TP compilation : IMP

Le langage IMP introduit des instructions de contrôle structurées, à savoir les branchements conditionnels et les boucles.

1 Description du langage IMP

La principale nouveauté du langage IMP par rapport au langage IMP est le remplacement partiel des instructions de saut `jump` et `jump/when` par des instructions `if` et `while`.

1.1 Spécification

1.1.1 Expressions

Les expressions de IMP sont identiques aux expressions de ART.

1.1.2 Instructions

Les instructions `nop`, `exit`, `print` et `:=` conservent la même forme que dans ART (et doivent être suivies d'un point-virgule).

Les nouvelles instructions `if` et `while` sont ajoutées à la grammaire des instructions selon les formes suivantes :

$$
\begin{array}{rcl}
\mathit{instr} & ::= & \ldots \\
& | & \texttt{if (}\ \mathit{expr}\ \texttt{) \{}\ \mathit{instr}^*\ \texttt{\} else \{}\ \mathit{instr}^*\ \texttt{\}} \\
& | & \texttt{while (}\ \mathit{expr}\ \texttt{) \{}\ \mathit{instr}^*\ \texttt{\}}
\end{array}
$$

où les cinq symboles `if`, `else`, `while`, `{` et `}` sont de nouveaux symboles terminaux, et où instr^* désigne une succession d'un nombre arbitraire d'instructions. Contrairement aux instructions de base, ces deux formes ne sont pas suivies du point-virgule.

Enfin, on conserve la possibilité de définir des étiquettes selon la forme habituelle, ainsi qu'une instruction `goto` de saut inconditionnelle, équivalent à l'instruction précédente `jump` mais avec la syntaxe légèrement modifiée suivante.

$$
\begin{array}{rcl}
\mathit{instr} & ::= & \ldots \\
& | & \texttt{goto (}\ \mathit{l\_expr}\ \texttt{) ;}
\end{array}
$$

1.1.3 Programmes

Un programme IMP à la même forme qu'un programme ART, mais en appliquant la nouvelle définition des instructions.

1.2 Exemple

Voici un exemple de programme IMP.

```
.text
  while (a < 123) {
    print(a);
    print(10);
    a := a+1;
  }
  exit;
.data
```

```
a : 97
```

2 Structure du compilateur

On réalisera un compilateur de IMP vers ART en trois parties.

1. Un analyseur syntaxique, qui transforme un programme source IMP en un arbre de syntaxe abstraite (modules `IMPLexer` et `IMPParser`).
2. Un traducteur, qui traduit la syntaxe abstraite IMP en syntaxe abstraite ART (module `IMPtoART`).
3. Un afficheur, qui transforme la syntaxe abstraite ART en programme source ART.

Les syntaxes abstraites sont définies dans les modules suivants :

`Op`	opérateurs utilisés dans les expressions
`IMPExpr`	expressions, communes à IMP et ART
`IMPInstr`	instructions IMP
`IMP`	programmes IMP
`ART`	instructions et programmes ART

L'afficheur est réparti dans les mêmes modules que les syntaxes abstraites qu'il affiche, à savoir `ART`, `IMPExpr` et `Op`.

3 Objectifs du TP

3.1 Contrat de base

Compléter les fichiers présentés dans la section précédente pour obtenir un compilateur traduisant un programme IMP en un programme ART équivalent. En particulier :

1. Compléter les fichiers `IMPLexer.mll` et `IMPParser.mly` pour obtenir un analyseur qui à partir d'un programme source IMP produit un arbre de syntaxe abstraite tel que défini dans `IMP.ml`.
2. Compléter le fichier `IMPtoART.ml` pour effectuer une traduction de la syntaxe abstraite IMP vers la syntaxe abstraite ART.
3. Compléter le fichier `ART.ml` pour finaliser l'afficheur de syntaxe abstraite `ART`.

Les fichiers `IMP.ml`, `IMPInstr.ml`, `IMPExpr.ml` et `Op.ml` n'ont pas besoin d'être complétés.

3.2 Extensions

3.2.1 Boucle `for`

Étendre le langage IMP avec une boucle `for`. Cette nouvelle forme de boucle devra être ajoutée sous la forme d'un sucre syntaxique, c'est-à-dire sans rien ajouter à la syntaxe abstraite définie dans le module `IMPInstr`. Les seuls fichiers à modifier seront donc `IMPLexer.mll` et `IMPParser.mly`.

3.2.2 Instructions `break` et `continue`

Étendre le langage IMP avec les deux instructions suivantes :

— `break`, qui interrompt une boucle en cours,

— `continue`, qui passe au tour suivant d'une boucle en cours.

En cas de boucles imbriquées, chacune de ces deux instructions s'applique à la boucle la plus interne.

Cette fois il faut également modifier les fichiers `IMPInstr.ml` et `IMPtoART.ml`.