CMP2 – Construction des compilateurs

EPITA – Apprentis Promo 2010
Tous documents (notes de cours, polycopiés, livres) autorisés.
Tous dispositifs de calcul automatisé (calculatrice, téléphone portable, ordinateur, etc.) interdits.

Juin 2008 (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 Incontournables

Il n'est pas admissible d'échouer sur une des questions suivantes : **chacune induit une pénalité sur la note finale**. Répondez sur les feuilles de QCM qui vous sont remises (n'oubliez pas d'y inscrire votre nom ou login).

1. `sizeof(getc())` est une expression valide en C. A. Vrai/B. Faux ?
2. On peut coder jusqu'à 256 valeurs différentes avec un octet signé. A. Vrai/B. Faux ?
3. `type_t *x;` ne peut-être qu'une déclaration de variable en C. A. Vrai/B. Faux ?
4. En C++, on ne peut pas redéfinir un opérateur << en dehors d'une classe (c'est-à-dire, en tant que fonction non-membre). A. Vrai/B. Faux ?

2 Parallélisation du compilateur

Les machine modernes disposent de processeurs multi-cœur, et on souhaite tirer parti de ce parallélisme facilement accessible dans notre compilateur Tiger. Nous allons limiter la portée de l'exercice au *front-end*, car le *back-end* n'a pas été vu en cours.

Les réponses informelles (mais sensées) sont acceptées ; l'objectif est de réfléchir et de faire jouer votre connaissance du compilateur. Répondre simplement « oui » ou « non » n'est pas suffisant.

1. Que peut-on paralléliser dans le scanner et le parser de `tc` ?
2. Est-ce que la liaison des noms (*binding*) peut bénéficier d'une accélération via la parallélisation ? Comment ? (Bien se rappeler comment l'Abstract Syntax Tree (AST) est parcouru !)
3. De même, peut-on paralléliser le *type-checking* ? Comment ? (Même remarque.)
4. Comment peut-on paralléliser la traduction vers la représentation intermédiaire ?
5. Intuitivement, on se dit qu'une tâche nécessitant un temps d'exécution t sur un processeur mono-cœur s'effectuera *au mieux* en un temps $\frac{t}{n}$ sur un processeur à n cœurs (tous les cœurs étant identiques, bien entendu) : c'est une *accélération linéaire*. Il existe cependant certaines opération qui peuvent bénéficier d'une accélération encore meilleure, c'est-à-dire s'exécutant en un temps *inférieur ou égal* à $\frac{t}{n}$; on dit alors que l'accélération est *superlinéaire*. À votre avis, quels types de traitements rentrent dans ce cas ? Pourquoi ?

6. Quelle(s) tâche(s) parallélisable pourraient tirer parti d'une accélération superlinéaire dans notre compilateur Tiger ?

3 Pointeurs

Dans cet exercice, on s'intéresse à l'adjonction de *pointeurs* au langage Tiger.

1. Au cours de l'enseignement que vous avez suivi, vous avez été mis en garde plusieurs fois contre les dangers que peuvent présenter les pointeurs dans un langage comme le C ou le C++. Rappelez ces problèmes que peuvent poser les pointeurs.
2. Néanmoins, les pointeurs représentent une fonctionnalité utile dans un langage de programmation. Que peuvent-ils apporter ?
3. Beaucoup de langages modernes ont banni les pointeurs, et leur ont préféré un substitut plus sûr. Quel est-il ? En quoi ce remplaçant est-il plus sûr ?
4. Le substitut aux pointeurs de la question précédente remplace avantageusement ceux-ci dans la majorité des cas. Citez un cas où les pointeurs permettent de faire « plus de choses ». (N.B. : Cette question recoupe – à dessein – la question 2, et il se peut que votre réponse ait déjà été mentionnée à cette précédente question, mais ce n'est pas grave.)
5. **Grammaire.** Comment étendez-vous la grammaire de `tc` pour lui ajouter le support des pointeurs ?
6. **AST.** Comment modifiez-vous les classes de la syntaxe abstraite ? (*Hint* : Des diagrammes UML sont parfois plus parlant que du code ou qu'une longue explication.)
7. **Noms.** Que faut-il changer dans le calcul de liaisons des noms ?
8. **Typage.** Que faut-il ajouter et/ou changer dans l'étape de vérification des types ?
9. **Traduction.** L'ajout des pointeurs à `tc` a-t-il une incidence sur la traduction vers la représentation intermédiaire ?
10. **Question bonus.** À votre avis, est-ce que les pointeurs nécessitent une modification du *back-end* (génération de code, allocation de registres, etc.) ? Si oui, laquelle/lesquelles ? Si non, pour quelle(s) raison(s) ?

4 À propos de ce cours

Pour terminer cette épreuve, nous vous invitons à répondre à un petit questionnaire.

Les renseignements ci-dessous ne serons bien entendu pas utilisés pour noter votre copie. Ils ne sont pas anonymes, car nous souhaitons pouvoir confronter réponses et notes. En échange, quelques points seront attribués pour avoir répondu. Merci d'avance.

Sauf indication contraire, vous pouvez cocher plusieurs réponses par question. Répondez sur les feuilles de QCM qui vous sont remises. N'y passez pas plus de dix minutes.

Le cours

5. Quelle a été votre implication dans le cours (CCMP & TYLA) ?
   - **A** Rien.
   - **B** Bachotage récent.
   - **C** Relu les notes entre chaque cours.
   - **D** Fait les annales.
   - **E** Lu d'autres sources.
6. Le cours
   - **A** Est incompréhensible et j'ai rapidement abandonné.
   - **B** Est difficile à suivre mais j'essaie.
   - **C** Est facile à suivre une fois qu'on a compris le truc.
   - **D** Est trop élémentaire.
7. Ce cours
   - **A** Ne m'a donné aucune satisfaction.
   - **B** N'a aucun intérêt dans ma formation.
   - **C** Est une agréable curiosité.
   - **D** Est nécessaire mais pas intéressant.
   - **E** Je le recommande.
8. La charge générale du cours (relecture de notes, compréhension, recherches supplémentaires, etc.) est
   - **A** Légère (quelques minutes par semaine).
   - **B** Supportable (environ une heure de travail par semaine).
   - **C** Lourde (plusieurs heures par semaine).
   - **D** Telle que je n'ai pas pu suivre du tout.

Les formateurs

9. L'enseignant de CCMP & TYLA
   - **A** N'est pas pédagogue.
   - **B** Parle à des étudiants qui sont au dessus de mon niveau.
   - **C** Me parle.
   - **D** Se répète vraiment trop.
   - **E** Se contente de trop simple et devrait pousser le niveau vers le haut.
10. Les assistants
    - **A** Ne sont pas pédagogues.

B Parlent à des étudiants qui sont au dessus de mon niveau.

C M'ont aidé à avancer dans le projet.

D Ont résolu certains de mes gros problèmes, mais ne m'ont pas expliqué comment ils avaient fait.

E Pourraient viser plus haut et enseigner des notions supplémentaires.

Le projet Tiger

11. Vous avez contribué au développement du compilateur de votre groupe (une seule réponse attendue) :

    A Presque jamais.

    B Moins que les autres.

    C Équitablement avec vos pairs.

    D Plus que les autres.

    E Pratiquement seul.

12. La charge générale du projet Tiger est

    A (J'ai été dispensé du projet.)

    B Légère (une ou deux heures par semaine).

    C Supportable (plusieurs heures de travail par semaine).

    D Lourde (plusieurs jours de travail par semaine).

    E Telle que je n'ai pas pu suivre du tout.

13. Y a-t-il de la triche dans le projet Tiger ? (Une seule réponse attendue.)

    A Pas à votre connaissance.

    B Vous connaissez un ou deux groupes concernés.

    C Quelques groupes.

    D Dans la plupart des groupes.

    E Dans tous les groupes.

Questions 14-20 Le projet Tiger vous a-t-il bien formé aux sujets suivants ? Répondre selon la grille qui suit. (Une seule réponse attendue par question.)

A Pas du tout

B Trop peu

C Correctement

D Bien

E Très bien

14. Formation au C++
15. Formation à la modélisation orientée objet et aux *design patterns*.
16. Formation à l'anglais technique.
17. Formation à la compréhension du fonctionnement des ordinateurs.
18. Formation à la compréhension du fonctionnement des langages de programmation.
19. Formation au travail en collaboration.
20. Formation aux outils de développement (contrôle de version, systèmes de construction, débogueurs, générateurs de code, etc.

Questions 22-33 Comment furent les étapes du projet (ne pas répondre à celles que vous n'avez pas faites). Répondre selon la grille suivante. (Une seule réponse attendue par question.)

A Trop facile.
B Facile.
C Nickel.
d Difficile.
E Trop difficile.

21. Mini-projet en Tiger (LZW)

22. `TC-0`, Scanner & Parser.
23. `TC-1`, Scanner & Parser en C++, Autotools.
24. `TC-2`, Construction de l'AST.
25. `TC-3`, Liaison des noms.
26. `TC-4`, Typage.
27. Désucrage des constructions objets (transformation Tiger → Panther)
28. `TC-5`, Traduction vers représentation intermédiaire.

29. Option `TC-E`, Calcul des échappements.
30. Option `TC-A`, Surcharge des fonctions.
31. Option `TC-D`, Suppression du sucre syntaxique (boucles `for`, comparaisons de chaînes de caractères).
32. Option `TC-B`, Vérification dynamique des bornes de tableaux.
33. Option `TC-I`, Mise en ligne du corps des fonctions.