Cahier des charges pour la réalisation d'un compilateur Lobo

Jin-Kao Hao, Jean-Michel Richer
M1 Informatique, UE1
Année 2012-2013

1 But du projet

L'objectif du projet vise à mettre en pratique les connaissances acquises dans le cadre des modules de l'UE1 (Génie Logiciel, Gestion de Projet, Test et Qualité du logiciel).

2 Sujet

On désire réaliser un compilateur pour le langage `Lobo` (un dérivé du langage Logo). Le compilateur devra prendre en entrée un programme Lobo et le traduire en un programme C++ qui sera compilé puis exécuté.

Le compilateur prend en entrée un fichier `a.lobo` et génère un fichier `a.cpp`. Le fichier c++ doit pouvoir être compilé simplement par `g++ -o a.exe a.cpp`. On indiquera le cas échéant s'il faut ajouter d'autres paramètres ou on fournira un utilitaire pour la compilation.

Un programme **Lobo** est une liste de fonctions et d'instructions dont voici une grammaire :

```
program := { instruction }
instruction := definition-fonction | appel-fonction |
    repete | instruction-simple ';'

definition-fonction := FONCTION identifiant PARAMETRES { identifiant }
DEBUT { instructions ';' } RETOUR expression ';' FIN_FONCTION
appel-fonction := APPEL identifiant [ parametres ] ';'
parametres := { parametre }
parametre := identifiant | nombre
repete := nombre DEBUT { instruction } FIN_REPETE
instruction-simple := declare-variable |
    affecte-variable |
    affiche-chaine |
    affiche-variable

declare-variable := DECLARE identifiant
affecte-variable := AFFECTE identifiant expression
affiche-chaine := AFFICHE '"' .... '"'
affiche-variable := AFFICHE identifiant

expression := nombre | operateur-binaire expression expression

operateur-binaire := '+' | '-' | '*' | '/'
```

```
identifiant := [a-zA-Z]+[a-zA-Z_]*
nombre := [0-9]+
```

Voici un exemple de programme **Lobo** qui affiche i^3 pour des valeurs de i variant de 1 à 10 :

```
# fichier cube.golo
# fonction sans argument
FONCTION nouvelle_ligne
PARAMETRES
DEBUT
    AFFICHE "\n" ;
    RETOUR 0 ;
FIN_FONCTION

# fonction avec un argument, affiche le cube
FONCTION affiche_cube
PARAMETRES n
DEBUT
    DECLARE c ;
    AFFECTE c * n * n n ;
    AFFICHE n ;
    AFFICHE " au cube = " ;
    AFFICHE c ;
    APPEL nouvelle_ligne ;
    RETOUR 0 ;
FIN_FONCTION

DECLARE i ;
AFFECTE i 1 ;
REPETE 10
DEBUT
    APPEL affiche_cube i ;
    AFFECTE i + i 1 ;
FIN_REPETE
```

La traduction en C++ de ce programme est :

```
# fichier cube.cpp
#include <iostream>
using namespace std;

int nouvelle_ligne() {
    cout << "\n";
    return 0;
}
```

```
int affiche_cube(int n) {
    int c;
    c = (n*n)*n;
    cout << n;
    cout << " au cube ";
    cout << c;
    nouvelle_ligne();
    return 0;
}

int main() {
    int i;
    i = 1;
    for (int loop = 1; loop <= 10; ++loop) {
        affiche_cube(i);
        i = i + 1;
    }
    return 0;
}
```

On notera que :

- les seules données manipulées sont des entiers et des chaînes pour l'instruction AFFICHE
- certaines fonctions peuvent ne pas avoir de paramètres
- il n'y a pas de structure if-then
- les expressions sont au format *notation polonaise inverse* avec des opérateurs binaires uniquement
- la structure de boucle REPETE peut être imbriquée, c'est à dire qu'on peut avoir un REPETE à l'intérieur d'un autre REPETE comme dans l'exemple suivant :

```
REPETE 10
DEBUT
    REPETE 5
    DEBUT
        AFFICHE "*" ;
    FIN_REPETE
FIN_REPETE
```

3 Outils et méthodologie

Nous conseillons, afin de faciliter le développement, d'utiliser les outils suivants :

- un environnement intégré de développement style KDevelop ou Eclipse
- réaliser des tests unitaires (cpp unit)
- utiliser make pour la compilation automatique

4 Documents à rendre et notation

Vous obtiendrez 3 notes concernant :

- pour la partie **Génie Logicel** : diagrammes de cas d'utilisation, diagrammes de classes, architecture du site
- pour la partie **Test et qualité** : tests unitaires, fiabilité du logiciel
- pour la partie **Gestion de Projet** : la rédaction d'un document de synthèse relatif à son ingénierie (avec notamment une présentation de l'équipe, de l'organisation des environnements de développement, du planning, du PERT, des outils adoptés pour le suivi et le pilotage...)

Contraintes annexes :

- le projet (compilateur et document) est à rendre pour le 20 décembre 2012 dernier délai en me l'envoyant par email sous la forme d'une archive `.tgz` obtenue en archivant le répertoire de votre projet grâce à la commande `tar -cvzf ...`
- vous pouvez travailler par groupe de 2 à 4 personnes maximum
- le compilateur doit pouvoir être compilé avec g++ et exécuté sous Linux Ubuntu 12.04