TYPO & CMP
Typologie des Langages
Construction des Compilateurs

EPITA – Promo 2007 – **Tous documents autorisés** *

Mai 2005 (1h30)

Une copie synthétique, bien orthographiée, avec un affichage clair des résultats, sera toujours mieux notée qu'une autre demandant une quelconque forme d'effort de la part du correcteur. Une argumentation informelle mais convaincante, sera souvent suffisante.

1 Typologie des Langages

1. Pourquoi les pointeurs sont-ils dangereux en C ?
2. Pourquoi les références en C++ sont-elles des pointeurs policés ?
3. Pourquoi les unions sont-elles dangereuses en C ?
4. Quelle restrictions sur les membres d'une union le C++ pose-t-il ?
5. Qu'est-ce qui motive cette limitation ?
6. Proposer un mécanisme généralisant les unions du C pour fonctionner pour « tout » le C++. On s'intéressera particulièrement à (i) l'installation, (ii) la copie, (iii) la destruction.
7. À quoi ressembleraient des fonctions unaires qui prennent en argument une telle union généralisée ?

2 Construction des Compilateurs : Désucrage sucré

1. Qu'est-ce que le sucre syntaxique ?
2. Qu'est-ce que le désucrage ?
3. Citer les 5 phases d'une partie frontale d'une compilateur comme `tc`.
4. Discuter deux emplacements possibles d'une phase de désucrage, et en comparer les mérites respectifs.
5. On considère l'ajout de + entre chaînes de caractères dans `tc` comme sucre pour la primitive `concat`. On autorise par exemple `print (dir + "/" + file)`.
   Où placer sa phase de désucrage ?
6. Proposer une implémentation pour désucrer ce cas. On ne demande pas le code, mais les grandes lignes de la mise en œuvre.
7. Quelle difficulté introduit l'appel à une fonction ou à une primitive ? On pensera particulièrement à la nécessité pour la suite du compilateur de ne pas avoir à se préoccuper du fait qu'il y ait eu, ou pas eu, de phase de désucrage : il compte sur la liaison des noms.

*"Tout document autorisé" signifie que notes de cours, livres, annales, etc. sont explicitement consultables pendant l'épreuve. Le zèle de la part des surveillants n'a pas lieu d'être, mais dans un tel cas contacter le LRDE au 01 53 14 59 22.

8. Pourquoi les macros du C ne sont pas (toujours) une bonne réponse pour introduire du sucre ?
9. Traduire

   ```
   for i := l to u do b
   ```

   en une boucle `while`, en conservant à l'esprit les points suivants.
   - Les métavariables *i*, *l*, *u*, et *b* représentent du *code*, pas nécessairement des variables Tiger. Elles sont en quelque sorte les AST fils du `ForExp`.
   - Si $l > u$ le corps de la boucle n'est pas évalué.
   - Si *l* ou *u* vaut `MAX_INT`, le comportement doit être correct et ne doit pas boucler indéfiniment sous prétexte que `MAX_INT + 1 = MIN_INT`.
   - Introduire de nouveaux noms comporte le risque de masquer ceux de l'utilisateur.
   - 9 vient après 8.
   - Le correcteur aime voir *une* réponse, pas une succession de tentatives rayées, blancotées, stabilotées, ciseautées, agrafées, tronçonneuseotées.
10. Pour votre information, notre première implémentation de cette transformation comprend 104 lignes de pénible instantiation à la main de nœuds de la hiérarchie `ast::Ast`. Elle est donc bien sûr difficile à lire, donc à contrôler. Que proposez-vous pour faciliter cette transformation ? On demande ici les grands principes d'une implémentation tenant compte de tous les problèmes soulevés dans la question 9.
11. Critiquer la solution que vous proposez à la question 10.