GRAMÁTICA PARA O COMPILADOR miniC

Abaixo está a definição do **miniC** em EBNF, onde os símbolos terminais que são compostos por caracteres especiais estão entres apóstrofos e as palavras reservadas estão em letras maiúsculas. `IDENTIFIER` e `CONSTANT_INT` são símbolos terminais adicionais (são tokens e não eles mesmo) que correspondem aos identificadores e constantes inteiras.

A linguagem possui: o tipo de dado inteiro; funções com um identificador, resultado inteiro e número de parâmetros arbitrários; variáveis globais, e locais dentro de funções; as estruturas de controle `if-else` e `while`; além de um grande número de operadores.

Considere que as palavras reservadas são todas minúsculas.

Implemente a gramática abaixo em ANTLR 4. Faça os ajustes que você julgar necessário. Inclua as definições das palavras chaves, ou, inclua as strings diretamente na gramática.

O que deve ser entregue:

- O arquivo com a gramática;
- O programa Python que instancia os analisadores léxicos e sintáticos, gera e imprime a árvore de análise sintática (*parse tree*). A entrada do código deve ser via arquivo.

A gramática em EBNF está abaixo.

```
program
      : definition { definition }

definition
      : data_definition
      | function_definition

data_definition
      : INT declarator { ',' declarator } ';'

declarator
      : Identifier

function_definition
      : [ INT ] function_header function_body

function_header
      : declarator parameter_list

parameter_list
      : '(' [ parameter_declaration ] ')'

parameter_declaration
      : INT declarator { ',' declarator }

function_body
```

```
      : ‘{‘ { data_definition } { statement } ‘}‘

statement
      : [ expression ] ‘;‘
      | IF ‘(‘ expression ‘)‘ statement [ ELSE statement ]
      | WHILE ‘(‘ expression ‘)‘ statement
      | BREAK ‘;‘
      | CONTINUE ‘;‘
      | RETURN [ expression ] ‘;‘


expression
      : binary { ‘,‘ binary }

binary
      : Identifier ‘=‘ binary
      | Identifier ‘+=‘ binary
      | Identifier ‘-=‘ binary
      | Identifier ‘*=‘ binary
      | Identifier ‘/=‘ binary
      | Identifier ‘%=‘ binary
      | binary ‘==‘ binary
      | binary ‘!=‘ binary
      | binary ‘<‘ binary
      | binary ‘<=‘ binary
      | binary ‘>=‘ binary
      | binary ‘>=‘ binary
      | binary ‘+‘ binary
      | binary ‘-‘ binary
      | binary ‘*‘ binary
      | binary ‘/‘ binary
      | binary ‘%‘ binary
      | unary

unary
      : ‘++‘ Identifier
      | ‘--‘ Identifier
      | primary

primary
      : IDENTIFIER
      | CONSTANT_INT
      | ‘(‘ expression ‘)‘
      | Identifier ‘(‘ [ argument_list ] ‘)‘

argument_list
      : binary { ‘,‘ binary }
```