Travaux Pratiques d'algorithmique n°3

Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

Récursivité

L'objet de cette séance est d'introduire la notion de récursivité. Dans un premier temps, on s'intéressera à plusieurs exemples basiques qui permettent de se familiariser avec cette technique. Ensuite, nous appliquerons cette méthode à des problèmes plus complexes. Vous devrez présenter clairement les algorithmes, ainsi que **le calcul de leur complexité**.

▸ **Exercice 1.** *Écrire une fonction* `Puissance` *en appliquant la définition suivante :*

$$b^0 = 1\ , \qquad b^n = b\, b^{n-1}\ .$$

Écrire une fonction `PuissanceBis` *qui tient compte des relations suivantes :*

$$\begin{cases} b^n = (b^{n/2})^2 & \textit{si } n \textit{ est pair ,} \\ b^n = b\,(b^{(n-1)/2})^2 & \textit{si } n \textit{ est impair ,} \\ b^n = 1 & \textit{si } n = 0\ . \end{cases}$$

Évaluer la complexité dans les deux cas.

▸ **Exercice 2.** *Écrire une fonction de prototype* `void Binaire (int n)` *qui affiche la décomposition binaire de l'entier* `n`*. Par exemple :*

pour `n` $= 21$*, l'affichage sera* 10101.

Pour cela, on utilisera une version ***récursive*** *de l'algorithme décrit ci-dessous :*

On effectue la division entière de `n` *par* 2*. On trouve un quotient* q_0 *et un reste* r_0*. On recommence avec* q_0*. On construit ainsi une suite de restes* $(r_0, r_1, r_2, \ldots)$*. On arrête le processus dès que l'on trouve un quotient* q_k *égal à* 0*. La décomposition de* `n` *en base* 2 *est alors* $(r_k, r_{k-1}, \ldots, r_2, r_1, r_0)$*. Par exemple pour* $n = 10$ *:*

$$\begin{array}{ccccccc} 10 & = & 5 \times 2 & + & 0 & \longrightarrow & r_0 = 0 \\ 5 & = & 2 \times 2 & + & 1 & \longrightarrow & r_1 = 1 \\ 2 & = & 1 \times 2 & + & 0 & \longrightarrow & r_2 = 0 \\ 1 & = & 0 \times 2 & + & 1 & \longrightarrow & r_3 = 1 \end{array}$$

La représentation de 10 *en base* 2 *est donc* 1010.

▸ **Exercice 3. (somme des chiffres d'un entier)** *On veut calculer la somme des chiffres d'un entier* `n`*. Par exemple, la somme des chiffres formant* 3221 *est* 8.

- *Trouver un algorithme récursif pour calculer la somme des chiffres formant un entier quelconque ;*
- *en déduire une fonction* `SommeChiffre` *prenant en argument un entier* `n` *et qui implémente l'algorithme précédent.*

On considère la décomposition sur la base 10 *d'un entier* `n` *:*

$$n = \sum_{i=0}^{k} 10^i c_i \ .$$

Les critères de divisibilité suivants peuvent se déduire facilement (essayez pour voir...) :

$$\begin{aligned} n \ \equiv \ 0 \ [3] \ \Leftrightarrow \ \sum_{i=0}^{k} c_i \ \equiv \ 0 \ [3] \ , \\ n \ \equiv \ 0 \ [11] \ \Leftrightarrow \ \sum_{i=0}^{k} (-1)^i c_i \ \equiv \ 0 \ [11] \ . \end{aligned}$$

En déduire les fonctions `NonMultde3` *et* `NonMultde11` *qui testent si un entier* `n` *n'est pas multiple de* 3 *et de* 11 *(on utilisera une nouvelle fonction* `SommeAlternée`*).*

▸ **Exercice 4.** *Concrêtement, toute chaîne de caractères sera représentée par un tableau de caractères. Pour cela, on définit le type* `Chaine` *de la façon suivante :*

```
typedef char * Chaine;
```

- *Écrire une fonction qui permet la saisie au clavier d'une chaîne de caractères (le prototype de la fonction sera* `Chaine SaisieChaine(void)`*). L'allocation dynamique de la mémoire nécessaire sera effectuée par la fonction* `malloc` *(voir le* `man`*) ;*
- *écrire une fonction qui affiche une chaîne de caractères (on demande une version itérative et une version récursive) ;*
- *écrire une fonction* `IsEqual` *qui teste l'égalité de deux chaînes de caractères (on demande une version itérative et une version récursive).*