RoboKit

ESTRUCTURAS DE CÓDIGO DEL COMPILADOR

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

Estructuras de Declaraciones de puerto (DP)	
(Puerto) (Tipo puerto) (Identificador) (Fin de línea)	port_A1 proximity proximitySensor!
Estructuras de Declaraciones con valor (DCV)	
(Declaración) (Tipo dato) (Identificador) (Operador asignación) (**VALORES**) (Fin de línea)	var string holaMundo = 'HolaMundo'!
Estructuras de Declaraciones sin valor (DSV)	
(Declaración) (Tipo de dato) (Identificador) (Fin de línea)	const string number3 !
Estructura de Asignaciones (A)	
(Identificador) (Operador asignación) (**VALORES**) (Fin de línea)	number2 = 3.3 !
Estructura de funciones de los motores con valor (FMCV)	
(Motor) (Signo agrupación “(”) (Identificador) (Signo puntuación “,”) (**VALORES**) (Signo puntuación “,”) (Identificador) (Signo agrupación “)”) (Fin De línea)	move(motor1,180,boton1)!
Estructura de funciones de los motores sin valor (FMSV)	
(Motor) (Signo agrupación “(”) (Identificador) (Signo puntuación “,”) (Identificador) (Signo agrupación “)”) (Fin De línea)	start(motor1,boton1)!
Estructura de Método de impresora a consola (MIC)	
(Método “print”) (Signo agrupación “(”) (Método “console”) (Signo puntuación “,”) (**VALORES** | Identificador) (Signo	print(console,'El motor1 se esta moviendo')!

agrupación “)”) (Fin de línea)	
Estructura de Método de impresora a LCD (MIL)	
(Método “print”) (Signo agrupación “(”) (Identificador) (Signo puntuación “,”) (**VALORES** | Identificador) (Signo agrupación “)”) (Fin de línea)	print(Hola,'El motor1 se esta moviendo')!
Estructura de Operaciones (O)	
(Método “operation”) (Identificador) (Operador asignación) (Identificador | **VALORES**) (Operador aritmético) (Identificador | **VALORES**) (Fin de línea)	operation result = a + b!
Estructura de las funciones sin parámetros (FSP)	

(Estructura control “function”) (Identificador) (Signo agrupación “(”) (Signo agrupación “)”) (Signo agrupación “{”) (**FMCV | FMSV**)* (**MIC** | **MIL**)* (Signo agrupación “}”)	function moverMotor(){
	move(motor1,180,boton1)!
	print(console,'El motor1 se esta moviendo')!
	}

Estructura de las funciones con parámetros (FCP)	

(Estructura control “function”) (Tipo dato) (Identificador) (Signo agrupación “(”) (Tipo dato) (Identificador) ((Signos puntuación “,”) (Tipo dato) (Identificador))* (Signo agrupación “)”) (Signo agrupación “{”) (**O**)* (Método “return”) (Identificador) (Fin de línea) (Signo agrupación “}”)	function int sumaNumerosInt(int a, int b){
	operation result = a + b!
	return result!
	}

Estructura de llamadas de funciones con valor (LFCV)	
(Método “call”) (Identificador) (Signo agrupación “(”) (**VALORES**) ((Signo puntuación “,”) (**VALORES**))* (Signo agrupación “)”) (Fin de línea)	call moverMotor(5)!
Estructura de llamadas de funciones sin valor (LFSV)	
(Método “call”) (Identificador) (Signo agrupación “(”) (Signo agrupación “)”) (Fin de línea)	call moverMotor()!
Estructura de control Begin (B)	

(Estructura control “begin”) (Signo agrupación “{”) ((**LFCV**) | (**LFSV**) | (**FMCV**) | **FMSV**)* (Signo agrupación “}”)	begin{
	call moverMotor()!
	call detenerMotor()!
	call escribirLCD(5)!
	}

Estructura de Método de los sensores (MS)	
(Método “call”) (Identificador) (Operador asignación) (Identificador) (Signo puntuación “.”) (Método sensor) (Signo agrupación “(”) (Signo agrupación “)”) (Fin de línea)	call Distancia = puerto1.distance()!
Estructura de Método Delay (MD)	
(Método “delay”) (Signo agrupación “(”) ((Identificador) | (**VALORES**)) (Signo agrupación “)”) (Fin de línea)	delay(1000)!
Estructura de Método Encender (ME)	
(Metodo “ligther”) (Signo agrupación “(”) (Identificador) (Signos puntuación “,”) (Identificador) (Signo agrupación “)”) (Fin de línea)	ligther(led1,boton1)!
Estructura de control If Relacional (IR)	

(Estructura control “if”) (Signo agrupación “(”) ((Identificador) | (**VALORES**)) ((Operador relacional) | (Operador asignación)) ((Identificador) | (**VALORES**)) (Signo agrupación “)”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**) | (**ME**))* (Signo agrupación “}”)	if(number1>number2){
	call moverMotor()!
	}

Estructura de control If Lógico con dos valores (IL2V)	

(Estructura control “if”) (Signo agrupación “(”) ((Identificador) | (**VALORES**)) (Operador lógico) ((Identificador) | (**VALORES**)) (Signo agrupación “)”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**) | (**ME**))* (Signo agrupación “}”)	if(number1 and number2){
	call moverMotor()!
	}

Estructura de control If Lógico con un valor (IL1V)	

(Estructura control “if”) (Signo agrupación “(”) ((Identificador) | (**VALORES**)) (Signo agrupación “)”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**) | (**ME**))* (Signo agrupación “}”)	if(number1){
	call moverMotor()!
	}

Estructura de control Else Relacional (ER)	

(Estructura control “if”) (Signo agrupación “(”) ((Identificador) | (**VALORES**)) ((Operador relacional) | (Operador asignación)) ((Identificador) | (**VALORES**)) (Signo agrupación “)”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**) | (**ME**))* (Signo agrupación “}”) (Estructura control “else”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**) | (**ME**))* (Signo agrupación “}”)	if(number1>number2){
	call moverMotor()!
	}else{
	call detenerMotor()!
	}

Estructura de control Else Lógico con dos valores (EL2V)	

(Estructura control “if”) (Signo agrupación “(”) ((Identificador) | (**VALORES**)) (Operador lógico) ((Identificador) | (**VALORES**)) (Signo agrupación “)”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**) | (**ME**))* (Signo agrupación “}”) (Estructura control “else”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**) | (**ME**))* (Signo agrupación “}”)	if(number1 and number2){
	call moverMotor()!
	}else{
	call detenerMotor()!
	}

Estructura de control Else Lógico con un valor (EL1V)	

(Estructura control “if”) (Signo agrupación “(”) ((Identificador) | (**VALORES**)) (Signo agrupación “)”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**) | (**ME**))* (Signo agrupación “}”) (Estructura control “else”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV**) | (**MD**) | (**ME**))* (Signo agrupación “}”)	if(number1){
	call moverMotor()!
	}else{
	call detenerMotor()!
	}

Estructura de control Loop (L)	

(Estructura control “loop”) (Signo agrupación “{”) ((**MS**) | (**MIC**) | (**MIL**) | (**A**) | (**DCV**) | (**DSV**) | (**FMCV**) | (**FMSV**) | (**LFCV**) | (**LFSV** | **IR** | **IL2V** | **IL1V** | **ER** | **EL2V** | **EL1V**) | (**MD**) | (**ME**))* (Signo agrupación “}”)	loop{
	if(number1>number2){
	call moverMotor()!
	}else{
	call detenerMotor()!
	}
	call Distancia = puerto1.distance()!
	number2 = 25.0 !
	}