Devoir maison n°9

À rendre lundi 09/02

⋆ ⋆ ⋆

Transpilation

1 Présentation du problème

On s'intéresse au langage de programmation fictif OCalme, dont l'extension de fichier est `.oklm`, et dont le lexique et la syntaxe sont simples et décrits ci-après :

- les **caractères blancs** sont les caractères espace (`' '`), tabulation (`'\t'`), nouvelle ligne (`'\n'`) et retour chariot (`'\r'`). Les lexèmes sont des suites de caractères entre deux suites de caractères blancs consécutives ;
- les lexèmes sont les suivants :
  - `init`, `print`, `while`, `end` sont des mots-clés du langage ;
  - `+`, `-` et `=` sont des symboles du langage ;
  - les variables sont des lexèmes commençant par la lettre `'v'` et suivies par une suite de chiffres ;
  - les entiers naturels sont des lexèmes ne contenant que des chiffres.

Par ailleurs, la syntaxe, et l'interprétation associée, sont données de la manière suivante :

- une suite de caractères blancs est interprétée comme un seul caractère espace : l'indentation et les sauts de ligne ne sont là que pour améliorer la lisibilité ;
- pour une variable `var` (dont le nom vérifie les contraintes précédentes) :
  - `init var` déclare la variable et l'initialise à 0 ;
  - `print var` affiche la valeur de la variable et revient à la ligne ;
  - `+ var` incrémente la variable ;
  - `- var` décrémente la variable ;
  - si `w` est une autre variable, `= var w` affecte à la variable `var` la valeur actuelle de `w` ;
  - si `n` est un entier naturel, `= var n` affecte à la variable `var` la valeur de n ;
  - si *prog* est un programme (donc une suite d'instructions parmi celles décrites ici), `while var` *prog* `end` exécute le contenu du programme *prog* tant que la variable `var` est strictement positive.

Par exemple, le programme suivant calcule et affiche la somme des entiers de 1 à 100 (donc 5050) :

```
init v1
init v2
init v3
= v1 100
while v1
    - v1
    + v2
    init v4
    = v4 v2
    while v4
        - v4
        + v3
    end
end
print v3
```

La variable `v1` correspond au compteur garantissant que la boucle est de taille 100. La variable `v2` correspond à l'entier courant (entre 1 et 100) qu'on ajoute à la somme. La variable `v3` correspond à la somme en cours de calcul. La variable `v4` correspond à la variable `v2`, mais qu'on s'autorise à modifier le temps de l'ajouter à `v3`.

2 L'objectif

On cherche à écrire **en OCaml** un **transpilateur** du langage OCalme **vers C**. La transpilation est simplement le nom donné à la traduction entre deux langages de haut niveau (contrairement à la compilation, qui désigne plutôt la traduction d'un langage haut niveau vers un langage bas niveau, comme l'assembleur ou le langage machine).

En pratique, on veut écrire un fichier `transpilateur.ml` (donc en OCaml) tel que, s'il existe un fichier `prog.oklm`, l'exécution dans la console de l'interprétation :

```
ocaml transpilateur.ml prog
```

ou de la compilation et exécution :

```
ocamlc -o transpilateur transpilateur.ml && ./transpilateur prog
```

crée un fichier `prog.c` dont la compilation et l'exécution correspondent à l'exécution du programme écrit dans le fichier `prog.oklm`.

3 Ce qu'il faut rendre

Pour ce devoir, il faut rendre une (courte) copie papier et un fichier `transpilateur_NOM_Prenom.ml` (avec votre nom et prénom) appliquant le principe décrit précédemment.

La copie papier devra contenir des explications de code et les réponses aux questions suivantes.

1. Décrire comment est effectuée l'analyse lexicale.
2. Décrire la grammaire hors-contexte utilisée pour faire l'analyse syntaxique.
3. Expliquer l'algorithme d'analyse syntaxique utilisé.
4. Expliquer l'algorithme permettant de convertir l'arbre d'analyse syntaxique en programme C.
5. Pour chacun des fichiers `.oklm` fournis sur le dépôt github (le fichier `prog1.oklm` correspondant au programme décrit précédemment) :
   (a) indiquer lesquels contiennent des erreurs lexicales ou des erreurs syntaxiques, et les corriger de la manière qui vous semble la plus logique/simple le cas échéant ;
   (b) déterminer empiriquement (ou en analysant le code) ce que calculent ces programmes (corrigés !) en fonctions de la valeur initiale donnée à la variable `v1`.

⋆ ⋆ ⋆