**PARADIGMES ET LANGAGES DE PROGRAMMATION**

**Rapport**

**Projet**

Christen & Mottier

# Objectif

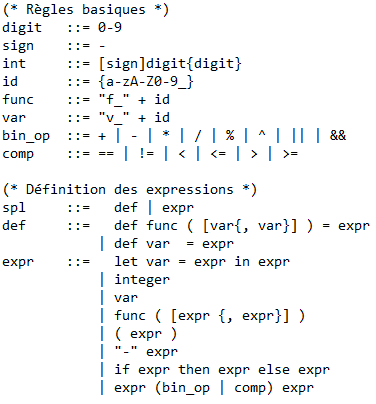
Produire un programme Haskell exécutable permettant de lire des expressions et des définitions de fonction et de les interpréter.

# Fonctionnalités

Notre langage de programmation propose :

* Des opérations arithmétiques habituels et originales,
* Des opérations relationnelles (comparaisons),
* La notion de variable et des énoncés de type « let »,
* La notion de fonction (dont les fonctions « factorielle » et « Fibonacci »)

# Grammaire



Quelque chose à dire ?

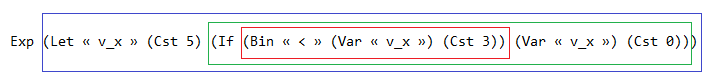
# Résultats intermédiaires

Pour la ligne « let v\_x = 5 in if v\_x < 3 then v\_x else 0 », nous avons :

* Liste des lexèmes :

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TLet | TVar « v\_x » | TSym « = » | TInt 5 | TIn | TIf | TVar « v\_x » | TComp « < » | TInt 3 | TThen | TVar « v\_x » | TElse | TInt 0 |

* Arbre syntaxique :



* Résultat de l’expression : **0**



i - Exemple de résultat d'un "let"

# Résultats du calcul des fonctions

Exemple de nos fonctions :

* Factorielle :



* Fibonacci :

