Syntaxe algorithmique

1 Structure d'un algorithme

Écriture standard :

```
ALGO nomAlgo
ENTREES     → Initialisées avant l'exécution de cet algo.
    <nomE> : <typeE>
    ...
SORTIES     → Transmises à l'algorithme appelant.
    <nomS> : <typeS>
    ...
ENTREES/SORTIES
    <nomES> : <typeES>
    ...
VARIABLES   → Connues uniquement de cet algorithme.
    <nomV> : <typeV>
    ...
DEBUT
    <instructions>
FIN
```

Écriture condensée :

```
ALGO nomAlgo (<nomE> : <typeE> PAR VALEUR,
<nomES> : <typeES> PAR REFERENCE, ... ) : <typeS>, ...
VARIABLES
    <nomV> : <typeV>
    ...
DEBUT
    <instructions>
    ...
FIN
```

---

2 Appel d'un algorithme par un autre algorithme

Définition de l'algorithme `sommeDiff` :

```
ALGO sommeDiff
ENTREES
    operande1, operande2 : NUMERIQUE
SORTIES
    somme, diff : NUMERIQUE
DEBUT
    somme ← operande1+operande2
    diff ← operande1-operande2
FIN
```

Appel de l'algorithme `sommeDiff` par `algoAppelant` :

```
ALGO algoAppelant
VARIABLES
    laSomme,laDiff : NUMERIQUE
DEBUT
    laSomme,laDiff ← sommeDiff(3,4)
FIN
```

`laSomme` vaudra 7 et `laDiff` vaudra -1.

---

3 Les variables

Nommage

En minuscule avec une majuscule à chaque mot.
Exemple 1 : maVariableEstBienNommee
Exemple 2 : MavariableestmalnommeE

Pas de valeur

Une variable non initialisée a pour valeur " ?"
Ce symbole peut être utilisé pour signaler un résultat indéfini. Par exemple, le carré de -2 vaut " ?".

4 Le type `BOOLEEN`

Valeurs : `VRAI` ou `FAUX`
Opérations : `ET, OU, NON, =, !=, <>`
Commentaire :

- "Différent" se note `!=` ou `<>`
- Priorités classiques de la logique
- Préférer le parenthésage total.

---

5 Le type `NUMERIQUE`

Valeurs : les entiers et les décimaux, positifs et négatifs
Opérations : `+, -, ×, *, ; /, ÷, DIV, MOD`
Comparaisons : `=, !=, <>, <, ≤, <=, >, ≥, >=`
Commentaire :

`*` et × sont la multiplication
`^` est la puissance
`/` et ÷ sont la division flottante
`DIV` est la division entière
`MOD` est le reste de la division entière
Priorités classiques

6 Les `TABLEAU`

Syntaxe : `TABLEAU DE <entier(s)> <type>`
Exemple 1D : `TABLEAU DE 5 NUMERIQUE`
→ [20, 5, -3, 2, 104]
Exemple 2D : `TABLEAU DE 3x2 CHAINE`
→ [["Li", "Be"], ["Na", "Mg"], ["K", "Ca"]]
Les tableaux ne sont pas redimensionnables

7 Le type CHAINE

Comparaisons : `=, !=, <>`

Les caractères sont des chaines de longueur 1.
Concaténation avec le symbole “+”.
Peuvent être concaténé avec des `NUMERIQUE`.
Exemple : “J’ai”+2+“ mains gauches”

```
ALGO longueur      → Calcule la longueur de la chaine.
ENTREES
    chaine : CHAINE
SORTIES
    longueur : NUMERIQUE
```

```
ALGO sousChaine    → Extrait la sous-chaine d'une chaine
                     à partir d'un indice et d'une longueur.
ENTREES
    chaine : CHAINE
    depart,longueur : NUMERIQUE
SORTIES
    souschaine : CHAINE
        → Vaut ? si les arguments sont incompatibles
```

```
ALGO trouve        → Calcule la position d'une sous-chaine
                                              dans une chaine.
ENTREES
    chaine, sousChaine : CHAINE
SORTIES
    position : NUMERIQUE
        → Position du 1er caractère de souchaine dans
            chaine. Vaut ? si souschaine n'est pas trouvée.
```

---

8 Les conditions

```
SI <expression booleenne> ALORS
    <instructions>
FINSI
```

```
SI <expression booleenne> ALORS
    <instructions>
SINON
    <instructions>
FINSI
```

```
SI <expression booleenne> ALORS
    <instructions>
SINONSI <expression booleenne> ALORS
    <instructions>
SINON
    <instructions>
FINSI
```

9 Les boucles

```
TANTQUE <expression booleenne> FAIRE
    <instructions>
FINTANTQUE
```

```
FAIRE
    <instructions>
TANTQUE <expression booleenne>
```

```
POUR <variable> ALLANT DE debut A fin  FAIRE
    <instructions>
FINPOUR
```

```
POUR <variable> ALLANT DE debut A fin PAR PAS DE <valeur>
FAIRE
    <instructions>
FINPOUR
```