Programmation Fonctionnelle I, Printemps 2017 – TD8

`http://deptinfo.unice.fr/~roy`

*On rappelle qu'en programmation fonctionnelle pure, les fonctions **itératives** sont implémentées par des **récurrences terminales** ! Ne cherchez pas de mot while, for, etc*

Exercice 8.1 Programmez <u>itérativement</u> la fonction `(que-les-impairs L)` qui prend une liste de nombres entiers `L`, et retourne la sous-liste des entiers impairs, dans l'ordre où ils apparaissent dans `L` . Exemple :

`(que-les-impairs '(3 2 4 9 8 7 6 5))` → `(3 9 7 5)`

Exercice 8.2 a) L'itération est souvent intéressante pour programmer des **fonctions à plusieurs résultats**. Vérifiez-le en programmant de manière itérative la fonction `(long&som L)` retournant deux résultats : la longueur `n` de la liste et la somme `s` de ses éléments, sous la forme d'une liste à deux éléments `(n s)`. *Indication : deux résultats, donc deux accumulateurs…*
b) En déduire la fonction `(moyenne L)` retournant la moyenne inexacte de la liste `L`.

Exercice 8.3 Complétez le texte [cours 8 page 11] de la version *itérative* de `($expt x n)` calculant x^n par dichotomie, avec `n` entier ≥ `0`. Comparez la difficulté de l'écriture de la version itérative par-rapport à celle de la version récursive enveloppée du cours ! Pour un gain négligeable, puisque *l'algorithme* est rapide [ils ont tous deux la même complexité logarithmique].

Indication : Essayez de <u>généraliser</u> cette fonction !

Exercice 8.4 *[PCPS § 6.7]* **L'arithmétique de Peano.** Vous allez vous <u>*interdire*</u> d'utiliser les fonctions arithmétiques primitives usuelles :

`(+ a b)`, `(- a b)`, `(* a b)`, `(/ a b)`, `(= a b)`, `(< a b)` , etc.

Vous allez travailler uniquement dans les **entiers naturels N** et n'aurez droit qu'aux trois fonctions primitives `(zero? a)`, `(add1 a)` qui retourne `a+1`, et `(sub1 a)` qui retourne `a-1`. Vous allez définir des fonctions, donc utiliserez `if`, `cond`, `and`, `local`, etc. bref les constructions linguistiques usuelles de Scheme. Et bien entendu le *principe de récurrence* dans **N** !

a) On considère la fonction `foo` ci-dessous. Dites pourquoi cette fonction n'est *pas* itérative.

```
(define (foo a b)       ; a et b entiers ≥ 0
  (if (zero? a)
      b
      (add1 (foo (sub1 a) b))))
```

b) Déroulez le calcul de `(foo 3 5)`. Quel semble être le résultat de `(foo a b)` en général ?

c) Démontrez-le par récurrence ! *Par récurrence sur qui, sur `a` ou sur `b` ?...*

d) Quelle est la *complexité* du calcul de `(foo a b)` si l'on mesure le nombre d'appels aux primitives `add1` et `sub1` ? Vous noterez $c_{a,b}$ cette complexité, et en vous appuyant sur le texte de `foo`, vous écrirez des équations sur la suite $(c_{a,b})$, que vous résoudrez pour obtenir l'ordre de grandeur du terme général pour `a` et `b` tendant vers +∞.

e) En utilisant `time`, on trouve par exemple que `(foo 20000 20000)` prend 5 ms. A quoi vous attendez-vous [environ] pour les temps de calcul de `(foo 40000 20000)`, `(foo 20000 40000)` et `(foo 40000 40000)` ?

f) Programmez une version itérative `(foo-it a b)` de la fonction `foo`. Elle sera sans doute un peu plus rapide, même si elle a la même complexité [*O(2n) et O(n) sont dans la même classe de complexité : linéaire !*].

g) Programmez la multiplication `(mul a b)` de deux entiers `a` et `b`. Calculez-en la complexité en tenant compte de celle trouvée en d). Quelle est la complexité de votre fonction `mul` ? Pouvez-vous faire mieux ?…

Exercice 8.5 Cherchez de manière itérative une approximation des coordonnées du point M(x,y) sur lequel la fonction $f(x) = x^5 - x^2 - 6x + 1$ atteint son minimum absolu sur l'intervalle [0 , 2]. *Réponse : (1.134, -5.215)*

N.B. i) Aucun rapport avec les dérivées...

ii) Tâchez de généraliser le problème à n'importe quelle fonction. Ceci doit être de l'ordre du réflexe !

Exercice complémentaire

Exercice 8.6 Calculez avec une boucle le <u>numérateur</u> du rationnel *S* ci-dessous (rép: 1446...711).

$$S = \sum_{k=1}^{100} \frac{1}{k}$$

Programmation Fonctionnelle I, Printemps 2017 – TP8

`http://deptinfo.unice.fr/~roy`

On rappelle qu'en programmation fonctionnelle pure, les fonctions ***itératives*** *sont implémentées par des* ***récurrences terminales*** *! Ne cherchez pas de mot while, for, etc qui ne sont pas dans le domaine mathématique, donc en théorie inutiles.*

Exercice 8.1 Programmez itérativement la fonction `(somme L)`, qui retourne la somme des éléments d'une liste de nombres.

Exercice 8.2 Vérifier que l'algorithme `rac2` du cours 8 page 16 donne le résultat avec une précision h = 10^{-8} en 7 tours de boucle seulement ! *Indication : votre fonction rendra non seulement le résultat mais aussi le nombre de tours de boucle pour l'obtenir !*

Exercice 8.3 **Méthode de Newton (déjà codée au S1 en Python).** a) Vous allez généraliser le problème de la recherche de la racine carrée approchée à la Newton [cours 8 pages 14-16], en programmant une fonction itérative `(une-racine f a h)` prenant une fonction dérivable `f` et retournant une valeur approchée d'une solution [supposée exister] de l'équation `f(x) = 0`, en partant d'une approximation `a`, et avec une précision de `h`, c'est à dire une valeur réelle `r` telle que `|f(r)| < h`. Vous conserverez la même architecture de fonction qu'à la page 16, avec trois fonctions locales. La technique utilisée est décrite à la page 14 [calculez l'équation de la tangente], en remplaçant la courbe $y = x^2 - r$ par la courbe $y = f(x)$.

b) *A.N.* Avec `f` bien choisie, calculez une valeur approchée de π [3.141…], puis de $\sqrt{2}$ [1.414…] puis de $\sqrt[3]{2}$ [1.259…] et enfin vous chercherez le point fixe du cosinus, solution de `cos(x) = x` [Rép : 0.739…].

Exercice 8.4 Programmez itérativement une fonction `(miro n)` prenant un entier n ≥ 0 et retournant une image noire de dimensions 200 × 200 dans laquelle sont placés de manière aléatoire n petits carrés de côté 20 et de couleur aléatoire jaune, rouge, ou vert. *Indication : l'accumulateur sera une image.*

Exercices complémentaires mais ô combien gratifiants !

Exercice 8.5 Me promenant dans les rues de Boston[1] il y a quelques années, j'avais vu sur un mur de Kendall Square un graffiti représentant le polynôme $p(x) = x^2 + x + 41$. Intrigué, j'ai calculé sur mon iPhone (avec un toplevel Scheme bien entendu[2]) les premières valeurs $p(0)$, $p(1)$ etc jusqu'à $p(8)$ et quelle ne fut pas ma surprise : toutes ces valeurs étaient des nombres premiers. J'ai pensé que ce polynôme ne fournissait que des nombres premiers mais cela m'a paru impossible[3]. J'ai alors calculé le plus petit entier $n > 0$ tel que $p(n)$ ne soit <u>pas</u> premier. Calculez cette valeur avec une boucle montante et la fonction `(premier? n)` vue dans le cours 7 page 13.

Exercice 8.6 Une ***expérience de Monte-Carlo*** [simulation probabiliste]. Ecrivez une fonction `(fléchette n)` chargée de calculer une approximation du nombre π avec l'expérience suivante. On considère un carré de côté `200` dans lequel est inscrit un cercle de rayon `100`. On jette `n` fois et de manière aléatoire une fléchette dans le carré et on compte le nombre de fois `k` qu'elle tombe dans le cercle. La probabilité de tomber dans le cercle peut être calculée de deux manières : soit par un rapport d'aires, soit par un quotient succès/total. En égalant ces deux manières de calculer, on peut en déduire une valeur approchée de la constante π. Programmez la fonction `(fléchette n)` retournant comme résultat de l'expérience l'approximation de π calculée à partir de `n` et de `k` [la longueur du côté n'intervient pas]. *A.N. n=10000, la convergence est lente !*

Pour en savoir plus : `http://fr.wikipedia.org/wiki/M%e9thode_de_Monte-Carlo`

Programmez l'animation graphique cette expérience ? Ceci a été fait en Python au S1...

Exercice 8.7 Utilisez votre savoir en calcul intégral pour mesurer avec une fonction Scheme la **longueur approchée de l'ellipse** d'équation $\frac{x^2}{2} + \frac{y^2}{9} = 1$ de centre *0*. Le problème est mathématiquement très difficile...

Indication : Vous commencerez par trouver les équations paramétriques x = a cos(t), y = b sin(t) puis vous intègrerez l'élément d'arc ds le long de la courbe, en calculant le ds^2 *(Pythagore + dérivée)... Et vous savez calculer numériquement une intégrale en Scheme (TP5), alors...*

---

[1] Boston est une ville des USA connue en particulier pour la présence du célèbre MIT (Massachusetts Institute of Technology).

[2] Il s'agit de « Gambit REPL », hélas pas de Racket. Mais Gambit est un excellent Scheme canadien !

[3] En fait, le mathématicien russe Yuri Matijasevič a montré en 1970 qu'il existait un polynôme $p(x_1,...x_n)$, à coefficients entiers, dont l'ensemble des valeurs possibles contient <u>tous</u> les nombres premiers ! Mais son résultat ne permet pas de fabriquer concrètement des nombres premiers...

$\boxed{\textbf{Exercice 8.8}}$ La courbe de la fonction $x \mapsto \sin x - x^5$ présente un seul maximum dans l'intervalle [0,1]. Pouvez-vous programmer le calcul d'une bonne approximation de l'abscisse et de l'ordonnée de ce maximum ? *Rép : x ≈ 0.6336, y ≈ 0.4899*