

## Techniques de programmation

### Série d'exercices n° 1

#### Buts

- Appliquer les techniques de spécification et d'analyse
  - Maîtriser les concepts de base de l'algorithmique
  - Coder des algorithmes en Langage C
- 
- 

#### Exercice 1

---

Ecrire l'algorithme/programme **Bissextile** qui lit un entier naturel A qui représente une année et qui détermine si cette année est bissextile.

On précise que si A n'est pas divisible par 4, l'année n'est pas bissextile. Si A est divisible par 4, l'année est bissextile sauf si A est divisible par 100 et pas par 400.

#### Exercice 2

---

Ecrire un algorithme/programme **Somme** qui pour un entier naturel n, calcule et affiche la somme :

$$S_n = 1 + 2 + \dots + n$$

#### Exercice 3

---

Ecrire un algorithme/programme **Som\_Carres** qui calcule pour un entier n la somme des carrées

$$\text{Som\_carres} = 1 + 4 + \dots + n^2$$

#### Exercice 4

---

La fonction factorielle peut être définie de la manière suivante :

$$0! = 1,$$

$n! = 1 * 2 * \dots * n$  pour  $n \geq 1$ .

1. Ecrire un algorithme/programme qui calcule  $n!$
2. Modifier cet algorithme/programme **Factorielle** pour calculer et afficher les factorielles de 0 à 10.

## Exercice 5

---

Ecrire l'algorithme/programme **Pythagore** qui recherche et compte tous les triplets pythagoriciens  $(x,y,z)$  tels que  $1 \leq x \leq y \leq \max$ , où  $\max$  est un entier entré par l'utilisateur. On rappelle qu'un triplet pythagoricien est un triplet d'entiers naturels  $(x, y, z)$  tels que  $x^2 + y^2 = z^2$

## Exercice 6

---

Un nombre **d'Armstrong** est entier naturel qui est égal à la somme des cubes de chiffres qui le composent (en base 10). Par exemple, 153 est un nombre d'Armstrong car  $153 = 1^3 + 5^3 + 3^3$ . En revanche 25 n'est pas un nombre d'Armstrong car  $25 \neq 2^3 + 5^3$

Ecrire l'algorithme/programme **Armstrong** qui affiche tous les nombres d'Armstrong inférieurs à 1000.

## Exercice 7

---

1. Ecrire un algorithme/programme **diviseurs** qui prend en paramètre un entier  $n$  et qui affiche la liste de ses diviseurs. Testez cet algorithme pour  $n = 6$ ,  $n = 17$ , et  $n = 36$ .
2. Écrire un algorithme/programme qui lit un entier positif à plusieurs reprises, détermine si l'entier est **déficient**, **parfait** ou **abondant**, et écrit en sortie le nombre avec sa classification. Un entier positif  $n$  est parfait si la somme de ses diviseurs propres est égale au nombre lui-même. (Les diviseurs propres comprennent 1 mais non le nombre lui-même.) Si la somme est inférieure à  $n$ , le nombre est déficient, et si la somme est supérieure à  $n$ , le nombre est abondant.

## Exercice 8

---

La suite de **Fibonacci** associée à un entier naturel est définie par récurrence de la manière suivante :

$$\begin{cases} U_0 = U_1 = 1 \text{ et pour } n \geq 2 \\ U_n = U_{n-2} + U_{n-1} \end{cases}$$

Ecrire un algorithme/programme nommé **Fibonacci** qui reçoit un entier naturel en paramètre, calcule et retourne le terme  $U_n$  de la suite de Fibonacci

## Exercice 9

---

Ecrire l'algorithme/programme MinMoyMax qui demande à l'utilisateur d'entrer une suite d'entiers positifs dont le nombre est à priori inconnu et qui affiche le minimum, la moyenne et le maximum de ces entiers.

## Exercice 10

---

Pour exprimer un polynôme de degré  $n$  à coefficients et variable réels on utilise les deux schémas suivants :

$$\text{Schéma usuel} \quad : P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0 x^0$$

$$\text{Schéma de Horner} \quad : P(x) = (\dots(((a_n)x + a_{n-1})x + a_{n-2})x + \dots + a_1)x + a_0$$

Pour les deux schéma  $P(x)$  est défini par le tableau de coefficients réels  $A$  et par une variable réelle  $x$  (on suppose de degré  $n$  limité à 10)

1. Déterminer le nombre de multiplications nécessaires pour évaluer  $P(x)$  pour les deux schémas
2. Ecrire l'algorithme/programme itérative **Eval\_iterative** pour évaluer  $P(x)$  en utilisant le schéma de Horner.