4TPM205U: Algorithmique des tableaux: feuille 1

Itérer dans un intervalle avec la boucle `for`

Travaux dirigés sur machine

Dans toutes les fonctions de la feuille on réalisera les itérations en utilisant la boucle `for`.
Par la suite, chaque fois qu'une fonction calcule un résultat, elle doit le renvoyer.

Exercice 1 – Soient `a` et `b` deux nombres entiers, écrire une fonction `affichePairs(a,b)` qui affiche tous les nombres pairs de l'intervalle **fermé** $[a, b]$.
Pouvez vous écrire la fonction en utilisant au maximum une seule instruction `if` avant la boucle ?
(au maximum signifie qu'on peut aussi ne pas l'utiliser du tout ...)

Exercice 2 – Soit `n` un nombre naturel, écrire une fonction `puissance(x,n)` qui calcule x^n.

Exercice 3 – Écrire et exécuter la fonction `fibonacci(n)` définie dans la feuille 1 de TD.
L'appel `fibonacci(9)` doit renvoyer 55.

Exercice 4 – Écrire une fonction `afficheTableMult(n)` qui affiche une table de multiplication de taille n.
Par exemple si `n` vaut 6, on doit obtenir l'affichage :

```
1   2   3   4   5   6
2   4   6   8  10  12
3   6   9  12  15  18
4   8  12  16  20  24
5  10  15  20  25  30
6  12  18  24  30  36
```

Exercice 5 – Écrire et exécuter la fonction `sommeFactorielles(n)` définie dans la feuille 1 de TD. Tester les deux versions demandées.

Exercice 6 – Comment calculer une approximation de $\sqrt{2}$? Une méthode simple est d'utiliser la suite u_n suivante :

$$\begin{cases} u_0 & = 2 \\ u_{n+1} & = \frac{u_n}{2} + \frac{1}{u_n} \end{cases}$$

Avec cette méthode, l'erreur entre $\sqrt{2}$ et u_n diminue à chaque itération du calcul.

1. Écrire une fonction `suite(n)` qui calcule et retourne le terme u_n.

2. Utilisez une boucle pour afficher les 10 premiers termes de la suite. Une valeur approchée de $\sqrt{2}$ est $1.414213562373095048[...]$. On constate effectivement que la suite converge vers cette valeur, mais sans pouvoir toutefois l'atteindre : le nombre de chiffres des valeurs de l'ordinateur est limité !
3. Plus généralement, pour calculer $\sqrt{x}$, on peut utiliser la suite

$$\begin{cases} u_0 & = x \\ u_{n+1} & = \frac{1}{2}(u_n + \frac{x}{u_n}) \end{cases}$$

où u_0 est une première estimation grossière de $\sqrt{x}$.
Écrire une fonction `sqrt(x)` qui retourne une approximation de $\sqrt{x}$ en calculant u_{10} et imprimant les valeurs intermédiaires u_i au passage. Tester avec $x = \{2, 3, 4, 10\}$ et $n = 10$.

Exercice 7 – Exercice plus difficile à traiter en dernier.

Un nombre s'écrit comme une suite de chiffres (ou de digits). On appelle premier chiffre (ou premier digit) le chiffre (ou digit) le plus à gauche dans l'écriture du nombre. Par exemple, le premier chiffre du nombre 714 est 7, le second chiffre est 1 et le troisième chiffre est 4.

On cherche un nombre composé de 9 chiffres (ou digits) tous différents de 0 tel que :

- le nombre formé par le premier chiffre soit divisible par 1
- le nombre formé par les deux premiers chiffres soit divisible par 2
- le nombre formé par les trois premiers chiffres soit divisible par 3
- le nombre formé par les quatre premiers chiffres soit divisible par 4
- le nombre formé par les cinq premiers chiffres soit divisible par 5
- le nombre formé par les six premiers chiffres soit divisible par 6
- le nombre formé par les sept premiers chiffres soit divisible par 7
- le nombre formé par les huit premiers chiffres soit divisible par 8
- le nombre formé par les neuf premiers chiffres soit divisible par 9

De tels nombres, il y en a 416 et le plus petit est 123252561.

En n'utilisant pas d'autres instructions d'itération que la boucle *for* est-ce que l'on peut écrire une fonction (itérative !) qui affiche (et qui compte) tous les nombres ayant la propriété décrite ci-dessus ?

Dans un premier temps, vous ne chercherez pas forcément à minimiser le nombre d'itérations. Puis dans un second temps, vous pourrez améliorer votre code pour minimiser le nombre d'itérations qui sera exécuté.

(Suggestion : commencer par essayer d'écrire une fonction qui cherche un nombre de trois chiffres satisfaisant les trois premières conditions.)