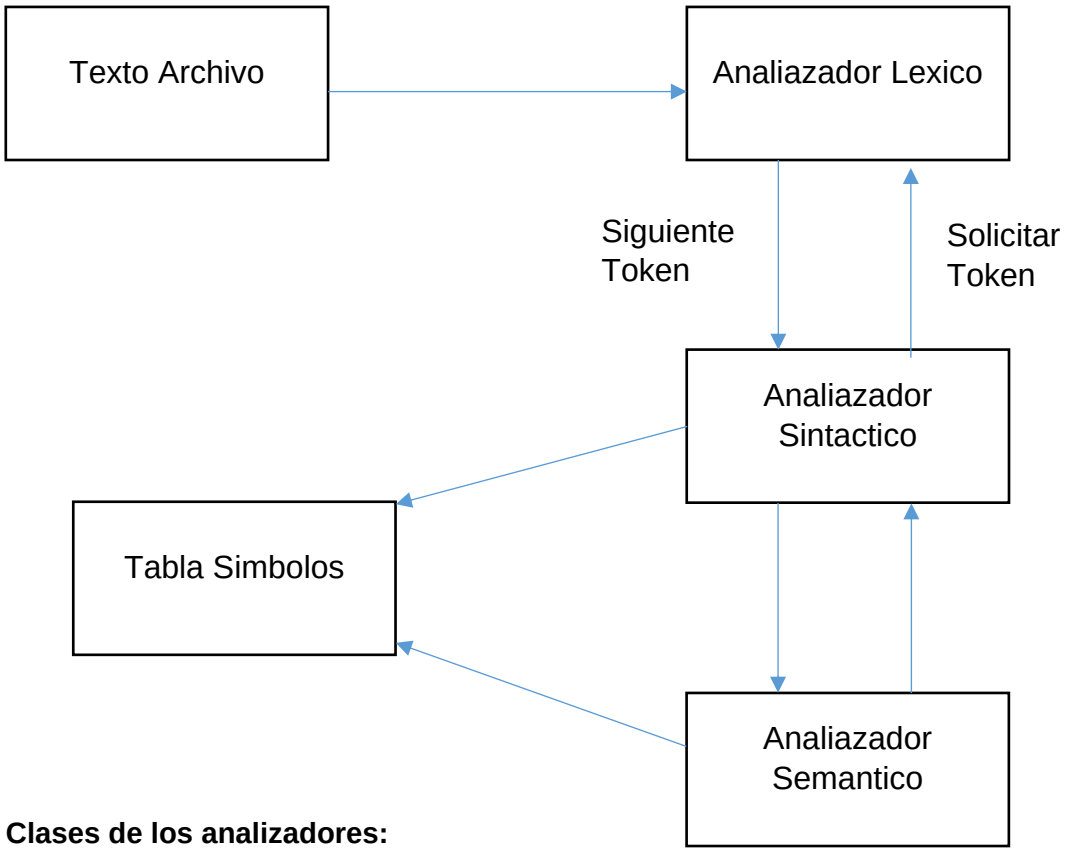
MANUAL TECNICO
(FASE 1)

DESCRIPCION DEL PROGRAMA

El programa cuenta con un analizador léxico y sintáctico. Estos se usan para determinar los objetos que se crearan para integracion del componente del programa. Cuenta con cuatro analizadores de estos tipos, una para el VB, uno paja java, uno para python y el ultimo para el programa principal.

Los archivos de lectura para los analizadores son de extensión .lmlg la gramática del archivo tiene una estructura según sea el lenguaje. De esta manera se implementa el sistema de lectura de los analizadores. Ya que el programa está hecho el Lenguaje Java, se utilizan las herramientas de jflex y cup para la creación de los analizadores.



Clases de los analizadores:

AnalizadorLexicoVB

AnalizadorSintacticoVB

AnalizadorLexicoJAVA

AnalizadorSintacticoJAVA

AnalizadorLexicoPY

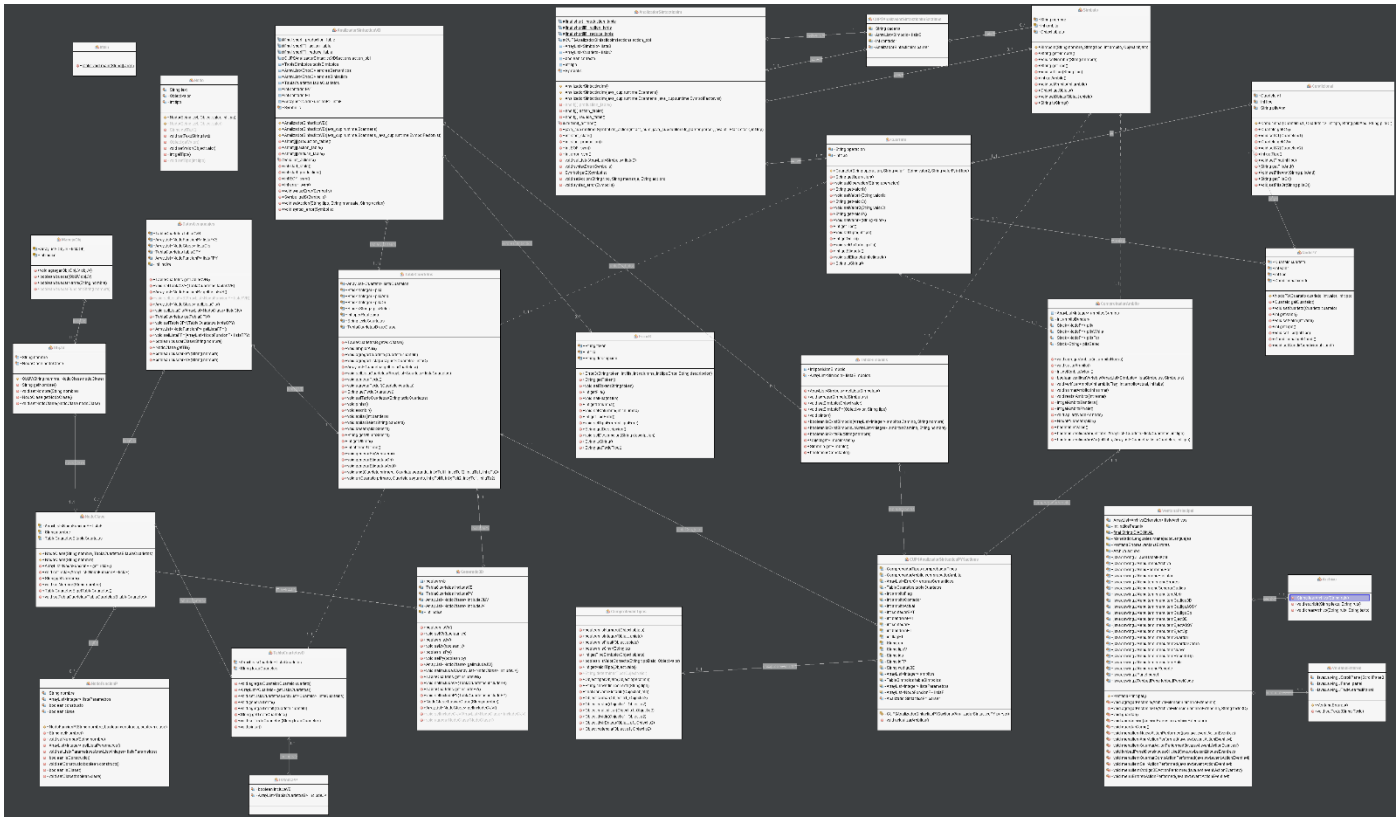
AnalizadorSintacticoPY

AnalizadorLexicoCPP

AnalizadorSintacticoCPP

Implementación del IDE

El ide cuenta con una ventana, en esta se agrega el código fuente, para posteriormente este sea analizada, y en caso de que existan errores abrirá una ventana de errores.



El IDE fue hecho con el programa NetBeans IDE.

Se utilizo el lenguaje de Java en su versión 8.

En el código fuente o el proyecto se encuentran las carpetas de las src utilizadas para el programa.

Se utilizo la plataforma de Windows y Linux como prueba del funcionamiento del sistema.

GRAMATICAS

GRAMATICAS

GRAMATICA VB

Léxico

Reservada	Nombre
&	AND_RESRV
And	AND
Or	OR
Not	NOT
+	MAS
-	MENOS
*	POR
/	DIV
\	DIV_ENTERO
<	MENOR_QUE
>	MAYOR_QUE
<=, =<	MENOR_IGUAL
>=, =>	MAYOR_IGUAL
=	IGUAL
<>	DIFERENTE
Is	IS
IsNot	ISNOT
Like	LIKE
MsgBox(	MSG
MessageBox(	MSG
Console.WriteLine(	CONSOLE_WRT
intinput	INTINPUT
floatinput	FLOATINPUT
charinput	CHARINPUT
For	FOR
To	TO
Step	STEP
Next	NEXT
While	WHILE

End	END
Do	DO
Until	UNTIL
Continue	CONTINUE
Exit	EXIT
Loop	LOOP
If	IF
Then	THEN
Else	ELSE
ElseIf	ELSE_IF
end	END
Select	SELECT
Case	CASE
Public	PUBLIC
Dim	DIM
As	AS
Function	FUNCTION
Return	RETURN
Module	MODULE
Sub	SUB
[	CORCHETE_A
]	CORCHETE_C
{	LLAVES_A
}	LLAVES_C
(	PARENTESIS_A
)	PARENTESIS_C
,	COMA

Sintáctico

INICIO ::= SEPARADOR_VB codigo SEPARADOR_PROGRAMA

|WHILE;

codigo ::= struc_funciones codigo

|struc_procedimientos codigo

|COMENTARIO codigo

|SALTO codigo

|;

bloque_declararion_var ::= DIM IDENTIFICADOR tipo_declaracion
|DIM IDENTIFICADOR IGUAL s4
|IDENTIFICADOR IGUAL s4 ;

s4 ::= operacion
|inputs_dato;

tipo_declaracion ::= var_identificadores
|AS tipo_datos IGUAL s4;

var_identificadores ::= COMA IDENTIFICADOR var_identificadores
|AS tipo_datos;

tipo_datos ::= INTEGER
|DECIMAL_R
|CHART;

valor ::= NUMERO
|DECIMAL
|VALOR;

struc_funciones ::= FUNCTION IDENTIFICADOR s5 RETURN operacion SALTO END FUNCTION SALTO;

struc_modulos ::= PUBLIC MODULE IDENTIFICADOR SALTO s7 END MODULE;

struc_procedimientos ::= PUBLIC SUB IDENTIFICADOR s6 END SUB SALTO;

s5 ::= PARENTESIS_A parametros PARENTESIS_C AS tipo_datos SALTO sentencias

|PARENTESIS_A PARENTESIS_C AS tipo_datos SALTO sentencias ;

s6 ::= PARENTESIS_A parametros PARENTESIS_C SALTO sentencias

|PARENTESIS_A PARENTESIS_C SALTO sentencias;

s7 ::= bloque_declararion_var s7

|struc_funciones s7

|struc_procedimientos s7

|COMENTARIO s7

|SALTO

|;

s9 ::= SALTO s9

|;

parametros ::= IDENTIFICADOR AS tipo_datos COMA parametros

|IDENTIFICADOR AS tipo_datos;

struc_ciclos ::= struc_for

|struc_while;

struc_for ::= FOR IDENTIFICADOR IGUAL operacion TO operacion opcion_step SALTO s8 NEXT IDENTIFICADOR SALTO;

opcion_step ::= STEP operacion

|;

```
struc_while ::=  WHILE condicionales SALTO s8 WEND SALTO
            |DO SALTO s8 LOOP WHILE condicionales SALTO
            |DO SALTO s8 LOOP UNTIL condicionales SALTO
            |DO WHILE condicionales SALTO s8 LOOP
            |DO UNTIL condicionales SALTO s8 LOOP;

struc_if ::= IF condicionales THEN SALTO s8 struc_else_if END IF SALTO;

struc_else_if ::=    ELSE SALTO s8 struc_else_if
                |ELSE_IF condicionales THEN SALTO s8 struc_else_if
                |;

struc_select ::= SELECT IDENTIFICADOR SALTO casos END SELECT SALTO
            |SELECT CASE IDENTIFICADOR SALTO casos END SELECT SALTO;

casos ::=    CASE operacion s2 SALTO s8 s1
          |CASE operacion TO operacion SALTO s8 s1
          |CASE ELSE SALTO s8;

s1 ::=   casos
       |;

s2 ::=   COMA operacion s2
       |;

condicionales ::=    operacion MENOR_QUE operacion condicion_xtra
                |operacion MAYOR_QUE operacion condicion_xtra
```

|operacion MENOR_IGUAL operacion condicion_xtra

|operacion MAYOR_IGUAL operacion condicion_xtra

|operacion IGUAL operacion condicion_xtra

|operacion DIFERENTE operacion condicion_xtra;

condicionales_1 ::= operacion MENOR_QUE operacion

|operacion MAYOR_QUE operacion

|operacion MENOR_IGUAL operacion

|operacion MAYOR_IGUAL operacion

|operacion IGUAL operacion

|operacion DIFERENTE operacion;

condicion_xtra ::= AND condicionales_1 condicion_xtra

|OR condicionales_1 condicion_xtra

|NOT condicionales_1 condicion_xtra

|;

operacion ::= operacion MAS operacion

|operacion AND_RESRV operacion

|operacion MENOS operacion

|operacion POR operacion

|operacion DIV operacion

|operacion DIV_ENTERO operacion

|operacion POT operacion

|PARENTESIS_A operacion PARENTESIS_C

|CORCHETE_A operacion CORCHETE_C

|LLAVES_A operacion LLAVES_C

|IDENTIFICADOR

```
            |NUMERO
            |DECIMAL
            |VALOR;


sentencias_1 ::=   mensajes SALTO
            |bloque_declararion_var SALTO
            |struc_ciclos
            |struc_if
            |struc_select
            |COMENTARIO
            |SALTO;

sentencias ::=   sentencias mensajes
            |sentencias bloque_declararion_var SALTO
            |sentencias struc_ciclos
            |sentencias struc_if
            |sentencias struc_select
            |sentencias COMENTARIO
            |sentencias SALTO
            |;

mensajes ::= MSG operacion PARENTESIS_C SALTO
          |CONSOLE_WRT operacion PARENTESIS_C SALTO
          |PRINT operacion SALTO;

inputs_dato ::=  INTINPUT PARENTESIS_A s3 PARENTESIS_C
            |FLOATINPUT PARENTESIS_A s3 PARENTESIS_C
```

```
            |CHARINPUT PARENTESIS_A s3 PARENTESIS_C;

s3 ::=   VALOR
       |;

s8 ::=   s8 sentencias_1
       |;
```

GRAMATICA JAVA

Léxico

Reservada	Nombre
&	AND_RESRV
And	AND
Or	OR
Not	NOT
+	MAS
-	MENOS
*	POR
/	DIV
\	DIV_ENTERO
<	MENOR_QUE
>	MAYOR_QUE
<=, =<	MENOR_IGUAL
>=, =>	MAYOR_IGUAL
=	IGUAL
<>	DIFERENTE
Is	IS
IsNot	ISNOT
Like	LIKE
MsgBox(	MSG
MessageBox(	MSG
Console.WriteLine(	CONSOLE_WRT
intinput	INTINPUT
floatinput	FLOATINPUT
charinput	CHARINPUT
For	FOR
To	TO
Step	STEP
Next	NEXT
While	WHILE
End	END
Do	DO
Until	UNTIL
Continue	CONTINUE
Exit	EXIT
Loop	LOOP

If	IF
Then	THEN
Else	ELSE
ElseIf	ELSE_IF
end	END
Select	SELECT
Case	CASE
Public	PUBLIC
Dim	DIM
As	AS
Function	FUNCTION
Return	RETURN
Module	MODULE
Sub	SUB
[	CORCHETE_A
]	CORCHETE_C
{	LLAVES_A
}	LLAVES_C
(	PARENTESIS_A
)	PARENTESIS_C
,	COMA

Sintáctico

INICIO ::= SEPARADOR_JAVA codigo_inicial SEPARADOR_PROGRAMA ;

codigo_inicial ::= struc_clase;

codigo ::= struc_vars codigo
|struc_function codigo
|mensaje codigo
|comentarios codigo
|;

codigo_funcion ::= struc_vars codigo_funcion

|struc_ciclos codigo_funcion

|struc_if codigo_funcion

|struc_switch codigo_funcion

|mensaje codigo_funcion

|inputs codigo_funcion

|comentarios codigo_funcion

|;

comentarios ::= COMENTARIO_SIMPLE

|COMENTARIO_VARIOS;

struc_clase ::= PUBLIC CLASS IDENTIFICADOR LLAVES_A codigo LLAVES_C struc_clase

|;

struc_vars ::= INT struc_asig_vars_n PUNTO_COMA

|FLOAT struc_asig_vars_n PUNTO_COMA

|CHAR struc_asig_vars_c PUNTO_COMA

|IDENTIFICADOR IGUAL valor PUNTO_COMA;

struc_asig_vars_n ::= IDENTIFICADOR IGUAL valor_n COMA struc_asig_vars_n

|IDENTIFICADOR COMA struc_asig_vars_n

|IDENTIFICADOR IGUAL valor_n

|IDENTIFICADOR;

struc_asig_vars_c ::= IDENTIFICADOR IGUAL valor_c COMA struc_asig_vars_c

|IDENTIFICADOR COMA struc_asig_vars_c

|IDENTIFICADOR IGUAL valor_c

|IDENTIFICADOR;

struc_ciclos ::= struct_for

|struct_do_while

|struct_while;

struc_ciclos_return ::= struct_for_return

|struct_do_while_return

|struct_while_return;

struct_for ::= FOR PARENTESIS_A struc_indices PARENTESIS_C LLAVES_A codigo_funcion LLAVES_C ;

struct_for_return ::= FOR PARENTESIS_A struc_indices PARENTESIS_C LLAVES_A codigo_funcion_return LLAVES_C;

struc_indices ::= IDENTIFICADOR IGUAL valor_n PUNTO_COMA IDENTIFICADOR valor_comprobacion PUNTO_COMA IDENTIFICADOR struc_a

|INT IDENTIFICADOR IGUAL valor_n PUNTO_COMA IDENTIFICADOR valor_comprobacion PUNTO_COMA IDENTIFICADOR struc_a;

valor_c ::= valor_c MAS valor_c

|valor_c MENOS valor_c

|valor_c POR valor_c

|valor_c DIV valor_c

|PARENTESIS_A valor_c PARENTESIS_C

|IDENTIFICADOR

|VALOR

|NUMERO

|DECIMAL;

valor_n ::= valor_n MAS valor_n

|valor_n MENOS valor_n

|valor_n POR valor_n

|valor_n DIV valor_n

|PARENTESIS_A valor_n:a PARENTESIS_C

|IDENTIFICADOR

|NUMERO

|DECIMAL;

valor_comprobacion ::= MENOR_QUE valor_n

|MAYOR_QUE valor_n

|MENOR_IGUAL valor_n

|MAYOR_IGUAL valor_n

|IGUAL_IGUAL valor_n

|DIFERENTE valor_n;

struc_a ::= IGUAL valor_n

|MAS_MAS

|MENOS_MENOS;

struct_while ::= WHILE PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion LLAVES_C;

struct_while_return ::= WHILE PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion_return LLAVES_C ;

struct_do_while ::= DO LLAVES_A codigo_funcion LLAVES_C WHILE PARENTESIS_A struc_condicional PARENTESIS_C PUNTO_COMA;

struct_do_while_return ::= DO LLAVES_A codigo_funcion_return LLAVES_C WHILE PARENTESIS_A struc_condicional PARENTESIS_C PUNTO_COMA;

valor_condicional ::= valor MENOR_QUE valor
|valor MAYOR_QUE valor
|valor MENOR_IGUAL valor
|valor MAYOR_IGUAL valor
|valor IGUAL_IGUAL valor
|valor DIFERENTE valor;

valor ::= valor MAS valor
|valor MENOS valor
|valor POR valor
|valor DIV valor
|PARENTESIS_A valor PARENTESIS_C
|IDENTIFICADOR
|VALOR
|NUMERO
|DECIMAL;

struc_condicional ::= struc_logico;

struc_logico ::= struc_logico AND struc_logico
|struc_logico OR struc_logico
|PARENTESIS_A struc_logico PARENTESIS_C

|NOT PARENTESIS_A struc_logico PARENTESIS_C

|valor_condicional;

struc_logico_not ::= NOT

|;

struc_if ::= IF PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion LLAVES_C struc_else ;

struc_if_return ::= IF PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion_return LLAVES_C struc_else_return ;

struc_else ::= ELSE IF PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion LLAVES_C struc_else

|ELSE LLAVES_A codigo_funcion LLAVES_C

| ;

struc_else_return ::= ELSE IF PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion_return LLAVES_C struc_else_return

|ELSE LLAVES_A codigo_funcion_return LLAVES_C

| ;

struc_switch ::= SWITCH PARENTESIS_A IDENTIFICADOR PARENTESIS_C LLAVES_A struc_case LLAVES_C;

struc_case ::= CASE valor_case DOS_PUNTOS codigo_funcion BREAK PUNTO_COMA struc_case

|DEFAULT DOS_PUNTOS codigo_funcion

|;

valor_case ::= VALOR

|NUMERO

|DECIMAL;

struc_function ::= PUBLIC VOID IDENTIFICADOR PARENTESIS_A struc_params PARENTESIS_C LLAVES_A codigo_funcion LLAVES_C

|PUBLIC vars IDENTIFICADOR PARENTESIS_A struc_params PARENTESIS_C LLAVES_A codigo_funcion_return LLAVES_C;

struc_params ::= struc_vars_params params

|;

params ::= COMA struc_vars_params params

|;

struc_vars_params ::= INT IDENTIFICADOR

|FLOAT IDENTIFICADOR

|CHAR IDENTIFICADOR;

vars ::= INT

|FLOAT

|CHAR;

mensaje ::= SOUT valor_m PARENTESIS_C PUNTO_COMA;

valor_m ::= valor_m MAS valor_m

|valor_m MENOS valor_m

|valor_m POR valor_m

|valor_m DIV valor_m

```
        |PARENTESIS_A valor_m PARENTESIS_C
        |IDENTIFICADOR
        |TEXTO
        |VALOR
        |NUMERO
        |DECIMAL;

inputs ::=   CHARINPUT
        |FLOATINPUT
        |INTINPUT;

codigo_funcion_return ::=    struc_vars codigo_funcion_return
                    |struc_ciclos_return codigo_funcion_return
                    |struc_if_return codigo_funcion_return
                    |struc_switch codigo_funcion_return
                    |mensaje codigo_funcion_return
                    |inputs codigo_funcion_return
                    |comentarios codigo_funcion_return
                    |RETURN valor PUNTO_COMA
                    |;
```

GRAMATICA PY

Léxico

Reservada	Nombre
def	DEF
and	AND
or	OR
not	NOT
+	MAS
-	MENOS
*	POR
/	DIV
\	DIV_ENTERO
<	MENOR_QUE
>	MAYOR_QUE
<=	MENOR_IGUAL
>=	MAYOR_IGUAL
==	IGUAL
¡=	DIFERENTE
Is	IS
println(	MSG
print(	MSG
intinput	INTINPUT
floatinput	FLOATINPUT
charinput	CHARINPUT
in	IN
for	FOR
Range	RANGE
if	IF
else	ELSE
elIf	ELSE_IF
end	END
switch	SWITCH
Case	CASE
public	PUBLIC
void	VOID
Return	RETURN

[	CORCHETE_A
]	CORCHETE_C
{	LLAVES_A
}	LLAVES_C
(	PARENTESIS_A
)	PARENTESIS_C
,	COMA

Sintáctico

INICIO ::= SEPARADOR_PY codigo_inicio SEPARADOR_PROGRAMA ;

declaracion_var ::= IDENTIFICADOR IGUAL valor_tipo SALTO;

valor_tipo ::= valor
|solicitud;

codigo_inicio ::= tabs codigo
|;

codigo ::= funcion codigo_inicio
|declaracion_var codigo_inicio
|ciclos codigo_inicio
|condicionales codigo_inicio
|mensajes codigo_inicio
|solicitud SALTO codigo_inicio
|SALTO codigo_inicio;

valor ::= valor MAS valor
|valor COMA valor
|valor MENOS valor
|valor POR valor

```
        |valor DIV valor
        |valor DIV_ENTERO valor
        |valor POT valor
        |PARENTESIS_A valor PARENTESIS_C
        |IDENTIFICADOR
        |VALOR
        |TEXTO
        |NUMERO;

funcion ::= DEF IDENTIFICADOR PARENTESIS_A parametros PARENTESIS_C
DOS_PUNTOS SALTO;

parametros ::=   IDENTIFICADOR params
            |;

params ::=   COMA IDENTIFICADOR
        |;

codigo_def ::= declaracion_var;

tabs ::= TAB tabs
      |;

ciclos ::=   struct_for
        |struct_while;

struct_for ::=   FOR IDENTIFICADOR IN datos_for DOS_PUNTOS SALTO
            |FOR GUION_BAJO IN datos_for DOS_PUNTOS SALTO;
```

```
datos_for ::=    TEXTO
              |NUMERO
              |CORCHETE_A dato mas_datos CORCHETE_C;

dato ::= TEXTO
       |NUMERO;

mas_datos ::= COMA dato mas_datos
           | ;

struct_while ::= WHILE condicion DOS_PUNTOS SALTO;


condicion ::= not condiciones pre;

condiciones ::=  valor MENOR_QUE valor
              |valor MAYOR_QUE valor
              |valor MENOR_IGUAL valor
              |valor MAYOR_IGUAL valor
              |valor IGUAL_IGUAL valor
              |valor DIFERENTE valor
              |PARENTESIS_A condiciones PARENTESIS_C;

pre ::= AND not condiciones pre
       |OR not condiciones pre
       |;

not ::=  NOT
```

```
        |;

condicionales ::= struc_if;

struc_if ::= IF condicion DOS_PUNTOS SALTO
           |ELSE_IF condicion DOS_PUNTOS SALTO
           |ELSE;

mensajes ::= PRINT val PARENTESIS_C SALTO;

solicitud ::=    INTINPUT PARENTESIS_A val PARENTESIS_C
               |FLOATINPUT PARENTESIS_A val PARENTESIS_C
               |CHARINPUT PARENTESIS_A val PARENTESIS_C;

val ::= valor
        |;
```

GRAMATICA CPP

Léxico

Reservada	Nombre
class	CLASE
&&	AND
||	OR
¡	NOT
+	MAS
-	MENOS
*	POR
/	DIV
\	DIV_ENTERO
<	MENOR_QUE
>	MAYOR_QUE
<=	MENOR_IGUAL
>=	MAYOR_IGUAL
==	IGUAL
¡=	DIFERENTE
Is	IS
printf(	MSG
getch(	GETCH
scanf(	SCANF
Clrast()	CLRS
intinput	INTINPUT
floatinput	FLOATINPUT
charinput	CHARINPUT
for	FOR
do	DO
if	IF
else	ELSE
else If	ELSE_IF
end	END
switch	SWITCH
Case	CASE
public	PUBLIC
void	VOID
Return	RETURN

Sub	SUB
[	CORCHETE_A
]	CORCHETE_C
{	LLAVES_A
}	LLAVES_C
(	PARENTESIS_A
)	PARENTESIS_C
,	COMA

Sintáctico

INICIO ::= SEPARADOR_PROGRAMA includs_code codigo struc_function;

includs_code ::= includs includs_code

|;

codigo ::= struc_vars codigo

|mensaje codigo

|comentarios codigo

|;

codigo_funcion ::= struc_vars codigo_funcion

|struc_ciclos codigo_funcion

|struc_if codigo_funcion

|struc_switch codigo_funcion

|mensaje codigo_funcion

|comentarios codigo_funcion

|SCANF TEXTO:a struc_scanf PARENTESIS_C PUNTO_COMA codigo_funcion

|GETCH PUNTO_COMA codigo_funcion

|ANDPERSAND codigo_funcion

|CLEARS PUNTO_COMA codigo_funcion

|struc_objeto codigo_funcion

|struc_leng codigo_funcion

|;

struc_llamada ::= PUNTO IDENTIFICADOR struc_llamada

|PUNTO IDENTIFICADOR PARENTESIS_A parametros_llamada PARENTESIS_C;

struc_objeto ::= JV PUNTO IDENTIFICADOR IDENTIFICADOR struc_var_obj PUNTO_COMA

|JV PUNTO IDENTIFICADOR IDENTIFICADOR PARENTESIS_A params_llamada PARENTESIS_C PUNTO_COMA

|JV PUNTO IDENTIFICADOR:a PUNTO IDENTIFICADOR:b PARENTESIS_A params_llamada PARENTESIS_C PUNTO_COMA

struc_var_obj ::= struc_id_ob;

struc_id_ob ::= COMA IDENTIFICADOR struc_id_ob

|;

struc_leng ::= VB PUNTO IDENTIFICADOR PARENTESIS_A params_llamada PARENTESIS_C PUNTO_COMA

|PY:c PUNTO IDENTIFICADOR PARENTESIS_A params_llamada PARENTESIS_C PUNTO_COMA ;

params_llamada ::= valor COMA params_llamada

|valor

|;

```
parametros_llamada ::= valor val_llamada
                    |;

val_llamada ::= COMA valor val_llamada
             |;

comentarios ::=  COMENTARIO_SIMPLE
             |COMENTARIO_VARIOS;


includs ::=  INCLUDE MENOR_QUE IDENTIFICADOR identificador_include
MAYOR_QUE
          |INCLUDE PY_ALL
          |INCLUDE VB_ALL
          |INCLUDE JV_ALL
          |INCLUDE JV_ONE

identificador_include ::= PUNTO IDENTIFICADOR parentesis
                    |PUNTO IDENTIFICADOR identificador_include
                    |PUNTO IDENTIFICADOR;

parentesis ::= PARENTESIS_A PARENTESIS_C;

struc_vars ::=   INT struc_asig_vars_n PUNTO_COMA
             |FLOAT struc_asig_vars_n PUNTO_COMA
             |CHAR struc_asig_vars_c PUNTO_COMA
             |CONSTANTE struc_constante
             |IDENTIFICADOR IGUAL var_senten:b PUNTO_COMA
```

|IDENTIFICADOR struc_dim_arreglo IGUAL var_senten PUNTO_COMA;

var_senten ::= valor:a

|GETCH

|VB:c PUNTO IDENTIFICADOR:a PARENTESIS_A params_llamada PARENTESIS_C

|PY:c PUNTO IDENTIFICADOR:a PARENTESIS_A params_llamada PARENTESIS_C

|JV:c PUNTO IDENTIFICADOR:a PUNTO IDENTIFICADOR:b PARENTESIS_A params_llamada PARENTESIS_C

struc_constante ::= INT struc_asig_vars_n PUNTO_COMA

|FLOAT struc_asig_vars_n PUNTO_COMA

|CHAR struc_asig_vars_c PUNTO_COMA;

struc_dim_arreglo ::= CORCHETE_A valor CORCHETE_C struc_dim_arreglo

|CORCHETE_A valor CORCHETE_C;

struc_asig_vars_n ::= IDENTIFICADOR IGUAL s_n

COMA struc_asig_vars_n

|IDENTIFICADOR

COMA struc_asig_vars_n

|IDENTIFICADOR s_n

|IDENTIFICADOR struc_arreglo

|IDENTIFICADOR

s_n ::= valor_n:a

|GETCH

|VB:c PUNTO IDENTIFICADOR:a PARENTESIS_A params_llamada PARENTESIS_C

|JV:c PUNTO IDENTIFICADOR:a PUNTO IDENTIFICADOR:b PARENTESIS_A params_llamada PARENTESIS_C

struc_asig_vars_c ::= IDENTIFICADOR:b IGUAL s_c:a

COMA struc_asig_vars_c

|IDENTIFICADOR

COMA struc_asig_vars_c

|IDENTIFICADOR:b IGUAL s_c

|IDENTIFICADOR

s_c ::= valor_c

|GETCH

|VB PUNTO IDENTIFICADOR PARENTESIS_A params_llamada PARENTESIS_C

|PY PUNTO IDENTIFICADOR PARENTESIS_A params_llamada PARENTESIS_C

|JV PUNTO IDENTIFICADOR PUNTO IDENTIFICADOR PARENTESIS_A params_llamada PARENTESIS_C

struct_while ::= WHILE PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion LLAVES_C;

struct_while_return ::= WHILE PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion_return LLAVES_C ;

struct_do_while ::= DO LLAVES_A codigo_funcion LLAVES_C WHILE PARENTESIS_A struc_condicional PARENTESIS_C PUNTO_COMA;

struct_do_while_return ::= DO LLAVES_A codigo_funcion_return LLAVES_C WHILE PARENTESIS_A struc_condicional PARENTESIS_C PUNTO_COMA;

valor_condicional ::= valor MENOR_QUE valor
|valor MAYOR_QUE valor
|valor MENOR_IGUAL valor
|valor MAYOR_IGUAL valor
|valor IGUAL_IGUAL valor
|valor DIFERENTE valor;

valor ::= valor MAS valor
|valor MENOS valor
|valor POR valor
|valor DIV valor
|PARENTESIS_A valor PARENTESIS_C
|IDENTIFICADOR
|VALOR
|NUMERO
|DECIMAL;

struc_condicional ::= struc_logico;

struc_logico ::= struc_logico AND struc_logico
|struc_logico OR struc_logico
|PARENTESIS_A struc_logico PARENTESIS_C
|NOT PARENTESIS_A struc_logico PARENTESIS_C
|valor_condicional;

struc_logico_not ::= NOT

|;

struc_if ::= IF PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion LLAVES_C struc_else ;

struc_if_return ::= IF PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion_return LLAVES_C struc_else_return ;

struc_else ::= ELSE IF PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion LLAVES_C struc_else

|ELSE LLAVES_A codigo_funcion LLAVES_C

| ;

struc_else_return ::= ELSE IF PARENTESIS_A struc_condicional PARENTESIS_C LLAVES_A codigo_funcion_return LLAVES_C struc_else_return

|ELSE LLAVES_A codigo_funcion_return LLAVES_C

| ;

struc_switch ::= SWITCH PARENTESIS_A IDENTIFICADOR PARENTESIS_C LLAVES_A struc_case LLAVES_C;

struc_case ::= CASE valor_case DOS_PUNTOS codigo_funcion BREAK PUNTO_COMA struc_case

|DEFAULT DOS_PUNTOS codigo_funcion

|;

valor_case ::= VALOR

|NUMERO

|DECIMAL;

struc_function ::= PUBLIC VOID IDENTIFICADOR PARENTESIS_A struc_params PARENTESIS_C LLAVES_A codigo_funcion LLAVES_C

|PUBLIC vars IDENTIFICADOR PARENTESIS_A struc_params PARENTESIS_C LLAVES_A codigo_funcion_return LLAVES_C;

struc_params ::= struc_vars_params params

|;

params ::= COMA struc_vars_params params

|;

struc_vars_params ::= INT IDENTIFICADOR

|FLOAT IDENTIFICADOR

|CHAR IDENTIFICADOR;

vars ::= INT

|FLOAT

|CHAR;

mensaje ::= SOUT valor_m PARENTESIS_C PUNTO_COMA;

valor_m ::= valor_m MAS valor_m

|valor_m MENOS valor_m

|valor_m POR valor_m

|valor_m DIV valor_m

|PARENTESIS_A valor_m PARENTESIS_C

|IDENTIFICADOR

|TEXTO

```
        |VALOR
        |NUMERO
        |DECIMAL;

inputs ::=   CHARINPUT
        |FLOATINPUT
        |INTINPUT;
```