F -> "with Ada.Text_IO; use Ada.Text_IO;" procedure <ident> is D* "begin" I+ "end" <ident>? ";"

D -> "type" <ident> ";"
| "type" <ident> "is" "access" <ident>
| "type" <ident> "is" "record" C+ "end" "record;"
| <ident>+ ":" T (":=" E?);"
| "procedure" <ident> P? "is" D* "begin" I+ "end" <ident>? ";"
| “function” <ident> P ? “return” T “is” D* “begin” I+ “end” <ident>? ”;”

C -> (<ident>’,’) + ":" T ";"

T -> <ident>
| "access" <ident>

P -> <ident>+: M? T

M -> "in"
| "in out"

E -> <entier>
| <caractère>
| "true"
| "false"
| "null"
| "(" E ")"
| A
| E O E
| "not" E
| "-" E
| "new" <ident>
| "<ident>" "(" (E,)+ ")"
| "character ' val (" E ")"

I -> A ":=" E
| <ident> ";"
| <ident> "(" (E,)+ ") ;"
| "return" E? ";"
| "begin" I+ "end" ";"
|”if “ <ident> then I+ (elsif E then I+)* (else I+)? End if;
|”for” <ident> in reverse? E “..” E loop I+ end loop;
|”while” E loop I+ end loop;

O -> "="
| "/="
| "<"
| "<="
| ">"
| ">="
| "+"
| "-"
| "*"
| "/"
| "rem"
| "and"
| "and then"
| "or"
| "or else"

A -> <ident>
| E . <ident>

ChatGPT :

Program -> "with Ada.Text_IO; use Ada.Text_IO;" ProcedureDeclaration

ProcedureDeclaration -> "procedure" Identifier "is" Declarations "begin" StatementSequence "end" Identifier ";"
| "function" Identifier Parameters "return" Type "is" Declarations "begin" StatementSequence "end" Identifier ";"

Declarations -> Declarations Declaration | ε

Declaration -> "type" Identifier ";"
| "type" Identifier "is" "access" Identifier
| "type" Identifier "is" "record" Components "end" "record;"
| Identifier ":" Type (":=" Expression?) ";"
| "procedure" Identifier ProcedureParameters "is" Declarations "begin" StatementSequence "end" Identifier ";"

Parameters -> "(" ParameterList ")" | ε

ParameterList -> Parameter ParameterTail

ParameterTail -> "," Parameter ParameterTail | ε

Parameter -> Identifier ParameterMode? Type

ParameterMode -> "in" | "in out"

Components -> Component ";" Components | ε

Component -> Identifier+ ":" Type ";"

Type -> Identifier | "access" Identifier

StatementSequence -> Statement StatementSequence | ε

Statement -> AssignmentStatement
| ProcedureCall
| ReturnStatement
| IfStatement
| ForLoop
| WhileLoop

AssignmentStatement -> Identifier ":=" Expression ";"

ProcedureCall -> Identifier "(" ArgumentList ")" ";"

ArgumentList -> Argument ArgumentTail

ArgumentTail -> "," Argument ArgumentTail | ε

Argument -> Expression

ReturnStatement -> "return" Expression? ";"

IfStatement -> "if" Expression "then" StatementSequence ElseIfPart? ElsePart? "end" "if" ";"

ElseIfPart -> "elsif" Expression "then" StatementSequence ElseIfPart | ε

ElsePart -> "else" StatementSequence | ε

ForLoop -> "for" Identifier "in" Reverse? Expression ".." Expression "loop" StatementSequence "end" "loop" ";"

WhileLoop -> "while" Expression "loop" StatementSequence "end" "loop" ";"

Expression -> SimpleExpression (RelationalOperator SimpleExpression)?

RelationalOperator -> "=" | "/=" | "<" | "<=" | ">" | ">="

SimpleExpression -> Term (AdditionOperator Term)*

AdditionOperator -> "+" | "-"

Term -> Factor (MultiplicationOperator Factor)*

MultiplicationOperator -> "*" | "/" | "rem"

Factor -> Primary (AndOperator Primary)*

AndOperator -> "and" | "and then"

Primary -> Identifier | Literal | "(" Expression ")"

Identifier -> <ident>

Literal -> IntegerLiteral | CharacterLiteral | BooleanLiteral | "null"

IntegerLiteral -> <entier>

CharacterLiteral -> <caractère>

BooleanLiteral -> "true" | "false"

Reverse -> "reverse"