CMP – Construction des compilateurs

EPITA – Apprentis promotion 2012
Tous documents (notes de cours, polycopiés, livres) autorisés.
Calculatrices et ordinateurs interdits.

Juin 2010 (1h30)

Bien lire les questions, chaque mot est important. Écrire court, juste, et bien. Une argumentation informelle mais convaincante est souvent suffisante.

Une lecture préalable du sujet est recommandée.

1 Programmation orientée objet

1. Définissez les termes suivants :
   (a) méthode
   (b) méthode polymorphe
   (c) multiméthode
2. Donner un cas d'utilisation de multiméthode. Quelques exemples :
3. Citer un langage qui dispose de multiméthodes.
4. Comment peut-on simuler le mécanisme d'une multiméthode (à deux arguments) dans un langage qui en est dépourvu ?
5. Qu'appelle-t-on un objet-fonction en C++ ?
6. Écrivez la définition d'un objet-fonction C++ représentant la fonction mathématique

$$f_a : \left\{ \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & \sin(ax) \end{array} \right. \quad \text{avec } a \in \mathbb{R}.$$

2 Langages

1. De quel type de passage par argument dispose-t-on en langage C ?
2. Même question avec le langage Tiger.
3. En C++, quels sont les bénéfices apportés par un passage par référence ?
4. Qu'appelle-t-on polymorphisme de surcharge ? Quelle différence y a-t-il avec le polymorphisme d'inclusion ?
5. Pour écrire (resp. lire) des données dans (resp. depuis) un flux, on utilise l'opérateur '<<' (resp. '>>') en C++. Ces opérateurs sont définis sous forme de fonctions surchargées (à deux arguments), et non de méthodes surchargées (à un argument). Pour quelle raison ?
6. Pourquoi les opérateurs de flux '<<' et '>>' sont-ils plus sûrs que la fonction `printf` ?

3 Compilation

Voici un programme Tiger incorrect :

```
let
 function fib (n : int) : int =
  if n <= 1
  then 1;
  else fib (n − 1) + fib (n − 2);
in
 fib (10);
end
```

1. Quelle est la nature du problème ici ? (Dit autrement : quel composant du compilateur va détecter qu'il y a une erreur ?)
2. Proposez une correction de ce programme.
3. Le comportement par défaut d'un couple scanner-parser lors d'une erreur lexicale ou grammaticale est d'arrêter l'analyse. Lors du projet Tiger, vous avez dû utiliser une technique disponible dans Bison permettant malgré tout d'obtenir (le plus souvent) un arbre de syntaxe abstraite (évidemment faux), représentant partiellement le programme en entrée.
   (a) Comment s'appelle cette technique ?
   (b) Comment la met-on en œuvre dans un parser ?
   (c) Comment fonctionne-t-elle ?
4. Quelle différence faites-vous entre un arbre de dérivation et un arbre de syntaxe abstraite (ou Abstract Syntax Tree (AST)) ?
5. Donnez une représentation de l'AST du programme Tiger ci-dessus (après correction). Un aide-mémoire comportant une liste (partielle) des nœuds de la syntaxe abstraite de Tiger est disponible en annexe A.
6. Sur le schéma de la question précédente, indiquer les liens entre sites d'utilisations et sites de définitions à l'aide de flèches partant des premiers et allant vers les seconds.
7. L'implémentation de `fib` ci-dessus est foncièrement inefficace. Pourquoi ? Proposez une implémentation plus rapide.

4 À propos de ce cours

Pour terminer cette épreuve, nous vous invitons à répondre à un petit questionnaire. Les renseignements ci-dessous ne seront 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 la feuille de QCM qui vous est remise. N'y passez pas plus de dix minutes.

Cette épreuve

1. Sans compter le temps mis pour remplir ce questionnaire, combien de temps ce partiel vous a-t-il demandé (si vous avez terminé dans les temps), ou combien vous aurait-il demandé (si vous aviez eu un peu plus de temps pour terminer) ?
   A Moins de 30 minutes.
   B Entre 30 et 60 minutes.
   C Entre 60 et 90 minutes.
   D Entre 90 et 120 minutes (estimation).
   E Plus de 120 minutes (estimation).
2. Ce partiel vous a paru
   A Trop difficile.
   B Assez difficile.
   C D'une difficulté normale.
   D Assez facile.
   E Trop facile.

Le cours

3. Quelle a été votre implication dans les cours de Construction des compilateurs ?
   A Rien.
   B Bachotage récent.
   C Relu les notes entre chaque cours.
   D Fait les annales.
   E Lu d'autres sources.
4. Ce 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.
5. 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.

6. La charge générale du cours en sus de la présence en amphi (relecture de notes, compréhension, recherches supplémentaires, etc.) est
   A Telle que je n'ai pas pu suivre du tout.
   B Lourde (plusieurs heures par semaine).
   C Supportable (environ une heure de travail par semaine).
   D Légère (quelques minutes par semaine).

Les formateurs

7. L'enseignant
   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.
8. 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

9. 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.
10. La charge générale du projet Tiger est
    A Telle que je n'ai pas pu suivre du tout.
    B Lourde (plusieurs jours de travail par semaine).
    C Supportable (plusieurs heures de travail par semaine).
    D Légère (une ou deux heures par semaine).
    E J'ai été dispensé du projet.
11. 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 12-18 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

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

Questions 19-30 Comment furent les étapes du projet ? Répondre selon la grille suivante. (Une seule réponse attendue par question ; ne pas répondre pour les étapes que vous n'avez pas faites.)

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

19. Rush `.tig` : mini-projet en Tiger (Bistromatig).
20. `TC-0`, Scanner & Parser.
21. `TC-1`, Scanner & Parser, Tâches, Autotools.
22. `TC-2`, Construction de l'AST et pretty-printer.
23. `TC-3`, Liaison des noms et renommage.
24. `TC-4`, Typage et calcul des échappements.
25. `TC-5`, Traduction vers représentation intermédiaire.
26. Option `TC-D`, Suppression du sucre syntaxique (boucles `for`, comparaisons de chaînes de caractères).
27. Option `TC-I`, Mise en ligne du corps des fonctions.
28. Option `TC-B`, Vérification dynamique des bornes de tableaux.
29. Option `TC-A`, Surcharge des fonctions.
30. Option `TC-O`, Désucrage des constructions objets (transformation Tiger → Panther). .

A Classes de la syntaxe abstraite du langage Tiger

Voici une copie du fichier 'src/ast/README' fourni avec le code de tc.

```
Tiger Abstract Syntax Tree nodes with their principal members.
Incomplete classes are tagged with a '*'.

/Ast/               (Location location)
  /Dec/             (symbol name)
    FunctionDec     (VarDecs formals, NameTy result, Exp body)
      MethodDec     ()
    RuleDec         (Exp match, Exp build)
    TypeDec         (Ty ty)
    VarDec          (NameTy type_name, Exp init)

  /Exp/             ()
*   /Var/
      CastVar       (Var var, Ty ty)
*     FieldVar
      SimpleVar     (symbol name)
      SubscriptVar  (Var var, Exp index)

*   ArrayExp
*   AssignExp
*   BreakExp
*   CallExp
*     MethodCallExp
    CastExp         (Exp exp, Ty ty)
    ForExp          (VarDec vardec, Exp hi, Exp body)
*   IfExp
    IntExp          (int value)
*   LetExp
    MetavarExp      ()
    NilExp          ()
*   ObjectExp
    OpExp           (Exp left, Oper oper, Exp right)
*   RecordExp
*   SeqExp
*   StringExp
    WhileExp        (Exp test, Exp body)

  /Ty/              ()
    ArrayTy         (NameTy base_type)
    ClassTy         (NameTy super, DecsList decs)
    NameTy          (symbol name)
*   RecordTy

  DecsList          (decs_type decs)

  Field             (symbol name, NameTy type_name)

  FieldInit         (symbol name, Exp init)
```

```
Some of these classes also inherit from other classes.

/Escapable/
  VarDec            (NameTy type_name, Exp init)

/Metavariable/      (unsigned id)
  MetavarExp        ()

/Typable/
  /Dec/             (symbol name)
  /Exp/             ()
  /Ty/              ()

/TypeConstructor/
  /Ty/              ()
  FunctionDec       (VarDecs formals, NameTy result, Exp body)
  TypeDec           (Ty ty)
```