Projet de compilation: présentation du langage FLO

1 Objectif

L'objectif de ce projet est de concevoir en binôme un compilateur qui prend en entrée le langage Flo et donne en sortie du langage assembleur `nasm` (qui sera lui-même compilé pour être utilisé comme programme exécutable). Vous êtes en théorie libre de programmer le compilateur dans le langage de votre choix, mais les TP vous guideront pour le réaliser en `python` ou en `c`.

2 Le langage FLO

Le langage FLO est un langage de programmation minimaliste inventé spécialement pour ce cours. Voici ses principales caractéristiques (qui pourraient éventuellement évoluer) :

Fichier source Un programme FLO tient dans exactement un fichier $.flo$.

Programme Un programme FLO est une suite de définitions de fonctions (éventuellement vide), suivi d'une liste d'instructions.

Types Le langage FLO connaît deux types de variables :
- Le type `entier` qui correspond aux nombres entiers positifs ou négatifs.
- Le type `booleen` qui correspond à Vrai ou Faux.

Opérateurs Le langage Flo connaît les opérateurs suivants :
- arithmétiques : `+`, `-`, `*`, `/` (division entière), `%` (modulo)
- comparaison : ==, ! =, <, >, <=, >=
- logiques : `et`, `ou`, `non`

Instructions Le langage FLO connaît les instructions suivantes :
- Déclaration `Type nomVariable;` où `Type` peut-être booleen ou entier.
- Affectation `nomVariable = expression;`
- Déclaration-Affectation `Type nomVariable = expression;`
- Instruction conditionnelle :

```
si(condition1){
listeInstructions1
}
sinon si(condition2){
listeInstructions2
}
...
sinon{
listeInstructionN
}
```

  où `condition`i est une expression booléenne et avec les `sinon si` et `sinon` optionnels.
- Instruction boucle `tantque(condition) { listeInstructions }`.
- Instruction `retourner` *expression* `;` (dans une fonction uniquement)
- Appel de fonction `nomFonction(arg1,...,argn)`

Fonctions — Le type de la fonction peut être entier ou booléen.
- — Le passage des arguments se fait par valeur.
- — Les fonctions peut posséder des variables locales.
- — Une fonction ne peut pas être déclarée à l'intérieur d'une autre.
- — On peut ignorer le résultat rendu par une fonction.

Portée des variables Il n'y a pas de variables globales en FLO. À l'intérieur d'une fonction, on peut simplement utiliser les variables définies dans la fonction et les arguments de la fonction. Plus précisément, si une variable est définie dans un bloc d'instructions (dans une boucle `tantque` , à l'intérieur d'un si), elle n'est visible que dans ce bloc.

Fonctions prédéfinies Les entrées-sorties de valeurs entières se font à l'aide de deux fonctions prédéfinies, `maVariable = lire();` et `ecrire(maVariable);`.

Commentaires Il est possible d'ajouter des commentaires à du code FLO. Ces commentaires commencent par un caractère dièse (`#`) et terminent à la fin de la ligne. Le caractère dièse et tout ce qui suit jusqu'à la fin de la ligne (`\n`) doit être ignoré par le compilateur.

Voici un exemple de programme en FLO :

```
entier fibo(entier n){
si(n<=1){
  retourner 1;
}
retourner fibo(n-1)+fibo(n-2);
}

entier n = lire();
ecrire(fibo(n));
```