RÈGLES POUR L'ÉCRITURE DES PSEUDO-CODES (Programmation par objets)

Consignes générales :

1) Chaque algorithme est bien identifié.

2) Les mots clés doivent être utilisés selon la même syntaxe que ci-dessous. Ce sont généralement des verbes à l'infinitif et sont en lettres majuscules.

3) La disposition doit être observée. Les énoncés sont alignés sur la gauche et selon leur niveau. Lorsqu'il y a “décalage” d'une ligne par rapport à la précédente, ce décalage est constant (ex : 4 ou 6 caractères).

4) Le nom des variables et des méthodes débute par une lettre minuscule et s’il utilise plus d’un mot, ceux-là débutent par une majuscule. Exemples : compteur, degreFahrenheit.

5) Le nom des constantes est en **majuscules**.

1. *Opérations d'entrée/sortie*

LIRE / ÉCRIRE *Nom de la variable*
dans *Nom du fichier , Nom de la variable*

OUVRIR *Nom du fichier* en LECTURE / ÉCRITURE (“fichier physique”)

FERMER *Nom du fichier*

2. *Arrêt du programme*

FIN (*module*)

EXEMPLE :

Algorithme : Produire un résultat

Variables locales :
donnee (entier)
resultat (entier)
Instructions :
LIRE donnee
[énoncé(s) du traitement]
ÉCRIRE resultat
FIN (produireResultat)

3. *Opérations d'affectation (transfert)*

← c'est-à-dire : “prend la valeur de...”

var1 ← var2 le contenu de la variable var2 est affecté à la variable var1. Donc, var1 prend la valeur de var2

var1 ← “texte”

var1 ← nombre

var1 ← expression

EXEMPLE : Somme de 2 nombres entiers

Algorithme : Faire la somme de 2 nombres entiers

Variables locales :
nombre1, nombre2 (entiers)
somme (entier)
fichier (fichier de texte)
Instructions :
OUVRIR fichier en LECTURE ("donnees.dat")
LIRE dans fichier, nombre1, nombre2
somme ← nombre1 + nombre2
ÉCRIRE somme
FERMER fichier
FIN (somme2nombres)

4. *Structures alternatives*

4.1 ***Structure alternative simple*** (un seul branchement)

SI condition
ALORS énoncé(s)(la condition est vraie)

EXEMPLE : Afficher “positif” si le nombre lu est plus grand que 0.

Algorithme : Afficher “positif” si nombre lu > 0

Variables locales :
nombre (entier)
Instructions :
LIRE nombre
SI nombre > 0
ALORS ÉCRIRE “positif”
FIN (Afficher)

4.2 *Structure alternative complexe*

4.2.1 *Sélection binaire simple*

```
SI condition
    ALORS énoncé(s)        (la condition est vraie)
    SINON énoncé(s)        (la condition est fausse)
```

EXEMPLE : Afficher “positif” si le nombre lu est plus grand que 0 sinon afficher “négatif”

```
Algorithme :  Afficher “positif” si nombre lu > 0 sinon afficher “négatif”
Variables locales :
        nombre (entier)
        resultat (chaîne de 7 caractères)
Instructions :
        LIRE nombre
        SI nombre > 0
            ALORS  resultat ← “positif”
            SINON   resultat ← “négatif”
        ECRIRE resultat
        FIN (Afficher)
```

4.2.2 *Emboîtement des conditions*

```
SI condition
    ALORS SI condition
                ALORS énoncé(s)
                SINON  SI condition
                            ALORS énoncé(s)
                            SINON énoncé(s)
    SINON  énoncé(s)
```

Note : Le bloc ALORS ou le bloc SINON peut être omis s'il n'y a aucune action correspondante à effectuer.

EXEMPLE : Vérifier quelle est la plus petite valeur parmi 3 valeurs différentes

```
Algorithme : Afficher la plus petite valeur parmi 3 différentes
        Variables locales :
                donnee1, donnee2, donnee3 (entiers)
                resultat (entier)
        Instructions :
                LIRE donnee1, donnee2, donnee3
                SI donnee1 < donnee2
                    ALORS SI donnee1 < donnee3
                                ALORS resultat ← donnee1
                                SINON resultat ← donnee3
                    SINON   SI donnee2 > donnee3
                                ALORS resultat ← donnee3
                                SINON resultat ← donnee2
                ÉCRIRE resultat
                FIN (afficherResultat)
```

4.3 ***Structure alternative multiple***

```
SELON expression
        valeur 1 :   action(s) correspondante(s)
        valeur 2 :   action(s) correspondante(s)
        valeur 3 :   action(s) correspondante(s)
        ...
        SINON :      action(s) correspondante(s)
```

Note : SINON est facultatif. S'il est omis, aucune action n'est effectuée si aucune des conditions n'est vraie.

EXEMPLE : Attribuer une valeur à la variable **resultat** selon la donnée lue.

```
Algorithme : Attribuer une valeur à resultat

Variables locales :
        donnee, resultat (entiers)
Instructions :
        LIRE donnee
        SELON donnee
                0 :     resultat ← 1
                1 :     resultat ← 0
                2 :     resultat ← 5
        SINON:          resultat ← 10
        ÉCRIRE resultat
```

5. *Structures itératives*

5.1 ***Boucle conditionnelle***

5.1.1 TANT QUE condition
énoncé(s) (les énoncés sont répétés tant que la condition est vraie)

5.1.2 RÉPÉTER
énoncé(s)
TANT QUE condition (les énoncés sont répétés tant que la condition est vraie)

5.1.3 RÉPÉTER
énoncé(s)
JUSQU'À condition (les énoncés sont répétés tant que la condition est fausse)

EXEMPLE : Compter combien de données ont été lues. La lecture se termine si la valeur lue est plus petite ou égale à 0.

Algorithme : Compter le nombre de valeurs lues

```
Variables locales :
    compteur, donnee (entiers)
Instructions :
    compteur ← 0
    LIRE donnee
    TANT QUE donnee > 0
        compteur ← compteur + 1
        LIRE donnee
    ÉCRIRE compteur
    FIN (compterValeurs)
```

Variante :

Algorithme : Compter le nombre de valeurs lues

```
Variables locales :
    compteur, donnee (entiers)
Instructions :
    compteur ← 0
    RÉPÉTER
        LIRE donnee
        SI donnee > 0  ALORS
            compteur ← compteur + 1
    JUSQU'À donnee ≤ 0
    ÉCRIRE compteur
    FIN (CompterValeurs)
```

5.2 ***Boucle avec variable de contrôle (compteur)***

POUR compteur ← val.départ, val.finale, PAS val.incrément
énoncé(s)
(les énoncés sont répétés jusqu'à ce que la valeur du compteur dépasse la valeur finale. Le nombre de répétitions dépend du pas)

EXEMPLE : Faire la somme de 10 nombres.
Algorithme : Obtenir la somme de 10 nombres

Variables locales :
somme, compteur, nombre (entiers)
Instructions :
somme ← 0
POUR compteur ← 1, 10, PAS 1
LIRE nombre
somme ← somme + nombre
ÉCRIRE somme
FIN (obtenirSomme)

6. *Appel de méthodes*

Appel de méthode :
nom de l'objet.nom de la méthode(paramètres)

EXEMPLE *aucun paramètre* : calcul.afficher( )
EXEMPLE *avec paramètres* : calcul.additionner(nombre1, nombre2)
resultat ← nombres.calculer (nombre1, nombre2)

Appel de méthode statique :

Nom de la classe.nom de la méthode (paramètres)

7. *Pseudo-code d'une méthode*

Nom de la méthode ( ) :
Description (ligne de commentaire qui explique ce que fait le module) :
En-tête : avec paramètres (↓,↑,↔) et valeur de retour
Corps :
Variables locales
énoncé(s)
RETOURNER Valeur (ou RETOUR seul si pas de valeur retournée)

Note : L'instruction RETOUR marque la fin de la méthode. Elle signifie que le contrôle est remis au programme ou méthode appelant.

EXEMPLE :
Nom de la méthode( ) : Somme
Description : Fonction servant à fournir une somme à partir de deux entiers.
En-tête : Entier Somme (nombre1 ↓, nombre2 ↓)
Variables locales :
somme (entier)
Instructions :
somme ← nombre1 + nombre2
RETOURNER somme