**ANÁLISIS SINTÁCTICO**

El análisis sintáctico es la segunda fase de un compilador y se encarga de generar estructuras de código, estas estructuras de código son la concatenación de componentes léxicos, por ende necesitamos los componentes léxicos generados, las estructuras que posiblemente se generen y con ello identificar y generar los posibles errores que pueden generar dichas estructuras, por ende los elementos anteriormente mencionados se definen para este compilador de la siguiente manera.

**COMPONENTES LÉXICOS DEL COMPILADOR**

Los componentes léxicos son aquellas cadenas de caracteres que concuerdan con un patrón, dicho patrón describe la forma en cómo se forma dicho componente léxico, por lo tanto los componentes léxicos de este compilador estarán formados por el lenguaje del mismo o otros componentes léxicos, y los componentes léxicos se muestran en la siguiente tabla:

|  |  |
| --- | --- |
| **COMPONENTE LÉXICO** | **PATRÓN** |
| **NUMERO\_ENTERO** | ({Digito})({Digito})\* |
| **NUMERO\_REAL** | ({Digito})+.({Digito})({Digito})\* |
| **CONDICIONAL** | (true | false) |
| **IDENTIFICADOR\_CADENA** | ({Comilla})({Letra} | {Dígito})({Letra} | {Dígito} | {Símbolo})\*({Comilla}) |
| **IDENTIFICADOR** | ({Letra})({Letra} | {Digito})\* |
| **VALORES** | (NUMERO\_ENTERO | NUMERO\_REAL | IDENTIFICADOR\_CADENA | CONDICIONAL) |
| **PUERTO** | (port\_A0 | port\_A1 | port\_A2 | port\_A3 | port\_A4 | port\_A5 | port\_B0 | port\_B1 | port\_B2 | port\_B3 | port\_B4 | port\_B5 | port\_B6 | port\_B7 | port\_C2 | port\_C3 | port\_C4 | port\_C5 | port\_C6 | port\_C7 | port\_D0 | port\_D1 | port\_D2 | port\_D3 | port\_D4 | port\_D5 | port\_D6 | port\_D7) |
| **TIPO\_PUERTO** | (proximity | temperature | LED | LCD | button | motor) |
| **TIPO\_DATO** | (int | string | decimal | boolean) |
| **MOTOR** | (move | restart | start) |
| **METODO\_SENSOR** | (distance | state | time | degree) |
| **ESTRUCTURA\_DE\_CONTROL** | (function | begin | loop | if | else) |
| **PALABRA\_RESERVADA** | (ligther | delay | operation | call | return | print | console | turn\_off) |
| **DECLARACION** | (var | const) |
| **OPERADOR\_LOGICO** | (and | or | not) |
| **OPERADOR\_RELACIONAL** | (> | < | >= | <= | ==) |
| **OPERADOR\_ARITMETICO** | (+ | - | \* | /) |
| **OPERADOR\_ASIGNACION** | (=) |
| **SIGNO\_AGRUPACION** | ( { | } | ( | )) |
| **SIGNO\_PUNTUACION** | (, | .) |
| **FIN\_DE\_LINEA** | (!) |

**ESTRUCTURAS DE CÓDIGO DEL COMPILADOR**

Las estructuras de código son generadas a través de los componentes léxicos, dónde son concatenaciones de los componentes léxicos que siguen un patrón dicha estructura, por ende las estructuras de este compilador también son producciones y las que se pueden generar son las que están en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| **Producción Declaraciones de puerto (DP)** | | |
| PUERTO TIPO\_PUERTO IDENTIFICADOR FIN\_DE\_LINEA | port\_A1 proximity proximitySensor! | |
| **Producción Declaraciones con valor (DCV)** | | |
| DECLARACION TIPO\_DATO IDENTIFICADOR OPERADOR\_ASIGNACION **VALORES** FIN\_DE\_LINEA | var string holaMundo = 'HolaMundo'! | |
| **Producción Declaraciones sin valor (DSV)** | | |
| DECLARACION TIPO\_DATO IDENTIFICADOR FIN\_DE\_LINEA | const string number3 ! | |
| **Producción Asignaciones (A)** | | |
| IDENTIFICADOR OPERADOR\_ASIGNACION **VALORES** FIN\_DE\_LINEA | number2 = 3.3 ! | |
| **Producción funciones de los motores con valor (FMCV)** | | |
| MOTOR SIGNO\_AGRUPACION\_3 IDENTIFICADOR SIGNO\_PUNTUACION\_1 **VALORES** SIGNO\_PUNTUACION\_1 IDENTIFICADOR SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | move(motor1,180,boton1)! | |
| **Producción funciones de los motores sin valor (FMSV)** | | |
| MOTOR SIGNO\_AGRUPACION\_3 IDENTIFICADOR SIGNO\_PUNTUACION\_1 IDENTIFICADOR SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | start(motor1,boton1)! | |
| **Producción Método de impresora a consola (MIC)** | | |
| PALABRA\_RESERVADA\_6 SIGNO\_AGRUPACION\_3 PALABRA\_RESERVADA\_7 SIGNO\_PUNTUACION\_1 (**VALORES** | IDENTIFICADOR) SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | print(console,'El motor1 se esta moviendo')! | |
| **Producción Método de impresora a LCD (MIL)** | | |
| PALABRA\_RESERVADA\_6 SIGNO\_AGRUPACION\_3 IDENTIFICADOR SIGNO\_PUNTUACION\_1 (**VALORES** | IDENTIFICADOR) SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | print(Hola,'El motor1 se esta moviendo')! | |
| **Producción Operaciones (O)** | | |
| PALABRA\_RESERVADA\_3 IDENTIFICADOR OPERADOR\_ASIGNACION (IDENTIFICADOR | **VALORES**) OPERADOR\_ARITMETICO (IDENTIFICADOR | **VALORES**) FIN\_DE\_LINEA | operation result = a + b! | |
| **Producción funciones sin parámetros (FSP)** | | |
| ESTRUCTURA\_DE\_CONTROL\_1 IDENTIFICADOR SIGNO\_AGRUPACION\_3 SIGNO\_AGRUPACION\_4 SIGNO\_AGRUPACION\_1 (**FMCV | FMSV** | **MIC** | **MIL**)\* SIGNO\_AGRUPACION\_2 | function moverMotor(){  move(motor1,180,boton1)!  print(console,'El motor1 se esta moviendo')!  } | |
| **Producción funciones con parámetros (FCP)** | | |
| ESTRUCTURA\_DE\_CONTROL\_1 TIPO\_DATO IDENTIFICADOR SIGNO\_AGRUPACION\_3 TIPO\_DATO IDENTIFICADOR (SIGNO\_PUNTUACION\_1 TIPO\_DATO IDENTIFICADOR)\* SIGNO\_AGRUPACION\_4 SIGNO\_AGRUPACION\_1 (**O**)\* PALABRA\_RESERVADA\_5 IDENTIFICADOR FIN\_DE\_LINEA SIGNO\_AGRUPACION\_2 | function int sumaNumerosInt(int a, int b){  operation result = a + b!  return result!  } | |
| **Producción llamadas de funciones con valor (LFCV)** | | |
| PALABRA\_RESERVADA\_4 IDENTIFICADOR SIGNO\_AGRUPACION\_3 **VALORES** (SIGNO\_PUNTUACION\_1 **VALORES**)\* SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | call moverMotor(5)! | |
| **Producción llamadas de funciones sin valor (LFSV)** | | |
| PALABRA\_RESERVADA\_4 IDENTIFICADOR SIGNO\_AGRUPACION\_3 SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | call moverMotor()! | |
| **Producción Begin (B)** | | |
| ESTRUCTURA\_DE\_CONTROL\_2 SIGNO\_AGRUPACION\_1 (**LFCV** | **LFSV** | **FMCV** | **FMSV**)\* SIGNO\_AGRUPACION\_2 | begin{  call moverMotor()!  call detenerMotor()!  call escribirLCD(5)!  } | |
| **Producción Método de los sensores (MS)** | | |
| PALABRA\_RESERVADA\_4 IDENTIFICADOR OPERADOR\_ASIGNACION IDENTIFICADOR SIGNO\_PUNTUACION\_2 METODO\_SENSOR SIGNO\_AGRUPACION\_3 SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | call Distancia = puerto1.distance()! | |
| **Producción Método Delay (MD)** | | |
| PALABRA\_RESERVADA\_2 SIGNO\_AGRUPACION\_3 (IDENTIFICADOR | **VALORES**) SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | | delay(1000)! |
| **Producción Método Encender (ME)** | | |
| PALABRA\_RESERVADA\_1 SIGNO\_AGRUPACION\_3 IDENTIFICADOR SIGNO\_PUNTUACION\_1 IDENTIFICADOR SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | | ligther(led1,boton1)! |
| **Producción Método Apagar (MA)** | | |
| PALABRA\_RESERVADA\_8 SIGNO\_AGRUPACION\_3 IDENTIFICADOR SIGNO\_PUNTUACION\_1 IDENTIFICADOR SIGNO\_AGRUPACION\_4 FIN\_DE\_LINEA | | turn\_off(led1,boton1)! |
| **Producción If Relacional (IR)** | | |
| ESTRUCTURA\_DE\_CONTROL\_4 SIGNO\_AGRUPACION\_3 (IDENTIFICADOR | **VALORES**) OPERADOR\_RELACIONAL (IDENTIFICADOR | **VALORES**) SIGNO\_AGRUPACION\_4 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 | if(number1>number2){  call moverMotor()!  } | |
| **Producción If Lógico con dos valores (IL2V)** | | |
| ESTRUCTURA\_DE\_CONTROL\_4 SIGNO\_AGRUPACION\_3 (IDENTIFICADOR | **VALORES**) OPERADOR\_LOGICO (IDENTIFICADOR | **VALORES**) SIGNO\_AGRUPACION\_4 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 | if(number1 and number2){  call moverMotor()!  } | |
| **Producción If Lógico con un valor (IL1V)** | | |
| ESTRUCTURA\_DE\_CONTROL\_4 SIGNO\_AGRUPACION\_3 (IDENTIFICADOR | **VALORES**) SIGNO\_AGRUPACION\_4 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 | if(number1){  call moverMotor()!  } | |
| **Producción Else Relacional (ER)** | | |
| ESTRUCTURA\_DE\_CONTROL\_4 SIGNO\_AGRUPACION\_3 (IDENTIFICADOR | **VALORES**) OPERADOR\_RELACIONAL (IDENTIFICADOR | **VALORES**) SIGNO\_AGRUPACION\_4 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 ESTRUCTURA\_DE\_CONTROL\_5 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 | if(number1>number2){  call moverMotor()!  }else{  call detenerMotor()!  } | |
| **Producción Else Lógico con dos valores (EL2V)** | | |
| ESTRUCTURA\_DE\_CONTROL\_4 SIGNO\_AGRUPACION\_3 (IDENTIFICADOR | **VALORES**) OPERADOR\_LOGICO (IDENTIFICADOR | **VALORES**) SIGNO\_AGRUPACION\_4 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 ESTRUCTURA\_DE\_CONTROL\_5 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 | if(number1 and number2){  call moverMotor()!  }else{  call detenerMotor()!  } | |
| **Producción Else Lógico con un valor (EL1V)** | | |
| ESTRUCTURA\_DE\_CONTROL\_4 SIGNO\_AGRUPACION\_3 (IDENTIFICADOR | **VALORES**) SIGNO\_AGRUPACION\_4 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 ESTRUCTURA\_DE\_CONTROL\_5 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 | if(number1){  call moverMotor()!  }else{  call detenerMotor()!  } | |
| **Producción Loop (L)** | | |
| ESTRUCTURA\_DE\_CONTROL\_3 SIGNO\_AGRUPACION\_1 (**MS** | **MIC** | **MIL** | **A** | **DCV** | **DSV** | **FMCV** | **FMSV** | **LFCV** | **LFSV** | **IR** | **IL2V** | **IL1V** | **ER** | **EL2V** | **EL1V** | **MD** | **ME** | **MA**)\* SIGNO\_AGRUPACION\_2 | loop{  if(number1>number2){  call moverMotor()!  }else{  call detenerMotor()!  }  call Distancia = puerto1.distance()!  number2 = 25.0 !  } | |

**ERRORES SINTÁCTICOS DEL COMPILADOR**

Los errores sintácticos son aquellas producciones que no son aceptadas en el compilador, esto se da por no cumplir con las reglas de escritura de las producciones, por lo tanto los errores son las concatenaciones de componentes léxicos que no son aceptados por el compilador y para este, los errores son los siguientes:

|  |  |
| --- | --- |
| **NOMBRE DEL ERROR** | **DESCRIPCIÓN** |
| **Error\_sintáctico\_0** | Se esperaba un puerto (port\_A0 | port\_A1 | port\_A2 | port\_A3 | port\_A4 | port\_A5 | port\_B0 | port\_B1 | port\_B2 | port\_B3 | port\_B4 | port\_B5 | port\_B6 | port\_B7 | port\_C2 | port\_C3 | port\_C4 | port\_C5 | port\_C6 | port\_C7 | port\_D0 | port\_D1 | port\_D2 | port\_D3 | port\_D4 | port\_D5 | port\_D6 | port\_D7) en la declaración de puerto |
| **Error\_sintáctico\_1** | Se esperaba un tipo de puerto (proximity | temperature | LED | LCD | button | motor) en la declaración de puerto |
| **Error\_sintáctico\_2** | Se esperaba un identificador en la declaración de puerto |
| **Error\_sintáctico\_3** | Se esperaba el fin de línea (!) en la declaración de puerto |
| **Error\_sintáctico\_4** | Hay dos o más componentes léxicos fuera de lugar en la declaración de puerto |
| **Error\_sintáctico\_5** | Faltan dos o más componentes léxicos en la declaración de puerto |
| **Error\_sintáctico\_6** | El puerto (port\_A0 | port\_A1 | port\_A2 | port\_A3 | port\_A4 | port\_A5 | port\_B0 | port\_B1 | port\_B2 | port\_B3 | port\_B4 | port\_B5 | port\_B6 | port\_B7 | port\_C2 | port\_C3 | port\_C4 | port\_C5 | port\_C6 | port\_C7 | port\_D0 | port\_D1 | port\_D2 | port\_D3 | port\_D4 | port\_D5 | port\_D6 | port\_D7) no está asociado a una declaración de puerto |
| **Error\_sintáctico\_7** | El tipo de puerto (proximity | temperature | LED | LCD | button | motor) no está asociado a una declaración de puerto |
| **Error\_sintáctico\_8** |  |
| **Error\_sintáctico\_9** |  |
| **Error\_sintáctico\_10** |  |
| **Error\_sintáctico\_11** |  |
| **Error\_sintáctico\_12** |  |
| **Error\_sintáctico\_13** |  |
| **Error\_sintáctico\_14** |  |