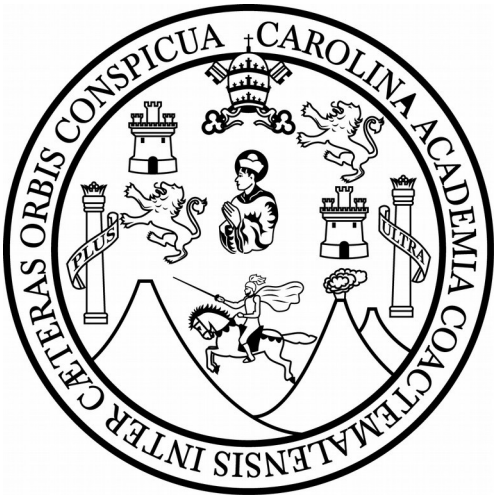
CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISIÓN CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA CIENCIAS Y SISTEMAS



LABORATORIO DE ORGANIZACIÓN LENGUAJES Y COMPILADORES 2

“SEXTO SEMESTRE”

ING.: JOSÉ MOISÉS GRANADOS GUEVARA

AUX.: EDVIN TEODORO GONZÁLEZ RAFAEL

ESTUDIANTE: Bryan René Gómez Gómez – 201730919

PROYECTO: “Manual Técnico – Segundo Proyecto, Fase - 1”

FECHA: 15 de Octubre de 2,020

Analizador Léxico

TOKEN	TIPO
int	**Palabra Reservada**
float	**Palabra Reservada**
char	**Palabra Reservada**
getch	**Palabra Reservada**
#include	**Palabra Reservada**
VB	**Palabra Reservada**
PY	**Palabra Reservada**
JAVA	**Palabra Reservada**
scanf	**Palabra Reservada**
%d	**Palabra Reservada**
%c	**Palabra Reservada**
%f	**Palabra Reservada**
“==”	**Palabra Reservada**
!=	**Palabra Reservada**
<	**Palabra Reservada**
>	**Palabra Reservada**
<=	**Palabra Reservada**
>=	**Palabra Reservada**
!	**Palabra Reservada**
&&	**Palabra Reservada**
->->	**Palabra Reservada**
clrsrc	**Palabra Reservada**
void	**Palabra Reservada**
main	**Palabra Reservada**
const	**Palabra Reservada**
if	**Palabra Reservada**
else	**Palabra Reservada**
else if	**Palabra Reservada**
switch	**Palabra Reservada**
case	**Palabra Reservada**
default	**Palabra Reservada**
break	**Palabra Reservada**
for	**Palabra Reservada**
do	**Palabra Reservada**

printf	**Palabra Reservada**
Console	**Palabra Reservada**
System.out.print	**Palabra Reservada**
System.out.println	**Palabra Reservada**
while	**Palabra Reservada**
=	**Palabra Reservada**
	Signo punto y coma
:	**Signo dos puntos**
(	**Signo Paréntesis Abrir**
)	**Signo Paréntesis Cerrar**
-	**Signo menos**
*	**Signo por**
+	**Signo mas**
[	**Llave Abir**
]	**Llave Cerrar**
?	**Interrogación Cerrada**
,	**Coma**
&	**Ampersand**
(Digito)+	**Expresión Numero**
(Digito)+((.)(Digito)+)*	**Expresión Decimal**
LetraMinuscula	**Expresión Terminal**
LetraMayuscula	**Expresión No Terminal**
(Letra -> "_") → (Letra -> "_" -> Dígito)*	**Expresión Identificador**
(Dígito)+	**Expresión Número**
Comillas → (Letra -> Símbolo -> Dígito) + → Comillas	**Expresión Cadena**
"/*" → (Letra -> Símbolo -> Dígito) + → ""	**Expresión Comentario Completo**
"//" → (Letra -> Símbolo -> Dígito) +	**Expresión Comentario de Linea**
Comillas Simples → (Letra -> Símbolo -> Dígito) + → Comillas Simple	**Expresión Caracter**
" VB "	**Libreria VB**
" Java "	**Libreria Java**
" PY "	**Libreria Python**
< ^>	**Libreria C**

ANALIZADOR SINTÁCTICO
PRINCIPAL

G = [N, T, P, S **]**

Terminal (T):

COMENTARIO, COD_VB, COD_PY, COD_JAVA, GETCH, INCLUDE, LIBRERIA_CLASES, LIBRERIA_JAVA, LIBRERIA_PYTHON, LIBRERIA_VB, LIBRERIA_C, CARACTER, VB, PY, JAVA, INT, FLOAT, CHAR, MAS, MENOS, POR, DIV, MOD, MAS_MAS, MENOS_MENOS, IGUAL, CLRSCR, AMPERSAND, IGUAL_IGUAL, DIFERENTE, MENOR, MAYOR, MENOR_IGUAL, MAYOR_IGUAL, NOT, AND, OR, SCANF, LEER_INT, LEER_CHAR, LEER_FLOAT, PROGRAMA, VOID, MAIN, CONST, IF, ELSE_IF, ELSE, SWITCH, CASE, DEFAULT, BREAK, FOR, WHILE, DO, PRINTF, COMILLAS, PUNTO, COMA, PUNTO_COMA, DOS_PUNTOS, LLAVE_A, LLAVE_C, CORCHETE_A, CORCHETE_C, PARENTESIS_A, PARENTESIS_C, NUMERO, REAL, ID, ERROR.

No Terminal (N):

inicio, main, estructura_principal, txt, cod_vb, cod_py, cod_java, texto_cadena_p, constante, cv_p, dato, texto_cadena, tipo_dato, switch_p, do_while_p, librerias, librerias_p, getch, operaciones_aritmeticas, estruct_op, operaciones_logicas, tipos_comparacion, condicional, and, or, condicional_or, condicional_and, condicional_not, declaracion_variables, declaracion_variables_g, lista_id_valores, lista_id_p, lista_ids, asignacion_var, asignacion, arreglo, dimension, while_n, while_p, if, else, else_if, else_if_p, estructura_if, switch, caso_default, cantidad_casos, caso, while, do_while, for, var, vp, parametros, parametros_p, llamada_fp, instancia_constructor, for_var, for_condicional, for_asignacion, scanf, texto, texto_p, tipo_dato_almacenar, printf, valores, ii, ie, ee, caso_p, i_ins, instancia_p, instancia_pp, estructuras, estructuras_p, operaciones_asignación.

S → inicio

Producciones (P):

inicio ->

estructura_principal

librerias_p

cv_p

main

estructuras_p

CORCHETE_C

main ->

VOID MAIN PARENTESIS_A PARENTESIS_C CORCHETE_A

estructura_principal ->

cod_vb

cod_java

cod_py

PROGRAMA

cod_vb ->

COD_VB

cod_py ->

COD_PY

cod_java ->

COD_JAVA

constante ->

CONST tipo_dato ID IGUAL dato PUNTO_COMA

-> declaracion_variables_g PUNTO_COMA

-> COMENTARIO

-> tipo_dato arreglo PUNTO_COMA

-> error PUNTO_COMA

cv_p → constante cv_p

dato → CARACTER

-> operaciones_aritmeticas

tipo_dato → INT

-> FLOAT

-> CHAR

librerias ->

INCLUDE LIBRERIA_VB

-> INCLUDE LIBRERIA_JAVA

-> INCLUDE LIBRERIA_CLASES

-> INCLUDE LIBRERIA_PYTHON

-> INCLUDE LIBRERIA_C

librerias_p → librerias librerias_p

operaciones_aritmeticas ->

operaciones_aritmeticas MAS operaciones_aritmeticas

-> operaciones_aritmeticas MENOS operaciones_aritmeticas

-> operaciones_aritmeticas POR operaciones_aritmeticas

-> operaciones_aritmeticas DIV operaciones_aritmeticas

-> operaciones_aritmeticas MOD operaciones_aritmeticas

-> MENOS estruct_op

-> estruct_op

estruct_op ->

PARENTESIS_A operaciones_aritmeticas PARENTESIS_C

-> ID

-> NUMERO

-> REAL

-> arreglo

operaciones_logicas ->

tipos_comparacion IGUAL_IGUAL:c tipos_comparacion

-> tipos_comparacion DIFERENTE:c tipos_comparacion

-> tipos_comparacion MENOR:c tipos_comparacion

-> tipos_comparacion MENOR_IGUAL:c tipos_comparacion

-> tipos_comparacion MAYOR:c tipos_comparacion

-> tipos_comparacion MAYOR_IGUAL:c tipos_comparacion

tipos_comparacion ->

operaciones_aritmeticas

-> CARACTER

condicional ->

condicional_or

and → AND

or → OR

condicional_or ->

condicional_or or:c condicional_or

-> condicional_and

condicional_and ->

condicional_and and:c condicional_and

-> condicional_not

condicional_not ->

NOT PARENTESIS_A operaciones_logicas PARENTESIS_C

-> operaciones_logicas

declaracion_variables_g ->

tipo_dato lista_ids

declaracion_variables ->

tipo_dato lista_ids

-> tipo_dato arreglo

lista_id_valores ->

ID IGUAL asignacion

-> ID

lista_ids ->

lista_id_valores lista_id_p

-> lista_id_valores

lista_id_p ->

COMA lista_id_valores

-> lista_id_p COMA lista_id_valores

asignacion_var ->

ID IGUAL asignacion

-> arreglo IGUAL asignacion

asignacion ->

dato

-> llamada_fp

arreglo -> ID dimension

dimension ->

LLAVE_A operaciones_aritmeticas LLAVE_C

-> LLAVE_A operaciones_aritmeticas LLAVE_C dimension

ii ->

IF PARENTESIS_A condicional PARENTESIS_C CORCHETE_A

ie ->

ELSE_IF PARENTESIS_A condicional PARENTESIS_C CORCHETE_A

ee -> ELSE CORCHETE_A

if ->

ii

estructuras_p

CORCHETE_C

else -> ee

else_if ->

ie

estructuras_p

CORCHETE_C

else_if_p ->

else_if else_if_p

estructura_if ->

if else_if_p

-> if else_if_p else

switch_p ->

SWITCH PARENTESIS_A dato PARENTESIS_C CORCHETE_A

switch ->

switch_p

cantidad_casos

caso_default

CORCHETE_C

caso_default ->

DEFAULT DOS_PUNTOS

estructuras_p

BREAK PUNTO_COMA

cantidad_casos ->

caso

-> caso cantidad_casos

caso_p ->

CASE dato DOS_PUNTOS

caso ->

caso_p

estructuras_p

BREAK PUNTO_COMA

-> caso_p

estructuras_p

while_n -> WHILE

while_p ->

while_n PARENTESIS_A condicional PARENTESIS_C CORCHETE_A

while ->

while_p

estructuras_p

CORCHETE_C

do_while_p ->

DO CORCHETE_A

do_while ->

do_while_p

estructuras_p

CORCHETE_C WHILE PARENTESIS_A condicional PARENTESIS_C PUNTO_COMA

for_var ->

FOR PARENTESIS_A var PUNTO_COMA

for_condicional ->

for_var condicional PUNTO_COMA

for_asignacion ->

for_condicional operaciones_asignacion PARENTESIS_C CORCHETE_A

for ->

for_asignacion

estructuras_p

CORCHETE_C

var ->

ID IGUAL operaciones_aritmeticas

-> tipo_dato ID IGUAL dato

operaciones_asignacion ->

ID IGUAL operaciones_aritmeticas

-> ID MAS_MAS

-> ID MENOS_MENOS

parametros ->

operaciones_aritmeticas parametros_p

-> operaciones_aritmeticas

->

parametros_p ->

COMA operaciones_aritmeticas

-> parametros_p COMA operaciones_aritmeticas

llamada_fp ->

PY PUNTO ID PARENTESIS_A parametros PARENTESIS_C

-> VB PUNTO ID PARENTESIS_A parametros PARENTESIS_C

-> JAVA PUNTO ID PUNTO ID PARENTESIS_A parametros PARENTESIS_C

instancia_constructor -> JAVA PUNTO ID instancia_p

instancia_p ->

i_ins

-> i_ins instancia_pp

instancia_pp ->

COMA i_ins

-> COMA i_ins instancia_pp

i_ins ->

ID

-> ID PARENTESIS_A parametros PARENTESIS_C

scanf ->

SCANF PARENTESIS_A COMILLAS texto_cadena tipo_dato_almacenar COMILLAS COMA AMPERSAND ID PARENTESIS_C

-> SCANF PARENTESIS_A COMILLAS tipo_dato_almacenar COMILLAS COMA AMPERSAND ID PARENTESIS_C

texto_p ->

texto_p txt

-> txt

txt ->

texto_cadena tipo_dato_almacenar

-> tipo_dato_almacenar

tipo_dato_almacenar ->

LEER_INT

-> LEER_CHAR

-> LEER_FLOAT

printf ->

PRINTF PARENTESIS_A COMILLAS texto_p COMILLAS COMA valores PARENTESIS_C

-> PRINTF PARENTESIS_A COMILLAS texto_cadena COMILLAS PARENTESIS_C

valores ->

ID

-> ID vp

-> AMPERSAND ID

-> AMPERSAND ID vp

vp ->

COMA ID

-> vp COMA ID

-> COMA AMPERSAND ID

-> vp COMA AMPERSAND ID

estructuras ->

scanf PUNTO_COMA

-> while

-> do_while

-> for

-> switch

-> asignacion_var PUNTO_COMA

-> getch

-> CLRSCR PARENTESIS_A PARENTESIS_C PUNTO_COMA

-> estructura_if

-> declaracion_variables PUNTO_COMA

-> instancia_constructor PUNTO_COMA

-> printf PUNTO_COMA

-> llamada_fp PUNTO_COMA

-> COMENTARIO

getch ->

GETCH PARENTESIS_A PARENTESIS_C PUNTO_COMA

-> tipo_dato:t ID IGUAL GETCH PARENTESIS_A PARENTESIS_C PUNTO_COMA

-> ID IGUAL GETCH PARENTESIS_A PARENTESIS_C PUNTO_COMA

estructuras_p → estructuras estructuras_p

texto_cadena ->

texto_cadena texto

-> texto

texto ->

VB

-> PY

-> JAVA

-> INT

-> FLOAT

-> CHAR

-> GETCH

-> CLRSCR

-> MENOS

-> POR

-> DIV

-> IGUAL

-> IGUAL_IGUAL

-> DIFERENTE

-> MENOR

-> MAYOR

-> MENOR_IGUAL

-> MAYOR_IGUAL

-> NOT

-> AND

-> OR

-> SCANF

-> PROGRAMA

-> VOID

-> MAIN

-> CONST

-> IF

-> ELSE_IF

-> ELSE

-> SWITCH

-> CASE

-> DEFAULT

-> BREAK

-> FOR

-> WHILE

-> DO

-> PRINTF

-> PUNTO

-> PUNTO_COMA

-> DOS_PUNTOS

-> LLAVE_A

-> LLAVE_C

-> CORCHETE_A

-> CORCHETE_C

-> PARENTESIS_A

-> PARENTESIS_C

-> NUMERO

-> REAL

-> ID

-> ERROR

ANALIZADOR SINTÁCTICO JAVA

G = [N, T, P, S **]**

Terminal (T):

CADENA, CARACTER,CONSOLA, THIS, PUNTO, INT, FLOAT, CHAR, MAS, MENOS, POR, DIV, MOD, IGUAL, DOS_PUNTOS, IGUAL_IGUAL, DIFERENTE, MENOR, MAYOR, MENOR_IGUAL, MAYOR_IGUAL, NOT, AND, OR, WHILE, DO, BREAK, FOR, IF, ELSE_IF, ELSE, SWITCH, DEFAULT, RETURN, CASE, PUBLIC, VOID, CLASS, MAS_MAS, MENOS_MENOS, PUNTO_COMA, COMA, CORCHETE_A, CORCHETE_C, PARENTESIS_A, PARENTESIS_C, IINPUT, FINPUT, CINPUT, CONSOLA_LINEA, NUMERO, REAL, ID.

No Terminal (N):

inicio, clases, clases_p, estructura_mf, estructura_mf_p, return, return_dato, metodos_p, tipo_dato, p_parentesis, constructor_p, operaciones_aritmeticas, estruct_op, operaciones_asignacion, operaciones_logicas, tipos_comparacion, condicional, condicional_or, condicional_and, condicional_not, concatenacion, and, or, declaracion_variables_g, mensaje_terminal, funcion_p, declaracion_variables, lista_id_valores, lista_id_p, lista_ids, asignacion_variables, asignacion, solicitud_datos, clases_s, while, while_n, while_p, do_while, do_while_p, asignacion_variables_d, for, var, for_var, for_condicional, for_asignacion, if, else, else_if, else_if_p, estructura_if, ii, ie, ee, switch_p, switch, caso_default, cantidad_casos, caso, caso_p, funcion, parametros, parametros_p, metodos, contructor, estructuras_p, estructuras, variables_locales, estruct

S → inicio

Producciones (P):

inicio ->

clases_p

clases_s ->

PUBLIC CLASS ID CORCHETE_A

clases ->

clases_s

variables_locales

contructor

estructura_mf_p

CORCHETE_C

-> clases_s

variables_locales

estructura_mf_p

CORCHETE_C

clases_p → clases clases_p

constructor_p →

PUBLIC ID PARENTESIS_A parametros PARENTESIS_C CORCHETE_A

contructor ->

constructor_p

estructuras_p:c

CORCHETE_C

variables_locales ->

variables_locales estruct

estruct ->

PUBLIC declaracion_variables_g PUNTO_COMA

-> declaracion_variables_g PUNTO_COMA

-> error PUNTO_COMA

estructura_mf ->

metodos

-> funcion

-> error CORCHETE_C

estructura_mf_p ->

estructura_mf estructura_mf_p

->

return ->

RETURN return_dato

return_dato ->

-> CARACTER

-> operaciones_aritmeticas

tipo_dato ->

INT

-> FLOAT

-> CHAR

operaciones_aritmeticas ->

operaciones_aritmeticas MAS operaciones_aritmeticas

-> operaciones_aritmeticas MENOS operaciones_aritmeticas

-> operaciones_aritmeticas POR operaciones_aritmeticas

-> operaciones_aritmeticas DIV operaciones_aritmeticas

-> operaciones_aritmeticas MOD operaciones_aritmeticas

-> MENOS estruct_op

-> estruct_op

estruct_op ->

PARENTESIS_A operaciones_aritmeticas PARENTESIS_C

-> ID

-> NUMERO

-> REAL

operaciones_asignacion ->

ID IGUAL operaciones_aritmeticas

-> ID MAS_MAS

-> ID MENOS_MENOS

operaciones_logicas ->

tipos_comparacion IGUAL_IGUAL:c tipos_comparacion

-> tipos_comparacion DIFERENTE:c tipos_comparacion

-> tipos_comparacion MENOR:c tipos_comparacion

-> tipos_comparacion MENOR_IGUAL:c tipos_comparacion

-> tipos_comparacion MAYOR:c tipos_comparacion

-> tipos_comparacion MAYOR_IGUAL:c tipos_comparacion

tipos_comparacion ->

operaciones_aritmeticas

-> CARACTER

condicional → condicional_or

and → AND

or → OR

condicional_or ->

condicional_or or condicional_or

-> condicional_and

condicional_and ->

condicional_and and:c condicional_and

-> condicional_not

condicional_not ->

NOT PARENTESIS_A operaciones_logicas PARENTESIS_C

-> operaciones_logicas

concatenacion ->

concatenacion MAS concatenacion

-> CADENA

-> estruct

mensaje_terminal ->

CONSOLA PARENTESIS_A concatenacion PARENTESIS_C

-> CONSOLA_LINEA PARENTESIS_A concatenacion PARENTESIS_C

declaracion_variables_g -> tipo_dato lista_ids

declaracion_variables -> tipo_dato lista_ids

lista_id_valores ->

asignacion_variables_d

-> ID

lista_ids ->

lista_id_valores

-> lista_id_valores lista_id_p

lista_id_p ->

COMA lista_id_valores

-> lista_id_p COMA lista_id_valores

asignacion_variables_d ->

ID IGUAL asignacion

asignacion_variables ->

ID IGUAL asignacion

asignacion ->

return_dato

-> solicitud_datos

solicitud_datos ->

IINPUT p_parentesis

-> FINPUT p_parentesis

-> CINPUT p_parentesis

p_parentesis ->

PARENTESIS_A CADENA PARENTESIS_C

-> PARENTESIS_A PARENTESIS_C

while_n -> WHILE

while_p -> while_n PARENTESIS_A condicional PARENTESIS_C CORCHETE_A

while ->

while_p

estructuras_p

CORCHETE_C

do_while_p ->

DO CORCHETE_A

do_while ->

do_while_p

estructuras_p

CORCHETE_C WHILE PARENTESIS_A condicional PARENTESIS_C PUNTO_COMA

for_var ->

FOR PARENTESIS_A var PUNTO_COMA

for_condicional ->

for_var condicional PUNTO_COMA

for_asignacion ->

for_condicional operaciones_asignacion PARENTESIS_C CORCHETE_A

for ->

for_asignacion

estructuras_p

CORCHETE_C

var ->

ID IGUAL operaciones_aritmeticas

-> tipo_dato ID IGUAL return_dato

ii ->

IF PARENTESIS_A condicional PARENTESIS_C CORCHETE_A

ie ->

ELSE_IF PARENTESIS_A condicional PARENTESIS_C CORCHETE_A

ee ->

ELSE CORCHETE_A

if ->

ii

estructuras_p

CORCHETE_C

else ->

ee

estructuras_p

CORCHETE_C

else_if ->

ie

estructuras_p

CORCHETE_C

else_if_p ->

else_if else_if_p

->

estructura_if ->

if else_if_p

-> if else_if_p else

switch_p ->

SWITCH PARENTESIS_A return_dato PARENTESIS_C CORCHETE_A

switch ->

switch_p

cantidad_casos

caso_default

CORCHETE_C

caso_default ->

DEFAULT DOS_PUNTOS

estructuras_p

BREAK PUNTO_COMA

->

cantidad_casos ->

caso

-> caso

cantidad_casos

caso_p ->

CASE return_dato DOS_PUNTOS

caso ->

caso_p

estructuras_p

BREAK PUNTO_COMA

-> caso_p

estructuras_p

funcion_p ->

PUBLIC tipo_dato ID PARENTESIS_A parametros PARENTESIS_C CORCHETE_A

funcion ->

funcion_p

estructuras_p

CORCHETE_C

parametros ->

tipo_dato ID

-> tipo_dato ID parametros_p

->

parametros_p ->

COMA tipo_dato ID

-> parametros_p COMA tipo_dato ID

metodos_p ->

PUBLIC VOID ID PARENTESIS_A parametros PARENTESIS_C CORCHETE_A

{:parser.mc.addCuartetoProcedimientoJava(a, "JAVA", e, null)

RESULT = new Procedimiento(a, e):}

metodos ->

metodos_p

estructuras_p

CORCHETE_C

estructuras ->

mensaje_terminal PUNTO_COMA

-> solicitud_datos PUNTO_COMA

-> declaracion_variables PUNTO_COMA

-> asignacion_variables PUNTO_COMA

-> while

-> do_while

-> for

-> estructura_if

-> switch

-> return PUNTO_COMA

-> THIS PUNTO ID IGUAL asignacion PUNTO_COMA

-> error PUNTO_COMA

-> error CORCHETE_C

estructuras_p ->

estructuras estructuras_p

->

ANALIZADOR SINTÁCTICO VISUAL BASIC

G = [N, T, P, S **]**

Terminal (T):

CADENA, CARACTER, CONSOLA, CONSOLA_LINEA, INTEGER, DECIMAL, CHAR, MAS, MENOS, POR, DIV, MODULO, IGUAL, COMENTARIO, DISTINTO, MENOR, MAYOR, MENOR_IGUAL, MAYOR_IGUAL, NOT, AND, OR, AMPERSAND, DIM, AS, END, RETURN, WHILE, DO, UNTIL, LOOP, TO, STEP, FOR, NEXT, IF, THEN, ELSEIF, ELSE, SELECT, CASE, PUBLIC, FUNCTION, SUB, PARENTESIS_A, PARENTESIS_C, SALTO, COMA, IINPUT, FINPUT, CINPUT, NUMERO, REAL, ID

No Terminal (N):

inicio,saltos, estructura_fs, estructura_fs_p, return, return_dato, tipo_dato, operaciones_aritmeticas, operaciones_logicas, tipos_comparacion, concatenacion, case_p, mensaje_terminal, do_while_p, declaracion_variables, lista_ids, lista_id_p, asignacion_variables, asignacion, condicional, and, or, solicitud_datos, for_p, while, while_p, while_n, p_parentesis, do_while, until_while, for, var, step, ii, ie, if_linea, if_line, if_linea_p, if_m, else_m, if_else_mult, elseif, else_if_p, switch, switch_p, caso_else, cantidad_casos, caso, casos, estructuras, estructuras_p, estruct, function, condicional_or, condicional_not, condicional_and, sub, ee, sub_s, function_s, parametros, parametros_p

S → inicio

Producciones:

inicio ->

estructura_fs_p

estructura_fs ->

function

-> sub

-> PUBLIC function

-> PUBLIC sub

-> error SUB

-> error END FUNCTION

estructura_fs_p ->

estructura_fs

-> estructura_fs_p saltos estructura_fs

-> estructura_fs_p saltos

->

saltos ->

SALTO

-> saltos SALTO

return ->

RETURN return_dato

return_dato ->

CARACTER

-> operaciones_aritmeticas

tipo_dato ->

INTEGER {:RESULT = Constantes.INT_VAR_VB_PY:}

-> DECIMAL {:RESULT = Constantes.FLOAT_VAR_VB_PY:}

-> CHAR {:RESULT = Constantes.CHAR_VAR_VB_PY:}

operaciones_aritmeticas ->

operaciones_aritmeticas MAS operaciones_aritmeticas

-> operaciones_aritmeticas POR operaciones_aritmeticas

-> operaciones_aritmeticas DIV operaciones_aritmeticas

-> operaciones_aritmeticas MODULO operaciones_aritmeticas

-> MENOS estruct

-> estruct

estruct ->

PARENTESIS_A operaciones_aritmeticas PARENTESIS_C

-> ID

-> NUMERO

-> REAL

operaciones_logicas ->

tipos_comparacion IGUAL:c tipos_comparacion

-> tipos_comparacion DISTINTO:c tipos_comparacion

-> tipos_comparacion MENOR:c tipos_comparacion

-> tipos_comparacion MENOR_IGUAL:c tipos_comparacion

-> tipos_comparacion MAYOR:c tipos_comparacion

-> tipos_comparacion MAYOR_IGUAL:c tipos_comparacion

tipos_comparacion ->

operaciones_aritmeticas

-> CARACTER

condicional ->

condicional_or

and → AND

or → OR

condicional_or ->

condicional_or or:c condicional_or

-> condicional_and

condicional_and ->

condicional_and and:c condicional_and

-> condicional_not

condicional_not ->

NOT PARENTESIS_A operaciones_logicas PARENTESIS_C

-> operaciones_logicas

concatenacion ->

concatenacion MAS concatenacion

-> concatenacion AMPERSAND concatenacion

-> CADENA

-> estruct

mensaje_terminal ->

CONSOLA PARENTESIS_A concatenacion PARENTESIS_C

-> CONSOLA_LINEA PARENTESIS_A concatenacion PARENTESIS_C

declaracion_variables ->

DIM lista_ids AS tipo_dato

-> DIM ID AS tipo_dato

-> DIM ID AS tipo_dato IGUAL asignacion

-> DIM asignacion_variables

lista_ids ->

ID lista_id_p

lista_id_p ->

COMA ID

-> lista_id_p COMA ID

asignacion_variables ->

ID IGUAL asignacion

asignacion ->

return_dato

-> solicitud_datos

solicitud_datos ->

IINPUT p_parentesis

-> FINPUT p_parentesis

-> CINPUT p_parentesis

p_parentesis ->

PARENTESIS_A CADENA PARENTESIS_C

-> PARENTESIS_A PARENTESIS_C

while_n ->

WHILE

while_p ->

while_n condicional saltos

while ->

while_p

estructuras_p

END WHILE

do_while_p ->

DO saltos

do_while ->

do_while_p

estructuras_p

LOOP until_while condicional

until_while ->

UNTIL

-> WHILE

for_p ->

FOR var TO operaciones_aritmeticas step saltos

for ->

for_p

estructuras_p

NEXT ID

step ->

STEP operaciones_aritmeticas

->

var ->

ID IGUAL operaciones_aritmeticas

-> ID AS tipo_dato IGUAL operaciones_aritmeticas

if_linea ->

if_line if_linea_p

if_line ->

ii THEN estructuras

if_linea_p ->

ee estructuras

->

if_m ->

ii THEN saltos

estructuras_p

-> ii saltos

estructuras_p

else_m ->

ee saltos estructuras_p

ee → ELSE

ii ->

IF condicional

if_else_mult ->

if_m

else_if_p

else_m

END IF

-> if_m

else_if_p

END IF

elseif ->

ie THEN saltos

estructuras_p

-> ie saltos

estructuras_p

ie ->

ELSEIF condicional

else_if_p ->

elseif else_if_p

->

switch_p ->

SELECT CASE operaciones_aritmeticas saltos

switch ->

switch_p

cantidad_casos

END SELECT

-> switch_p

cantidad_casos

caso_else

END SELECT

caso_else ->

CASE ELSE saltos

estructuras_p

cantidad_casos ->

cantidad_casos caso

-> caso

case_p ->

CASE casos saltos

caso ->

case_p

estructuras_p

casos ->

operaciones_aritmeticas

function_s ->

FUNCTION ID PARENTESIS_A parametros PARENTESIS_C AS tipo_dato saltos

function ->

function_s

estructuras_p

return saltos

END FUNCTION

sub_s ->

SUB ID PARENTESIS_A parametros PARENTESIS_C saltos

sub ->

sub_s

estructuras_p

END SUB

parametros ->

ID AS tipo_dato

-> ID AS tipo_dato parametros_p

parametros_p ->

COMA ID AS tipo_dato

-> parametros_p COMA ID AS tipo_dato

estructuras ->

mensaje_terminal

-> solicitud_datos

-> declaracion_variables

-> asignacion_variables

-> while

-> do_while

-> for

-> if_linea

-> if_else_mult

-> switch

-> error saltos

-> error END IF

-> error END WHILE

estructuras_p ->

estructuras saltos estructuras_p

-> estructuras saltos

ANALIZADOR SINTÁCTICO PYTHON

G = [N, T, P, S **]**

Terminal (T):

CADENA, CARACTER, CONSOLA, CONSOLA_LINEA, INTEGER, DECIMAL, CHAR, MAS, MENOS, POR, DIV, MODULO, IGUAL, COMENTARIO, DISTINTO, MENOR, MAYOR, MENOR_IGUAL, MAYOR_IGUAL, NOT, AND, OR, AMPERSAND, DIM, AS, END, RETURN, WHILE, DO, UNTIL, LOOP, TO, STEP, FOR, NEXT, IF, THEN, ELSEIF, ELSE, SELECT, CASE, PUBLIC, FUNCTION, SUB, PARENTESIS_A, PARENTESIS_C, SALTO, COMA, IINPUT, FINPUT, CINPUT, NUMERO, REAL, ID

No Terminal (N):

inicio,saltos, estructura_fs, estructura_fs_p, return, return_dato, tipo_dato, operaciones_aritmeticas, operaciones_logicas, tipos_comparacion, concatenacion, case_p, mensaje_terminal, do_while_p, declaracion_variables, lista_ids, lista_id_p, asignacion_variables, asignacion, condicional, and, or, solicitud_datos, for_p, while, while_p, while_n, p_parentesis, do_while, until_while, for, var, step, ii, ie, if_linea, if_line, if_linea_p, if_m, else_m, if_else_mult, elseif, else_if_p, switch, switch_p, caso_else, cantidad_casos, caso, casos, estructuras, estructuras_p, estruct, function, condicional_or, condicional_not, condicional_and, sub, ee, sub_s, function_s, parametros, parametros_p

S → inicio

Producciones:

inicio ->

SALTO estructuras_def

-> estructuras_def

estructuras_def ->

def estructuras_def

->

return ->

RETURN return_dato

return_dato ->

CARACTER

-> operaciones_aritmeticas

operaciones_aritmeticas ->

operaciones_aritmeticas MAS operaciones_aritmeticas

-> operaciones_aritmeticas MENOS operaciones_aritmeticas

-> operaciones_aritmeticas POR operaciones_aritmeticas

-> operaciones_aritmeticas DIV operaciones_aritmeticas

-> operaciones_aritmeticas MOD operaciones_aritmeticas

-> MENOS estruct

-> estruct

estruct ->

PARENTESIS_A operaciones_aritmeticas PARENTESIS_C

-> ID

-> NUMERO

-> REAL

operaciones_logicas ->

tipos_comparacion IGUAL_IGUAL:c tipos_comparacion

-> tipos_comparacion DISTINTO:c tipos_comparacion

-> tipos_comparacion DIFERENTE:c tipos_comparacion

-> tipos_comparacion MENOR:c tipos_comparacion

-> tipos_comparacion MENOR_IGUAL:c tipos_comparacion

-> tipos_comparacion MAYOR:c tipos_comparacion

-> tipos_comparacion MAYOR_IGUAL:c tipos_comparacion

tipos_comparacion ->

operaciones_aritmeticas

-> CARACTER

condicional → condicional_or

and → AND

or → OR

condicional_or ->

condicional_or or:c condicional_or

-> condicional_and

condicional_and ->

condicional_and and:c condicional_and

-> condicional_not

condicional_not ->

NOT PARENTESIS_A operaciones_logicas PARENTESIS_C

-> operaciones_logicas

concatenacion ->

concatenacion MAS concatenacion

-> concatenacion COMA concatenacion

-> CADENA

-> estruct

mensaje_terminal ->

PRINT PARENTESIS_A concatenacion PARENTESIS_C

asignacion_variables ->

ID IGUAL asignacion

asignacion ->

return_dato

-> solicitud_datos

soliciud_datos ->

IINPUT solicitud_p

-> FINPUT solicitud_p

-> CINPUT solicitud_p

solicitud_p ->

PARENTESIS_A CADENA PARENTESIS_C

-> PARENTESIS_A PARENTESIS_C

while_n ->

WHILE

while_p ->

while_n:w condicional DOS_PUNTOS INDENT

while ->

while_p

estructuras_p

SALTO

-> while_p

estructuras_p

DEDENT

var ->

ID

for_p ->

FOR var IN RANGE PARENTESIS_A rango PARENTESIS_C DOS_PUNTOS INDENT

for ->

for_p

estructuras_p

SALTO

-> for_p

estructuras_p

DEDENT

rango ->

return_dato

-> return_dato COMA return_dato

-> return_dato COMA return_dato COMA return_dato

if ->

ii DOS_PUNTOS INDENT

estructuras_p

DEDENT

-> ii DOS_PUNTOS INDENT

estructuras_p

SALTO

ii ->

IF condicional

ie ->

ELIF condicional

ee ->

ELSE

else_if ->

ie DOS_PUNTOS INDENT

estructuras_p

DEDENT

-> ie DOS_PUNTOS INDENT

estructuras_p

SALTO

else ->

ee DOS_PUNTOS INDENT

estructuras_p

DEDENT

-> ee DOS_PUNTOS INDENT

estructuras

SALTO

else_if_p ->

else_if else_if_p

->

estructura_if ->

if else_if_p

-> if else_if_p else

def_p ->

DEF ID PARENTESIS_A parametros PARENTESIS_C DOS_PUNTOS INDENT

def ->

def_p

estructuras_p

DEDENT

-> def_p

estructuras_p

space ->

SALTO

->

parametros ->

ID

-> ID parametros_p

->

parametros_p ->

COMA ID

-> parametros_p COMA ID

estructuras ->

mensaje_terminal space

-> asignacion_variables space

-> while

-> for

-> estructura_if

-> return space

-> error SALTO

-> error DEDENT

estructuras_p ->

estructuras estructuras_p

-> estructuras

MÉTODOS Y CLASES

Paquetes

proyecto.ctdoa
proyecto.ctdoa.backed.analizador.c
proyecto.ctdoa.backed.analizador.java
proyecto.ctdoa.backed.analizador.python
proyecto.ctdoa.backed.analizador.visual_basic
proyecto.ctdoa.backed.manejadores
proyecto.ctdoa.backed.objetos
proyectoanalizador.gui

ProyectoAnalizador

Clase Main

Clase Generador de archivos.

Generador():
Genera los archivos del analizador

Clase ManejadorArchivosBinarios

boolean crearArchivo(T tipoObjeto, String tipoArchivo, String identificadorUnico, String extensionArchivo)
Crea un archivo binario de un tipo de objeto.

T leerArchivo(String pathInicial, String pathNombreArchivo, String tipoDeArchivoPath)
Busca un archivo binario

List<T> leerListaArchivos(String extensionArchivo)
Carga a memoria los archivos con extension len para cargarlos en el repositorio.

Clase ManejadorEntrada

String entradaTexto(String entrada)
Lee el archivo de entrada de la carga de un lenguaje, para posteriormente realizar un analisis.

Frame.java

Constructor de la ventana
public Principal()

private void menuGuardarActionPerformed(java.awt.event.ActionEvent evt)
Método para guardar los cambios realizados por cada uno de los archivos en el editor de texto.

private void realizarAnalisisLCT(String comprobacion)
Método para realizar el análisis sintáctico y léxico de un archivo.

private void menuAcercaDeActionPerformed(java.awt.event.ActionEvent evt)
Método que muestra la información del desarrollador.

private void menuGuardarActionPerformed(java.awt.event.ActionEvent evt)
Método para exportar los lienzo al formato indicado.

Clase ManejadorSintactico

Constructor de la Clase
private ManejadorSintactico() {}

Instancia de la clase
public static ManejadorSintactico getInstancia()

public void clear()
Método que realiza la limpieza dela lista de símbolos y la tabla de símbolos, para poder realizar un nuevo análisis.

public void errorSintax(int left, int right, String value, String mensaje)
Método que agrega al área de texto errores la descripción de un error sintactico.

public void errorSemantico(int left, int right, String valor, String mensaje)
Método que agrega al área de texto errores la descripción de un error semantico.

public Object addTabla(Object object, int left, int right, String tipo)

Agrega un nuevo símbolo a la tabla de símbolos

public Object declararVariablesGlobales(Simbolo s, int l, int r)

Metodo que declara una variable sin el tipo de id, y comprueba que el valor asignado sea el correspondiente

@param s valor que se le asignara a la variable

@param l left del simbolo

@param r right del simbolo

@return La declaracion de la variable

public Simbolo declaraUnaVariableParametro(Tipo tipo, String id, int l, int r)

Metodo que declara una variable, y comprueba que el valor asignado sea el correspondiente

@param tipo tipo de dato de la variable

@param id nombre de la variable

@param l left del simbolo

@param r right del simbolo

@return La declaracion de la variable

public Object declararVariables(Object a, int l, int r)

Metodo que declara varias variables

@param a Objeto que contiene las declaraciones

@param l left del simbolo

@param r right del simbolo

@return La declaracion de la variable

public Simbolo metodoBuscarID(String id, int l, int r)

Funcion para buscar un id en la tabla de simbolos

@param id nombre de la variable

@param l left del simbolo

@param r right del simbolo

@return Retorna el simmbolo

public Simbolo modificarVariables(Object s, int l, int r) throws CloneNotSupportedException

Funcion para modiicar el valor de una variable (Hacerla negativa)

@param s Simbolo

@param l left del simbolo

@param r right del simbolo

@return Retorna el valor modificado

public boolean comprobarCompatibilidadTipos(Object op1, Object op2, int l1, int r1, int l2, int r2)

Funcion que comprueba si dos operandos son del mismo tipo

@param op1 OOperando 1

@param op2 Operando 2

@param l1 left del simbolo

@param r2 right del simbolo

@param r1 right del simbolo

@param l2 left del simbolo

@return Retorna true si pueden ser operados, de lo contrario false

@param 2: right del simbolo

public Simbolo realizarOperaciones(Object op1, Object op2, int l1, int r1, int l2, int r2, int tipoOperacion)

Funcion que realiza operaciones

@param op1 OOperando 1

@param op2 Operando 2

@param l1 left del simbolo

@param r2 right del simbolo

@param r1 right del simbolo

@param l2 left del simbolo

@param tipoOperacion tipo de operacion que se realizara 1. suma, 2. resta, 3. multiplicacion, 4. division, 5. mod

public Simbolo asignacionVariables(Object s, int l, int r)

Metodo que asigna una vaiable

@param s valor que se le asignara a la variable

@param l left del simbolo

@param r right del simbolo

public void asignacionVariablesThis(String id, Object asignacion, int l, int r)

Metodo que asigna una vaiable

@param id id de la variable

@param asignacion valor asignacion

@param l left del simbolo

@param r right del simbolo

public Simbolo comprobarOperacionesLogicas(Object op1, Object op2, int tipoOperacion, int l, int r)

Metodo que comprueba si dos operandos pueden ser validados

1. Igual

2. Distinto

3. Menor

4. Menor Igual

5. Mayor

6. Mayor Igual

@param op1 Operando 1

@param op2 Operando 2

@param l left del simbolo

@param tipoOperacion tipo de la operacion condicional

@param r right del simbolo

@return Retorna el valor false o true si la comapracion de los operandos es coherente, de lo contrario null

public Simbolo contruirEstructuraCiloDW(Object statement, Object condicional, int tipo)

Metodo que contruye una estructura

1.Estructura WHILE

2.Estructura DO WHILE

3. Estructura FOR

@param statement Intrucciones

@param condicional

@param tipo

@return Estructura de un ciclo while o do while

public Simbolo construirCondicionalIf(Object simboloIf, Object listaElseIf, Object simboloElse)

Funcion que construye una condicional Validando cada una de las instrucciones

@param simboloIf Estructura condicional If

@param listaElseIf Estructuras condicional Else If

@param simboloElse Estructura condicional Else

Objetos:

- Case
- CicloDoWhile
- CicloFor
- CicloWhile
- Condicional
- CondicionalIf
- CondicionalSwitch
- Constantes
- Cuarteto
- Estructuras
- For
- Funcion
- Metodo
- NumeroLinea
- Pestania
- Print
- Procedimiento
- Simbolo
- TablaSimbolos
- Tipo
- While

NOTA: Cada uno con sus respectivo constructor, getter's y setter's.

Analizador Léxico y Sintáctico

- LexicoC.flex
- SintacticoC.cup
- SintacticoC.java
- sym.java
- LexicoC.java

- LexicoJava.flex
- SintacticoJava.cup
- SintacticoJava.java
- sym.java
- LexicoJava.java

- LexicoPython.flex
- SintacticoPython.cup
- SintacticoPython.java
- sym.java
- LexicoPython.java

- LexicoVisualBasic.flex
- SintacticoVisualBasic.cup
- SintacticoVisualBasic.java
- sym.java
- LexicoVisualBasic.java

Librerías:

- Jflex.jar
- java-cup-11b-runtime.jar
- java-cup-11b.ja

REQUISITOS MINIMOS PARA PROGRAMAR EN JAVA:
REQUERIMIENTOS DE HARDWARE PARA PROGRAMAR EN JAVA

Para las aplicaciones generadas se debe tener un mínimo de 32MB de RAM, se recomienda que se tengan 48MB o más. Para compilar utilizando el SDK de Microsoft es un mínimo de 32MB de RAM, y para JDK 48MB mínimo. El procesador en principio no es tan crítico como la memoria RAM, pero se recomienda utilizar al menos un Pentium de 133 para compilar/ejecutar las aplicaciones.

Requerimientos de SOFTWARE
Para trabajar con el Generador Java es necesario cierto software adicional para la compilación y ejecución de las aplicaciones generadas.

Java Development Kit - Compilador y Máquina Virtual
Para la compilación y ejecución de los programas es necesario tener el JDK. En cuanto a las versiones se recomienda en general utilizar las últimas liberadas de cada uno de ellos. En caso de que alguna versión sea menos estable que su versión anterior, se avisara.

Se requieren las versiones siguientes de .NET Framework:
SQL Server 2008 en Windows Server 2003 64 bits IA64: .NET Framework 2.0 SP2SQL Server Express: .NET Framework 2.0 SP2. Todas las demás ediciones de SQL Server 2008: .NET Framework 3.5 SP1, etc.

I**DE NetBeans 8.2 requisitos mínimos:**

Microsoft Windows Vista SP1 / Windows 7 Professional:
Procesador: Varel Pentium III 800MHz o equivalente
Memoria: 512 MB
Espacio en disco: 750 MB de espacio libre en disco

Ubuntu 9.10:
Procesador: Varel Pentium III 800MHz o
equivalente Memoria: 512 MB
Espacio en disco: 650 MB de espacio libre en disco

Macvarosh OS X 10.7
Varel: Procesador: Varel
Dual-Core Memoria: 2 GB
Espacio en disco: 650 MB de espacio libre en disco

Microsoft Windows 7 Professional / Windows 8 / Windows 8.2 / Windows 10:
Procesador: Varel Core i5 o
equivalente Memoria: 2 GB 32 bits,
4GB 64 bits
Espacio en disco: 1,5 GB de espacio libre en disco

Ubuntu 15.04 y Superior:
Procesador: Varel Core i5 o
equivalente Memoria: 2 GB 32 bits, 4
GB 64 bits
Espacio en disco: 1,5 GB de espacio libre en disco

OS X 10.10 Varel:
Procesador: Varel Dual-
Core Memoria: 4 GB
Espacio en disco: 1,5 GB de espacio libre en disco