esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

INF1032
Méthodes de conception d'algorithmes
Enseignants
Arnaud Bannier, Nicolas Bodin, Matthieu Le Berre

TP n°1

Récursivité

1 Mesure de performances

L'objectif de cette section est de comparer les versions itératives et récursives d'un algorithme. Vous aurez l'occasion de vous rendre compte des pouvoirs de la récursivité : une puissance phénoménale mais extrêmement dangereuse si elle tombe entre de mauvaises mains !

Exercice. Je suis ton pair !
Nous étudions dans cet exercice plusieurs moyens de calculer la parité d'un entier. L'objectif est de trouver le plus efficace.

1) La première méthode utilise l'arithmétique. Quelle est-elle et quelle est sa complexité ?
2) La seconde méthode est purement informatique et particulièrement rapide. Quelle est-elle ?
3) La dernière méthode est récursive mutuelle. Elle consiste en la rédaction de deux fonctions nommées `pair(int n)` et `impair(int n)` s'appelant mutuellement. Lorsque `n` vaut 0, la fonction `pair()` renvoie VRAI et la fonction `impair()` renvoie FAUX. Rédigez donc un programme renvoyant VRAI si un entier `n` est pair, FAUX sinon. Remarquons que si `n` est pair, alors `n-1` est impair.

2 Chaines de caractères en récursif

Les chaines de caractères sont particulièrement bien adaptées à la programmation récursive, du fait de leur marqueur de fin. De façon récursive, implémentez les fonctions suivantes. N'oubliez pas que le décalage d'adresse est de rigueur !

1) `char *PremiereOccurence (char *str, char occ)` : Renvoie un pointeur sur la première occurrence du caractère `occ` dans la chaine `str`, NULL si ce caractère n'apparait pas. Comparez votre résultat avec la fonction `strchr()`.
2) `int CompareChaine (char *str1, char *str2)` : Renvoie 1 si la chaine `str1` est après `str2` dans l'ordre lexicographique, -1 si elle est avant, 0 si les chaines sont identiques. Comparez le résultat de votre fonction avec `strcmp()`.
3) `int NbOcc (char *str, char c)` : Calcule et renvoie le nombre d'occurrences du caractère `c` dans la chaine de caractères `str`. Utilisez le décalage d'adresse pour arriver à vos fins. Pour calculer son temps d'exécution, nous vous conseillons de l'appeler de nombreuses fois sur une très longue chaine de caractères. Pourquoi la longueur de la chaine importe-t-elle ?
4) Implémentez une fonction vous permettant d'inverser une chaine de caractères de façon récursive. Par exemple la chaine `"bonjour"` devient `"ruojnob"` Encore une fois, c'est à vous de trouver le prototype.

3 Modèle proie-prédateurs

Les équations de Lotka-Volterra introduites en 1925 et 1926 permettent de simuler simplement l'évolution de population d'individus selon un modèle proie-prédateur. Dans ce modèle, les proies se développent naturellement grâce à une ressource illimitée. Ils ne meurent qu'à cause de leurs prédateurs qui eux ne se développent que grâce à la profusion des proies. Ce modèle peut être mis en place pour illustrer l'évènement survenu durant la première guerre mondiale en mer Adriatique.

Avant le début de la guerre, la pêche intensive des sardines permettait de trouver un équilibre entre les populations de sardines et de requins. Le sardines se développaient grâce au plancton en ressource illimitée, et les requins mangeaient alors les sardines. Durant la guerre, il n'y a pas eu de pêche et la population des sardines a alors explosé. Au retour de la guerre, les pêcheurs reviennent, et ont alors la mauvaise surprise de ne pêcher que très peu de sardines ! Comment expliquer ce phénomène ?

Sans prédateur, la population de sardines évolue selon l'équation $S_n = (1 + a)S_{n-1}$, avec a un réel strictement positif permettant de décrire l'augmentation (exponentielle) de la population de sardines. Sans nourriture les requins disparaissent suivant la formule $R_n = (1 - k)R_{n-1}$, avec k un réel strictement positif. Lorsque les deux espèces cohabitent, les sardines disparaissent à cause des requins, et perdent à chaque instant t_n $bS_{n-1}R_{n-1}$ individus, un nombre proportionnel à la population de requins et de sardines à l'instant t_{n-1}. De façon analogue, la population de requins augmente de $cS_{n-1}R_{n-1}$ individus à chaque instant.

Les équations de Lotka-Volterra illustrent alors ces populations pour un instant t_n donné :

$$\begin{cases} S_{n+1} &= S_n(1 + a - bR_n) \\ R_{n+1} &= R_n(1 - k + cS_n) \end{cases} \tag{1}$$

Vous devez implémenter deux fonctions récursives permettant de calculer le nombre de requins et le nombre de sardines à un instant donné. Ces fonctions sont récursives mutuelles et utilisent les formules de Lotka-Volterra vues ci-dessus. Vous pouvez vérifier votre code avec Libre Office, en réalisant une feuille de calcul qui vous permettra de tracer les graphes de ces deux populations au cours du temps. Vous pouvez utiliser les valeurs initiales suivantes pour tester votre code : $a = 0,03$ qui est le taux de croissance de la population de sardines, $b = c = 0,0005$ qui sont les facteurs d'évolution d'une population en fonction de l'autre population et $k = 0,06$ qui est le taux de mortalité des requins.