Forestier Quentin & Herzig Melvyn

PLP – 14.01.2021

Compilateur

Laboratoire 13 - 14

# /Users/Yves/Desktop/Logo HEIG-VD_CMJN 9,6x20_4L.pdf

# Grammaire

|  |  |  |  |
| --- | --- | --- | --- |
| **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 | |

# Fonctionnement

Afin de produire le code objet d’un programme, lancer la commande start Compilation fichier Compiler.hs



La fonction invoquée lit le fichier « input.txt » qui contient le code à traduire puis elle écrit le code objet dans le fichier « codeObjet.txt ». Le fichier « input.txt » doit être situé dans le même répertoire que « Compiler.hs ».

Exécuter la machine abstraite disponible dans le fichier « main.cpp » en passant le fichier « codeObjet.txt » en argument du programme.

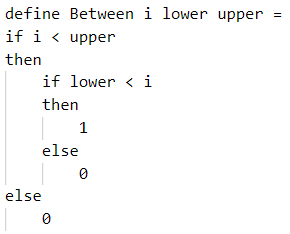
# Annexes

* Fonction personnalisée (dans ce document)
* Fonction factorielle (dans ce document)
* Programme source (dans ce document et fichier input.txt )
* Code objet profuit (dans ce document et fichier codeObje.txt)t

# Fonction personnalisée

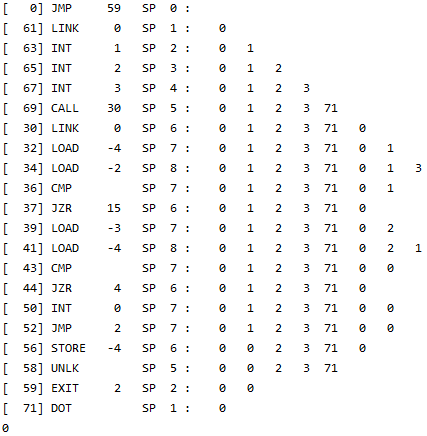
Nous avons mis en place une fonction « Between i lower upper » qui vérifie si « i » est strictement inclus entre lower et upper. Si c’est le cas retourne 1 sinon 0.

## Définition



## Trace

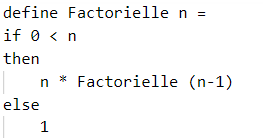
Pour « Between(1 2 3) »



# Fonction Factorielle

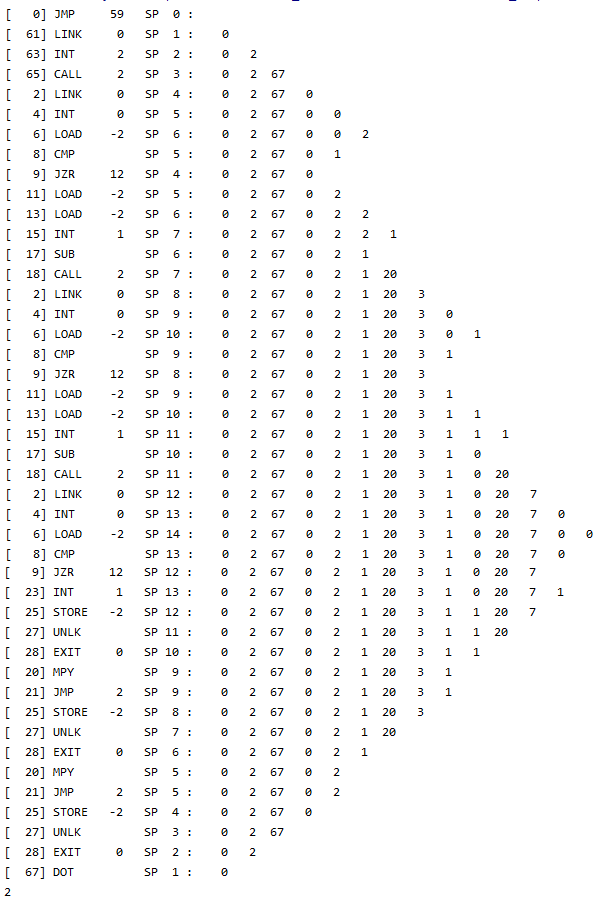
Nous avons implémenté la fonction « Factorielle n » qui effectue la factorielle de « n ».   
Si « n » est <= 0, le fonction retourne 1.

## Définition



## Trace

Pour « Factorielle (2) » dais raisons de lisibilité.



# Programme complet et code objet

|  |  |
| --- | --- |
| define Factorielle n = if 0 < n  then   n \* Factorielle (n-1) else  1  define Between i lower upper = if i < upper then  if lower < i  then   1  else  0  else   0  let x = Factorielle(3) in if x  then 1  else 0 | JMP 59  LINK 0  INT 0  LOAD (-2)  CMP  JZR 12  LOAD (-2)  LOAD (-2)  INT 1  SUB  CALL 2  MPY  JMP 2  INT 1  STORE (-2)  UNLK  EXIT 0  LINK 0  LOAD (-4)  LOAD (-2)  CMP  JZR 15  LOAD (-3)  LOAD (-4)  CMP  JZR 4  INT 1  JMP 2  INT 0  JMP 2  INT 0  STORE (-4)  UNLK  EXIT 2  LINK 1  INT 3  CALL 2  STORE 1  LOAD 1  JZR 4  INT 1  JMP 2  INT 0  DOT  HALT |