# Interprétation des programmes – TP 1 : MiniHopix, analyse lexicale, syntaxique et interprétation

Université Paris Diderot – Master 1

(2016-2017)

Cette séance de travaux pratiques a pour objectifs :
— De vous faire découvrir l'arbre de sources du projet.
— De vous amener à comprendre puis modifier un analyseur lexical, un analyseur syntaxique et un interpréteur.
Vous devez faire des *commits* réguliers (à chaque modification de votre code) pour que nous puissions suivre votre avancement.

**Exercice 1 (Découverte de flap)** *Explorez le code source de* FLAP *pour répondre aux questions suivantes.*

1. *Quel exécutable est produit par la compilation de l'arbre source ? Quelles sont ses options ?*
2. *Quel est le module qui décide quelles actions effectuées en fonction des options ?*
3. *Quelle différence faites-vous entre le mode batch et le mode interatif d'utilisation du compilateur ?*
4. *Quel est le rôle du module* `Languages` *? Dans ce compilateur, quels sont tous les constituants qui caractérisent un langage ?*
5. *Quel est le rôle du module* `Compilers` *?*
6. *Quels sont les modules qui définissent le langage* HOPIX *? Quel est leur rôle respectif ?*
7. *Reproduisez et expliquez la séquence d'utilisation suivante :*

```
        Flap version 16.1

flap> val x = 2 + 3 * 2
x = 8
flap> val y = 2 / 3
y = 5
flap> val z = 3 - 3
z = 9
```

*(Corrigez le programme pour obtenir des résultats plus corrects.)*

□

**Exercice 2 (Premiers contacts avec l'analyseur lexical)**

1. *Modifiez l'analyseur lexical et l'analyseur syntaxique pour reconnaître les mots-clés* ***true***, ***false***, ***if***, ***then*** *et* ***else****.*
2. *Les commentaires en* HOPIX *sont similaires à ceux du langage* C *: on peut les introduire avec /∗ et les clore avec ∗/ et chose importante, on peut les imbriquer ! Modifiez l'analyseur lexical pour qu'il ignore les commentaires du code source.*

□

**Exercice 3 (Premiers contacts avec l'analyseur syntaxique)**

1. *Modifiez le type de l'arbre de syntaxe abstraite pour y intégrer un constructeur* `LBool of bool` *dans le type des litéraux.*
2. *Modifiez la règle d'analyse syntaxique des litéraux pour reconnaître les litéraux* ***true*** *et* ***false****.*

3. *Modifiez la règle d'analyse syntaxique des expressions pour y intégrer la reconnaissance de la règle suivante :*

   ```
   expression ::= 'if' expression 'then' expression 'else' expression
   ```

□

**Exercice 4 (Premiers contacts avec l'interpréteur)**

1. *Modifiez le type des valeurs produites par l'interpréteur en y rajoutant un constructeur* `VBool of true` *correspondant à une valeur booléenne. Utilisez le typechecker d'OCaml pour mettre à jour le code impacté par ce changement.*
2. *Modifiez l'évaluateur des expressions pour traiter les litéraux* ***true*** *et* ***false*** *ainsi que le* ***if-then-else****.*

□

**Exercice 5 (En avant pour les conflits)**

1. *Modifiez la grammaire pour reconnaître les opérateurs binaires* <=, >=, =, < *et* > *d'une façon similaire à celle de* +, −, ∗ *et* /.
2. *Quels sont les warnings produits par* MENHIR *?*
3. *À l'aide de la documentation de* MENHIR*, trouvez une façon de supprimer ces warnings.*

□