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 :

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

$$
\begin{aligned}
< operateur > & \rightarrow = \\
& \rightarrow \ /= \\
& \rightarrow < \\
& \rightarrow <= \\
& \rightarrow > \\
& \rightarrow >= \\
& \rightarrow \ + \\
& \rightarrow \ - \\
& \rightarrow \ * \\
& \rightarrow \ / \\
& \rightarrow \ rem
\end{aligned}
$$

Par :

$$
\begin{array}{ll}
<expr> & \rightarrow T+<expr> \\
 & \rightarrow T-<expr> \\
 & \rightarrow \text{T} \\
 & \\
\text{T} & \rightarrow \text{I * T} \\
 & \rightarrow \text{I / T} \\
 & \rightarrow \text{I} \\
 & \\
\text{I} & \rightarrow \text{F = I} \\
 & \rightarrow \text{F /= I} \\
 & \rightarrow F < I \\
 & \rightarrow F <= I \\
 & \rightarrow F > I \\
 & \rightarrow F >= I \\
 & \rightarrow \text{F rem I} \\
 & \rightarrow \text{I} \\
 & \\
\text{F} & \rightarrow \text{P} \\
 & \rightarrow \text{-P} \\
 & \rightarrow \text{not P} \\
 & \\
\text{P} & \rightarrow <entier> \\
 & \rightarrow <caractere> \\
 & \rightarrow true \\
 & \rightarrow false \\
 & \rightarrow null \\
 & \rightarrow <acces> \\
 & \rightarrow new <ident> \\
 & \rightarrow <ident> (<expr>^{+}_{,}) \\
 & \rightarrow character\,'\,val\,(<expr>) \\
 & \rightarrow (<expr>)
\end{array}
$$

En factorisant, on obtient.

$$
\begin{array}{ll}
< expr > & \rightarrow T < expr >_1 \\
\\
< expr >_1 & \rightarrow + < expr > \\
 & \rightarrow - < expr > \\
 & \rightarrow \wedge \\
\\
\mathrm{T} & \rightarrow \mathrm{I}\ T_1 \\
\\
T_1 & \rightarrow \mathrm{*\ T} \\
 & \rightarrow \mathrm{/\ T} \\
 & \rightarrow \wedge \\
\\
\mathrm{I} & \rightarrow \mathrm{F}\ I_1 \\
\\
I_1 & \rightarrow \mathrm{=\ I} \\
 & \rightarrow \mathrm{/=\ I} \\
 & \rightarrow < I \\
 & \rightarrow <= I \\
 & \rightarrow > I \\
 & \rightarrow >= I \\
 & \rightarrow \mathrm{rem\ I} \\
 & \rightarrow \wedge \\
\\
\mathrm{F} & \rightarrow \mathrm{P} \\
 & \rightarrow \mathrm{-P} \\
 & \rightarrow \mathrm{not\ \ P} \\
\\
\mathrm{P} & \rightarrow < entier > \\
 & \rightarrow < caractere > \\
 & \rightarrow true \\
 & \rightarrow false \\
 & \rightarrow null \\
 & \rightarrow < acces > \\
 & \rightarrow new < ident > \\
 & \rightarrow < ident > (< expr >^{+}_{,}) \\
 & \rightarrow character\ '\ val\ (< expr >) \\
 & \rightarrow (< expr >)
\end{array}
$$

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

$$\begin{array}{ll} \text{P} & \rightarrow < acces > \\ < instr > & \rightarrow < acces > := < expr >; \end{array}$$

On obtient les règles :

$$\begin{array}{ll} < instr > & \rightarrow < ident > := < expr >; \\ & \rightarrow < expr > . < ident > := < expr >; \\ & \\ P & \rightarrow < ident > \\ & \rightarrow < expr > . < ident > \end{array}$$

La règle $P \rightarrow \langle expr \rangle$. $\langle ident \rangle$ offre la possibilité de finir une expression par . $\langle ident \rangle$. Or, une expression se termine toujours par l'une des parties droites des règles $P \rightarrow$ et ces parties droites terminent toujours une expression.

Donc, on peut remplacer :
$P \rightarrow \langle expr \rangle$. $\langle ident \rangle$

Par :
$P \rightarrow P$. $\langle ident \rangle$

On développe la grammaire pour ensuite 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

$< expr > \quad \rightarrow T < expr >_1$

$< expr >_1 \rightarrow + < expr >$
$\rightarrow - < expr >$
$\rightarrow \wedge$

T $\rightarrow$ I T_1

T_1 $\rightarrow$ * T
$\rightarrow$ / T
$\rightarrow \wedge$

I $\rightarrow$ F I_1

I_1 $\rightarrow$ = I
$\rightarrow$ /= I
$\rightarrow < I$
$\rightarrow <= I$
$\rightarrow > I$
$\rightarrow >= I$
$\rightarrow$ rem I
$\rightarrow \wedge$

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

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

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

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

On n'a plus de récursivité à droite.

On factorise la grammaire et on numérote les règles.

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

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

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

$r_4 :< fichier >_3 \rightarrow$; EOF

$r_5 : \rightarrow < ident >$; EOF

$r_6 :< decl > \rightarrow$ type $< ident > < decl >_{11}$

$r_7 : \rightarrow$ procedure $< ident > < decl >_{21}$

$r_8 : \rightarrow$ function $< ident > < decl >_{31}$

$r_9 : \rightarrow < ident >^{+}_{,} \,:\, < type > < decl >_{12}$

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

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

$r_{12} :< decl >_{12} \rightarrow$;

$r_{13} : \rightarrow := < expr >$;

$r_{14} :< decl >_{13} \rightarrow$ access $< ident >$;

$r_{15} : \rightarrow$ record $< champs >^{+}$ end record;

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

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

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

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

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

$r_{21} : \rightarrow < ident >$;

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

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

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

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

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

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

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

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

$r_{30} : \ \rightarrow < \mathit{mode} > < \mathit{type} >$

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

$r_{32} : < \mathit{mode} >_1 \ \rightarrow \mathrm{out}$

$r_{33} : \ \rightarrow \wedge$

$r_{34} : < \mathit{expr} > \ \rightarrow T < \mathit{expr} >_1$

$r_{35} : < \mathit{expr} >_1 \ \rightarrow + < \mathit{expr} >$

$r_{36} : \ \rightarrow - < \mathit{expr} >$

$r_{37} : \ \rightarrow \wedge$

$r_{38} : T \ \rightarrow I \ T_1$

$r_{39} : T_1 \ \rightarrow *T$

$r_{40} : \ \rightarrow /T$

$r_{41} : \ \rightarrow \wedge$

$r_{42} : I \ \rightarrow F \ I_1$

$r_{43} : I_1 \ \rightarrow = T$

$r_{44} : \ \rightarrow = / \ T$

$r_{45} : \ \rightarrow < T$

$r_{46} : \ \rightarrow <= T$

$r_{47} : \ \rightarrow > T$

$r_{48} : \ \rightarrow >= T$

$r_{49} : \ \rightarrow remT$

$r_{50} : \ \rightarrow \wedge$

$r_{51} : F \ \rightarrow P$

$r_{52} : \ \rightarrow -P$

$r_{53} : \ \rightarrow \mathit{not} \ P$

$r_{54} : P \quad \rightarrow < entier > P_{recur}$

$r_{55} : \quad \rightarrow < caractere > P_{recur}$

$r_{56} : \quad \rightarrow true\ P_{recur}$

$r_{57} : \quad \rightarrow false\ P_{recur}$

$r_{58} : \quad \rightarrow null\ P_{recur}$

$r_{59} : \quad \rightarrow \text{new} < ident > P_{recur}$

$r_{60} : \quad \rightarrow \text{character ' val } (< expr)\ P_{recur}$

$r_{61} : \quad \rightarrow (< expr >)\ P_{recur}$

$r_{62} : \quad \rightarrow < ident > P_1$

$r_{63} : P_{recur} \quad \rightarrow .\ < ident > P_{recur}$

$r_{64} : \quad \rightarrow \wedge$

$r_{65} : P_1 \quad \rightarrow P_{recur}$

$r_{66} : \quad \rightarrow (< expr >^{+}_{,})\ P_{recur}$

$r_{67} :< instr > \quad \rightarrow < ident > < instr >_1$

$r_{68} : \quad \rightarrow < expr > .\ < ident > := < expr >;$

$r_{69} : \quad \rightarrow \text{return} < instr >_2$

$r_{70} : \quad \rightarrow \text{begin} < instr >^{+} \text{end;}$

$r_{71} : \quad \rightarrow \text{if} < expr > \text{then} < instr >^{+} < instr >_3$

$r_{72} : \quad \rightarrow \text{for} < ident > \text{in} < instr >_4$

$r_{73} : \quad \rightarrow \text{while} < expr > \text{loop} < instr >^{+} \text{end loop;}$

$r_{74} :< instr >_1 \rightarrow := < expr >;$

$r_{75} : \quad \rightarrow ;$

$r_{76} : \quad \rightarrow (< expr >^{+}_{,});$

$r_{77} :< instr >_2 \rightarrow < expr >;$

$r_{78} : \quad \rightarrow ;$

$r_{79} :< instr >_3 \rightarrow \text{end if;}$

$r_{80} : \quad \rightarrow \text{else} < instr >^{+} \text{end if;}$

$r_{81} : \quad \rightarrow < elsif >^{+} < instr >_3$

$r_{82} :< instr >_4 \rightarrow < expr > ..\ < expr > \text{loop} < instr >^{+} \text{end loop;}$

$r_{83} : \quad \rightarrow \text{reverse} < expr > ..\ < expr > \text{loop} < instr >^{+} \text{end loop;}$

$r_{84} :< instr >^{+} \quad \rightarrow < instr > < instr >^{+}_{1}$

$r_{85} :< instr >^{+}_{1} \quad \rightarrow < instr >^{+}$
$r_{86} : \quad \rightarrow \wedge$

$r_{87} :< decl >^{+} \quad \rightarrow < decl > < decl >^{+}_{1}$

$r_{88} :< decl >^{+}_{1} \quad \rightarrow < decl >^{+}$
$r_{89} : \quad \rightarrow \wedge$

$r_{90} :< champs >^{+} \rightarrow < champs > < champs >^{+}_{1}$

$r_{91} :< champs >^{+}_{1} \rightarrow < champs >^{+}$
$r_{92} : \quad \rightarrow \wedge$

$r_{93} :< ident >^{+}_{,} \quad \rightarrow < ident > < ident >^{+}_{,1}$

$r_{94} :< ident >^{+}_{,1} \quad \rightarrow , < ident >^{+}_{,}$
$r_{95} : \quad \rightarrow \wedge$

$r_{96} :< param >^{+}_{;} \quad \rightarrow < param > < param >^{+}_{;1}$

$r_{97} :< param >^{+}_{;1} \quad \rightarrow ; < param >^{+}_{;}$
$r_{98} : \quad \rightarrow \wedge$

$r_{99} :< expr >^{+}_{,} \quad \rightarrow < expr > < expr >^{+}_{,1}$

$r_{100} :< expr >^{+}_{,1} \quad \rightarrow , < expr >^{+}_{,}$
$r_{101} : \quad \rightarrow \wedge$

$r_{102} :< elsif >^{+} \quad \rightarrow \text{elsif} < expr > \text{then} < instr >^{+} < elsif >^{+}_{1}$

$r_{103} :< elsif >^{+}_{1} \quad \rightarrow < elsif >^{+}$
$r_{104} : \quad \rightarrow \wedge$

On injecte les règles de $< expr >$ dans la règle :

$$< instr > \rightarrow < expr > \ . < ident > := < expr >;$$

On obtient :

$$< instr > \rightarrow T < expr >_1 \ . < ident > := < expr >;$$

On injecte les règles de T dans la règle :

$$< instr > \rightarrow T < expr >_1 \ . < ident > := < expr >;$$

On obtient :

$$< instr > \rightarrow I\ T_1 < expr >_1 \ . < ident > := < expr >;$$

On injecte les règles de I dans la règle :

$$< instr > \rightarrow I\ T_1 < expr >_1 \ . < ident > := < expr >;$$

On obtient :

$$< instr > \rightarrow FI_1\ T_1 < expr >_1 \ . < ident > := < expr >;$$

On injecte les règles de F dans la règle :

$$< instr > \rightarrow FI_1\ T_1 < expr >_1 \ . < ident > := < expr >;$$

On obtient :

$$\begin{aligned} < instr > &\rightarrow P\ I_1\ T_1 < expr >_1 \ . < ident > := < expr >; \\ &\rightarrow -P\ I_1\ T_1 < expr >_1 \ . < ident > := < expr >; \\ &\rightarrow not\ P\ I_1\ T_1 < expr >_1 \ . < ident > := < expr >; \end{aligned}$$

On injecte les règles de P dans la règle :

$$< instr > \rightarrow P\ I_1\ T_1 < expr >_1 \ . < ident > := < expr >;$$

On obtient :

$$\begin{aligned}
< instr > &\rightarrow < entier > P_{recur}\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow < caractere > P_{recur}\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow true\ P_{recur}\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow false\ P_{recur}\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow null\ P_{recur}\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow new < ident > P_{recur}\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow character'val(< expr >)P_{recur}\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow (< expr >)P_{recur}\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow < ident > P_1\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;
\end{aligned}$$

On injecte les règles de P_1 dans la règle :

$$< instr > \rightarrow < ident > P_1\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;$$

On obtient :

$$\begin{aligned}
< instr > &\rightarrow < ident > . < ident >\ \ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow < ident > (< expr >_{,}^{+})\ P_{recur}\ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;\\
&\rightarrow < ident >\ \ I_1\ T_1 < expr >_1 .\ < ident > := < expr >;
\end{aligned}$$

On injecte les règles de $< instr >_1$ dans la règle :

$$< instr > \rightarrow < ident > < instr >_1$$

On obtient :

$$\begin{aligned}
< instr > &\rightarrow < ident > := < expr > ;\\
&\rightarrow < ident > ;\\
&\rightarrow < ident > (< expr >_{,}^{+});
\end{aligned}$$

On a au final :

$$\begin{aligned}
< instr > &\to < ident > := < expr > ; \\
&\to < ident > ; \\
&\to < ident > (< expr >_{,}^{+}); \\
&\to < ident > . < ident > \; I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to < ident > (< expr >_{,}^{+}) \; P_{recur} \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to < ident > \; I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to -P \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to not \; P \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to < entier > P_{recur} \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to < caractere > P_{recur} \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to true \; P_{recur} \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to false \; P_{recur} \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to null \; P_{recur} \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to new < ident > P_{recur} \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to character'val(< expr >)P_{recur} \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to (< expr >)P_{recur} \, I_1 \, T_1 < expr >_1 \, . < ident > := < expr >; \\
&\to \text{return} < instr >_2 \\
&\to \text{begin} < instr >^{+} \text{end;} \\
&\to \text{if} < expr > \text{then} < instr >^{+} < instr >_3 \\
&\to \text{for} < ident > \text{in} < instr >_4 \\
&\to \text{while} < expr > \text{loop} < instr >^{+} \text{end loop;}
\end{aligned}$$

En factorisant de nouveau, on a :

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

$r_2 :< fichier >_2 \rightarrow$ begin $< instr >^+$ end $< fichier >_3$

$r_3 : \rightarrow < decl >^+$ begin $< instr >^+$ end $< fichier >_3$

$r_4 :< fichier >_3 \rightarrow$; EOF

$r_5 : \rightarrow < ident >$; EOF

$r_6 :< decl > \rightarrow$ type $< ident > < decl >_{11}$

$r_7 : \rightarrow$ procedure $< ident > < decl >_{21}$

$r_8 : \rightarrow$ function $< ident > < decl >_{31}$

$r_9 : \rightarrow < ident >^+_, \,:\, < type > < decl >_{12}$

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

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

$r_{12} :< decl >_{12} \rightarrow$;

$r_{13} : \rightarrow :=< expr >;$

$r_{14} :< decl >_{13} \rightarrow$ access $< ident >$;

$r_{15} : \rightarrow$ record $< champs >^+$ end record;

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

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

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

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

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

$r_{21} : \rightarrow < ident >$;

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

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

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

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

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

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

$$\begin{array}{ll}
r_{28} :< param > & \to < ident >_{,}^{+} \, : \, < param >_2 \\
\\
r_{29} :< param >_2 & \to < type > \\
r_{30} : & \to < mode > < type > \\
\\
r_{31} :< mode > & \to \text{in} < mode >_1 \\
\\
r_{32} :< mode >_1 & \to \text{out} \\
r_{33} : & \to \wedge \\
\\
r_{34} :< expr > & \to T < expr >_1 \\
\\
r_{35} :< expr >_1 & \to + < expr > \\
r_{36} : & \to - < expr > \\
r_{37} : & \to \wedge \\
\\
r_{38} : T & \to I \; T_1 \\
\\
r_{39} : T_1 & \to *T \\
r_{40} : & \to /T \\
r_{41} : & \to \wedge \\
\\
r_{42} : I & \to F \; I_1 \\
\\
r_{43} : I_1 & \to = T \\
r_{44} : & \to = / \; T \\
r_{45} : & \to < T \\
r_{46} : & \to <= T \\
r_{47} : & \to > T \\
r_{48} : & \to >= T \\
r_{49} : & \to remT \\
r_{50} : & \to \wedge \\
\\
r_{51} : F & \to P \\
r_{52} : & \to -P \\
r_{53} : & \to not \; P
\end{array}$$

$r_{54} : P \quad \rightarrow < entier > P_{recur}$

$r_{55} : \quad \rightarrow < caractere > P_{recur}$

$r_{56} : \quad \rightarrow true\ P_{recur}$

$r_{57} : \quad \rightarrow false\ P_{recur}$

$r_{58} : \quad \rightarrow null\ P_{recur}$

$r_{59} : \quad \rightarrow \text{new} < ident > P_{recur}$

$r_{60} : \quad \rightarrow \text{character ' val} (< expr)\ P_{recur}$

$r_{61} : \quad \rightarrow (< expr >)\ P_{recur}$

$r_{62} : \quad \rightarrow < ident > P_1$

$r_{63} : P_{recur} \quad \rightarrow .\ < ident > P_{recur}$

$r_{64} : \quad \rightarrow \wedge$

$r_{65} : P_1 \quad \rightarrow P_{recur}$

$r_{66} : \quad \rightarrow (< expr >^{+}_{,})\ P_{recur}$

$r_{67} :< instr > \quad \rightarrow < ident > < instr >_1$

$r_{68} : \quad \rightarrow -P\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{69} : \quad \rightarrow not\ P\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{70} : \quad \rightarrow < entier > P_{recur}\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{71} : \quad \rightarrow < caractere > P_{recur}\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{72} : \quad \rightarrow true\ P_{recur}\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{73} : \quad \rightarrow false\ P_{recur}\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{74} : \quad \rightarrow null\ P_{recur}\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{75} : \quad \rightarrow new < ident > P_{recur}\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{76} : \quad \rightarrow character'val(< expr >)P_{recur}\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{77} : \quad \rightarrow (< expr >)P_{recur}\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr >;$

$r_{78} : \quad \rightarrow \text{return} < instr >_2$

$r_{79} : \quad \rightarrow \text{begin} < instr >^{+} \text{end;}$

$r_{80} : \quad \rightarrow \text{if} < expr > \text{then} < instr >^{+} < instr >_3$

$r_{81} : \quad \rightarrow \text{for} < ident > \text{in} < instr >_4$

$r_{82} : \quad \rightarrow \text{while} < expr > \text{loop} < instr >^{+} \text{end loop;}$

$r_{83} :< instr >_1 \quad \rightarrow := < expr >;$

$r_{84} : \quad \rightarrow ;$

$r_{85} : \quad \rightarrow (< expr >^{+}_{,}) < instr >_{11}$

$r_{86} : \quad \rightarrow .\ < ident > I_1\ T_1 < expr >_1\ .\ < ident > := < expr > ;$

$r_{87} : \quad \rightarrow I_1\ T_1 < expr >_1\ .\ < ident > := < expr > ;$

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

$r_{89} : \quad \rightarrow P_{recur}\ I_1\ T_1 < expr >_1\ .\ < ident > := < expr > ;$

$r_{90} :< \mathit{instr} >_2$	$\rightarrow < \mathit{expr} >;$
$r_{91} :$	$\rightarrow ;$
$r_{92} :< \mathit{instr} >_3$	$\rightarrow$ end if;
$r_{93} :$	$\rightarrow$ else $< \mathit{instr} >^+$ end if;
$r_{94} :$	$\rightarrow < \mathit{elsif} >^+ < \mathit{instr} >_3$
$r_{95} :< \mathit{instr} >_4$	$\rightarrow < \mathit{expr} > .. < \mathit{expr} >$ loop $< \mathit{instr} >^+$ end loop;
$r_{96} :$	$\rightarrow$ reverse $< \mathit{expr} > .. < \mathit{expr} >$ loop $< \mathit{instr} >^+$ end loop;
$r_{97} :< \mathit{instr} >^+$	$\rightarrow < \mathit{instr} > < \mathit{instr} >^+_1$
$r_{98} :< \mathit{instr} >^+_1$	$\rightarrow < \mathit{instr} >^+$
$r_{99} :$	$\rightarrow \wedge$
$r_{100} :< \mathit{decl} >^+$	$\rightarrow < \mathit{decl} > < \mathit{decl} >^+_1$
$r_{101} :< \mathit{decl} >^+_1$	$\rightarrow < \mathit{decl} >^+$
$r_{102} :$	$\rightarrow \wedge$
$r_{103} :< \mathit{champs} >^+$	$\rightarrow < \mathit{champs} > < \mathit{champs} >^+_1$
$r_{104} :< \mathit{champs} >^+_1$	$\rightarrow < \mathit{champs} >^+$
$r_{105} :$	$\rightarrow \wedge$
$r_{106} :< \mathit{ident} >^+_,$	$\rightarrow < \mathit{ident} > < \mathit{ident} >^+_{,1}$
$r_{107} :< \mathit{ident} >^+_{,1}$	$\rightarrow , < \mathit{ident} >^+_,$
$r_{108} :$	$\rightarrow \wedge$
$r_{109} :< \mathit{param} >^+_;$	$\rightarrow < \mathit{param} > < \mathit{param} >^+_{;1}$
$r_{110} :< \mathit{param} >^+_{;1}$	$\rightarrow ; < \mathit{param} >^+_;$
$r_{111} :$	$\rightarrow \wedge$

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

$r_{113} :< expr >^{+}_{,1} \rightarrow , < expr >^{+}_{,}$

$r_{114} : \qquad\qquad \rightarrow \wedge$

$r_{115} :< elsif >^{+} \rightarrow \text{elsif} < expr > \text{then} < instr >^{+} < elsif >^{+}_{1}$

$r_{116} :< elsif >^{+}_{1} \rightarrow < elsif >^{+}$

$r_{117} : \qquad\qquad \rightarrow \wedge$

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

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}	$-, not, < entier >, < caractère >, < ident >, true, false, null, new, character, ($
$< expr >_1$	r_{35}	$+$
$< expr >_1$	r_{36}	$-$
$< expr >_1$	r_{37}	$; ,), ., , , then, loop, ..$
T	r_{38}	$-, not, < entier >, < caractère >, < ident >, true, false, null, new, character, ($
T_1	r_{39}	$/$
T_1	r_{40}	$*$
T_1	r_{41}	$; ,), ., , , then, loop, .., +, -$
I	r_{42}	$-, not, < entier >, < caractère >, < ident >, true, false, null, new, character, ($
I_1	r_{43}	$=$
I_1	r_{44}	$= /$
I_1	r_{45}	$<$
I_1	r_{46}	$<=$
I_1	r_{47}	$>$
I_1	r_{48}	$>=$
I_1	r_{49}	rem

I_1	r_{50}	$;,),.,,,then,loop,..,+,-,*,/$
F	r_{51}	$< entier >,< caractère >,< ident >,true,false,null,new,character,($
F	r_{52}	$-$
F	r_{53}	not
P	r_{54}	$< entier >$
P	r_{55}	$< caractere >$
P	r_{56}	$true$
P	r_{57}	$false$
P	r_{58}	$null$
P	r_{59}	new
P	r_{60}	$character$
P	r_{61}	$($
P	r_{62}	$< ident >$
P_{recur}	r_{63}	.
P_{recur}	r_{64}	$=,/=,<,<=,>,>=,rem,*,/,+,-,;,),$. $,,,then,loop,..,$
P_1	r_{65}	$=,/=,<,<=,>,>=,rem,*,/,+,-,;,),.,,,then,loop,..,$
P_1	r_{66}	$($
$< instr >$	r_{67}	$< ident >$
$< instr >$	r_{68}	$-$
$< instr >$	r_{69}	not
$< instr >$	r_{70}	$< entier >$
$< instr >$	r_{71}	$< caractere >$
$< instr >$	r_{72}	$true$
$< instr >$	r_{73}	$false$
$< instr >$	r_{74}	$null$
$< instr >$	r_{75}	new
$< instr >$	r_{76}	$character$
$< instr >$	r_{77}	$($
$< instr >$	r_{78}	$return$
$< instr >$	r_{79}	$begin$
$< instr >$	r_{80}	if
$< instr >$	r_{81}	for
$< instr >$	r_{82}	$while$
$< instr >_1$	r_{83}	:=
$< instr >_1$	r_{84}	;
$< instr >_1$	r_{85}	(
$< instr >_1$	r_{86}	.
$< instr >_1$	r_{87}	., +, -, *, /, =, =/, <, <=, >, >=, rem
$< instr >_{11}$	r_{88}	;
$< instr >_{11}$	r_{89}	., +, -, *, /, =, =/, <, <=, >, >=, rem
$< instr >_2$	r_{90}	$< entier >$
$< instr >_2$	r_{91}	;
$< instr >_3$	r_{92}	end
$< instr >_3$	r_{93}	$else$
$< instr >_3$	r_{94}	elsif
$< instr >_4$	r_{95}	$< entier >,< caractere >,true,false,null,new,character,(,< ident >$
$< instr >_4$	r_{96}	reverse
$< instr >^+$	r_{97}	$< ident >$, $< entier >$, $< caractère >$, true, false, null, (, not, -, new, character, return, begin, if, for, while
$< instr >_1^+$	r_{98}	$< ident >$, $< entier >$, $< caractère >$, true, false, null, (, new, character, return, begin, if, for, while
$< instr >_1^+$	r_{99}	end, (,)
$< decl >^+$	r_{100}	type, procedure, function
$< decl >_1^+$	r_{101}	type, procedure, function

$< decl >_1^+$	r_{102}	begin
$< champs >^+$	r_{103}	$< ident >$
$< champs >_1^+$	r_{104}	$< ident >$
$< champs >_1^+$	r_{105}	end
$< ident >_,^+$	r_{106}	$< ident >$
$< ident >_{,1}^+$	r_{107}	,
$< ident >_{,1}^+$	r_{108}	:
$< param >_;^+$	r_{109}	$< ident >$
$< param >_{;1}^+$	r_{110}	;
$< param >_{;1}^+$	r_{111}	)
$< expr >_,^+$	r_{112}	$< ident >$, $< entier >$, $< caractère >$, true, false, null, (, not, -, new, character
$< expr >_{,1}^+$	r_{113}	,
$< expr >_{,1}^+$	r_{114}	)
$< elsif >^+$	r_{115}	elsif
$< elsif >_1^+$	r_{116}	elsif
$< elsif >_1^+$	r_{117}	end, else

Si problème, on infecte.