

## Travail dirigé No. 1

### Rédaction d'algorithme

**Objectifs :** Apprendre à rédiger correctement un algorithme

**Durée :** 1 semaine

**Remise du travail :** Avant 23h30 le dimanche 14 septembre 2025.

**Travail préparatoire :** Leçons 1 à 4 sur Moodle, et lecture des exercices.

**Documents à remettre :** Les algorithmes complétés.

---

#### Pour chaque exercice :

- **a) Décrivez l'algorithme de manière générale, en français**, sans tenir compte des contraintes du langage simple décrit en **b)**. Cette description est le plan que vous suivrez pour écrire la version raffinée en **b)**, elle doit donc être écrite **avant**. Dans cette description, identifiez clairement :

- ce que l'ordinateur doit afficher à l'utilisateur,
- ce que l'ordinateur doit lire de l'utilisateur,
- où sont les conditions,
- où sont les répétitions (qui n'ont pas à être sous forme « TANT QUE »),
- ce qui sera dans une fonction (pour les questions que ça concerne).

- **b) Puis** écrivez une version raffinée de l'algorithme exprimée uniquement à l'aide des opérations élémentaires suivantes :

- |                                 |                              |   |
|---------------------------------|------------------------------|---|
| • LIRE                          | • Opérateurs arithmétiques : | • Comparaisons : <, >, ≤, ≥, =, ≠   |
| • AFFICHER                      | • + (additionner)            | • Opérateurs booléens : et, ou, pas/non                                   |
| • = (affecter)                  | • - (soustraire)             | • Fonctions mathématiques : sinus, cosinus, valeur absolue, racine carrée |
| • TANT QUE <i>condition</i>     | • * (multiplier)             | • FONCTION <i>nom</i> ( <i>paramètres</i> )                               |
| FAIRE ...                       | • / (diviser)                | • RÉSULTAT/RETOUR <i>expression</i>                                       |
| • SI <i>condition</i> ALORS ... | • % (reste ou modulo)        |   |
| SINON ...                       |                              |   |

Les conditions et répétitions doivent être correctement indentées : les instructions dont l'exécution est contrôlée par une condition (SI/SINON) ou répétition (TANT QUE) sont en retrait vers la droite par rapport à cette condition/répétition. Les différents éléments d'une suite ou d'un texte sont référés avec les crochets, ainsi, *valeurs[n]* est l'élément à la position *n* de la suite ou texte. **Les index commencent à zéro, *valeurs[0]* est donc le premier élément/caractère.** « longueur de » permet de savoir combien de valeurs/caractères se trouvent dans une suite/texte. Pour initialiser tous les éléments à une valeur identique, on peut faire par exemple « variable = tableau de 0 » (variable sera un tableau où tous les éléments sont initialisés à zéro). Aussi, les lettres d'un texte sont modifiables, donc *mot[0]* = « a » permet de remplacer la première lettre d'un mot par la lettre « a ».

#### Exemples d'utilisation de suite/chaine :

Un mot est entré par l'utilisateur, puis chaque lettre du mot est affichée avec espaces entre :

- a) Demander (affichage) et lire le mot. Pour chaque lettre du mot, afficher la lettre suivie d'un espace.
- b) Afficher « Entrer un mot : »  
Lire mot  
position = 0  
TANT QUE position < longueur de mot FAIRE  
Afficher *mot[position]* « »  
position = position + 1

**Lire  $N$  valeurs dans une suite :**

- a) Pour (**répétition**) les positions de 0 à  $N$  exclu, **lire** la valeur et la placer à cette position dans la suite.
- b) position = 0  
TANT QUE position <  $N$  FAIRE  
    Lire valeur  
    suite[position] = valeur  
    position = position + 1

**Exemple de problème :** Écrire un algorithme qui vérifie si un nombre entré par l'utilisateur est premier ou non.

**Une solution possible :**

- a) Demander le nombre à l'utilisateur (**affichage**). **Lire** le nombre  $n$  de l'utilisateur.  
**Pour chaque** entier entre 2 et la racine carrée de  $n$ , vérifier (une **condition**) est-ce que cet entier divise  $n$ .  
**Si** aucun des entiers testés ne divise  $n$ , **afficher** que le nombre est premier, sinon **afficher** qu'il ne l'est pas.
- b) Afficher « Entrer le nombre à vérifier : »  
Lire  $n$   
 $i = 2$   
a trouvé un diviseur = Faux  
TANT QUE  $i * i \leq n$  FAIRE  
    SI  $n \% i == 0$  ALORS  
        a trouvé un diviseur = Vrai  
         $i = i + 1$   
    SI a trouvé un diviseur ALORS  
        Afficher « Le nombre n'est pas premier »  
SINON  
    Afficher « Le nombre est premier »

Note :  $i * i \leq n$  est équivalente à  $i \leq \sqrt{n}$  si  $i$  et  $n$  sont positifs, et n'a pas besoin de l'opération racine carrée.

**Exemple de problème :** Écrire une fonction qui vérifie si un nombre passé en paramètre est premier ou non.

**Une solution possible :**

- a) Fonction avec paramètre  $n$ .  
**Pour chaque** entier entre 2 et la racine carrée de  $n$ , vérifier (une **condition**) est-ce que cet entier divise  $n$ , le **résultat** est Faux si c'est le cas. Dans le cas où aucun diviseur n'est trouvé le **résultat** est Vrai.
- b) FONCTION est premier ( $n$ ) :  
 $i = 2$   
TANT QUE  $i * i \leq n$  FAIRE  
    SI  $n \% i == 0$  ALORS  
        RÉSULTAT Faux  
         $i = i + 1$   
RÉSULTAT Vrai      (vous pouvez utiliser le mot RÉSULTAT ou RETOUR, à votre choix, les deux sont équivalents, mais utilisez le même mot dans tous vos algorithmes raffinés)

**Exemple d'utilisation de cette fonction :**

```

Lire x
SI est premier (x) ALORS
    Afficher « Oui »

```

**1 – Orthogonal :** Écrire un algorithme qui détermine si deux vecteurs à deux dimensions sont orthogonaux ou non. Note : utiliser un produit scalaire ; les opérations sur les vecteurs, dont le produit scalaire, ne sont pas des opérations élémentaires disponibles pour l'algorithme raffiné.

*Exemple :* L'utilisateur entre les composantes des vecteurs (1 ; 0,5) et (-1 ; 2)

L'affichage attendu est : Les vecteurs sont orthogonaux.

**2 – Quadratique :** Écrire un algorithme qui calcule les valeurs  $y$  pour  $nPoints$  points également espacés en  $x$  entre  $x_{min}$  et  $x_{max}$  sur une fonction quadratique  $y = ax^2 + bx + c$ . Utilisez une fonction pour calculer  $y$  à partir de  $a$ ,  $b$ ,  $c$ , et  $x$ .

*Exemple :* L'utilisateur entre les valeurs de  $a$ ,  $b$  et  $c$  comme étant -1 ; 2 ; -5, et les valeurs de  $x_{min}$ ,  $x_{max}$  et  $nPoints$  comme étant 0 ; 1 ; 6

L'affichage attendu est :

Les valeurs (x ; y) des 6 points sont :

0 ; -5

0,2 ; -4,64

0,4 ; -4,36

0,6 ; -4,16

0,8 ; -4,04

1 ; -4

**3 – Graphique :** Reprendre l'algorithme écrit en 2 et afficher sous forme textuelle les points pour faire un « graphique » en utilisant des espaces et des étoiles. L'algorithme aura aussi besoin des valeurs  $y_{min}$ ,  $y_{max}$  et  $tailleY$ . Le nombre d'espaces à afficher doit augmenter linéairement pour une valeur de  $y$  entre  $y_{min}$  et  $y_{max}$ , soit aucun espace lorsque  $y = y_{min}$ , et  $tailleY$  espaces lorsque  $y = y_{max}$ .

Un point dont le  $y$  n'est pas dans l'intervalle  $[y_{min}, y_{max}]$  sera affiché comme une ligne vide. Utiliser l'opération « Afficher finDeLigne » pour indiquer où sont les fins de lignes.

*Exemple :* L'utilisateur entre les mêmes valeurs qu'à l'exercice 2, puis les valeurs de  $y_{min}$ ,  $y_{max}$  et  $tailleY$  comme étant -5 ; -4 ; 20

L'affichage attendu est : (les «    » représentent des espaces et les « ↵ » les fins de lignes)

```
*↵
      *↵
    *↵
  *↵
*↵
```

**4 – Recherche :** Écrire une fonction dont le résultat est la position où se trouve le texte « INF » dans une phrase qui lui est passée ; elle doit avoir le résultat -1 dans le cas où il n'y est pas. La position résultante doit être celle où se trouve la première lettre du « INF » dans la phrase, la position zéro étant la première lettre de la phrase.

Note : chaque caractère compte comme une position, incluant les espaces et les ponctuations. La fonction « longueur de (phrase) » permet de connaître le nombre de caractères dans la phrase (voir exemple en p.1).

*Exemple :* La phrase passée à la fonction est « J'ai un cours d'INF1005C », le résultat attendu est 16.

**5 – Moyenne :** Écrire un algorithme qui calcule la moyenne entre des valeurs positives entrées par l'utilisateur. Le nombre de valeurs n'est pas connu à l'avance, l'utilisateur entrera la valeur -1 pour indiquer qu'il a terminé.

*Exemple :* L'utilisateur entre les valeurs 1 ; 7 ; 11 ; -1.

L'affichage attendu est : La moyenne des 3 valeurs est 6,33.

**6 – Logarithme :** Écrire une **fonction** pour calculer le logarithme naturel d'un nombre réel  $x$  situé dans l'intervalle  $]0, 2[$ . Vous devez vérifier que le nombre respecte bien l'intervalle. S'il ne le respecte pas, affichez « Le nombre n'est pas valide » et terminer le programme. La méthode pour calculer le logarithme sera d'utiliser la série définie comme :

$$\ln(x) = (x - 1) - \frac{(x - 1)^2}{2} + \frac{(x - 1)^3}{3} - \frac{(x - 1)^4}{4} + \frac{(x - 1)^5}{5} - \dots$$

Lorsque le nombre de termes tend vers l'infini, cette série converge vers le logarithme naturel de  $x$ , si  $x$  est dans l'intervalle  $]0, 2[$  (notez que numériquement ça ne converge pas très bien pour  $x$  très près de 0 ou de 2). L'estimation de l'erreur pour la série ayant  $n$  termes est la valeur absolue du  $n$ ième terme (sans le signe, les termes sont décroissants sur l'intervalle de convergence). L'algorithme doit arrêter lorsque cette estimation de l'erreur est inférieure à la précision voulue. La lecture des données et l'affichage doit être fait à l'extérieur de la **fonction**. La **fonction** doit donc uniquement retourner le logarithme naturel en fonction de  $x$  et de la précision voulue.

Note : ni la valeur absolue, ni l'exponentiation, ne sont des opérations élémentaires disponibles pour l'algorithme raffiné.

*Exemple :* L'utilisateur entre les valeurs de  $x$  et *précision* comme étant 1,5 et 0,05.

L'affichage attendu est : Le logarithme naturel de 1.5 est approximativement 0.416667.

Dans cet exemple, les valeurs des termes sont : 0.5, -0.125, 0.041666666 ; puisque ce dernier terme (en valeur absolue) est inférieur à la précision voulue, c'est la somme de ces termes qui est l'approximation acceptée.