Faculté des sciences Dhar EL-Mehraz
Département d'Informatique
SMI - S3

Algorithmique II
Exercices de TD 2

Les Fonctions (Série 2)

Prof. Noureddine En-nahnahi

2020/2021

Objectifs : Être capable d'analyser un problème suivant l'approche descendante.

1. Problème

L'algorithme de Kaprekar a été découvert en 1949 par le mathématicien indien D.R. Kaprekar pour les nombres de quatre chiffres mais il peut être généralisé à tous les nombres. Pour simplifier, on se limitera ici à l'application de cet algorithme à des nombres de 4 chiffres (c'est-à-dire compris entre 1000 et 9999).

L'algorithme de Kaprekar consiste à associer à un nombre *N* (entier positif à 4 chiffres) un nombre *K* généré de la façon suivante :

- On considère les 4 chiffres de *N*. On forme le nombre *n1* en arrangeant ces chiffres dans l'ordre croissant et le nombre *n2* en les arrangeant dans l'ordre décroissant.
- On calcule *K*= *n2* - *n1*.

On itère ensuite le processus en remplaçant la valeur de *N* par celle de *K*. On arrête les itérations lorsque l'une des 3 situations suivantes est atteinte :

- le nombre *K* obtenu à l'issue d'une itération est nul,
- les nombres *K* obtenus à l'issue de 2 itérations successives sont identiques,
- le nombre d'itérations dépasse une valeur maximale prédéfinie.

Dans la situation 2 (et pour un nombre *N* à 4 chiffres), l'algorithme de Kaprekar produit le nombre *K* = 6174 qui n'évolue plus au fil des itérations.

Exemple : à partir du nombre *N* = 5463 on obtient les itérations suivantes :

K = 6543 – 3456 = 3087
K = 8730 – 378 = 8352
K = 8532 – 2358 = 6174
K = 7641 – 1467 = 6174

Remarque : pour vérifier la propriété, on décomposera toujours le nombre obtenu au fil des itérations en 4 chiffres (999 par exemple se décompose en 0, 9, 9, 9).

2. Programme à réaliser : Analyse descendante et algorithme principal

On veut concevoir et mettre au point par analyse descendante un programme permettant de tester l'algorithme de Kaprekar sur un nombre à 4 chiffres. Prévoir une manière simple de modifier le programme pour qu'il puisse traiter des nombres à 3, 5 ou 6 chiffres.

2.1 Proposer une structure de données permettant de manipuler les chiffres composant un nombre c'est à dire permettant par la suite de les trier par ordre croissant et/ou décroissant.

2.2 Ecrire l'algorithme principal du programme en s'appuyant sur une décomposition logique en sous-programmes.

2.3 Définir les jeux de données qui seront utilisés pour tester le programme (tests fonctionnels).

2.4 Spécifier tous les sous-programmes identifiés lors de la décomposition (type du sous-programme, liste des paramètres avec leur mode de passage et leur type, commentaire précisant le but du sous-programme et la signification des paramètres).

2.5 Vérifier la cohérence entre l'appel du sous-programme et sa spécification.

3. Algorithmes des sous-programmes

3.1 Ecrire l'algorithme associé à chaque sous-programme (décomposition d'un nombre en chiffres, recomposition, etc.) excepté celui correspondant au sous-programme de tri.

3.2 Préciser les jeux de données qui permettront de tester individuellement chaque sous-programme.