

COMPLEJIDAD COMPUTACIONAL

PRÁCTICA 3 Maximiliano Onofre Martínez 422054438

EJERCICIO 1

```
G = \{N, \Sigma, P, S\}
P = {
 programa \rightarrow declaraciones sentencias
 declaraciones \rightarrow declaraciones declaracion | declaracion
 declaracion → tipo lista_var;
 	ext{tipo} 
ightarrow int | float
 	ext{lista\_var} 
ightarrow 	ext{lista\_var}, identificador | identificador
 sentencias → sentencias sentencia | sentencia
 sentencia \rightarrow identificador = expresion;
 if (expresion) sentencias else sentencias | while (expresion) sentencias
 expression \rightarrow expression + expression | expression - expression |
 expresion * expresion | expresion | identificador | numero
 expresion \rightarrow (expresion)
N = \{programa, declaraciones, declaracion, tipo, lista\_var, sentencias, \}
 sentencia, expresion}
\Sigma = \{int, float, identificador, numero, if, else, while, +, -, *, /, =, \}
 ; , , , (, )}
S = program
```

EJERCICIO 2: ELIMINACIÓN DE LA AMBIGÜEDAD

```
sentencia_completa 	o if (expresion) sentencia_completa else sentencia_completa | while (expresion) sentencias | identificador = expresion; sentencia_incompleta 	o if (expresion) sentencia_completa | while (expresion) sentencias;
```

EJERCICIO 3: ELIMINACIÓN DE LA RECURSIVIDAD IZQUIERDA

```
\label{eq:declaraciones} \begin{split} & \operatorname{declaraciones} \to \operatorname{declaracion declaraciones}' \\ & \operatorname{declaraciones}' \to \operatorname{declaracion declaraciones}' | \, \epsilon \\ & \operatorname{sentencias} \to \operatorname{sentencia sentencias}' \\ & \operatorname{sentencias}' \to \operatorname{sentencia sentencias}' | \, \epsilon \end{split}
```

EJERCICIO 4: FACTORIZACIÓN IZQUIERDA

```
expresion → expresion_suma
expresion_suma → expresion_suma + expresion_producto |
expresion_suma − expresion_producto | expresion_producto
expresion_producto → expresion_producto * factor |
expresion_producto/factor | factor
factor → identificador | numero | (expresion)
```

EJERCICIO 5

```
P = {
 programa \rightarrow declaraciones sentencias
 declaraciones \rightarrow declaracion declaraciones'
 declaraciones' \rightarrow declaracion declaraciones' | \epsilon
 declaracion → tipo lista_var;
 \mathsf{tipo} 	o int \mid float
 	ext{lista\_var} 
ightarrow 	ext{lista\_var}, identificador | identificador
 sentencias → sentencia sentencias'
 sentencias' \rightarrow sentencia sentencias' \mid \epsilon
 sentencia → sentencia_completa | sentencia_incompleta
 sentencia_completa 
ightarrow if (expresion) sentencia_completa else
 sentencia_completa | while (expresion) sentencias |
 identificador = expresion;
 sentencia_incompleta 	o if (expresion) sentencia_completa |
 while (expresion) sentencias;
 expression \rightarrow expression + expression | expression - expression |
 expresion * expresion | expresion | identificador | numero
 expresion \rightarrow (expresion)
N = \{programa, declaraciones, declaraciones', declaracion, tipo, \}
 lista_var, sentencias, sentencias', sentencia, sentencia_completa,
 sentencia_incompleta, expresion}
```