

## Travail dirigé No. 1

### Rédaction d'algorithme

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

**Durée :** ½ semaine

**Remise du travail :** Avant 23h30 le 13 mai 2019.

**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 <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

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

```

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

```

1. Écrire un algorithme qui calcule la puissance dissipée par une résistance comme suit :

$$P = V^2 / R$$

Note : l'exponentiation n'est pas une opération élémentaire disponible pour l'algorithme raffiné (voir première page).

*Exemple* : L'utilisateur entre les valeurs de V et R comme étant 5 (volts) et 1000 (ohms)

L'affichage attendu est : La puissance dissipée est 0,025 watts.

2. 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.

3. 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  $y = a x^2 + b x + c$ .

*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

4. Écrire un algorithme qui indique à quelle position se trouve le texte « INF » dans une phrase ; il doit indiquer « ne s'y trouve pas » dans le cas où il n'y est pas. La position affichée 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(phrase) permet de connaître le nombre de caractères dans la phrase.

*Exemple* : L'utilisateur entre la phrase « J'ai un cours d'INF1005C ».

L'affichage attendu est : INF se trouve à la position 16.

5. 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. Vous déménagez et vous désirez savoir le nombre de vos d'objets qui peuvent remplir un camion. Écrire l'algorithme avec un booléen qui demande la capacité du camion, le volume des objets et qui affiche le nombre d'objets pouvant être contenus dans le camion. Concrètement, vous devez demander, en boucle, à l'utilisateur d'entrer le volume d'un objet à ajouter au camion et arrêter si l'objet demandé ne rentre pas dans le camion (est plus gros que l'espace restant).

7. Écrire un algorithme pour calculer la racine carrée d'un nombre réel positif  $x$ . La méthode sera d'utiliser la suite définie comme :

$$y_0 = x$$

$$y_{n+1} = (y_n + x / y_n) / 2$$

Lorsque  $n$  tend vers l'infini, cette suite converge vers la racine carrée de  $x$ . L'estimation de l'erreur au terme  $y_n$ , par rapport à la véritable racine carrée, sera  $e_n = |y_n - y_{n-1}|$  (soit la valeur absolue de la différence entre deux termes qui se suivent dans la suite). L'algorithme doit arrêter, et afficher la valeur de  $y_n$ , dès que cette erreur estimée est inférieure à *epsilon*.

Exemple : L'utilisateur entre les valeurs de  $x$  et *epsilon* comme étant 2 et 0,01.

L'affichage attendu est : La racine de 2 est approximativement 1,414215686.

Dans cet exemple, les valeurs des  $y$  sont :  $y_0 = 2$  ;  $y_1 = 1,5$  ;  $y_2 = 1,41666666$  ;  $y_3 = 1,414215686$ . La différence entre  $y_3$  et  $y_2$  est de  $\sim 0,002$ , qui est inférieur au *epsilon* de 0,01, d'où l'affichage de la valeur de  $y_3$  comme approximation acceptée de la racine de 2.