TD3-1 : Java et algorithmique
V1.6.0

Document en ligne : www.mickael-martin-nevot.com

1 Exercices

Écrivez en Java les algorithmes définis ci-après (vus au `TD1-1 : Initiation à l'algorithmique`) en respectant la convention de programmation donnée en cours ; puis testez-les.

1.1 Somme

Affiche la somme de deux nombres entiers saisis par l'utilisateur.

1.2 Permutation

Échange les valeurs de deux réels saisis par l'utilisateur.

1.3 Tarifs

Demande à l'utilisateur de saisir le prix de base du billet et l'âge d'un passager, puis affiche le prix après réduction.

Contraintes

Gratuit pour les enfants de moins de 2 ans ; moitié prix pour les enfants de moins de 10 ans ; réduction de 10 % pour les personnes de moins de 27 ans et celles de plus de 70 ans.

1.4 Minimum

Affiche le minimum d'une suite de dix réels saisis par l'utilisateur.

1.5 Changement de base

Affiche la valeur en base b ($b \in \mathbb{N}$) (compris entre 2 et 9) d'un entier n ($n \in \mathbb{N}$) décimal (en base 10).

1.6 Factoriel

Affiche le résultat de la fonction mathématique factorielle appliquée à n ($n \in \mathbb{N}$).

Contrainte

Faire un algorithme **récursif**.

Équation 1 – Factorielle

$$n! = \prod_{k=1}^{n} k = 1 \times 2 \times 3 \times 4 \times \cdots \times (n-1) \times n$$

1.7 Conjecture de Syracuse

Applique la « célèbre » fonction de conjecture de Syracuse à un nombre et s'arrête lorsque le calcul devient égal à 1.

La suite de Syracuse de N ($N \in \mathbb{N}^*$) est définie par récurrence, de la manière suivante :

Équation 2 – Conjecture de Syracuse

$$\forall\, n \in \mathbb{N} : \begin{cases} u_0 = N \\ u_{n+1} = \begin{cases} \dfrac{u_n}{2}, & si\ u_n\ est\ pair \\ \dfrac{3u_n + 1}{2}, & si\ u_n\ est\ impair \end{cases} \end{cases}$$

Faites plusieurs essais en changeant la valeur de test. Voyez-vous un problème algorithmique ?

1.8 Monnayeur

Affiche le nombre chaque de billets/pièces nécessaires pour composer une somme donnée en euros sans centimes (par exemple 1949), grâce à un algorithme glouton (le nombre de billets/pièces ne sera donc pas forcément minimal).

1.9 Test de divisibilité par 11

Affiche le résultat du test de divisibilité suivant : en ajoutant et soustrayant en alternance les chiffres d'un nombre (en commençant par n'importe quelle extrémité) et ce, de manière itérative jusqu'à ce que la valeur absolue du résultat soit strictement inférieure à 11, nous pouvons affirmer que le nombre est divisible par 11 si le résultat est égal à 0 ou qu'il ne l'est pas sinon.

Exemples de nombres divisibles par 11 :
- 121 (+ 1 - 2 + 1 = 0) ;
- 9251 (-9 + 2 – 5 + 1 = - 11 [+1 – 1 = 0]).

1.10 Recherche dichotomique dans un tableau ordonné

Pour un tableau `t` de 100 nombres entiers deux à deux distincts rangés par ordre croissant et un nombre v, détermine soit le rang de v dans `t`, soit le fait que v ne figure pas dans `t`.

1.11 Tri par insertion

Trie par la méthode suivante (soit un tableau `t` de quinze nombres entiers) ; à l'étape i ($i \in \mathbb{N}^*$) :
- on suppose que les i premiers éléments de `t` sont triés ;
- on considère l'élément `t[i]` :
  - l'insérer parmi `t[0], …, t[i - 1]` à la bonne place (en décalant progressivement tous les éléments vers la droite) ;
  - incrémenter `i`.