Nom : Djebien
Prénom : Tarik
Groupe : 2
Sujet : COMPIL-TP1- Analyseur Lexical du langage AVA.
Date : 22/09/2010

Le Langage AVA.

1. Variables, types et expressions.

i. *Les variables AVA.*

déclaration des variables :
$nomDeVariable \in L([A-Za-z\_].[A-Za-z0-9\_]*)$
Unicité :
$\forall (i,j) \in [\![1,n]\!]^2,\ (i \neq j) \Rightarrow (vari \neq varj).$
Initialisation par défaut :
$\forall i \in [\![1,n]\!],\ (\text{int}\ vari = 0;) \wedge (boolean\ vari = FALSE;).$

ii. *Les types AVA*

.
$avaTypes \in \{\text{int}, boolean\}$
$\text{int}\ vari \Rightarrow vari \in IDENT$
$boolean\ varj \Rightarrow varj \in IDBOOL$

iii. *Les expressions AVA.*

a) les expressions Arithmétique :

$E \rightarrow E+E \mid E-E \mid E*E \mid E\ /\ E \mid E\ mod\ E \mid -E \mid (E) \mid$
$E=E \mid E\ /= E \mid E <= E \mid E < E \mid E > E \mid E >= E \mid IDENT \mid ENTIER$
$IDENT \rightarrow [:jletter:].[:jletterdigit:]*$
$ENTIER \rightarrow [:digit:]+$

Priorités des opérateurs arithmétiques noté prio $\{Op_a\}$:
$prio\{()\} \geq prio\{-_u\} \geq prio\{mod\} \geq prio\{*, /\} \geq prio\{+, -_b\}$

Priorités des opérateurs de comparaisons noté prio $\{Op_c\}$:
$prio\{=\} = prio\{/=\} = prio\{<=, >=\} = prio\{<, >\}$

a) les expressions Booléenne :

$B \rightarrow B\ \text{and}\ B \mid B\ \text{or}\ B \mid \text{not}\ B \mid (B) \mid IDBOOL \mid BOOLEEN$

Priorités des opérateurs booléens noté prio $\{Op_b\}$:
$prio\{not\} \geq prio\{\text{and}\} \geq prio\{\text{or}\}$

Finalement, on a :
$prio\{Op_a\} \geq prio\{Op_c\} \geq prio\{Op_b\}$

2. **Instructions AVA**

Définitions :
Soit « expr » une expression AVA et on définit : $X_{expr} = \{expr\}$
Soit « exprA » une expression AVA Arithmétique et on définit : $A_{expr} = \{exprA\}$
Soit « exprB » une expression AVA Booléenne et on définit : $B_{expr} = \{exprB\}$
D'où : $X_{expr} = A_{expr} \cup B_{expr}$

i. *Affectation :*

On a la syntaxe suivante :

nomDeVariable := expr ;
avec $expr \in X_{expr}$ *et* $var \in \{IDENT \cup IDBOOL\}$

$(v \in IDENT) \wedge (v := expr) \Rightarrow expr \in A_{expr}$
$(w \in IDBOOL) \wedge (w := expr) \Rightarrow expr \in B_{expr}$

ii. *Impression sur la sortie standard (StdOut)*

write(%i , exprEnt) ;	avec	$exprEnt \in A_{expr}$
write(%b , exprBool) ;	avec	$exprBool \in B_{expr}$
write(%s , chaineDeCaracteres) ;	avec	$chaineDeCaracteres \simeq Java.lang.String$

%i, %b, %s sont appelés « formats d'impressions » .
\ : caractère d'échappement.
\n : retour à la ligne.

$writeln() \Leftrightarrow (write() + \text{\n})$

iii. *Lecture sur l'entrée standard (StdIn)*

Soit nomDeVariable $\in$ *IDENT alors on a la syntaxe suivante* :
read nomDeVariable ;

iv. *Structure Conditionnelle*

Soit $exprBool \in B_{expr}$ *alors on a la syntaxe suivante* : ,

if exprBool then listeInstruction end if ;
if exprBool then listeInstruction else listeInstruction end if ;

Et *listeInstruction* $\rightarrow$ *INSTAVA listeInstruction* | *INSTAVA*

v. *Structure Itérative*

Soit $exprBool \in B_{expr}$ *alors on a la syntaxe suivante* : ,
while exprBool loop listeInstruction end loop ;

3. Structure d'un programme AVA

ENTETE OBLIGATOIRE :
- mot clé '**program**'.
- littéral de type **chaineDeCaractere** qui soit **UNIQUE** dans tout le programme.
- un marqueur de fin d'instruction **;**

```
program

        "Fact"
;
```

COMMENTAIRE FACULTATIF
- - mon Commentaire **End Of Line**
Un commentaire peut apparaître n'importe où !

```
-- calcul de Factorielle
```

UNE LISTE FACULTATIVE DE DECLARATIONS

```
int x, xbis ;
boolean  Fini ;
int  res ;
```

UNE LISTE FACULTATIVE D' INSTRUCTIONS
- Les 'espaces' et '\n' **ne sont pas significatif** dans tout le programme AVA.

```
write (%s, "entrer un entier positif \n ") ;
read x ;
xbis := x ; -- memorisation de x
res := 1 ; -- resultat
Fini :=    x = 1 ;

if   x /= 0 then
   while not Fini loop
          res :=  res * x ;
          x := x – 1 ;
          Fini :=   x = 1 ;
   end loop ;
end if ;

write ( %s , " Factorielle( ") ; write( %i, xbis) ;
write(%s , " ) = " ) ;  write(%i , res ) ;
writeln;
```