CMP1 – Construction des compilateurs

EPITA – Apprentis promo 2010
Tous documents (notes de cours, polycopiés, livres) autorisés

Février 2008 (1h00)

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**.

1. La surcharge de fonctions en C++ est un mécanisme qui dépend des types à l'exécution. Vrai ou faux ?
2. L'utilisation de fonctions virtuelles en C++ est incompatible avec la compilation séparée. Vrai ou faux ?
3. Parmi ces propositions, laquelle (lesquelles) est (sont) un pré-requis pour le typage statique fort dans un langage orienté objet ?
   (a) une liaison des noms résolue à la compilation ;
   (b) l'absence d'instructions de transtypage (*cast*) ;
   (c) l'absence de méthodes polymorphes abstraites (appelées fonctions membres virtuelles pures en C++) ;
   (d) la présence automatique d'une surclasse au sommet de toute hiérarchie de classes (`Object`, par exemple).
4. Quelle est le type (de Chomsky) du langage engendré par

$$S \to X|Y \qquad X \to Zp \qquad Y \to o \qquad Z \to pS$$

2 Langages, grammaires et compilation

1. Quel est l'avantage de l'algorithme GLR sur LALR ?
2. Pourquoi est-ce aussi un désavantage dangereux ?
3. Que signifie AST en anglais et en français ?
4. Donner la syntaxe abstraite (soit sous la forme d'une grammaire, soit sous la forme d'une hiérarchie orientée objet typée) de la grammaire suivante.

```
<term> ::= <term> ( <term> )        -- Application
         | λ <var> . <term>         -- Abstraction
         | <var>                    -- Variable
<var>  ::= <identifier>
```

On utilise `--` pour introduire des commentaires qui bien sûr n'appartiennent pas à la grammaire.

5. Pourquoi un langage comme le C inclut les "prédéclarations", et pas un autre comme Java ?

6. Étant donné que le programme Tiger suivant est valide, expliquer quelle politique de gestion mémoire le compilateur doit utiliser pour les types.

```
let
  var box :=
    let
      type box = { val: string }
    in
      box { val = "42\n" }
    end
in
  print (box.val)
end
```

7. Soient deux fractions $\frac{a}{b}$, $\frac{c}{d}$, à quelle surprise s'expose-t-on si l'on programme $\frac{a}{b} \leq \frac{c}{d}$ comme `(a/b) <= (c/d)`.

3 Un peu de C++

1. Qu'affiche le programme suivant :

```
#include <iostream>
struct Foo
{
  void m1 () const
  {
    std::cout << "Foo::m1()" << std::endl;
  }
  virtual void m2 () const
  {
    std::cout << "Foo::m2()" << std::endl;
  }
};
struct Bar : public Foo
{
  void m1 () const
  {
    std::cout << "Bar::m1()" << std::endl;
  }
  virtual void m2 () const
  {
    std::cout << "Bar::m2()" << std::endl;
  }
};
```

```
int main ()
{
 const Bar& a = Bar (); a.m1 (); a.m2 ();
 const Foo& b = Bar (); b.m1 (); b.m2 ();
 const Foo& c = Foo (); c.m1 (); c.m2 ();
}
```

2. Écrire un *class template* `Pair` qui stocke deux éléments, `a` et `b`, d'un même type, avec un constructeur prenant les deux valeurs.

3. Utiliser la classe précédente pour déclarer une variable `pi` qui contienne la paire $(3, 14159)$.

4. Qu'est-ce qu'un `const_iterator` dans STL ?

5. Écrire une fonction `product` qui prenne une liste STL de `float` par référence, et en retourne le produit des éléments. On prendra garde à la constance et au `namespace`.