Programmation Fonctionnelle I, Printemps 2017 – TD9

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

*Mot d'ordre : **faire abstraction** de la nature concrète des données, et des traitements particuliers…*

Exercice 9.1 Un programmeur a besoin de « vecteurs du plan ». Il vous demande de lui programmer un *type abstrait* fournissant les fonctions suivantes. Testez sur les vecteurs $\vec{u}\begin{pmatrix}0\\1\end{pmatrix}$ et $\vec{v}\begin{pmatrix}3\\2\end{pmatrix}$.

```
(vec2d x y)          ; retourne un vecteur de coordonnées x et y  (définissez les deux vecteurs u et v pour les tests)
(xcor v)             ; l'abscisse du vecteur v
(ycor v)             ; l'ordonnée du vecteur v
(vec2d? obj)         ; l'objet Scheme obj est-il un « vecteur » ?
------------------------------------------- le type abstrait s'arrête ici ----------------------------------------------
(vec2d->string v)    ; retourne une chaîne de caractères représentant v, par exemple "<x, y>"
(vec+ v1 v2)         ; l'addition vectorielle de v1 et v2
(ext* k v)           ; multiplication d'un réel k par un vecteur v
(prodscal v1 v2)     ; produit scalaire de deux vecteurs
(norme v)            ; longueur d'un vecteur
(unit v)             ; le vecteur unitaire de même sens et direction que le vecteur v
```

Exercice 9.2 **a)** Programmez une version itérative `(sigma5 a f s fini?)` de la fonction `(sigma4 a f s fini?)` développée dans le cours 9 page 15.

b) Pourquoi se limiter à faire des *additions* ? Programmez une dernière généralisation de `sigma5`, que vous nommerez `(accumulation ...)`, dont vous trouverez les bons paramètres, et permettant de faire aussi des multiplications ou toute autre opération. Elle calculera bien entendu aussi tout ce que `sigma5` pouvait calculer auparavant.

c) Comment programmeriez-vous la fonction `(intervalle a b)` prenant deux entiers `a ≤ b`, et retournant la liste des entiers de `[a,b]`, en une ligne, avec `accumulation` ? Exemple :

`(intervalle 10 20)` → `(10 11 12 13 14 15 16 17 18 19 20)`

d) Et le produit des logarithmes des entiers impairs de l'intervalle `[10,30]` ? *Rép : 47383.151…*

Exercice 9.3 **a)** Reprenez la fonction `tri-ins` du tri par insertion (cours 6), et rédigez-la en une ligne sous la forme d'un schéma `reduce`.

b) Reprenez le système de particules du TP 7 et recodez la fonction `dessiner` avec `reduce`.

Exercice complémentaire

Exercice 9.4 **a)** En complément de l'exercice 9.1 :

```
(rot v alpha)        ; le vecteur obtenu par rotation de v d'un angle alpha dans le sens direct
(proj v1 v2)         ; le vecteur projeté orthogonal de v1 sur v2
```

b) Transformez la fonction `(rot v alpha)` en une fonction `(rotation alpha)` retournant la rotation d'angle `alpha`.

*Ce processus de passage d'une fonction à deux paramètres à une fonction à un paramètre se nomme la **curryfication**. Elle peut être automatisée [cf livre de cours PCPS page 117].*

Programmation Fonctionnelle I, Printemps 2017 – TP9

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

Exercice 9.1 **a)** Calculez en une ligne une approximation de la constante e à l'aide d'une série de Taylor avec une quinzaine de termes. *Rép : 2.71828…*

Mathématiquement, il s'agit d'utiliser le développement de e^x en 0 :

$$e^x \approx \sum_{k=0}^{15} \frac{x^k}{k!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^{15}}{15!}$$

b) La formule de Gregory [issue du développement limité de `atan x`] permet de calculer une approximation de π :

$$\pi/4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + \ldots$$

Programmez en une ligne une fonction `(gregory n)` calculant cette somme en utilisant `n` termes. Vérifiez en faisant n=10, 100, 1000, que cette somme converge très lentement…

c) Ecrire à l'ordre supérieur [en utilisant `apply`, `map`…] la fonction `(integrale f a b n)` retournant une valeur approchée de l'intégrale de `f` sur `[a,b]`, en découpant à la Riemann `[a,b]` en `n` sous-intervalles de même largeur. Vérifiez que l'intégrale du sinus sur [0, π] est bien proche de 2.

Exercice 9.2 Si l'on représente un vecteur de $\mathbf{R}^n$ par une **liste** de `n` coordonnées directement sans passer par un type abstrait, comment écririez-vous en une ligne l'addition `($vec+ v1 v2)`, la loi externe `($ext* k v)`, ainsi que le produit scalaire `($prodscal v1 v2)` ? Exemple : `($prodscal '(3 1 2 -1) '(1 1 -3 1))` → `-3`.

Exercice 9.3 Comment programmeriez-vous la fonction primitive `(filter pred? L)` si elle n'existait pas en Scheme ?

Deux solutions : récursive enveloppée, puis itérative.

Exercice 9.4 Programmez en une ligne la fonction `(maxListe L)` retournant le plus grand élément d'une liste L de nombres réels. Exemple : `(maxListe '(7 3 4 8 2 1 5))` → `8`

Exercice 9.5 Programmez les fonctions `(andmap pred L)` et `(ormap pred L)` si elles n'existaient pas en Racket ? Il s'agit de l'exercice 8.13.21 du livre de cours PCPS.

N.B. Ces deux fonctions `andmap` *et* `ormap` *sont extraordinairement pratiques pour programmer la fonction `final?` d'un système de particules...*

Exercices complémentaires

Exercice 9.6 Prendre la **n-ème itérée** d'une fonction `f` consiste à l'appliquer successivement `n` fois de suite. Par exemple l'image de `x` par la 3-ème itérée de `f` est `f(f(f(x)))`.

a) Définissez la fonction `(repeated f n)` retournant la `n`-ème itérée de la fonction `f`. Le résultat est donc une fonction.

b) Utilisez `repeated` pour définir en une ligne la fonction $n \mapsto 2^n$ avec `n` entier ≥ 0.

c) Utilisez `repeated` pour calculer la valeur de la fraction :

$$K = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{7}}}}$$

d) Utilisez `repeated` pour calculer une approximation de la limite de la suite $(u_n)_{n\geq 0}$ définie par $u_0 = 2015$ et $u_n = \dfrac{1}{1+u_{n-1}}$.

Exercice 9.7 Etudiez l'éditeur de polygone à la souris développé pages 141-142 du livre de cours PCPS *(question d'examen).*