Algorithme

-- Déclarations

Les CONSTANTES :

Le mot CONSTANTE ne peut apparaître qu'une seule fois dans un algorithme
La déclaration de constantes est facultative et doit précéder les déclarations de types et variables
Constante :
CONSTANTE PI = 3.14 ;
TVA = 0.196 ;
TITRE = 'CECI EST UN ESSAI' ;

Les VARIABLES :

Le mot VARIABLE ne peut apparaître qu'une seule fois dans un algorithme

Variable : VARIABLES NomVariable1 : NomType1 ;
NomVariable2, NomVariable3 : NomType2 ;
NomVariable4 : NomType3 ;

Les TYPES :

- type ENTIER :
- type REEL :
- type CARACTERE : 'A'
- type CHAINE DE CARACTERES : 'Chaine de caractere'
- type BOOLEEN : VRAI ou FAUX
- autre : SEXE = (FEMININ, MASCULIN) ;

Affectation variables :

Identificateur de variable <- Expression ;
ex : A <- 5 ;

Les Opérateurs Numériques :

+ : addition
- : soustraction
* : multiplication
/ : division
DIV : division entiere
MOD : reste d'une division entière

Les Opérateurs Logiques :

NON, ET, OU
NON : Priorité la plus forte
ET : Priorité supérieure vis à vis de OU
OU : Priorité la plus faible

Les Instruction de Lecture :

Pour que l'utilisateur puisse entrer une nouvelle valeur : LIRE (A) ;

Les Instructions d'Ecriture :

Pour afficher la valeur de A à l'écran : ECRIRE (A) ;

Exemple : ECRIRE ('Entrez la valeur de A : ') ;
LIRE (A) ;

La structure de selection simple :

```
   SI Condition
   ALORS
                Traitement 1
  SINON
                Traitement 2
  FIN SI ;

  Ou bien

  SI Condition
  ALORS
                Traitement 1 ;
  FIN SI ;
```

La structure de selection multiple :

```
SELON QUE Sélecteur VAUT
                Selecteur1 : Instruction1 ;
                Selecteur2 : Instruction2

                SINON  InstructionSinon;
FIN SELON ;
```

Les structures repetitives :

```
TANT QUE Condition FAIRE
                Instruction ;
FAIT ;

REPETER
                Instruction ;
JUSQU'A Condition ;

POUR I VARIANT DE 1 A N FAIRE
                Instruction ;
FAIT ;
```

Les Programmes, Fonctions et Procédures :

Programmes

```
PROGRAMME Exemple1 ;
VARIABLE
                -- Variables ici
```

```
DEBUT
                -- Programmes
FIN .
```

Fonction :

```
FONCTION NomFonction (Argument : ValeurArg) : RetourType ;
DEBUT {NomFonction}
                -- Fonction
FIN ;  {NomFonction}
```

Exemple Programme avec Fonction entier :

```
PROGRAMME Exemple7 ;
VARIABLE
        A : ENTIER ;
FONCTION Majeur(AGE : ENTIER) : BOOLEEN ;
DEBUT {Majeur}
                RENVOYER ( AGE ⩾ 18 ) ;
FIN ; {Majeur}
DEBUT
        ECRIRE ('Entrez votre âge : ') ;
        LIRE (A) ;
        SI ( Majeur (A) )
        ALORS
                ECRIRE ('Vous êtes majeur.') ;
        SINON
                ECRIRE ('Vous êtes mineur') ;
        FIN SI ;
FIN .
```

Exemple Procédure :

Une procédure possède plusieurs paramètres en entrée (données) et peut renvoyer plusieurs paramètres en sortie (résultats) et peut également **ne pas** renvoyer de paramètres en sortie.

```
PROCEDURE Affichage ( X : ENTIER ) ;
DEBUT {Affichage}
        ECRIRE ('La valeur est : ‘, x) ;
FIN ; {Affichage}

Procédure sans paramètres
PROCEDURE Bonjour1 ;
VARIABLE
                NOM : CHAINE DE CARACTERES ;
DEBUT {Bonjour1}
                ECRIRE (' Entrez votre nom ') ;
                LIRE (NOM)
                ECRIRE ('Bonjour' , NOMn ‘ ! ’) ;
FIN ; {Bonjour1}
```

Exemple Programme avec Procédure :

```
PROGRAMME Exemple ;
PROCEDURE Bonjour1 ;
VARIABLE
                NOM : CHAINE DE CARACTERES ;
DEBUT {Bonjour1}
                ECRIRE (' Entrez votre nom ') ;
                LIRE (NOM)
                ECRIRE ('Bonjour' , NOM,  ‘ ! ’) ;
FIN ; {Bonjour1}
DEBUT
        Bonjour1 ;
FIN .
```

Le mot clé VARIABLE :

Il permet de distinguer la transmission des paramètres par variable de la transmissions des paramètres par valeur

Exemple :

PROCEDURE RéponseOuiNon(MSG : CHAINE DE CARACTERES ;
VARIABLE REP : CARACTERE) ;

Transmission de parametres :
On a la transmission des paramètres par valeur et par variable (ou référence)

Exemple appel de procédure avec parametre

```
PROGRAMME Exemple10 ;
VARIABLES
 A, B : ENTIER ;
 PROCEDURE Echange1 (X, Y : ENTIER ) ;
 VARIALE
  AUX : ENTIER ;
 DEBUT {Echange1}
  AUX ← X ;
  X ← Y ;
  Y ← AUX ;
  ECRIRE ('X = ‘, X, ’  Y = ‘, Y) ;
 FIN ; {Echange1}

 DEBUT
   ECRIRE('Entrez la valeur de A : ') ;
   LIRE(A) ;
   ECRIRE('Entrez la valeur de B : ') ;
   LIRE(B) ;
   Echange1(A,B) ;
   ECRIRE('A = ‘, A, ’ B = ', B ) ;
  FIN.
```

La programmation récursive

A chaque appel d'une procédure récursive, de nouvelles variables locales sont engendrées. La récursivité implique l'empilement des résultats intermédiaires.
Si les variables locales étaient les mêmes à chaque appel, il n'y aurait pas possibilité de récursivité.

Exemple de fonction récursive permettant de calculer la somme des N premiers nombres :
SOMME(N) = 0 + 1 + 2 + ... + N

Une solution simple consisterait à programmer cela avec une boucle POUR, mais une autre manière de voir les choses serait de dire que :

SOMME (N) = N + SOMME(N - 1)
avec SOMME (0) = 0

Cette fonction effectue la somme du nombre passé en paramètre de la fonction avec la somme du nombre précédent. Et cette somme du nombre précédent va elle-même être calculée par la fonction SOMME.

Exemple :

```
FONCTION SOMME ( N : ENTIER ) : ENTIER ;
DEBUT
  SI ( N = 0 )
  ALORS
    RENVOYER 0
  SINON
    RENVOYER N + SOMME ( N - 1 ) ;
  FIN SI ;
FIN  ;
```

Les tableaux

Un tableau est une suite de valeurs repérées par un indice, il est donc nécessaire de préciser :
- d'une part, le type de l'indice
- d'une autre part, le type des éléments du tableau ou de la liste de valeurs

TYPE TYPTAB = TABLEAU [TYPEINDICE] DE TYPETABLEAU ;

TYPTAB est le nom du tableau, où TYPEINDICE est le type de l'indice et où TYPETABLEAU est le type des éléments du tableau.

soit :

```
TYPE
 TYPTAB = TABLEAU [ 1 ... N ] DE ENTIER ;
VARIABLE
  TAB : TYPTAB ;
```

Les tableaux à double dimensions

Les tris

Il y a plusieurs type de tris :
- Tri par sélection
- Tri par insertion

- Tri à bulles

Le tri par sélection est le plus simple mais le moins efficace, il cherche le minimum et le place en premier élement puis cherche le sedonc ainsi de suite en parcourant à chaque fois la chaîne.

Le tri par insertion, on prend le premier élement en indice 1, ensuite on prend le second élément on compare à l'élément avant puis on le range à sa place et ainsi de suite. on insère
ainsi les éléments successivement dans des sous-tableaus triés.

Le tri à bulles, on parcourt le tableau et compare les éléments consécutifs. Lorsque deux éléments consécutifs ne sont pas dans l'ordre ils sont permutés.

Algorithmes :

Programme Tri par sélection

```
PROGRAMME TriSélection ;
CONSTANTE
 MAX = 10 ;
TYPE
 TYPTAB = TABLEAU [ 1 .. MAX ] DE ENTIER ;
VARIABLES
 TAB : TYPTAB ;
 I, J : ENTIER ;
 MINI, POS : ENTIER ;
DEBUT

  POUR I VARIANT DE 1 A MAX FAIRe
    ECRIRE ('TAB[', I, '] = ') ;
    LIRE (TAB[I]) ;
  FAIT ;

  POUR I VARIANT DE 1 A (MAX - 1) FAIRE
   MINI ← TAB[I] ;
   POS ← I ;
   POUR J VARIANT DE (I + 1) A MAX FAIRE
    SI (TAB[J] < MINI)
    ALORS
     MINI ← TAB[J] ;
     POS ← J ;
    FIN SI ;
    FAIT ;
    TAB[POS] ← TAB[I] ;
    TAB[I] ← MINI ;
   FAIT ;

   POUR I VARIANT DE 1 A MAX FAIRE
    ECRIRE (TAB[I], '  ' ) ;
   FAIT ;
 FIN.
```

Programme Tri par insertion

```
PROGRAMME TriInsertion ;

CONSTANTE
 MAX = 10 ;
TYPE
 TYPTAB = TABLEAU [ 1 .. MAX ] DE ENTIER ;
VARIABLES
 TAB : TYPTAB ;
 I, J : ENTIER ;
 MEM : ENTIER ;
DEBUT
 POUR I VARIANT DE 1 A MAX FAIRE
  ECRIRE ('TAB [', I, '] = ' ) ;
  LIRE (TAB[I]) ;
 FAIT ;
 POUR I VARIANT DE 2 A MAX FAIRE
  MEM ← TAB [I] ;
  J ← I ;
  TANT QUE (( J > 1 ) ET ( TAB [ J - 1] > MEM )) FAIRE
    TAB [J] ← TAB [J - 1] ;
    J ← J - 1 ;
   FAIT ;
   TAB [J] ← MEM ;
 FAIT ;
 POUR I VARIANT DE 1 A MAX FAIRE
   ECRIRE(TAB[I], '  ') ;
 FAIT ;
FIN.
```

Programme Tri à bulles

```
PROGRAMME TriBulles ;
CONSTANTE
 MAX = 10 ;
TYPE
 TYPTAB = TABLEAU [ 1 .. MAX ] DE ENTIER ;
VARIABLES
 TAB : TYPTAB ;
 I, J : ENTIER ;
 MEM : ENTIER ;
DEBUT

POUR I VARIANT DE 1 A MAX FAIRE
 ECRIRE ('TAB[', I, '] = ' ) ;
 LIRE (TAB[I]) ;
FAIT ;

 POUR I VARIANT DE N A 1 FAIRE
  POUR J VARIANT DE 2 A I FAIRE
    SI (TAB[J - 1] > TAB[J])
    ALORS
     MEM ← TAB[J - 1] ;
```

```
      TAB[J - 1] ← TAB[J] ;
      TAB[J] ← MEM ;
     FIN SI ;
    FAIT ;
   FAIT ;

  POUR I VARIANT DE 1 A MAX FAIRE
   ECRIRE(TAB[I], ‘  ’ );
   FAIT ;
  FIN .
```