Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Organización de Lenguajes y Compiladores 2

Guillermo Alfredo Peitzner Estrada – 201504468.

Gramática Descendente

Producción	Reglas Semánticas
INIT -> main: CUERPO	INIT.val = Principal() + CUERPO.val
CUERPO -> INSTRUCCIONES	CUERPO.val = INSTRUCCIONES.val
CUERPO ->	CUERPO.val = None
INSTRUCCIONES -> INSTRUCCION INSTRUCCION_P	INSTRUCCIONES_P.add(INSTRUCCION); INSTRUCCIONES_P.reverse(); INSTRUCCIONES.val = INSTRUCCIONES_P
INSTRUCCIONES_P -> INSTRUCCION INSTRUCCIONES_P	INSTRUCCIONES_P1.add(INSTRUCCION); INSTRUCCIONES_P.val = INSTRUCCIONES_P1
INSTRUCCIONES_P ->	INSTRUCCIONES_P.val = new list()
INSTRUCCION -> ETIQUETA	INSTRUCCION.val = ETIQUETA.val
INSTRUCCION -> SALTO	INSTRUCCION.val = SALTO.val
INSTRUCCION -> ASIGNACION	INSTRUCCION.val = ASIGNACION.val
INSTRUCCION -> ASIGNACION_ARREGLO	INSTRUCCION.val = ASIGNACION_ARREGLO.val
INSTRUCCION -> ELIMINAR	INSTRUCCION.val = Eliminar.val
INSTRUCCION -> IMPRIMIR	INSTRUCCION.val = Imprimir.val
INSTRUCCION -> SI	INSTRUCCION.val = SI.val
INSTRUCCION -> SALIR	INSTRUCCION.val = SALIR.val
ETIQUETA -> identificador:	ETIQUETA.val = Etiqueta(identificador, t.lineno)
SALTO -> goto identificador;	SALTO.val = Salto(identificador, t.lineno)
ASIGNACION -> registro = EXPRESION;	ASIGNACION.val = Asignacion(registro, EXPRESION.val, t.lineno)
ASIGNACION_ARREGLO -> registro ACCESOS = EXPRESION;	ASIGNACION_ARREGLO.val = AsignacionArreglo(registro ', ACCESOS.val, EXPRESION.val, t.lineno)
ELIMINAR -> unset(EXPRESION);	ELIMINAR.val = Eliminar(EXPRESION.val, t.lineno)
IMPRIMIR -> print(EXPRESION);	IMPRIMIR.val = Imprimir(EXPRESION.val, t.lineno)
SI -> if (EXPRESION) goto identificador;	SI.val = Si(EXPRESION.val, identificador , t.lineno)

SALIR -> exit;	SALIR.val = Salir()
EXPRESION -> VALOR + VALOR **EXPRESION -> VALOR - VALOR** **EXPRESION -> VALOR * VALOR** **EXPRESION -> VALOR / VALOR** **EXPRESION -> VALOR % VALOR**	EXPRESION.val = OperacionAritmetica(VALOR.val, signo, VALOR.val)
EXPRESION -> VALOR && VALOR **EXPRESION -> VALOR || VALOR** **EXPRESION -> VALOR xor VALOR**	EXPRESION.val = OperacionLogica(VALOR.val, signo , VALOR.val)
EXPRESION -> VALOR & VALOR **EXPRESION -> VALOR | VALOR** **EXPRESION -> VALOR ^ VALOR** **EXPRESION -> VALOR << VALOR** **EXPRESION -> VALOR >> VALOR**	EXPRESION.val = OperacionBit(VALOR.val, signo, VALOR.val)
EXPRESION -> VALOR == VALOR **EXPRESION -> VALOR != VALOR** **EXPRESION -> VALOR >= VALOR** **EXPRESION -> VALOR <= VALOR** **EXPRESION -> VALOR > VALOR** **EXPRESION -> VALOR < VALOR**	EXPRESION.val = OperacionRelacional(VALOR.val, signo, VALOR.val)
EXPRESION -> - VALOR **EXPRESION -> & VALOR** **EXPRESION -> ! VALOR** **EXPRESION -> ~ VALOR**	EXPRESION.val = OperacionCasteo(signo, VALOR.val)
EXPRESION -> read() **EXPRESION -> abs(VALOR)** **EXPRESION -> array()**	EXPRESION.val = OperacionFuncion(función, VALOR.val)
EXPRESION -> (int) VALOR **EXPRESION -> (float) VALOR** **EXPRESION -> (char) VALOR**	EXPRESION.val = OperacionCasteo(casteo, VALOR.val)
EXPRESION -> VALOR	EXPRESION.val = VALOR.val
VALOR -> entero	VALOR.val = Entero(entero)
VALOR -> cadena	VALOR.val = Cadena(cadena)
VALOR -> caracter	VALOR.val = Caracter(caracter)
VALOR -> decimal	VALOR.val = Decimal(decimal)
VALOR -> registro	VALOR.val = Registro(registro)
ACCESOS -> ACCESO INSTRUCCION_P	ACCESOS_P.add(ACCESO); ACCESOS_P.reverse(); ACCESOS.val = ACCESOS_P
ACCESOS_P -> ACCESO ACCESOS_P	ACCESOS_P1.add(ACCESO); ACCESOS_P.val = ACCESOS_P1
ACCESOS_P ->	ACCESOS_P.val = new list()
ACCESO -> [EXPRESION]	ACCESO.val = EXPRESION.val