Compiladores

Alunos:

Maria Heloísa de Paula Souza (20203599)

Vitor Soares Moreira (202003622)

Gramática

- A gramática é de autoria própria.

A tabela a seguir com as regras de derivação foram geradas por IA:

Regra de Gramática	**Derivações**	**Explicação das Derivações**
PROGRAMA	-> LISTA_FUNCOES	Define que o programa consiste em uma lista de funções.
LISTA_FUNCOES	-> FUNCAO LISTA_FUNCOES	Define que uma lista de funções pode ser composta por uma função seguida de outras funções.
	-> FUNCAO	Alternativamente, uma lista de funções pode consistir apenas de uma única função.
FUNCAO	-> TOKEN_INICIO_FUNCAO FUNCAO_DECLARACAO TOKEN_FIM_FUNCAO	Define a estrutura de uma função com início, declaração, e fim.
FUNCAO_DECLARACAO	-> FUNCAO_NOME FUNCAO_RETORNO FUNCAO_ARGS FUNCAO_CORPO	Especifica os componentes de uma função, incluindo nome, tipo de retorno, argumentos e corpo.
FUNCAO_NOME	-> TOKEN_IDENTIFICADOR	Define que o nome da função é representado por um identificador único.
FUNCAO_RETORNO	-> TOKEN_TIPO FUNCAO_RETORNO_TIPO	Define o tipo de retorno da função.
	-> ε	Alternativamente, a função pode não ter tipo de retorno (ε).
FUNCAO_RETORNO_TIPO	-> TOKEN_INTEIRO	Define que o retorno pode ser do tipo inteiro.
	-> TOKEN_CARACTERE	Alternativamente, o retorno pode ser do tipo caractere.
	-> TOKEN_REAL	Alternativamente, o retorno pode ser do tipo real.
	-> TOKEN_LITERAL	Alternativamente, o retorno pode ser do tipo literal.
FUNCAO_ARGS	-> TOKEN_INICIO_ARGS FUNCAO_ARGS_CORPO TOKEN_FIM_ARGS	Define a lista de argumentos de uma função.
	-> ε	Alternativamente, a função pode não ter argumentos (ε).
FUNCAO_ARGS_CORPO	-> LISTA_ARGS	Define o corpo dos argumentos da função, composto por uma lista de declarações de argumentos.
LISTA_ARGS	-> ARG_DECLARACAO LISTA_ARGS	Define a lista de argumentos com uma declaração seguida de outras declarações.
	-> ARG_DECLARACAO	Alternativamente, a lista pode ter uma única declaração de argumento.
ARG_DECLARACAO	-> TIPO_ARG LISTA_IDENTIFICADORES TOKEN_PONTO_E_VIRG	Especifica a estrutura de um argumento, incluindo o tipo, identificadores e o ponto-e-vírgula de término.
TIPO_ARG	-> VARIAVEL_TIPO	Define o tipo de um argumento como um tipo de variável.
LISTA_IDENTIFICADORES	-> TOKEN_IDENTIFICADOR TOKEN_VIRGULA LISTA_IDENTIFICADORES	Especifica uma lista de identificadores separados por vírgula.

Regra de Gramática	Derivações	Explicação das Derivações
	-> TOKEN_IDENTIFICADOR	Alternativamente, a lista pode ter apenas um identificador.
FUNCAO_CORPO	-> LISTA_DECLARACOES	Define o corpo de uma função como uma lista de declarações.
LISTA_DECLARACOES	-> DECLARACAO LISTA_DECLARACOES	Define uma lista de declarações composta por uma declaração seguida de outras declarações.
	-> DECLARACAO	Alternativamente, a lista pode conter apenas uma única declaração.
DECLARACAO	-> ESCREVA_DECLARACAO	Define a declaração como uma escrita.
	-> CHAMA_DECLARACAO	Alternativamente, a declaração pode ser uma chamada de função.
	-> VARIAVEL_DECLARACAO	Alternativamente, a declaração pode ser uma declaração de variável.
	-> SE_DECLARACAO	Alternativamente, a declaração pode ser uma declaração condicional.
	-> ENQUANTO_DECLARACAO	Alternativamente, a declaração pode ser um loop "enquanto".
	-> ATRIBUICAO_DECLARACAO	Alternativamente, a declaração pode ser uma atribuição.
	-> RETORNA_DECLARACAO	Alternativamente, a declaração pode ser uma instrução de retorno.
ESCREVA_DECLARACAO	-> TOKEN_ESCREVA ESCREVA_CORPO TOKEN_PONTO_E_VIRG	Define a declaração de escrita com o token de escrita, corpo, e ponto-e-vírgula.
ESCREVA_CORPO	-> TOKEN_STRING_LITERAL	Define o corpo de escrita como uma string literal.
	-> TOKEN_IDENTIFICADOR	Alternativamente, o corpo de escrita pode ser um identificador.
CHAMA_DECLARACAO	-> CHAMADA_FUNCAO TOKEN_PONTO_E_VIRG	Define uma chamada de função seguida por ponto-e-vírgula.
CHAMADA_FUNCAO	-> TOKEN_CHAMA TOKEN_CHAMADA LISTA_CHAMA_IDENTIFICADORES	Define a estrutura de uma chamada de função.
LISTA_CHAMA_IDENTIFICADORES	-> TOKEN_IDENTIFICADOR LISTA_CHAMA_IDENTIFICADORES	Define uma lista de identificadores para a chamada da função.
	-> TOKEN_IDENTIFICADOR	Alternativamente, a lista pode conter apenas um identificador.
VARIAVEL_DECLARACAO	-> VARIAVEL_TIPO VARIAVEL_DECLARACAO_CORPO	Declara uma variável com tipo e corpo.
VARIAVEL_DECLARACAO_CORPO	-> LISTA_IDENTIFICADORES TOKEN_PONTO_E_VIRG	Define o corpo de uma declaração de variável como uma lista de identificadores.
	-> ATRIBUICAO_DECLARACAO	Alternativamente, o corpo pode ser uma atribuição de valor.
VARIAVEL_TIPO	-> TOKEN_INTEIRO	Define o tipo da variável como inteiro.
	-> TOKEN_REAL	Alternativamente, o tipo da variável pode ser real.
	-> TOKEN_CARACTERE	Alternativamente, o tipo da variável pode ser caractere.
	-> TOKEN_LITERAL	Alternativamente, o tipo da variável pode ser literal.
RETORNA_DECLARACAO	-> TOKEN_RETORNA RETORNA_CORPO TOKEN_PONTO_E_VIRG	Define uma declaração de retorno com o token de retorno, corpo e ponto-e-vírgula.
RETORNA_CORPO	-> TOKEN_IDENTIFICADOR	Define o corpo do retorno como um identificador.
	-> VALOR	Alternativamente, o corpo do retorno pode ser um valor específico.

Regra de Gramática	Derivações	Explicação das Derivações
VALOR	-> TOKEN_NUMERO	Define o valor como um número.
	-> TOKEN_STRING_LITERAL	Alternativamente, o valor pode ser uma string literal.
SE_DECLARACAO	-> TOKEN_SE SE_CABECALHO SE_CORPO SENAO_SE_BLOCO	Define a estrutura de uma declaração condicional com possibilidade de bloco "senão".
SE_CABECALHO	-> TOKEN_ABRE_PAR EXPRESSAO_BOOLEANA TOKEN_FECHA_PAR TOKEN_ENTAO	Define o cabeçalho da condição com parênteses e expressão booleana.
SENAO_SE_BLOCO	-> TOKEN_SENAO_SE SE_CABECALHO SE_CORPO SENAO_SE_BLOCO	Define o bloco "senão-se" como uma extensão da estrutura condicional.
	-> TOKEN_SENAO SE_CORPO TOKEN_FIM_SE	Alternativamente, o bloco pode ter apenas uma declaração “senão”.
	-> TOKEN_FIM_SE	Alternativamente, o bloco não ser utilizado.
SE_CORPO	-> LISTA_DECLARACOES	Define o corpo da condição com uma lista de declarações.
ENQUANTO_DECLARACAO	-> TOKEN_ENQUANTO TOKEN_ABRE_PAR EXPRESSAO_BOOLEANA TOKEN_FECHA_PAR TOKEN_FACA ENQUANTO_CORPO TOKEN_FIM_ENQUANTO	Define a estrutura de uma declaração de loop "enquanto", com condição e corpo.
ENQUANTO_CORPO	-> FUNCAO_CORPO	Define o corpo do loop “enquanto”.
ATRIBUICAO_DECLARACAO	-> TOKEN_IDENTIFICADOR TOKEN_ATRIBUICAO EXPRESSAO TOKEN_PONTO_E_VIRG	Define uma atribuição entre um identificador e uma expressão.
EXPRESSAO	-> CHAMADA_FUNCAO	Define uma expressão como uma chamada de função.
	-> TOKEN_IDENTIFICADOR	Alternativamente, a expressão pode ser um identificador.
	-> EXPRESSAO_MATEMATICA	Alternativamente, a expressão pode ser uma expressão matemática.
	-> VALOR	Alternativamente, a expressão pode ser um valor específico.
EXPRESSAO_MATEMATICA	-> OPERACAO_BINARIA	Define uma expressão matemática como uma operação binária.
	-> OPERACAO_UNARIA	Alternativamente, a expressão matemática pode ser uma operação unária.
OPERACAO_BINARIA	-> FATOR_MATEMATICO OPERACAO_MATEMATICA FATOR_MATEMATICO	Define uma operação binária entre dois fatores.
OPERACAO_MATEMATICA	-> TOKEN_SOMA	Define os operadores matemáticos, como a soma.
	-> TOKEN_SUB	Alternativamente, o operador pode ser a subtração.
	-> TOKEN_MULT	Alternativamente, o operador pode ser a multiplicação.
	-> TOKEN_DIVISAO	Alternativamente, o operador pode ser a divisão.
OPERACAO_UNARIA	-> SINAL_MATEMATICO FATOR_MATEMATICO	Define uma operação unária com sinal e fator matemático.
SINAL_MATEMATICO	-> TOKEN_SOMA	Define os sinais unários, como o positivo.
	-> TOKEN_SUB	Alternativamente, o sinal unário pode ser o negativo.
FATOR_MATEMATICO	-> TOKEN_IDENTIFICADOR	Define o fator matemático, podendo ser um identificador.
	-> TOKEN_NUMERO	Alternativamente, o fator pode ser um número.
EXPRESSAO_BOOLEANA	-> TERMO_BOOLEANO TOKEN_OP_RELACIONAL TERMO_BOOLEANO	Define uma expressão booleana com termos e operadores relacionais.

Regra de Gramática	Derivações	Explicação das Derivações
TERMO_BOOLEANO	-> TOKEN_IDENTIFICADOR	Define um termo booleano como um identificador.
	-> VALOR	Alternativamente, o termo booleano pode ser um valor literal.