**ANÁLISIS SEMÁNTICO**

El análisis semántico es la tercera fase de un compilador y se encarga de darle sentido a las estructuras generadas, estas estructuras se les da sentido evaluando los componentes léxicos por las que están hechas, por ende necesitamos los componentes léxicos generados, los tipos de datos que existen con su debido limite, las estructuras que tienen sentido acorde a sus componentes léxicos 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** |
| **Número entero** | ({Digito})({Digito})\* |
| **Número real** | ({Digito})+.({Digito})({Digito})\* |
| **Condicionales** | (true | false) |
| **Identificador cadena** | ({Comilla})({Letra} | {Dígito})({Letra} | {Dígito} | {Símbolo})\*({Comilla}) |
| **Identificador** | ({Letra})({Letra} | {Digito})\* |
| **VALORES** | (**Numero Entero** | **Número Real** | **Identificador Cadena** | **Condicionales**) |
| **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\_C0 | port\_C1 | 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 | LED\_RGB | LCD | button | motor) |
| **Tipo dato** | (int | string | decimal | boolean) |
| **Motor** | (move | restart | start) |
| **Metodo sensor** | (distance | state | time | degree) |
| **Estructura control** | (begin | loop | if | else | function) |
| **Método** | (delay | call | operation | return | print | console) |
| **Declaración** | (var | const) |
| **Operador lógico** | (and | or | not) |
| **Operador relacional** | (> | < | >= | <=) |
| **Operador aritmético** | (+ | - | \* | /) |
| **Operador asignación** | (=) |
| **Signo agrupación** | ( { | } | ( | )) |
| **signo puntuacion** | (, | .) |
| **Fin de línea** | (!) |

**TIPOS DE DATOS DEL COMPILADOR**

Los tipos de datos son aquellos valores que pueden tener las variables en este compilador, donde también cuentan con límite y espacio en memoria, por ende los tipos de datos del compilador que son aceptados están en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| **TIPO DE DATO** | **LIMITE** | **ESPACIO EN MEMORIA** |
| **int** | -500000 - 500000 | 6-7 Byte |
| **string** | Cadena de 98 caracteres | 100 Byte |
| **decimal** | -999.9999 – 999.9999 | 8-9 Byte |
| **boolean** | true,false | 4-5 Byte |
| **puerto** | proximity, temperature, LED, LED\_RGB, LCD, button, motor | 3-11 Byte |
| **sensor** | distance, state, time, degree | 4-8 Byte |

**ESTRUCTURAS DE CÓDIGO CON SENTIDO DEL COMPILADOR**

Las estructuras de código con sentido son generadas a través de los tipos de datos, dónde se evalúan las estructuras ya generadas para que tengan un sentido que es aceptado por este compilador, por ende las estructuras con sentido de este compilador que se pueden generar son las que están en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| **Estructuras de Declaraciones de puerto (DP) proximity** | | |
| (Puerto) (**proximity**) (Identificador) (Fin de línea) | port\_A1 proximity proximitySensor! | |
| **Estructuras de Declaraciones de puerto (DP) temperature** | | |
| (Puerto) (**temperature**) (Identificador) (Fin de línea) | port\_A2 temperature temperatureSensor! | |
| **Estructuras de Declaraciones de puerto (DP) LED** | | |
| (Puerto) (**LED**) (Identificador) (Fin de línea) | port\_A3 LED LEDSensor! | |
| **Estructuras de Declaraciones de puerto (DP) LED\_RGB** | | |
| (Puerto) (**LED\_RGB**) (Identificador) (Fin de línea) | port\_A4 LED\_RGB LED\_RGBSensor! | |
| **Estructuras de Declaraciones de puerto (DP) LCD** | | |
| (Puerto) (**LCD**) (Identificador) (Fin de línea) | port\_A4 LCD LCDSensor! | |
| **Estructuras de Declaraciones de puerto (DP) button** | | |
| (Puerto) (**button**) (Identificador) (Fin de línea) | port\_A5 button buttonSensor! | |
| **Estructuras de Declaraciones de puerto (DP) motor** | | |
| (Puerto) (**motor**) (Identificador) (Fin de línea) | port\_B6 motor motorSensor! | |
| **Estructuras de Declaraciones con valor (DCV) string** | | |
| (Declaración) (**string**) (Identificador) (Operador asignación) (**Cadena**) (Fin de línea) | var string holaMundo = 'HolaMundo'! | |
| **Estructuras de Declaraciones con valor (DCV) int** | | |
| (Declaración) (**int**) (Identificador) (Operador asignación) (**Numero entero**) (Fin de línea) | var int numero1 = 234! | |
| **Estructuras de Declaraciones con valor (DCV) decimal** | | |
| (Declaración) (**decimal**) (Identificador) (Operador asignación) (**Numero real**) (Fin de línea) | var decimal numero2 = 34.56! | |
| **Estructuras de Declaraciones con valor (DCV) boolean** | | |
| (Declaración) (**boolean**) (Identificador) (Operador asignación) (**Condicionales**) (Fin de línea) | var boolean valor3 = true! | |
| **Estructuras de Declaraciones sin valor (DSV) string** | | |
| (Declaración) (string) (Identificador) (Fin de línea) | const string valor1 ! | |
| **Estructuras de Declaraciones sin valor (DSV) int** | | |
| (Declaración) (int) (Identificador) (Fin de línea) | const int valor2 ! | |
| **Estructuras de Declaraciones sin valor (DSV) decimal** | | |
| (Declaración) (decimal) (Identificador) (Fin de línea) | const decimal valor3 ! | |
| **Estructuras de Declaraciones sin valor (DSV) boolean** | | |
| (Declaración) (boolean) (Identificador) (Fin de línea) | const boolean valor4 ! | |
| **Estructura de Asignaciones (A) Depende del tipo de dato que es la variable** | | |
| (**variable**) (Operador asignación) (**VALORES**) (Fin de línea) | valor3 = 3.3 ! | |
| **Estructura de funciones de los motores con valor (FMCV) move** | | |
| (**move**) (Signo agrupación “(” ) (**Variable de puerto asociado al motor**) (Signo puntuación “,” ) (**Numero entero**) (Signo agrupación “)” ) (Fin De línea) | move(motorSensor,34)! | |
| **Estructura de funciones de los motores con valor (FMCV) start** | | |
| (**start**) (Signo agrupación “(” ) (**Variable de puerto asociado al motor**) (Signo agrupación “)” ) (Fin De línea) | start(motorSensor)! | |
| **Estructura de funciones de los motores con valor (FMCV) restart** | | |
| (**restart**) (Signo agrupación “(” ) (**Variable de puerto asociado al motor**) (Signo agrupación “)” ) (Fin De línea) | restart(motorSensor)! | |
| **Estructura de Método de impresora a consola (MIC)** | | |
| (**print**) (Signo agrupación “(”) (**console**) (Signo puntuación “,” ) (**Cadena** | **Nombre de variable con valor cadena**) (Signo agrupación “)” ) (Fin de línea) | print(console,'El motor1 se esta moviendo')! | |
| **Estructura de Método de impresora a LCD (MIL)** | | |
| (**print**) (Signo agrupación “(”) (**Variable que está asociado a un puerto con el sensor LCD**) (Signo puntuación “,” ) (**Cadena** | **Nombre de variable con valor cadena**) (Signo agrupación “)” ) (Fin de línea) | print(LCDSensor,'El motor1 se esta moviendo')! | |
| **Estructura de Operaciones (O)** | | |
| (**operation**) (**Variable con el tipo de dato acorde al tipo de dato de la operación**) (Operador asignación) (**Variable declarada en los parámetros de la función o un valor acorde al tipo de dato de la variable donde se guardara**) (Operador aritmético) (**Variable declarada en los parámetros de la función o un valor acorde al tipo de dato de la variable donde se guardara**) (Fin de línea) | operation valor2 = a + b! | |
| **Estructura de las funciones sin parámetros (FSP)** | | |
| (**function**) (**Nombre de la función**) (Signo agrupación “(” ) (Signo agrupación “)” ) (Signo agrupación “{” ) (**FMCV | FMSV**)\* (**MIC** | **MIL**)\* (Signo agrupación “}” ) | function moverMotor(){  move(motorSensor,60)!  print(console,'El motor1 se esta moviendo')!  } | |
| **Estructura de las funciones con parámetros (FCP)** | | |
| (**function**) (**Tipo dato a retornar**) (**Nombre de la función**) (Signo agrupación “(” ) (Tipo dato) (**Nombre de la variable parámetro**) ((Signos puntuación “,”) (Tipo dato) (**Nombre de la variable parámetro**))\* (Signo agrupación “)” ) (Signo agrupación “{” ) (**O**)\* (**return**) (**Variable con el mismo tipo de dato a retornar de la función**) (Fin de línea) (Signo agrupación “}” ) | function int sumaNumerosInt(int a, int b){  operation valor2 = a + b!  return valor2!  } | |
| **Estructura de llamadas de funciones con valor (LFCV)** | | |
| (**call**) (**Nombre de la función a llamar**) (Signo agrupación “(” ) (**Valor con el mismo tipo de dato del parámetro de la función**) ((Signo puntuación “,”) (**Valor con el mismo tipo de dato del parámetro de la función**))\* (Signo agrupación “)” ) (Fin de línea) | call sumaNumerosInt (5,10)! | |
| **Estructura de llamadas de funciones sin valor (LFSV)** | | |
| (**call**) (**Nombre de la función a llamar**) (Signo agrupación “(” ) (Signo agrupación “)” ) (Fin de línea) | call moverMotor()! | |
| **Estructura de control Begin (B)** | | |
| (**begin**) (Signo agrupación “{” ) ((**LFCV**) | (**LFSV**) | (**FMCV**) | **FMSV**)\* (Signo agrupación “}” ) | begin{  call moverMotor()!  } | |
| **Estructura de Método de los sensores (MS)** | | |
| (**call**) (**Nombre de variable tipo cadena**) (Operador asignación) (**Variable de puerto asociado a algún sensor**) (Signo puntuación “.” ) (Método sensor) (Signo agrupación “(” ) (Signo agrupación “)” ) (Fin de línea) | call Distancia = proximitySensor.distance()! | |
| **Estructura de Método Delay (MD)** | | |
| (**delay**) (Signo agrupación “(” ) ((**Variable de tipo de dato Entero**) | (**Valor de tipo Numero entero**) (Signo agrupación “)” ) (Fin de línea) | | delay(1000)! |
| **Estructura de control If Relacional (IR)** | | |
| (**if**) (Signo agrupación “(” ) ((**Variable de tipo Entero o Decimal**) | (**Valor de tipo Entero o Decimal**)) ((Operador relacional) | (Operador asignación)) ((**Variable de tipo Entero o Decimal**) | (**Valor de tipo Entero o Decimal**)) (Signo agrupación “)” ) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**))\* (Signo agrupación “}” ) | if(numero1>valor2){  call moverMotor()!  } | |
| **Estructura de control If Lógico con dos valores (IL2V)** | | |
| (**if**) (Signo agrupación “(” ) ((**Variable de tipo Booleano**) | (**Valor de tipo Booleano**)) (Operador lógico) ((**Variable de tipo Booleano**) | (**Valor de tipo Booleano**)) (Signo agrupación “)” ) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**))\* (Signo agrupación “}” ) | if(valor3 and valor4){  call moverMotor()!  } | |
| **Estructura de control If Lógico con un valor (IL1V)** | | |
| (**if**) (Signo agrupación “(” ) ((**Variable de tipo Booleano**) | (**Valor de tipo Booleano**)) (Signo agrupación “)” ) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**))\* (Signo agrupación “}” ) | if(valor3){  call moverMotor()!  } | |
| **Estructura de control Else Relacional (ER)** | | |
| (**if**) (Signo agrupación “(” ) ((**Variable de tipo Entero o Decimal**) | (**Valor de tipo Entero o Decimal**)) ((Operador relacional) | (Operador asignación)) ((**Variable de tipo Entero o Decimal**) | (**Valor de tipo Entero o Decimal**)) (Signo agrupación “)” ) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**))\* (Signo agrupación “}” ) (**else**) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**))\* (Signo agrupación “}” ) | if(numero1>valor2){  call moverMotor()!  }else{    } | |
| **Estructura de control Else Lógico con dos valores (EL2V)** | | |
| (**if**) (Signo agrupación “(” ) ((**Variable de tipo Booleano**) | (**Valor de tipo Booleano**)) (Operador lógico) ((**Variable de tipo Booleano**) | (**Valor de tipo Booleano**)) (Signo agrupación “)” ) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**))\* (Signo agrupación “}” ) (**else**) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**))\* (Signo agrupación “}” ) | if(valor3 and valor4){    }else{  call moverMotor()!  } | |
| **Estructura de control Else Lógico con un valor (EL1V)** | | |
| (**if**) (Signo agrupación “(” ) ((**Variable de tipo Booleano**) | (**Valor de tipo Booleano**)) (Signo agrupación “)” ) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**))\* (Signo agrupación “}” ) (**else**) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**))\* (Signo agrupación “}” ) | if(valor3){  call moverMotor()!  }else{    } | |
| **Estructura de control Loop (L)** | | |
| (**loop**) (Signo agrupación “{” ) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV** | **IR** | **IL2V** | **IL1V** | **ER** | **EL2V** | **EL1V**) | (**MD**))\* (Signo agrupación “}” ) | loop{  if(valor3 and valor4){  call moverMotor()!  }else{    }  call Distancia = proximitySensor.distance()!    } | |

**ERRORES SEMÁNTICOS DEL COMPILADOR**

Los errores semánticos son aquellas estructuras que cuentan con valores incorrectos entre sí, no hay coherencia, los errores son los siguientes:

|  |  |
| --- | --- |
| **NOMBRE DEL ERROR** | **DESCRIPCIÓN** |
| Error\_123 | El tipo de dato no es el mismo que el valor asignado en la declaración |
| Error\_124 | El valor asignado en la declaración no está en rango |
| Error\_125 | La variable ya ha sido declarada |
| Error\_126 | La variable es una constante y no es posible modificar su valor |
| Error\_127 | La variable no ha sido declarada |
| Error\_128 | El tipo de dato del valor no es el mismo que el de la variable a asignar |
| Error\_129 | La variable no está asociado a un sensor de tipo motor |
| Error\_130 | La función del motor move no tiene valor |
| Error\_131 | La función del motor start o restart tiene valor |
| Error\_132 | La función del motor move tiene un valor diferente de tipo entero |
| Error\_133 | En el método impresora el valor no es de tipo string |
| Error\_134 | En el método de la impresora el valor de la variable no es de tipo string |
| Error\_135 | En el método de la impresora la variable no está asociada a un sensor de tipo LCD |
| Error\_136 | La variable del operando 1 no está declarada en los parámetros |
| Error\_137 | La variable del operando 2 no está declarada en los parámetros |
| Error\_138 | El valor de la operación no está en rango |
| Error\_139 | El tipo de dato de la variable es diferente a los operandos de la operación |
| Error\_140 | Ya existe una función con el mismo nombre |
| Error\_141 | El valor a retornar no es el mismo tipo de dato de la función |
| Error\_142 | No existe una función con ese nombre |
| Error\_143 | El valor dado en el parámetro no es del mismo tipo de dato del parámetro de la función |
| Error\_144 | La variable no es tipo cadena |
| Error\_145 | La variable no está asociada a un tipo de sensor |
| Error\_146 | El método del sensor es indefinido al sensor |
| Error\_147 | El valor de la variable no es de tipo de dato entero |
| Error\_148 | El valor no es de tipo de dato entero |
| Error\_149 | El valor 1 de la condición no es de tipo entero o decimal |
| Error\_150 | El valor 2 de la condición no es de tipo entero o decimal |
| Error\_151 | El valor de la variable 1 de la condición no es de tipo entero o decimal |
| Error\_152 | El valor de la variable 2 de la condición no es de tipo entero o decimal |
| Error\_153 | El valor 1 y el valor 2 de la condición son de diferentes tipos de datos |
| Error\_154 | El valor de la variable 1 y el valor de la variable 2 de la condición son de diferentes tipos de datos |
| Error\_155 | El valor 1 de la condición no es de tipo booleano |
| Error\_156 | El valor 2 de la condición no es de tipo booleano |
| Error\_157 | El valor de la variable 1 de la condición no es de tipo booleano |
| Error\_158 | El valor de la variable 2 de la condición no es de tipo booleano |
| Error\_159 | El valor de la condición no es de tipo booleano |
| Error\_160 | El valor de la variable de la condición no es de tipo booleano |
| Error\_161 | La variable no tiene valor asignado |
| Error\_162 | El valor no está en rango |