J1MI2013: Algorithmes et Programmes: feuille 3

Itérer avec la boucle while

Travaux pratiques

Se placer dans le répertoire `~/algoprog/`, télécharger l'archive `tp03.tar.gz` depuis le site de l'UE et en extraire le contenu.
Se placer dans le répertoire `tp03` et lancer l'éditeur de texte `emacs` en chargeant le fichier correspondant au premier exercice de TP (suivre toujours les indications fournies dans la feuille `MemoTP`).

Exercice 1. Dans le fichier `testSommesChiffres.c` :

1. Écrire la fonction `sommeChiffres` vue en TD.
2. Écrire une fonction `sommeChiffresRangPair` qui calcule et renvoie la somme des chiffres de rang pair d'un nombre entier `n` passé en paramètre (le chiffre le plus à droite est considéré comme étant de rang 0). Exemple : si `n` vaut 5741, la fonction retournera 8. (Ne pas confondre le **rang** (position) des chiffres, et leur **valeur**)
3. En utilisant la fonction exclusivement la fonction `sommeChiffresRangPair` écrire une fonction `sommeChiffresRangImpair` qui calcule et renvoie la somme des chiffres de rang impair d'un nombre `n` passé en paramètre.

Exercice 2. À tout nombre $n \geq 1$ on peut associer la *suite de Syracuse* $(u_k)_{k\geq 0}$ définie comme suit, $u_0 = n$ et :

$$u_{k+1} = \begin{cases} u_k / 2 & \text{si } u_k \text{ est pair} \\ 3 \times u_k + 1 & \text{si } u_k \text{ est impair} \end{cases}$$

Par exemple la suite associée à 5 est $5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, \ldots$.
La *conjecture de Syracuse* dit que quel que soit le nombre n choisi, la suite associée termine toujours par se stabiliser sur la séquence $1, 4, 2, 1, 4, 2, 1, \ldots$

Prouver ou infirmer cette conjecture est un problème ouvert en mathématiques depuis près de 70 ans.

Dans le fichier `syrac.c` fourni,

1. Écrire une fonction `syrac` qui affiche la suite de Syracuse associée à un entier n passé en paramètre et s'arrête quand la valeur 1 est atteinte en affichant le premier indice k tel que $u_k = 1$.
2. Étant donné un entier n, le *temps de vol* de la suite de Syracuse de n est le plus petit indice k tel que $u_k = 1$. Écrire une fonction `syracTemps` qui calcule et retourne le temps de vol de la suite de Syracuse associée à un entier n passé en paramètre.

3. Étant donné un entier n, le *temps de vol en altitude* de la suite de Syracuse de n est le plus petit indice $k > 0$ tel que $u_k \leq u_0$. Écrire une fonction `syracAltitude` qui calcule et retourne le temps de vol en altitude de la suite de Syracuse associée à un entier n passé en paramètre.
4. Étant donné un entier n, l'*altitude maximale* de la suite de Syracuse de n est sa valeur maximale. Écrire une fonction `syracMax` qui calcule et retourne l'altitude maximale de la suite de Syracuse associée à un entier n passé en paramètre.

Exemple : pour $n = 5$ on a
- temps de vol égal à 5 car $u_5 = 1$ et $u_k \neq 1$ pour $k < 5$,
- temps de vol en altitude égal à 3 car $u_3 = 4$ et $u_k > 5$ pour $0 < k < 3$,
- altitude maximale égale à 16.

Exercice complémentaire 1. Dans le fichier `afficheChiffres.c`,

1. Écrire une fonction `afficheChiffres` qui affiche la liste des chiffres d'un nombre entier `n` passé en paramètre. Pour cette première question on est autorisé à afficher un même chiffre plusieurs fois. Par exemple si `n = 541845619`, la fonction doit afficher,

   9 1 6 5 4 8 1 4 5

2. Écrire une fonction `classeChiffres` qui affiche la liste des chiffres d'un nombre entier `n` passé en paramètre sans répétitions et dans l'ordre croissant. Par exemple si `n = 541845619`, la fonction doit afficher,

   1 4 5 6 8 9