- S;
        Fsi
      Fsi
    Fait

    Si (MChange) alors
       Ecrire("Matrice M:");
       Pour i <-- 1 a N Faire
          Pour J <-- 1 a N Faire
             Ecrire(A[i, J]);
          Fait
       Fait
    Fsi
Fin.
```