INF1005C **Programmation procédurale**

Travail dirigé No. 1
Rédaction d'algorithme

Objectifs : Apprendre à rédiger correctement un algorithme
Durée : $^1/_2$ semaine
Remise du travail : Avant 23h30 le 9 mai.
Travail préparatoire : Lecture des exercices.
Documents à remettre : Les algorithmes complétés.

Pour chaque exercice :

- **Décrivez l'algorithme de manière générale, en français,** sans tenir compte des contraintes du langage simple décrit ci-dessous. Cette description servira de plan pour ensuite écrire l'algorithme raffiné ci-dessous; elle doit donc être écrite avant de faire l'algorithme raffiné, mais pourra être ajustée par la suite si des problèmes sont détectés lors du raffinement. 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 »).
- **Puis** écrivez une version raffinée de l'algorithme exprimée uniquement à l'aide des opérations élémentaires suivantes :

▪ Lire	▪ TANT QUE *condition* FAIRE …	▪ Opérateurs arithmétiques :
▪ Afficher	▪ SI *condition* ALORS … SINON …	▪ + (additionner)
▪ = (affecter)	▪ Comparaisons : <, >, ≤, ≥, ==, ≠	▪ − (soustraire)
	▪ Opérateurs booléens : et, ou, pas/non	▪ * (multiplier)
		▪ / (diviser)
		▪ % (reste ou modulo)

Les conditions et répétitions doivent être correctement indentées. Les opérations mathématiques suivantes sont permises : sinus, cosinus, valeur absolue, racine carré. Les différents éléments d'une suite ou d'une chaine de caractères sont référés avec les crochets, ainsi, *valeurs[n]* est le $n^{\text{ième}}$ élément de la suite. « longueur de » permet de savoir combien de valeurs/caractères se trouvent dans une suite/chaine.

Exemple : Écrire un algorithme qui vérifie si un nombre entré par l'usager est premier ou non.

Une solution possible :

Demander le nombre à l'usager (affichage). Lire le nombre *n* de l'usager.
Pour chaque entier (une répétition) entre 2 et la racine carrée de *n*, vérifier (une condition) est-ce que cet entier divise *n*.
Si (une condition) aucun des entiers testés ne divise *n*, afficher que le nombre est premier, sinon afficher qu'il ne l'est pas.

Algorithme raffiné :

```
Afficher « Entrer le nombre à vérifier : »
Lire n
i = 2
a trouvé un diviseur = faux
TANT QUE 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 »
```

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. Reprendre l'algorithme écrit en 3 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 3, 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)

```
*↲
␣␣␣␣␣␣␣␣*↲
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*↲
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*↲
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*↲
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*↲
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣*↲
```

5. 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.

Exemple : L'utilisateur entre la phrase « `J'ai un cours d'INF1005C` ».
L'affichage attendu est : `INF se trouve à la position 16`.

6. Écrire un algorithme qui calcule la moyenne entre des valeurs positives entrées par l'usager. 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 : `La moyenne des 3 valeurs est 6,33`.

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*.
Note : la valeur absolue n'est pas une opération élémentaire disponible pour l'algorithme raffiné.

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 : $y_0 = 2$; $y_1 = 1{,}5$; $y_2 = 1{,}41666666$; $y3 = 1{,}414215686$.
La différence entre y_3 et y_2 est de ~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.