HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Compilateur

LABORATOIRE 13 - 14

Forestier Quentin & Herzig Melvyn

PLP – 20.01.2021

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]	let
[a-z]	in
[0-9]	if
+ - * = <	then
( )	else
	define

Fonctionnement

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

```
*Compiler> startCompilation
```

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 produit (dans ce document et fichier codeObje.txt)

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

```
define Between i lower upper =
if i < upper
then
    if lower < i
    then
        1
    else
        0
else
    0
```

Trace

Pour « Between(1 2 3) »

```
[   0] JMP     59   SP  0 :
[  61] LINK     0   SP  1 :    0
[  63] INT      1   SP  2 :    0   1
[  65] INT      2   SP  3 :    0   1   2
[  67] INT      3   SP  4 :    0   1   2   3
[  69] CALL    30   SP  5 :    0   1   2   3  71
[  30] LINK     0   SP  6 :    0   1   2   3  71   0
[  32] LOAD    -4   SP  7 :    0   1   2   3  71   0   1
[  34] LOAD    -2   SP  8 :    0   1   2   3  71   0   1   3
[  36] CMP          SP  7 :    0   1   2   3  71   0   1
[  37] JZR     15   SP  6 :    0   1   2   3  71   0
[  39] LOAD    -3   SP  7 :    0   1   2   3  71   0   2
[  41] LOAD    -4   SP  8 :    0   1   2   3  71   0   2   1
[  43] CMP          SP  7 :    0   1   2   3  71   0   0
[  44] JZR      4   SP  6 :    0   1   2   3  71   0
[  50] INT      0   SP  7 :    0   1   2   3  71   0   0
[  52] JMP      2   SP  7 :    0   1   2   3  71   0   0
[  56] STORE   -4   SP  6 :    0   0   2   3  71   0
[  58] UNLK         SP  5 :    0   0   2   3  71
[  59] EXIT     2   SP  2 :    0   0
[  71] DOT          SP  1 :    0
0
```

Fonction Factorielle

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

Définition

```
define Factorielle n =
if 0 < n
then
    n * Factorielle (n-1)
else
    1
```

Trace

Pour « Factorielle (2) ».

```
[   0] JMP     59   SP  0 :
[  61] LINK     0   SP  1 :    0
[  63] INT      2   SP  2 :    0   2
[  65] CALL     2   SP  3 :    0   2  67
[   2] LINK     0   SP  4 :    0   2  67   0
[   4] INT      0   SP  5 :    0   2  67   0   0
[   6] LOAD    -2   SP  6 :    0   2  67   0   0   2
[   8] CMP          SP  5 :    0   2  67   0   1
[   9] JZR     12   SP  4 :    0   2  67   0
[  11] LOAD    -2   SP  5 :    0   2  67   0   2
[  13] LOAD    -2   SP  6 :    0   2  67   0   2   2
[  15] INT      1   SP  7 :    0   2  67   0   2   2   1
[  17] SUB          SP  6 :    0   2  67   0   2   1
[  18] CALL     2   SP  7 :    0   2  67   0   2   1  20
[   2] LINK     0   SP  8 :    0   2  67   0   2   1  20   3
[   4] INT      0   SP  9 :    0   2  67   0   2   1  20   3   0
[   6] LOAD    -2   SP 10 :    0   2  67   0   2   1  20   3   0   1
[   8] CMP          SP  9 :    0   2  67   0   2   1  20   3   1
[   9] JZR     12   SP  8 :    0   2  67   0   2   1  20   3
[  11] LOAD    -2   SP  9 :    0   2  67   0   2   1  20   3   1
[  13] LOAD    -2   SP 10 :    0   2  67   0   2   1  20   3   1   1
[  15] INT      1   SP 11 :    0   2  67   0   2   1  20   3   1   1   1
[  17] SUB          SP 10 :    0   2  67   0   2   1  20   3   1   0
[  18] CALL     2   SP 11 :    0   2  67   0   2   1  20   3   1   0  20
[   2] LINK     0   SP 12 :    0   2  67   0   2   1  20   3   1   0  20   7
[   4] INT      0   SP 13 :    0   2  67   0   2   1  20   3   1   0  20   7   0
[   6] LOAD    -2   SP 14 :    0   2  67   0   2   1  20   3   1   0  20   7   0   0
[   8] CMP          SP 13 :    0   2  67   0   2   1  20   3   1   0  20   7   0
[   9] JZR     12   SP 12 :    0   2  67   0   2   1  20   3   1   0  20   7
[  23] INT      1   SP 13 :    0   2  67   0   2   1  20   3   1   0  20   7   1
[  25] STORE   -2   SP 12 :    0   2  67   0   2   1  20   3   1   1  20   7
[  27] UNLK         SP 11 :    0   2  67   0   2   1  20   3   1   1  20
[  28] EXIT     0   SP 10 :    0   2  67   0   2   1  20   3   1   1
[  20] MPY          SP  9 :    0   2  67   0   2   1  20   3   1
[  21] JMP      2   SP  9 :    0   2  67   0   2   1  20   3   1
[  25] STORE   -2   SP  8 :    0   2  67   0   2   1  20   3
[  27] UNLK         SP  7 :    0   2  67   0   2   1  20
[  28] EXIT     0   SP  6 :    0   2  67   0   2   1
[  20] MPY          SP  5 :    0   2  67   0   2
[  21] JMP      2   SP  5 :    0   2  67   0   2
[  25] STORE   -2   SP  4 :    0   2  67   0
[  27] UNLK         SP  3 :    0   2  67
[  28] EXIT     0   SP  2 :    0   2
[  67] DOT          SP  1 :    0
2
```

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
```