IFT2035 – Travail pratique #2 – 3 novembre 2016

PROGRAMMATION FONCTIONNELLE

Marc Feeley

Le TP2 a pour but de vous faire pratiquer la programmation fonctionnelle et en particulier les concepts suivants : les fonctions récursives, la forme itérative, les continuations, et le traitement de liste. Vous devez reproduire fidèlement le comportement du programme du TP1 et comparer l'approche fonctionnelle avec l'approche impérative.

Pour ce travail vous devez utiliser uniquement le sous-ensemble fonctionnel de Scheme (en particulier, vous ne devez pas utiliser les formes-spéciales "`set!`" et "`begin`" dans votre programme, ni les fonctions prédéfinies contenant un point d'exclamation, comme "`set-car!`"). Le but du travail étant de vous faire pratiquer les concepts de programmation fonctionnelle, utilisez des fonctions d'ordre supérieur lorsque c'est approprié. L'élégance de votre codage est un facteur important dans l'évaluation de ce travail.

Vous avez à réaliser un programme en Scheme ainsi que rédiger un rapport contenant une analyse du programme.

---

1 Programmation

Vous devez réaliser en Scheme avec Gambit l'application spécifiée dans le TP1. Vous devez exploiter les forces du langage Scheme pour exprimer au mieux votre programme. Le dictionnaire contenant les variables et leur valeur doit être représenté avec une liste d'association.

Votre programme Scheme doit avoir le même comportement que celui spécifié dans le TP1, à l'exception de l'opérateur "`?`" qui n'a plus de sens en Scheme (car la gestion mémoire est faite automatiquement). D'autre part utilisez l'incitateur "`#`" au lieu de "`>`". Cela facilitera le déboguage car Gambit utilise l'incitateur "`>`" (donc vous pourrez facilement distinguer une interaction avec Gambit et une interaction avec votre programme).

D'autre part, s'il y a une erreur dans l'expression entrée par l'usager, aucune des affectations de variable ne doit être exécutées. Par exemple :

```
# 100 =a
100
# 999 =a b *
b n'a pas de valeur
# a
100
# b
b n'a pas de valeur
```

Votre programme doit être robuste. Les expressions peuvent être arbitrairement longues (votre programme ne doit donc pas imposer de limite sur leur longueur).

Votre programme Scheme doit être entièrement contenu dans un fichier dont le nom est "`tp2.scm`". Vous devez utiliser le fichier "`tp2.scm`" de la page Studium du cours comme point de départ et seulement modifier la première section. L'exécution du programme doit se faire au moyen de l'interprète `gsi` avec

la commande "`gsi tp2.scm`", ou vous pouvez faire "`chmod +x tp2.scm`" et ensuite lancer l'exécution avec "`./tp2.scm`". Pour déboguer le programme interactivement, démarrez l'interprète `gsi` puis faites `(load "tp2.scm")` et ensuite `(main)`. Vous avez avantage à utiliser `emacs` pour éditer vos fichiers et faire le déboguage. Les instructions relatives à l'utilisation de Gambit dans `emacs` sont données ici : `http://www.iro.umontreal.ca/~gambit/doc/gambit.html#Emacs-interface` .

2 Rapport

Vous devez rédiger un rapport qui:

1. Explique brièvement le fonctionnement général du programme (maximum de 1 page au total).
2. Explique comment les problèmes de programmation suivants ont été résolus (en 2 à 3 pages au total):
   (a) comment se fait l'analyse syntaxique d'une expression et comment est réalisé le traitement d'une expression de longueur quelconque
   (b) comment se fait le calcul de l'expression
   (c) comment se fait l'affectation aux variables
   (d) comment se fait l'affichage des résultats et erreurs
   (e) comment se fait le traitement des erreurs
3. Compare votre expérience de développement avec le TP1 (en 1 à 2 pages au total). Combien de lignes de code ont vos programmes C et Scheme? Sans tenir compte de votre niveau de connaissance des langages C et Scheme, quels sont les traitements qui ont été plus faciles et plus difficiles à exprimer en Scheme? Indépendamment des particularités de Scheme, pour quelles parties du programme l'utilisation du style de programmation fonctionnel a-t'il été bénéfique et pour quelles parties détrimental? Pour quelles parties avez vous utilisé des récursions en forme itérative? Pour quelles parties avez vous utilisé des continuations?

3 Évaluation

- Ce travail compte pour 15 points dans la note finale du cours. Indiquez vos noms clairement au tout début du programme. **Vous devez faire le travail par groupes de 2 personnes. Vous devez confirmer la composition de votre équipe (noms des coéquipiers) au démonstrateur. Si vous ne trouvez pas de partenaire d'ici quelques jours, venez me voir.**
- Le programme sera évalué sur 6 points et le rapport sur 9 points. Un programme qui plante à l'exécution, même dans une situation extrême, se verra attribuer zéro sur 6 (c'est un incitatif à bien tester votre programme). Assurez-vous de prévoir toutes les situations d'erreur.
- Vous devez remettre un fichier ".tar" qui contient uniquement deux fichiers : votre rapport (qui doit se nommer "rapport.pdf") et le programme (qui doit se nommer "tp2.scm"). La remise doit se faire au plus tard à 23h55 vendredi le 9 décembre sur le site Studium du cours. En supposant que vos deux fichiers sont dans le répertoire "tp2", vous pouvez créer le fichier ".tar" avec la commande "`tar cf tp2.tar tp2`".
- L'élégance et la lisibilité du code, l'exactitude et la performance, la lisibilité du rapport, et l'utilisation d'un français sans fautes sont des critères d'évaluation.