**INF1005C** **Programmation procédurale**

# Travail dirigé No. 1
# Rédaction d'algorithme

**Objectifs :** Apprendre à rédiger correctement un algorithme
**Durée :** 1 semaine
**Remise du travail :** Avant 23h30 le 20 mai 2020.
**Travail préparatoire :** Leçons 1 à 4 sur Moodle, et lecture des exercices.
**Documents à remettre :** Les algorithmes complétés dans un seul fichier en format docx ou pdf

---

DES EXEMPLES DE SOLUTIONS SONT DISPONIBLES DANS L'ANNEXE A!

## CONSIGNES

**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'usager,
  - ce que l'ordinateur doit lire de l'usager,
  - 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
- AFFICHER
- = (affecter)
- TANT QUE *condition* FAIRE …
- SI *condition* ALORS … SINON …

- Opérateurs arithmétiques :
  - + (additionner)
  - − (soustraire)
  - * (multiplier)
  - / (diviser)
  - % (reste ou modulo)

- Comparaisons : <, >, ≤, ≥, =, ≠
- Opérateurs booléens : et, ou, pas/non
- Fonctions mathématiques : sinus, cosinus, valeur absolue, racine carrée
- FONCTION *nom* ( *paramètres* )
- RÉSULTAT *expression*

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**. « longueurDe(texte) » 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 ».

# EXERCICES

**1 – Énergie cinétique :** Écrire un algorithme qui calcule l'énergie cinétique d'un objet comme suit :

$$E_C = \frac{1}{2}mv^2$$

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

*Exemple* : L'utilisateur entre les valeurs de m et v comme étant 5 (kg) et 10 (m/s), l'affichage attendu est :
`L'énergie cinétique de l'objet est 250 joules.`

**2 – Pair/Impair :** Écrire un algorithme qui détermine si au moins un des deux nombres entrés est pair. Cependant, vous ne devez effectuer qu'une seule vérification.

*Exemple :* L'utilisateur entre les nombres 5 et 4
L'affichage attendu est : `Au moins un des nombres entrés est pair.`

**3a – Bonds :** Écrire un algorithme qui affiche certaines valeurs d'une liste en fonction d'un bond. L'utilisateur doit d'abord entrer un nombre entier positif *n* (la vérification n'est pas nécessaire) correspondant au nombre de valeurs qu'il souhaite entrer. Il doit aussi préciser une valeur entière positive *b* de bond (aucune vérification nécessaire). Le programme doit ensuite lire les *n* nombres de l'utilisateur. Finalement, il doit afficher un nombre à chaque bond ainsi que la position de ce nombre dans la liste.

*Exemple* : L'utilisateur entre les valeurs de *n* et *b* comme étant 5 et 2. Il entre ensuite les 5 valeurs suivantes : **12**, 32, **4**, 8, **17**
L'affichage attendu est :

```
Les valeurs situées à un bond de 2 sont :
Position 0 : 12
Position 2 : 4
Position 4 : 17
```

**3b – Lignes :** Reprendre l'algorithme écrit en 3a et afficher une ligne pour chaque valeur sélectionnée plutôt que de l'afficher en chiffre. Vous devez donc créer une **fonction** afficherLigne qui permet d'afficher une ligne d'étoiles correspondant au nombre passé en paramètre.

*Exemple* : L'utilisateur entre les mêmes valeurs qu'à l'exercice 3.
L'affichage attendu est :

```
Les valeurs situées à un bond de 2 sont :
Position 0 : ************
Position 2 : ****
Position 4 : ****************
```

DES EXEMPLES D'AFFICHAGES SONT DISPONIBLES DANS L'ANNEXE B!

**4 – Séquence :** Écrire un algorithme qui trouve et affiche la première séquence entre deux lettres précisées dans un texte. L'utilisateur écrit d'abord la phrase. Ensuite, il fournit deux caractères (un de début et un de fin). L'algorithme doit d'abord repérer le premier caractère correspondant au caractère de début et afficher tous les caractères jusqu'au prochain caractère de fin inclusivement. Si le caractère de début n'est pas dans la séquence, afficher « `La séquence n'a pas été trouvée.` » Si le caractère de fin n'est pas trouvé, la phrase sera affichée à partir du caractère de début jusqu'à la fin.
Note : chaque caractère compte comme une position, incluant les espaces et les ponctuations. La fonction longueurDe(phrase) permet de connaître le nombre de caractères dans la phrase (voir exemple en p.4). Un « a » n'est pas le même caractère qu'un « A ».

*Exemple* : L'utilisateur entre la phrase « `Batman assure la sécurité des restaurants` » avec un début de « a » et une fin de « s ». L'affichage attendu est : `atman as.`

**5 – Écart :** Écrire un algorithme qui calcule l'écart entre le maximum et le minimum d'une liste de valeurs entre 0 et 100 entrées par l'usager (aucune vérification nécessaire). Le nombre de valeurs n'est pas connu à l'avance, l'usager entrera la valeur -1 pour indiquer qu'il a terminé.

*Exemple* : L'utilisateur entre les valeurs 1 ; 7 ; 11 ; -1.
L'affichage attendu est : `L'écart entre le maximum et le minimum est de 10`.

**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 :

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

Le nombre de termes de la série qui sera utilisé pour le calcul sera donné par l'utilisateur. 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 du nombre de termes.
*Exemple :* L'utilisateur entre les valeurs de *x* et *n* comme étant 1.5 et 3.
L'affichage attendu est : `Le logarithme naturel de 1.5 est approximativement 0.416667`.

Dans cet exemple, la valeur de l'expression est :

$$\ln(1.5) = (1.5-1) - \frac{(1.5-1)^2}{2} + \frac{(1.5-1)^3}{3}$$
$$= 0.5 - \frac{0.5^2}{2} + \frac{0.5^3}{3}.$$

DES EXEMPLES D'AFFICHAGES SONT DISPONIBLES DANS L'ANNEXE B!

# ANNEXE A : EXEMPLES AVEC SOLUTION

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

**Un mot est entré par l'usager, 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'usager est premier ou non.

**Une solution possible :**

a) Demander le nombre à l'usager (**affichage**). **Lire** le nombre *n* de l'usager.
**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 ≤ 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** et Vrai.

b) FONCTION est premier (*n*) :

```
i = 2
TANT QUE i * i ≤ 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 »
```

# ANNEXE B : EXEMPLES D'AFFICHAGES

## 1 – Énergie cinétique

```
Veuillez entrer la masse
5
Veuillez entrer la vitesse
10
L'energie cinetique de l'objet est 250 joules.
```

## 2 – Pair/impair

```
Veuillez entrer le premier nombre
4
Veuillez entrer le deuxieme nombre
5
Au moins un des nombres entres est pair
```

## 3a – Bonds

```
Combien de valeurs voulez-vous entrer?
5
Quel est le bond?
2
Veuillez entrer une valeur
3
Veuillez entrer une valeur
32
Veuillez entrer une valeur
1
Veuillez entrer une valeur
8
Veuillez entrer une valeur
5
Les valeurs situees α un bond de 2 sont:
Position 0 : 3
Position 2 : 1
Position 4 : 5
```

## 3b – Lignes

```
Combien de valeurs voulez-vous entrer?
5
Quel est le bond?
2
Veuillez entrer une valeur
3
Veuillez entrer une valeur
32
Veuillez entrer une valeur
1
Veuillez entrer une valeur
8
Veuillez entrer une valeur
5
Les valeurs situees a un bond de 2 sont:
Position 0 : ***
Position 2 : *
Position 4 : *****
```

### 4 – Séquences

```
Veuillez entrer une phrase
Batman assure la securite des restaurants
Veuillez entrer le caractere de debut de la sequence recherchee
a
Veuillez entrer le caractere de fin de la sequence recherchee
s
atman as
```

### 5 – Écart

```
Veuillez entrer une valeur ou -1 pour terminer
1
Veuillez entrer une valeur ou -1 pour terminer
11
Veuillez entrer une valeur ou -1 pour terminer
7
Veuillez entrer une valeur ou -1 pour terminer
-1
L'ecart entre le maximum et le minimum est de 10.
```

### 6 – Logarithme

```
Veuillez entrer un nombre
1.5
Veuillez entrer un nombre de termes
3
Le logarithme naturel de 1.5 est approximativement 0.416667
```