⟨*fichier*⟩ ::= `with Ada.Text_IO; use Ada.Text_IO;`
`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*⟩ | ⟨*caractère*⟩ | `true` | `false` | `null`
| `(` ⟨*expr*⟩ `)`
| ⟨*accès*⟩
| ⟨*expr*⟩ ⟨*opérateur*⟩ ⟨*expr*⟩
| `not` ⟨*expr*⟩ | `-` ⟨*expr*⟩
| `new` ⟨*ident*⟩
| ⟨*ident*⟩ `(` ⟨*expr*⟩+, `)`
| `character'val (` ⟨*expr*⟩ `)`

⟨*instr*⟩ ::= ⟨*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*⟩