TOKENS:

- `COMNT`: `//` Comentário de linha (estilo moderno/Delphi)
- `COMNTSTD`: `{ ... }` Comentário padrão do Pascal
- `COMTSTDS`: `(* ... *)` Comentário padrão alternativo
- `PROGRAM`: `program` Anuncio que o programa vai começar
- `BEGIN`: `begin` Início da parte que executa (pode-se comparar à main do `C`)
- `FUNCTION`: `function` Início da declaração de uma função
- `ENDPROGRAM`: `end.` Fim do programa
- `END`: `end` Fim de qualquer outro bloco
- `VAR`: `var` Declaração de uma variável
- `AND`: `and` AND lógico
- `ARRAY`: `array` Declaração de um array
- `DOTDOT`: `..` Separador de intervalos (ex: `1..10`)
- `CASE`: `case` Início de um bloco switch case
- `CONST`: `const` Declaração de uma constante
- `DIV`: `div` Divisão inteira
- `GE`: `>=` Simbolo maior ou igual
- `LE`: `<=` Simbolo menor ou igual
- `NE`: `<>` Simbolo de diferente
- `ATR`: `:=` Simbolo de atribuição
- `DO`: `do` Instrução para executar um bloco
- `ELSE`: `else` Instrução de caso contrário
- `FOR`: `for` Intrução para o início de um ciclo for
- `GOTO`: `goto` Instrução para ir para um label
- `IF`: `if` Instrução para testar uma condição
- `IN`: `in` Verifica se um elemento está num conjunto
- `LABEL`: `label` Declaração de rótulos para goto
- `MOD`: `mod` Resto da divisão inteira
- `NIL`: `nil` Ponteiro nulo
- `NOT`: `not` Negação lógica
- `OF`: `of` Preposição usada em arrays, cases, sets...
- `OR`: `or` OR lógico
- `PACKED`: `packed` Otimização de memória para arrays/records
- `PROCEDURE`: `procedure` Rotina sem retorno de valor
- `RECORD`: `record` Estrutura de dados heterogénea (struct)
- `SET`: `set` Declaração de um conjunto matemático
- `THEN`: `then` Então (usado com if)
- `TO`: `to` Para (usado em loops for crescentes)
- `DOWNTO`: `downto` Para (usado em loops for decrescentes)
- `TYPE`: `type` Definição de novos tipos de dados
- `REPEAT`: `repeat` Início de um ciclo repeat-until
- `UNTIL`: `until` Condição de paragem do ciclo repeat
- `WHILE`: `while` Ciclo enquanto a condição for verdadeira
- `WITH`: `with` Atalho para aceder a campos de um Record
- `ABS`: `abs` Palavra reservada para função valor absoluto

- `SQR`: `sqr` Palavra reservada para função quadrado
- `SQRT`: `sqrt` Palavra reservada para função raiz quadrada
- `ROUND`: `round` Palavra reservada para arredondamento
- `ODD`: `odd` Palavra reservada para verificar imparidade
- `INTEGER`: `integer` Palavra reservada para o tipo Inteiro
- `SMALLINT`: `smallint` Palavra reservada para Inteiro curto
- `LONGINT`: `longint` Palavra reservada para Inteiro longo
- `REAL`: `real` Palavra reservada para o tipo Real
- `BOOL`: `boolean` Palavra reservada para o tipo Boleano
- `STRING`: `string` Palavra reservada para o tipo String
- `CHAR`: `char` Palavra reservada para o tipo Caracter
- `BYTE`: `byte` Palavra reservada para o tipo Byte
- `FILE`: `file` Palavra reservada para tipo ficheiro
- `READ`: `read` Palavra reservada para leitura
- `READL`: `readln` Palavra reservada para leitura com quebra de linha
- `WRITE`: `write` Palavra reservada para escrita
- `WRITEL`: `writeln` Palavra reservada para escrita com quebra de linha
- `VARNAME`: Identificador (nome de variável, função, tipo, etc.)
- `SEMI`: `;` Ponto e vírgula

VALORES:

- `INTVALUE`: Um número inteiro literal (ex: `123`)
- `REALVALUE`: Um número real literal (ex: `12.34`)
- `BOOLVALUE`: Um valor boleano literal (`true` ou `false`)
- `STRINGVALUE`: Uma string literal (ex: `'texto'`)
- `CHARVALUE`: Um caracter literal (ex: `'c'`)

LITERALS (SÍMBOLOS):

- `=` Igualdade
- `(` Parêntesis esquerdo
- `)` Parêntesis direito
- `,` Vírgula (separador de listas)
- `:` Dois pontos (definição de tipo ou label)
- `+` Adição
- `-` Subtração
- `*` Multiplicação
- `/` Divisão real
- `<` Menor que
- `>` Maior que
- `[` Colchete esquerdo (arrays)
- `]` Colchete direito (arrays)
- `{` Chaveta esquerda (comentários)
- `}` Chaveta direita