[Nombre de la compañía]

REPORTE GRAMATICAL

COMPILADORES 2

Ricardo Menéndez Tobías
201602916

Indice

Contenido

Indice 1
Terminales 2
Palabras Reservadas 3
Gramática Ascendente 4
Gramática Descendente ..... **¡Error! Marcador no definido.**

Terminales

Token	ER	Descripción
SUMA	+	Sirve para la operación de Suma
RESTA	-	Sirve para la operación de Resta
MULTI	*	Sirve para la operación de Multiplicación
DIV	/	Sirve para la operación de División
PORCENTAJE	%	Sirve para la operación de Modulo
PUNTERO	&	Sirve para indicar que una variable es un puntero
NOT	!	Sirve para la operación de Negación
AND	&&	Sirve para la operación de AND
OR	||	Sirve para la operación de Or
SXOR	xor	Sirve para la operación de Xor
DIGUAL	==	Sirve para la operación de Igualdad
IGUAL	=	Sirve para indicar una asignación
DESIGUAL	!=	Sirve para la operación de Desigualdad
MAYORIGUAL	>=	Sirve para la operación de Mayor o Igual
MENORIGUAL	<=	Sirve para la operación de Menor o Igual
MAYOR	>	Sirve para la operación de Mayor que
MENOR	<	Sirve para la operación de Menor que
BNOT	~	Sirve para la operación de Negación de Bit a Bit
BAND	&	Sirve para la operación de AND de Bit a Bit
BOR	|	Sirve para la operación de OR de Bit a Bit
BXOR	^	Sirve para la operación de XOR de Bit a Bit
BLEFT	<<	Sirve para la operación de Corrimiento a la izquierda de Bit a Bit
BRIGHT	>>	Sirve para la operación de Corrimiento a la derecha de Bit a Bit
IZQPAR	(	Indica la parte de inicio de los parámetros del llamado a una función
DERPAR	)	Indica la parte del fin de los parámetros del llamado a una función
IZQLLAVE	[	Indica el inicio de una posición de un arreglo
DERLLAVE	]	Indica el final de la posición de un arreglo
PCOMA	;	Indica el final de una instrucción
DP	:	Indica que viene el cuerpo de una etiqueta
ID	[a-zA-Z_][a-zA-Z_0-9]*	Representa el nombre de una etiqueta
DOUBLE	d+.d+	Un valor con punto flotante
INTEGER	d+	Un valor entero
CORCHETES	{}	Indican el abrir y cerrar de un cuerpo de código.
STR	‘ .*? ‘	Una valor de cadena
COMENTARIO	//.*\n	Un comentario que se ignora

Palabras Reservadas

Token	Lexema	Descripción
MAIN	main	Indica el inicio del programa.
GOTO	goto	Indica un salto.
INT	int	Indica el tipo de dato Entero.
FLOAT	float	Indica el tipo de dato Flotante.
CHAR	char	Indica el tipo de dato Char.
PRINT	print	Representa la función nativa Print.
ARRAY	array	Representa la función nativa Array.
UNSET	unset	Representa la función nativa Unset.
IF	if	Indica el inicio de un salto condicional.
ABS	abs	Indica la función nativa de Valor Absoluto.
XOR	xor	Indica la operación de XOR.
RETURN	return	Indica la función que termina el metodo.
READ	readf	Representa la función nativa readf.
ELSE	else	Representa una vía alterna en la sentencia if.
WHILE	while	Representa el ciclo While
DO	do	Parte del ciclo Do-While
STRUCT	struct	palabra que representa un struct
SIZEOF	sizeof	paralbra que sirve para la expresión “sizeof”
INCRE	++	operador de auto incremento.
DECRE	--	operador de decremento.
MASIGUAL	+=	operador de suma directa.
MENOSIGUAL	-=	operador de resta directa.
PORIGUAL	*=	operador de multiplicación directa.
DIVIGUAL	/*	operador de división directa.

Gramática Ascendente

PRODUCCION	FUNCION
S -> LTags	Producción que representa el inicio del programa.
LTags -> LTags Tag | Tag	Producción que representa una lista de Etiquetas
Tag -> TYPE id (PARAMS2) { LInst } | struct id { Linst } | declaracion	Producción que representa que un método o un struct
LInst -> LInst Inst ; | Inst ;	Representa una lista de instrucciones.
Inst -> Asignacion |Iff | Declaracion |GOTO ID |CALL | For | While | ID : | DoWhile | ;	Representa todas las instrucciones que puede ejecutar el programa.
Asignacion -> Var2 = Exp	Producción que representa una asignación. Va de una variable a una expresión asignada.
Declaración ->TYPE Var2 = Exp	Sirve para ingresar una variable a la tabla de símbolos
For -> for (asig|dec ; exp ; exp) { LInst }	Ciclo for, pide una declaración o asignación como primer parámetro. Luego una expresión de comparación para ver el ciclo, y una expresión para iterar el ciclo.
While -> while (exp) { LInst }	Ciclo While, que recibe una expresión para evaluar si se ejeucta o no el cuerpo.
DoWhile -> Do { Linst } while (exp)	Lo mismo que el while, pero ejecuta primero y luego evalua la expresión.
Iff -> IF (Exp) { Linst } [else if (exp) { LInst}]* [else {Linst}]?	Producción que representa un if., Este puede venir a compañado de cero o varios else if o un else.
Exp -> E + E | E - E | E * E | E / E | E > E | E < E | E >= E | E <= E	Estas son todas las operaciones que puede realizar el lenguaje. No existe una precedencia de operadores debido a que no hay recursividad en la misma.

| E && E | E & E | E || E | E | E | E == E | E != E | E ^ E | E XOR E | E << E | E >> E | ~ E | ! E | ABS (E) | (TYPE) E	
E -> - E | SRT | ENTERO | DOUBLE | Var2 | { LEX }	Representan los valores que pueden tomar los datos de las expresiones
Var2 -> VAR | VAR Arr	Representa una variable junto a una lista de índices si estos existen.
Arr -> Arr [E] |Arr .VAR | . VAR | [E]	Crea una lista de índices de los arreglos.
TYPE -> int | float | double | char | void	Esta producción sirve para ver los diferentes tipos de datos, ya sea para declaraciones o casteos.
LEX -> LEX , E | E	Lista de expresiones
CALL -> id (PARAMS)	Llamado a una función
PARAMS -> PARAMS , exp | exp |	Lista de expresiones para parámetros
PARAMS -> PARAMS , TYPE Id | TYPE Id |	Lista de declaraciones de datos