Forestier Quentin & Herzig Melvyn

PLP – 05.12.2020

Interpréteur

Laboratoire 9-10-11



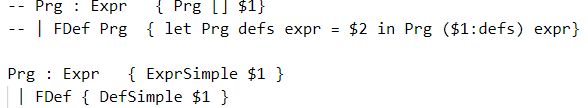
# Grammaire

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 **)**  | # 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 | |

# Parser

Conformément à notre grammaire, le parser retournait un Prg formé de 0 ou plusieurs définitions de fonctions et forcément une instruction. Pour faire fonctionner l’interpréteur nous avons modifié un Prg pour être soit une expression soit une définition de fonction.



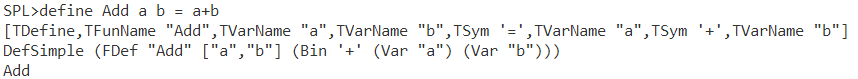


# Fonctionnement

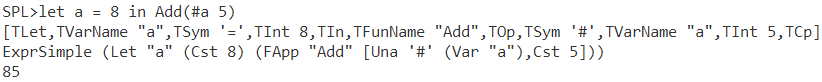
Lancement de l’interpréteur. La fonction launchRepl construit les fonctions par défauts et lance le vrai REPL.



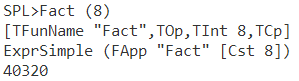
Test de définition d’une fonction.

Pour une simple fonction à deux paramètres qui les additionnes :  
  
Les lexèmes obtenus ainsi que l’arbre syntaxique sont corrects.

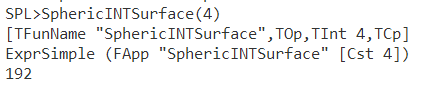
# Tests d’expressions



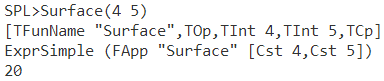
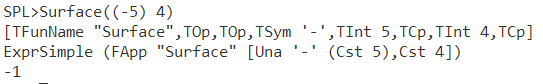
Cette expression est correctement traduite et interprétée. Nous pouvons y voir notre opérateur personnalisé ‘#’ qui, préfixé à une expression, multiplie par 10. L’appel de fonction est fonctionnel.



L’utilisation de la fonction factorielle est implémentée et fonctionnelle.



Comme fonction personnalisée nous avons défini qu’elle calculait l’air d’un rectangle. Si un des deux paramètres est inférieur à 0 elle retourne -1 sinon l’air issu des deux paramètres.



# Conclusion

Au terme de ce laboratoire, notre version de SPL est correctement interprétée. Les opérateurs unaires qui sont également binaire doivent être couplés à des parenthèses pour fonction (p.ex : *(-4)*). Les appels de fonction sans paramètre demandent tout de même l’utilisation de parenthèses vides. Un appel à une fonction qui fourni plus de paramètres que demandé, résulte en la non-évaluation des paramètres excédants. Notre langage fonctionne uniquement avec des expressions fermées. Si une variable demandée n’existe pas pour l’expression courante, nous la remplaçons par un 0 plutôt que de lever une exception.