Forestier Quentin & Herzig Melvyn

PLP – 03.12.2020

Interpréteur

Laboratoire 9-10-11



# Grammaire

Pour définir notre version de SPL nous avons dû définir une grammaire à notre langage. Voici sa structure :

|  |  |  |  |
| --- | --- | --- | --- |
| **prg** → { def } expr  **def** → **define** funName { var } = expr  **expr** → **let** var = expr **in** expr  | **if** expr **then** expr **else** expr  | expr (**+**|**-**|**\***|**/**|**<**|**#**) expr  | **(** (+|-) expr **)**   | integer  | varName  | funName **(** {expr} **)**  | **(** expr **)**  **varName** -> (a-z){A-Za-z0-9}  **funName** -> (A-Z){A-Za-z0-9}  **integer** -> (0-9){0-9} | Symbole de départ :  **prg**  Ensemble des symboles non terminaux :  **Prg**  **Def**  **Expr**  **VarName**  **funName**  **integer**  Ensemble des symboles terminaux :   |  |  | | --- | --- | | [A-Z]  [a-z]  [0-9]  + - \* / = < # ( ) | let in if then else define | |

# Lexèmes

Pour que le lexer puisse analyser le contenu des programmes nous avons défini les tokens suivants :

tokens :-  
 $white+ ;  
 let { \s -> TLet } -- Symbole let  
 in { \s -> TIn } -- Symbole in  
 if { \s -> TIf} -- Symbole if  
 then { \s -> TThen} -- Symbole then  
 else { \s -> TElse} -- Symbole else  
 define { \s -> TDefine} -- Symbole define  
 $digit+ { \s -> TInt (read s) } -- Nombres entiers positifs  
 [\=\+\\*\-\/\<\#] { \s -> TSym (head s) } -- Symboles  
 $low{1}$alphaNum\* { \s -> TVarName s } -- Noms de variables  
 $up{1}$alphaNum\* { \s -> TFunName s} -- Noms de fonctions  
 [\(] { \s -> TOp} -- Parenthèse ouvrante  
 [\)] { \s -> TCp} -- Parenthèse fermante

Nous avons ensuite implémenté le parser conformément à la grammaire démontrée et aux tokens.  
(C.F Parser.y)

# Tests