PCL GRAMMAIRE

<fichier> = with Ada.Text_I0; use Ada.Text_I0;
procedure **<ident>** is **<decl>***
begin **<instr>*** end **<ident>**? ; EOF

<decl> = type **<ident>** ;
| type **<ident>** is access **<ident>**;
| type **<ident>** is record **<champs>**+ end record ;
| **<ident>**+, : **<type>** (:= **<expr>**)?
| procedure **<ident>** **<params>**? is **<decl>***
begin **<instr>**+ end **<ident>**?
| function **<ident>** **<params>**? return **<type>** is **<decl>***
begin **<instr>**+ end **<ident>**?

<champs> = **<ident>**+, : **<type>**;

<type> = **<ident>**
| access **<ident>**

<params> = (**<param>**+;)

<param> = **<ident>**+, : **<mode>**? **<type>**

<mode> = in
| in out

<expr> = **<entier>** I **<caractère>** I true | false | null
| (**<expr>**)
| **<accès>**
| **<expr>** **<opérateur>** **<expr>**
| **<expr>**
| not **<expr>**
| new **<ident>**
| **<ident>** (**<expr>**+,)
| character' val (**<expr>**)

<intr> = **<accès>** := **<expr>**;
| **<ident>**
| **<ident>** (**<expr>**+,);
| return **<expr>**?;
| begin **<instr>**+ end

| if **<expr>** then **<instr>**+ (elsif **<expr>** then **<instr>**+)*
(else **<instr>** +)? end if;
| for **<ident>** in reverse? **<expr>** .. **<expr>**
loop **<instr>**+ end loop ;
| while **<expr>** loop **<instr>**+ end loop ;

<opérateur> ::= = | /= | *< | *<= | >** | >**=
| + | - | * | / | rem
| and | and then | or | or else

<accès> ::= **<ident>** | **<expr>** . **<ident>**

<ident> ::= **<alpha>** (**<alpha>** | **<chiffre>** | _)*

<alpha> ::= [a-z|A_Z]

<entier> ::= **<chiffre>**+

<chiffre> ::= [0-9]

<caractère> ::= '**<ASCII>**'

<ASCII> ::= SPC | ! | " | # | $ | % | & | ' | (|) | * | + | , | - | . | / | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | [| \ |] | ^ | _ | ` | a | b | c | d | e| f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | { | | | } | ~ |

	Premiers	**Suivants**
Fichiers	with	$
Decl	type, a-z, A-Z, procedure, function	type, a-z, A-Z, procedure, function
Champs	a-z, A-Z	end, a-z, A-Z
Type	a-z, A-Z , access	(, is, ;,
Params	(	
Param	a-z, A-Z	
Mode	in	
Expr	a-z, A-Z , 0-9, ', true, false, null, (, not, new, character'	
Instr	a-z, A-Z, 0-9, ', true, false, null, (, not, new, character' return, begin, if, for, while	
Opérateur	=, /=, <, <=, >, >=, +, -, *, , rem, and, and then, or, or else	
Accès	a-z, A-Z , 0-9, ', true, false, null, (, not, new, character'	

	Premiers	Suivants
Ident	a-z, A-Z	(, is, ;,
Alpha	a-z, A-Z	
Entier	0-9	
Chiffre	0-9	
Caractère	'	
ASCII	ASCII	