siunitx

On change les règles liés aux opérations pour intégrer dans ces règles l'associativité à gauche et la priorité des multiplication et division.

On change les règles :

$$
\begin{aligned}
< expr > \;&\rightarrow\, < entier > \\
&\rightarrow\, < caractère > \\
&\rightarrow \text{true} \\
&\rightarrow \text{false} \\
&\rightarrow \text{null} \\
&\rightarrow (< expr >) \\
&\rightarrow\, < acces > \\
&\rightarrow\, < expr > \,<operateur>\, < expr > \\
&\rightarrow \text{not} < expr > \\
&\rightarrow \text{-} < expr > \\
&\rightarrow \text{new} < ident > \\
&\rightarrow\, < ident > (< expr >^{+}_{,}) \\
&\rightarrow \text{character ' val} (< expr >) \\
\\
< operateur > \;&\rightarrow\, = \\
&\rightarrow\; /= \\
&\rightarrow\, < \\
&\rightarrow\, <= \\
&\rightarrow\, > \\
&\rightarrow\, >= \\
&\rightarrow\; + \\
&\rightarrow\; - \\
&\rightarrow\; * \\
&\rightarrow\; / \\
&\rightarrow\; rem
\end{aligned}
$$

Par :

$< expr >$ $\rightarrow < expr >$ + T
$\rightarrow < expr >$ - T
$\rightarrow$ T

T $\rightarrow$ T * F
$\rightarrow$ T / F
$\rightarrow$ F

F $\rightarrow$ P
$\rightarrow$ -P
$\rightarrow$ not P

P $\rightarrow < entier >$
$\rightarrow < caractère >$
$\rightarrow true$
$\rightarrow false$
$\rightarrow null$
$\rightarrow < acces >$
$\rightarrow new < ident >$
$\rightarrow < ident > (< expr >^{+}_{,})$
$\rightarrow character\ '\ val\ (< expr >)$
$\rightarrow < expr > < operateur > < expr >$
$\rightarrow (< expr >)$

$< operateur > \rightarrow =$
$\rightarrow\ /=$
$\rightarrow <$
$\rightarrow <=$
$\rightarrow >$
$\rightarrow >=$
$\rightarrow\ rem$

On développe la grammaire pour obtenir appliquer la dérécursivation :

$< fichier >$ $\rightarrow$ with Ada.Text_IO; use Ada.Text_IO; procedure $< ident >$ is begin $< instr >^+$ end; EOF
$\rightarrow$ with Ada.Text_IO; use Ada.Text_IO; procedure $< ident >$ is $< decl >^+$ begin $< instr >^+$ end; EOF
$\rightarrow$ with Ada.Text_IO; use Ada.Text_IO; procedure $< ident >$ is begin $< instr >^+$ end $< ident >$; EOF
$\rightarrow$ with Ada.Text_IO; use Ada.Text_IO; procedure $< ident >$ is $< decl >^+$ begin $< instr >^+$ end $< ident >$; EOF

$< decl >$ $\rightarrow$ type $< ident >$;
$\rightarrow$ type $< ident >$ is access $< ident >$;
$\rightarrow$ type $< ident >$ is record $< champs >^+$ end record;
$\rightarrow$ type $< ident >^+_,$: $< type >$;
$\rightarrow$ type $< ident >^+_,$: $< type >$ (:=$< expr >$);
$\rightarrow$ procedure $< ident >$ is begin $< instr >^+$ end;
$\rightarrow$ procedure $< ident >$ is begin $< instr >^+$ end $< ident >$;
$\rightarrow$ procedure $< ident >$ is $< decl >^+$ begin $< instr >^+$ end;
$\rightarrow$ procedure $< ident >$ is $< decl >^+$ begin $< instr >^+$ end $< ident >$;
$\rightarrow$ procedure $< ident >$ $< param >$ is begin $< instr >^+$ end;
$\rightarrow$ procedure $< ident >$ $< param >$ is begin $< instr >^+$ end $< ident >$;
$\rightarrow$ procedure $< ident >$ $< param >$ is $< decl >^+$ begin $< instr >^+$ end;
$\rightarrow$ procedure $< ident >$ $< param >$ is $< decl >^+$ begin $< instr >^+$ end $< ident >$;
$\rightarrow$ function $< ident >$ return $< type >$ is begin $< instr >^+$ end;
$\rightarrow$ function $< ident >$ return $< type >$ is begin $< instr >^+$ end $< ident >$;
$\rightarrow$ function $< ident >$ return $< type >$ is $< decl >^+$ begin $< instr >^+$ end;
$\rightarrow$ function $< ident >$ return $< type >$ is $< decl >^+$ begin $< instr >^+$ end $< ident >$;
$\rightarrow$ function $< ident >$ $< param >$ return $< type >$ is begin $< instr >^+$ end;
$\rightarrow$ function $< ident >$ $< param >$ return $< type >$ is begin $< instr >^+$ end $< ident >$;
$\rightarrow$ function $< ident >$ $< param >$ return $< type >$ is $< decl >^+$ begin $< instr >^+$ end;
$\rightarrow$ function $< ident >$ $< param >$ return $< type >$ is $< decl >^+$ begin $< instr >^+$ end $< ident >$;

$< champs >$ $\rightarrow$ $< ident >^+_,$: $< type >$;

$< type >$ $\rightarrow$ $< ident >$
$\rightarrow$ access $< ident >$

$< params >$ $\rightarrow$ ($< param >^+_;$)

$< param >$ $\rightarrow$ $< ident >^+_,$: $< type >$
$\rightarrow$ $< ident >^+_,$: $< mode >$ $< type >$

$< mode >$ $\rightarrow$ in
$\rightarrow$ in out

$< expr >$	$\rightarrow < expr >$ + T
	$\rightarrow < expr >$ - T
	$\rightarrow$ T
T	$\rightarrow$ T * F
	$\rightarrow$ T / F
	$\rightarrow$ F
F	$\rightarrow$ P
	$\rightarrow$ -P
	$\rightarrow$ not P
P	$\rightarrow < entier >$
	$\rightarrow < caractère >$
	$\rightarrow true$
	$\rightarrow false$
	$\rightarrow null$
	$\rightarrow < acces >$
	$\rightarrow new < ident >$
	$\rightarrow < ident > (< expr >^{+}_{,})$
	$\rightarrow character\,'\,val\ (< expr >)$
	$\rightarrow < expr > < operateur > < expr >$
	$\rightarrow (< expr >)$
$< operateur >$	$\rightarrow =$
	$\rightarrow\ /=$
	$\rightarrow <$
	$\rightarrow <=$
	$\rightarrow >$
	$\rightarrow >=$
	$\rightarrow\ rem$

$$
\begin{array}{ll}
<\mathit{instr}> & \rightarrow <\mathit{acces}> := <\mathit{expr}>; \\
& \rightarrow <\mathit{ident}>; \\
& \rightarrow <\mathit{ident}> (<\mathit{expr}>^{+}_{,}); \\
& \rightarrow \text{return}; \\
& \rightarrow \text{return} <\mathit{expr}>; \\
& \rightarrow \text{begin} <\mathit{instr}>^{+} \text{ end}; \\
& \rightarrow \text{if} <\mathit{expr}> \text{then} <\mathit{instr}>^{+} \text{ end if}; \\
& \rightarrow \text{if} <\mathit{expr}> \text{then} <\mathit{instr}>^{+} (\text{else} <\mathit{instr}>^{+}) \text{ end if}; \\
& \rightarrow \text{if} <\mathit{expr}> \text{then} <\mathit{instr}>^{+} <\mathit{elsif}>^{+} \text{ end if}; \\
& \rightarrow \text{if} <\mathit{expr}> \text{then} <\mathit{instr}>^{+} <\mathit{elsif}>^{+} (\text{else} <\mathit{instr}>^{+}) \text{ end if}; \\
& \rightarrow \text{for} <\mathit{ident}> \text{in} <\mathit{expr}> .. <\mathit{expr}> \text{loop} <\mathit{instr}>^{+} \text{ end loop}; \\
& \rightarrow \text{for} <\mathit{ident}> \text{in reverse} <\mathit{expr}> .. <\mathit{expr}> \text{loop} <\mathit{instr}>^{+} \text{ end loop}; \\
& \rightarrow \text{while} <\mathit{expr}> \text{loop} <\mathit{instr}>^{+} \text{ end loop};
\end{array}
$$

$$
\begin{array}{ll}
<\mathit{acces}> & \rightarrow <\mathit{ident}> \\
& \rightarrow <\mathit{expr}> . <\mathit{ident}>
\end{array}
$$

$$
\begin{array}{ll}
<\mathit{instr}>^{+} & \rightarrow <\mathit{instr}> <\mathit{instr}>^{+} \\
& \rightarrow <\mathit{instr}> \\
<\mathit{decl}>^{+} & \rightarrow <\mathit{decl}> <\mathit{decl}>^{+} \\
& \rightarrow <\mathit{decl}> \\
<\mathit{champs}>^{+} & \rightarrow <\mathit{champs}> <\mathit{champs}>^{+} \\
& \rightarrow <\mathit{champs}> \\
<\mathit{ident}>^{+}_{,} & \rightarrow <\mathit{ident}> , <\mathit{ident}>^{+}_{,} \\
& \rightarrow <\mathit{ident}>; \\
<\mathit{param}>^{+}_{;} & \rightarrow <\mathit{param}> ; <\mathit{param}>^{+}_{;} \\
& \rightarrow <\mathit{param}> \\
<\mathit{expr}>^{+}_{,} & \rightarrow <\mathit{expr}> , <\mathit{expr}>^{+}_{,} \\
& \rightarrow <\mathit{expr}> \\
<\mathit{elsif}>^{+} & \rightarrow \text{elsif} <\mathit{expr}> \text{then} <\mathit{instr}>^{+} <\mathit{elsif}>^{+} \\
& \rightarrow \text{elsif} <\mathit{expr}> \text{then} <\mathit{instr}>^{+}
\end{array}
$$

On enlève les récursivités à gauche qui posent problème. On choisit la numérotation suivante pour l'algorithme :

A_1	$< expr >$
A_2	T
A_3	F
A_4	P
A_5	$< instr >$
A_6	$< acces >$

$< expr > \quad \rightarrow T < expr >_{recur}$

$< expr >_{recur} \rightarrow +T < expr >_{recur}$
$\rightarrow -T < expr >_{recur}$
$\rightarrow \wedge$

T $\quad \rightarrow F\ T_{recur}$

$T_{recur} \quad \rightarrow *F\ T_{recur}$
$\rightarrow /F\ T_{recur}$
$\rightarrow \wedge$

F $\quad \rightarrow P$
$\rightarrow -P$
$\rightarrow not\ P$

P $\quad \rightarrow -P\ T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}$
$\rightarrow not\ P\ T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}$
$\rightarrow < entier > P_{recur}$
$\rightarrow < caractère > P_{recur}$
$\rightarrow true\ P_{recur}$
$\rightarrow false\ P_{recur}$
$\rightarrow null\ P_{recur}$
$\rightarrow < acces > P_{recur}$
$\rightarrow new < ident > P_{recur}$
$\rightarrow < ident > (< expr >^{+}_{,})\ P_{recur}$
$\rightarrow character\ '\ val\ (< expr >)P_{recur}$
$\rightarrow (< expr >)\ P_{recur}$
$P_{recur} \quad \rightarrow T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}$
$\rightarrow \wedge$

$< instr >$ $\to < acces > := < expr >;$
$\to < ident >;$
$\to < ident > (< expr >_{,}^{+});$
$\to$ return;
$\to$ return $< expr >;$
$\to$ begin $< instr >^{+}$ end;
$\to$ if $< expr >$ then $< instr >^{+}$ end if;
$\to$ if $< expr >$ then $< instr >^{+}$ (else $< instr >^{+}$) end if;
$\to$ if $< expr >$ then $< instr >^{+}$ $< elsif >^{+}$ end if;
$\to$ if $< expr >$ then $< instr >^{+}$ $< elsif >^{+}$ (else $< instr >^{+}$) end if;
$\to$ for $< ident >$ in $< expr >$ .. $< expr >$ loop $< instr >^{+}$ end loop;
$\to$ for $< ident >$ in reverse $< expr >$ .. $< expr >$ loop $< instr >^{+}$ end loop;
$\to$ while $< expr >$ loop $< instr >^{+}$ end loop;

$< acces >$ $\to -P\ T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to not\ P\ T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to < entier > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to < caractère > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to true\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to false\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to null\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to new < ident > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to < ident > (< expr >_{,}^{+})\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to character\ '\ val\ (< expr >) P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to (< expr >)\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to -P\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to not\ P\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$

$< acces >_{recur}$ $\to < acces > T_{recur} < expr >_{recur} . < ident >< acces >_{recur}$
$\to \wedge$

On injecte les règles de $< acces >$ dans les règles ci-dessous pour supprimer la règle $< acces >$ et ainsi résoudre des conflits :

$$\begin{aligned}
&\text{P} && \rightarrow < acces > P_{recur} \\
&< instr > && \rightarrow < acces > := < expr >;
\end{aligned}$$

On obtient les règles :

$$\begin{aligned}
P \quad & \rightarrow -P\ T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow -P\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow not\ P\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow not\ P\ T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow < entier > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow < caractère > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow true\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow false\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow null\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow new < ident > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow < ident > (< expr >^{+}_{,})\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow character\ '\ val\ (< expr >) P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur} \\
& \rightarrow (< expr >)\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur}\ P_{recur}
\end{aligned}$$

$$\begin{aligned}
< instr > & \rightarrow -P\ T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow -P\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow not\ P\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow not\ P\ T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr \\
& \rightarrow < entier > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow < caractère > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow true\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow false\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow null\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow new < ident > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow < ident > (< expr >^{+}_{,})\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow character\ '\ val\ (< expr >) P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >; \\
& \rightarrow (< expr >)\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;
\end{aligned}$$

On factorise la grammaire et on numérote les règles

$r_1 :< \mathit{fichier} >$ $\rightarrow$ with Ada.Text_IO; use Ada.Text_IO; procedure $< \mathit{ident} >$ is $< \mathit{fichier} >_2$

$r_2 :< \mathit{fichier} >_2$ $\rightarrow$ begin $< \mathit{instr} >^+$ end $< \mathit{fichier} >_3$
$r_3 :$ $\rightarrow$ $< \mathit{decl} >^+$ begin $< \mathit{instr} >^+$ end $< \mathit{fichier} >_3$

$r_4 :< \mathit{fichier} >_3$ $\rightarrow$; EOF
$r_5 :$ $\rightarrow$ $< \mathit{ident} >$; EOF

$r_6 :< \mathit{decl} >$ $\rightarrow$ type $< \mathit{ident} >$ $< \mathit{decl} >_{11}$
$r_7 :$ $\rightarrow$ procedure $< \mathit{ident} >$ $< \mathit{decl} >_{21}$
$r_8 :$ $\rightarrow$ function $< \mathit{ident} >$ $< \mathit{decl} >_{31}$
$r_9 :$ $\rightarrow$ $< \mathit{ident} >^+_{,}$: $< \mathit{type} >$ $< \mathit{decl} >_{12}$

$r_{10} :< \mathit{decl} >_{11}$ $\rightarrow$;
$r_{11} :$ $\rightarrow$ is $< \mathit{decl} >_{13}$

$r_{12} :< \mathit{decl} >_{12}$ $\rightarrow$;
$r_{13} :$ $\rightarrow$:=$< \mathit{expr} >$;

$r_{14} :< \mathit{decl} >_{13}$ $\rightarrow$ access $< \mathit{ident} >$;
$r_{15} :$ $\rightarrow$ record $< \mathit{champs} >^+$ end record;

$r_{16} :< \mathit{decl} >_{21}$ $\rightarrow$ is $< \mathit{decl} >_{22}$
$r_{17} :$ $\rightarrow$ $< \mathit{params} >$ is $< \mathit{decl} >_{22}$

$r_{18} :< \mathit{decl} >_{22}$ $\rightarrow$ begin $< \mathit{instr} >^+$ end $< \mathit{decl} >_{23}$
$r_{19} :$ $\rightarrow$ $< \mathit{decl} >^+$ begin $< \mathit{instr} >^+$ end $< \mathit{decl} >_{23}$

$r_{20} :< \mathit{decl} >_{23}$ $\rightarrow$;
$r_{21} :$ $\rightarrow$ $< \mathit{ident} >$;

$r_{22} :< \mathit{decl} >_{31}$ $\rightarrow$ return $< \mathit{type} >$ is $< \mathit{decl} >_{22}$
$r_{23} :$ $\rightarrow$ $< \mathit{params} >$ return $< \mathit{type} >$ is $< \mathit{decl} >_{22}$

$r_{24} :< \mathit{champs} >$ $\rightarrow$ $< \mathit{ident} >^+_{,}$: $< \mathit{type} >$;

$r_{25} :< \mathit{type} >$ $\rightarrow$ $< \mathit{ident} >$
$r_{26} :$ $\rightarrow$ access $< \mathit{ident} >$

$r_{27} :< \mathit{params} >$ $\rightarrow$ ($< \mathit{param} >^+_{;}$)

$r_{28} :< param >$ $\rightarrow < ident >^{+}_{,} \; : \; < param >_2$

$r_{29} :< param >_2$ $\rightarrow < type >$
$r_{30} :$ $\rightarrow < mode > < type >$

$r_{31} :< mode >$ $\rightarrow \text{in} < mode >_1$

$r_{32} :< mode >_1$ $\rightarrow \text{out}$
$r_{33} :$ $\rightarrow \wedge$

$r_{34} :< expr >$ $\rightarrow T < expr >_{recur}$

$r_{35} :< expr >_{recur}$ $\rightarrow +T < expr >_{recur}$
$r_{36} :$ $\rightarrow -T < expr >_{recur}$
$r_{37} :$ $\rightarrow \wedge$

$r_{38} : T$ $\rightarrow F \; T_{recur}$

$r_{39} : T_{recur}$ $\rightarrow *F \; T_{recur}$
$r_{40} :$ $\rightarrow /F \; T_{recur}$
$r_{41} :$ $\rightarrow \wedge$

$r_{42} : F$ $\rightarrow P$
$r_{43} :$ $\rightarrow -P$
$r_{44} :$ $\rightarrow not \; P$

$r_{45} : P$	$\rightarrow -P\ T_{recur} < expr >_{recur} P_{11}$
$r_{46} :$	$\rightarrow not\ P\ T_{recur} < expr >_{recur} P_{11}$
$r_{47} :$	$\rightarrow < entier > P_{recur}\ P_{12}$
$r_{48} :$	$\rightarrow < caract\grave{e}re > P_{recur}\ P_{12}$
$r_{49} :$	$\rightarrow true\ P_{recur}\ P_{12}$
$r_{50} :$	$\rightarrow false\ P_{recur}\ P_{12}$
$r_{51} :$	$\rightarrow null\ P_{recur}\ P_{12}$
$r_{52} :$	$\rightarrow new < ident > P_{recur}\ P_{12}$
$r_{53} :$	$\rightarrow < ident > (< expr >^{+}_{,})\ P_{recur}\ P_{12}$
$r_{54} :$	$\rightarrow character\ '\ val\ (< expr >) P_{recur}\ P_{12}$
$r_{55} :$	$\rightarrow (< expr >)\ P_{recur}\ P_{12}$
$r_{56} : P_{11}$	$\rightarrow < operateur >< expr > P_{recur}\ P_{12}$
$r_{57} :$	$\rightarrow .< ident >< acces >_{recur} P_{recur}$
$r_{58} : P_{12}$	$\rightarrow T_{recur} < expr >_{recur} .< ident >< acces >_{recur} P_{recur}$
$r_{59} :$	$\rightarrow \wedge$
$r_{60} : P_{recur}$	$\rightarrow T_{recur} < expr >_{recur} < operateur >< expr > P_{recur}$
$r_{61} :$	$\rightarrow \wedge$
$r_{62} : < operateur >$	$\rightarrow =$
$r_{63} :$	$\rightarrow /=$
$r_{64} :$	$\rightarrow <$
$r_{65} :$	$\rightarrow <=$
$r_{66} :$	$\rightarrow >$
$r_{67} :$	$\rightarrow >=$
$r_{68} :$	$\rightarrow \text{rem}$

$r_{69} :< instr > \quad \rightarrow -P\ T_{recur} < expr >_{recur} < instr >_6$

$r_{70} : \quad \rightarrow not\ P\ T_{recur} < expr >_{recur} < instr >_6$

$r_{71} : \quad \rightarrow < entier > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;$

$r_{72} : \quad \rightarrow < caractère > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;$

$r_{73} : \quad \rightarrow true\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;$

$r_{74} : \quad \rightarrow false\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;$

$r_{75} : \quad \rightarrow null\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;$

$r_{76} : \quad \rightarrow new < ident > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;$

$r_{77} : \quad \rightarrow character\ '\ val\ (< expr >) P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;$

$r_{78} : \quad \rightarrow (< expr >)\ P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;$

$r_{79} : \quad \rightarrow < ident > < instr >_1$

$r_{80} : \quad \rightarrow \text{return} < instr >_2$

$r_{81} : \quad \rightarrow \text{begin} < instr >^+ \text{end;}$

$r_{82} : \quad \rightarrow \text{if} < expr > \text{then} < instr >^+ < instr >_3$

$r_{83} : \quad \rightarrow \text{for} < ident > \text{in} < instr >_5$

$r_{84} : \quad \rightarrow \text{while} < expr > \text{loop} < instr >^+ \text{end loop;}$

$r_{85} : \quad \rightarrow \text{put}\ (< expr >)\ ;$

$r_{86} :< instr >_1 \quad \rightarrow ;$

$r_{87} : \quad \rightarrow (< expr >^+_,) < instr >_{11}$

$r_{88} :< instr >_{11} \rightarrow ;$

$r_{89} : \quad \rightarrow P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;);$

$r_{90} :< instr >_2 \quad \rightarrow ;$

$r_{91} : \quad \rightarrow < expr >;$

$r_{92} :< instr >_3 \quad \rightarrow \text{end if;}$

$r_{93} : \quad \rightarrow \text{else} < instr >^+ \text{end if;}$

$r_{94} : \quad \rightarrow < elsif >^+ < instr >_4$

$r_{95} :< instr >_4 \quad \rightarrow \text{end if;}$

$r_{96} : \quad \rightarrow \text{else} < instr >^+ \text{end if;}$

$r_{97} :< instr >_5 \quad \rightarrow < expr > .. < expr > \text{loop} < instr >^+ \text{end loop;}$

$r_{98} : \quad \rightarrow \text{reverse} < expr > .. < expr > \text{loop} < instr >^+ \text{end loop;}$

$r_{99} :< instr >_6 \quad \rightarrow < operateur >< expr > P_{recur}\ T_{recur} < expr >_{recur} . < ident >< acces >_{recur} := < expr >;$

$r_{100} : \quad \rightarrow . < ident >< acces >_{recur} := < expr >;$

$r_{101} :< \mathit{acces} >_{recur} \rightarrow < \mathit{acces} > T_{recur} < \mathit{expr} >_{recur} . < \mathit{ident} >< \mathit{acces} >_{recur}$
$r_{102} : \qquad\qquad \rightarrow \wedge$

$r_{103} :< \mathit{instr} >^{+} \rightarrow < \mathit{instr} > < \mathit{instr} >^{+}_{1}$

$r_{104} :< \mathit{instr} >^{+}_{1} \rightarrow < \mathit{instr} >^{+}$
$r_{105} : \qquad\qquad \rightarrow \wedge$

$r_{106} :< \mathit{decl} >^{+} \rightarrow < \mathit{decl} > < \mathit{decl} >^{+}_{1}$

$r_{107} :< \mathit{decl} >^{+}_{1} \rightarrow < \mathit{decl} >^{+}$
$r_{108} : \qquad\qquad \rightarrow \wedge$

$r_{109} :< \mathit{champs} >^{+} \rightarrow < \mathit{champs} > < \mathit{champs} >^{+}_{1}$

$r_{110} :< \mathit{champs} >^{+}_{1} \rightarrow < \mathit{champs} >^{+}$
$r_{111} : \qquad\qquad \rightarrow \wedge$

$r_{112} :< \mathit{ident} >^{+}_{,} \rightarrow < \mathit{ident} > < \mathit{ident} >^{+}_{,1}$

$r_{113} :< \mathit{ident} >^{+}_{,1} \rightarrow , < \mathit{ident} >^{+}_{,}$
$r_{114} : \qquad\qquad \rightarrow \wedge$

$r_{115} :< \mathit{param} >^{+}_{;} \rightarrow < \mathit{param} > < \mathit{param} >^{+}_{;1}$

$r_{116} :< \mathit{param} >^{+}_{;1} \rightarrow ; < \mathit{param} >^{+}_{;}$
$r_{117} : \qquad\qquad \rightarrow \wedge$

$r_{118} :< \mathit{expr} >^{+}_{,} \rightarrow < \mathit{expr} > < \mathit{expr} >^{+}_{,1}$

$r_{119} :< \mathit{expr} >^{+}_{,1} \rightarrow , < \mathit{expr} >^{+}_{,}$
$r_{120} : \qquad\qquad \rightarrow \wedge$

$r_{121} :< \mathit{elsif} >^{+} \rightarrow \text{elsif} < \mathit{expr} > \text{then} < \mathit{instr} >^{+} < \mathit{elsif} >^{+}_{1}$

$r_{122} :< \mathit{elsif} >^{+}_{1} \rightarrow < \mathit{elsif} >^{+}$
$r_{123} : \qquad\qquad \rightarrow \wedge$

$$P_{\wedge}(G) = \{< mode >_1, < expr >_{recur}, < acces >_{recur}, < instr >_1^+, < decl >_1^+, < champs >_1^+, < ident >_{,1}^+, < param >_{;1}^+, < expr >_{,1}^+, < elsif >_1^+\}$$

Non terminal gauche	**Règle**	**Symbole Directeur**
$< fichier >$	r_1	with
$< fichier >_2$	r_2	begin
$< fichier >_2$	r_3	type , procedure , function, $< ident >$
$< fichier >_3$	r_4	;
$< fichier >_3$	r_5	$< ident >$
$< decl >$	r_6	type
$< decl >$	r_7	procedure
$< decl >$	r_8	function
$< decl >$	r_9	$< ident >$
$< decl >_{11}$	r_{10}	;
$< decl >_{11}$	r_{11}	is
$< decl >_{12}$	r_{12}	;
$< decl >_{12}$	r_{13}	:=
$< decl >_{13}$	r_{14}	access
$< decl >_{13}$	r_{15}	record
$< decl >_{21}$	r_{16}	is
$< decl >_{21}$	r_{17}	(
$< decl >_{22}$	r_{18}	begin
$< decl >_{22}$	r_{19}	type , procedure , function, $< ident >$
$< decl >_{23}$	r_{20}	;
$< decl >_{23}$	r_{21}	$< ident >$
$< decl >_{31}$	r_{22}	return
$< decl >_{31}$	r_{23}	(
$< champs >$	r_{24}	$< ident >$
$< type >$	r_{25}	$< ident >$
$< type >$	r_{26}	access
$< params >$	r_{27}	(
$< param >$	r_{28}	$< ident >$
$< param >_2$	r_{29}	$< ident >$, access
$< param >_2$	r_{30}	in
$< mode >$	r_{31}	in
$< mode >_1$	r_{32}	out
$< mode >_1$	r_{33}	$< ident >$, access
$< expr >$	r_{34}	$-, not, < entier >, < caractère >, < ident >, true, false, null, new, character, ($
$< expr >_{recur}$	r_{35}	$+$
$< expr >_{recur}$	r_{36}	$-$
$< expr >_{recur}$	r_{37}	$; , , ,), *, /, +, -, .., loop, then, =, / =, <, <=, >, >=, rem$
T	r_{38}	$-, not, < entier >, < caractère >, < ident >, true, false, null, new, character, ($
T_{recur}	r_{39}	$/$
T_{recur}	r_{40}	$*$
T_{recur}	r_{41}	$; , , ,), *, /, +, -, .., loop, then, =, / =, <, <=, >, >=, rem$
F	r_{42}	$-, not, < entier >, < caractère >, < ident >, true, false, null, new, character, ($
F	r_{43}	$-$
F	r_{44}	not
P	r_{45}	$-$
P	r_{46}	not
P	r_{47}	$< entier >$
P	r_{48}	$< caractère >$
P	r_{49}	$true$

P	r_{50}	$false$
P	r_{51}	$null$
P	r_{52}	new
P	r_{53}	$< ident >$
P	r_{54}	$character$
P	r_{55}	$($
P_{11}	r_{56}	$=, / =, <, <=, >, >=, rem$
P_{11}	r_{57}	$.$
P_{12}	r_{58}	$*, /, +, -, .$
P_{12}	r_{59}	$*, /, +, -, =, / =, <, <=, >, >=, rem, .$
P_{recur}	r_{60}	$*, /, +, -, =, / =, <, <=, >, >=, rem$
P_{recur}	r_{61}	$*, /, +, -, =, / =, <, <=, >, >=, rem, .$
$< operateur >$	r_{62}	$=$
$< operateur >$	r_{63}	$/ =$
$< operateur >$	r_{64}	$<$
$< operateur >$	r_{65}	$<=$
$< operateur >$	r_{66}	$>$
$< operateur >$	r_{67}	$>=$
$< operateur >$	r_{68}	rem
$< instr >$	r_{69}	$-$
$< instr >$	r_{70}	not
$< instr >$	r_{71}	$< entier >$
$< instr >$	r_{72}	$<$ *caractère*
$< instr >$	r_{73}	$true$
$< instr >$	r_{74}	$false$
$< instr >$	r_{75}	$null$
$< instr >$	r_{76}	new
$< instr >$	r_{77}	$character$
$< instr >$	r_{78}	$($
$< instr >$	r_{79}	$< ident >$
$< instr >$	r_{80}	$return$
$< instr >$	r_{81}	$begin$
$< instr >$	r_{82}	if
$< instr >$	r_{83}	for
$< instr >$	r_{84}	$while$
$< instr >$	r_{85}	put
$< instr >_1$	r_{86}	$;$
$< instr >_1$	r_{87}	$($
$< instr >_{11}$	r_{88}	$;$
$< instr >_{11}$	r_{89}	$=, / =, <, <=, >, >=, rem, ., *, /, +, -$
$< instr >_2$	r_{90}	$;$
$< instr >_2$	r_{91}	$< entier >$, < *caractère* >, true , false , null , (, $< ident >$, not, -, new, character
$< instr >_3$	r_{92}	end
$< instr >_3$	r_{93}	else
$< instr >_3$	r_{94}	elsif
$< instr >_4$	r_{95}	end
$< instr >_4$	r_{96}	else
$< instr >_5$	r_{97}	$-, not, < entier >,$ < *caractère* >, $< ident >, true, false, null, new, character, ($
$< instr >_5$	r_{98}	$reverse$
$< instr >_6$	r_{99}	$=, / =, <, <=, >, >=, rem$
$< instr >_6$	r_{100}	$.$
$< acces >_{recur}$	r_{101}	$.$
$< acces >_{recur}$	r_{102}	=, /=, <, <=, >, >=, +, -, *, /, rem, :

$< instr >^+$	r_{103}	$< ident >$, $< entier >$, $< caractère >$, true, false, null, (, not, -, new, character, return, begin, if, for, while
$< instr >_1^+$	r_{104}	$< ident >$, $< entier >$, $< caractère >$, true, false, null, (, not, -, new, character, return, begin, if, for, while
$< instr >_1^+$	r_{105}	end, (,)
$< decl >^+$	r_{106}	type, procedure, function
$< decl >_1^+$	r_{107}	type, procedure, function
$< decl >_1^+$	r_{108}	begin
$< champs >^+$	r_{109}	$< ident >$
$< champs >_1^+$	r_{110}	$< ident >$
$< champs >_1^+$	r_{111}	end
$< ident >_,^+$	r_{112}	$< ident >$
$< ident >_{,1}^+$	r_{113}	,
$< ident >_{,1}^+$	r_{114}	:
$< param >_;^+$	r_{115}	$< ident >$
$< param >_{;1}^+$	r_{116}	;
$< param >_{;1}^+$	r_{117}	)
$< expr >_,^+$	r_{118}	$< ident >$, $< entier >$, $< caractère >$, true, false, null, (, not, -, new, character
$< expr >_{,1}^+$	r_{119}	,
$< expr >_{,1}^+$	r_{120}	)
$< elsif >^+$	r_{121}	(
$< elsif >_1^+$	r_{122}	(
$< elsif >_1^+$	r_{123}	end, (