Programa 2: Análisis sintáctico

Código fuente: *matrícula*_c2.c / *matrícula*_c2.cpp

El programa consiste en la implementación de la etapa del análisis sintáctico de un compilador para el lenguaje de programación visto en clase. La siguiente gramática del lenguaje está en la notación Backus-Naur extendida:

```
PROGRAMA = INCLUSIONES FUNCIONES FIN_ARCHIVO

INCLUSIONES = ε
INCLUSIONES = “include” LITERAL_CADENA “;” INCLUSIONES

FUNCIONES = ε
FUNCIONES = FUNCION FUNCIONES

FUNCION = “function” IDENTIFICADOR “(“ PARAMETROS “)” “{“ INSTRUCCIONES “}”

PARAMETROS = ε
PARAMETROS = IDENTIFICADOR
PARAMETROS = IDENTIFICADOR “,” PARAMETROS

INSTRUCCIONES = ε
INSTRUCCIONES = INSTRUCCIÓN INSTRUCCIONES

INSTRUCCIÓN = SENTENCIA_DECLARACION
INSTRUCCIÓN = SENTENCIA_IF
INSTRUCCIÓN = SENTENCIA_PRINT
INSTRUCCIÓN = SENTENCIA_RETURN

SENTENCIA_DECLARACION = “var” IDENTIFICADOR “=” EXPRESIÓN “;”

SENTENCIA_IF = “if” EXPRESIÓN “{“ INSTRUCCIONES “}” SENTENCIA_IF_CONT
SENTENCIA_IF_CONT = ε
SENTENCIA_IF_CONT = ELSE SENTENCIA_IF
SENTENCIA_IF_CONT = ELSE “{“ INSTRUCCIONES “}”

SENTENCIA_PRINT = “print” EXPRESIÓN “;”
SENTENCIA_RETURN = “return” EXPRESIÓN “;”

EXPRESIÓN = EXPRESIÓN_UNARIA EXPRESIÓN_CONT
EXPRESIÓN_CONT = ε
EXPRESIÓN_CONT = OPERADOR_BINARIO EXPRESIÓN_UNARIA EXPRESIÓN_CONT

OPERADOR_BINARIO = MAS | MENOS | POR | ENTRE | MENOR_QUE | MENOR_IGUAL_QUE |
                   MAYOR_QUE | MAYOR_IGUAL_QUE | IGUAL_QUE | DIFERENTE_QUE

EXPRESIÓN_UNARIA = OPERADORES_PREFIJOS EXPRESIÓN_PRIMARIA OPERADORES_POSFIJOS

OPERADORES_PREFIJOS = ε
OPERADORES_PREFIJOS = OPERADOR_PREFIJO OPERADORES_PREFIJOS
OPERADOR_PREFIJO = MAS | MENOS

OPERADORES_POSFIJOS = ε
OPERADORES_POSFIJOS = “(“ EXPRESIONES “)”

EXPRESIONES = ε
```

```
EXPRESIONES = EXPRESIÓN
EXPRESIONES = EXPRESIÓN “,” EXPRESIONES

EXPRESIÓN_PRIMARIA = IDENTIFICADOR
EXPRESIÓN_PRIMARIA = LITERAL_ENTERA
EXPRESIÓN_PRIMARIA = “(“ EXPRESIÓN “)”
```

Donde el símbolo no terminal inicial es `PROGRAMA` y los no terminales no definidos anteriormente coinciden con lo descrito en la especificación léxica del lenguaje.

Los operadores tienen la misma asociatividad y precedencia relativa que los operadores correspondientes del lenguaje C excepto que los operadores `==` y `!=` tienen la misma precedencia que el resto de los operadores relacionales.

Entrada: Código fuente que deberá ser leído desde un archivo llamado `codigo.txt`. Puede suponer que el archivo existe y que el análisis léxico no generará errores.

Salida: Código fuente que genere exactamente el mismo árbol sintáctico que el código de entrada o el mensaje `ERROR` en caso de un error sintáctico.

Ejemplo de entrada	Ejemplo de salida

`include "archivo.txt";`
 	
`function main( )`
`{`
`  if a == b {`
`  if b == c {`
`  var a = f( );`
`  }`
`  }`
`  else {`
`  var c = h( ) + h( );`
`  }`
`}`
 	
`function f(a, c)`
`{`
`  var a = (g + 2)( );`
`  var b = c + (5 + 1 * 3) ==`
`  +(k * k) + e;`
`}`
	`include "archivo.txt";`
	
	`function main()`
	`{`
	`  if (a)==(b) {`
	`  if (b)==(c) {`
	`  var a = (f)();`
	`  }`
	`  else {`
	`  }`
	`  }`
	`  else {`
	`  var c = ((h)())+((h)());`
	`  }`
	`}`
	
	`function f(a, c, )`
	`{`
	`  var a = ((g)+(2))();`
	`  var b = ((c)+((5)+((1)*(3))))==`
	`  ((+((k)*(k)))+(e));`
	`}`

Su programa sólo debe imprimir lo solicitado. El código fuente deberá ser enviado como archivo adjunto al correo al203305906@alumnos.azc.uam.mx. No se recibirán ejecutables y de ninguna otra forma.