MANUAL TÉCNICO

Introducción:

Descripción del Sistema: Esta aplicación consiste en un analizador léxico, sintáctico y semántico, analiza código Pascal, tienen la opción de poder visualizar los errores, tabla de símbolos, tabla de tipos, y el árbol de activaciones.

Requisitos Previos: Para entender este manual y la aplicación debe de tener conocimientos básicos de JAVA, y saber sobre las herramientas de flex y cup.

Instalación:

Editor de Código: El editor de código recomendado para este sistema es visual studio code,netbeans o Intellij IDEA.

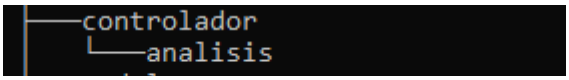
Configuración Inicial: Deberá tener instalada alguna versión de java, específicamente el JDK con la versión 17 en adelante para evitar problemas, además de tener instalado Graphviz y agregarlos a las variables de entorno para que se puedan ejecutar las imágenes correctamente .

Arquitectura del Sistema:

Arquitectura: La arquitectura aplicada para este sistema fue un patrón de diseño llamado MVC (Modelo,Vista,Controlador), los archivos del sistema están distribuidos de la siguiente manera. podemos observar el Modelo: este contiene otros paquetes que cada una maneja el análisis semántico correspondiente de cada bloque de código. Tenemos el Controlador, este es el archivo java que nos genera flex y cup para el análisis léxico y sintáctico. En el apartado de la vista podemos encontrar lo que sería la ventana principal de nuestro sistema.

Controlador:

En este apartado podemos encontrar todo los archivos generados por jflex y cup, estos nos van a servir para conectarlo con la vista y que haga uso de los componentes del modelo para realizar el análisis.

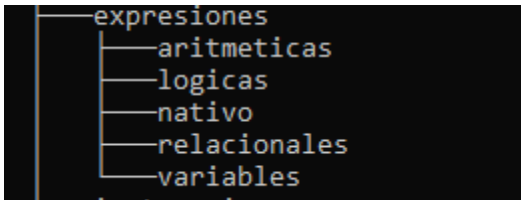


Modelo:

Como ya se había mencionado anteriormente aquí se hará el análisis semántico de todas las operaciones realizadas en nuestro sistema. este se divide en varios paquetes, cada uno correspondiente al tipo que realizará implementar.

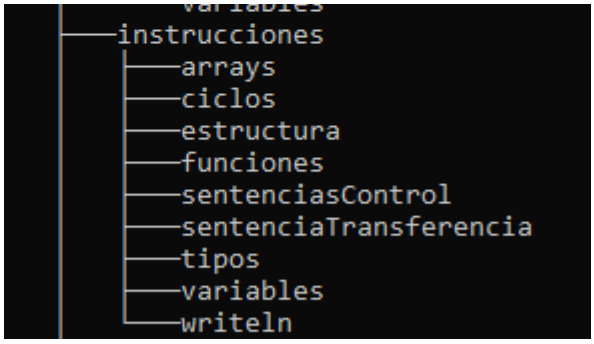
expresiones:

Este paquete contiene todas las clases que manejan la lógica de las expresiones de nuestro código, estas expresiones pueden ser aritméticas como lo son las sumas, restas, multiplicaciones etc, también podemos encontrar expresiones lógicas, relacionales y expresiones como variable, esta última hace referencia al acceso de una variable.



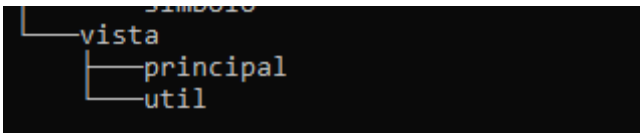
instrucciones:

En este paquete podemos encontrar la lógica para el análisis semántico en las instrucciones, como por ejemplo lo que son las declaraciones de arreglos, como asignar datos a un arreglo, y como poder acceder a ellos, tambien contamos con la lógica de los ciclos, como lo es el for, while y el repeat, funciones podemos encontrar el manejo de estas, como lo son los argumentos , que tipo de argumento recibe, etc, tenemos las sentencias de control, como son los if, y los cases, tenemos el paquete de tipos que nos permiten manejar la lógica de la creación de nuevos tipos, etc.



Vista:

Aquí encontraremos la ventana principal de nuestro sistema, además contamos con una segunda llamada reportes, esta se desplegará cuando elijamos la opción de reportes en nuestra ventana principal.



GRAMÁTICA

```
cuerpo ::= cuerpo:a estructura:
            | estructura:

estructura ::=  declaracion_program
                |estructura_tipos
                |estructura_const
                | estructura_var
                | declaracion_funcion
                | declaracion_procedimiento
                | estructura_main


estructura_tipos ::= TYPE declaraciones_tipos

estructura_const ::=CONST declaracion_constantes:

estructura_var ::= VAR declaraciones_variables:


estructura_main ::=BEGIN instrucciones:a  END PUNTO


instrucciones ::= instrucciones instruccion
                | instruccion
;

instruccion ::= writeln
                | readln
                | asignacion_variable
                | asignacion_array
                | asignacion_record
                | llamada_metodo FINCADENA
                | sentencia_if   FINCADENA
                | sentencia_case FINCADENA
                | sentencia_while  FINCADENA
                | sentencia_for FINCADENA
                | sentencia_repeat FINCADENA
                | transferencia_break   FINCADENA
                | transferencia_continue FINCADENA
                | error FINCADENA
                | error

instruccion_fin ::= writeln
                | readln
                | asignacion_variable
```

```
                | asignacion_array
                | asignacion_record
                | llamada_metodo
                | sentencia_if
                | sentencia_case
                | sentencia_while
                | sentencia_for
                | sentencia_repeat
                | transferencia_break
                | transferencia_continue

instruccion_else_fin ::= writeln
                | readln
                | asignacion_variable
                | asignacion_array
                | asignacion_record
                | llamada_metodo
                | sentencia_case
                | sentencia_while
                | sentencia_for
                | sentencia_repeat
                | transferencia_break
                | transferencia_continue
;

declaracion_program ::= PROGRAM ID:a FINCADENA

declaracion_procedimiento ::= PROCEDURE ID:a PAR1 lista_param_f:b PAR2 FINCADENA VAR
declaraciones_variables:d BEGIN instrucciones:e END FINCADENA

                    |PROCEDURE ID:a PAR1 lista_param_f:b PAR2 FINCADENA BEGIN
instrucciones:d END FINCADENA

           |PROCEDURE ID: PAR1 PAR2 FINCADENA VAR declaraciones_variables:d BEGIN
instrucciones:e END FINCADENA

       |PROCEDURE ID:a PAR1  PAR2 FINCADENA BEGIN instrucciones:d END FINCADENA

declaracion_funcion ::= FUNCTION ID:a PAR1 lista_param_f:b PAR2 DOSPUNTOS
tipos_dato_var:c FINCADENA VAR declaraciones_variables:d BEGIN instrucciones:e END
FINCADENA

                    |FUNCTION ID:a PAR1 lista_param_f:b PAR2 DOSPUNTOS tipos_dato_var:c
FINCADENA BEGIN instrucciones:d END FINCADENA

             |FUNCTION ID:a PAR1 PAR2 DOSPUNTOS tipos_dato_var:c FINCADENA VAR
declaraciones_variables:d BEGIN instrucciones:e END FINCADENA

                    |FUNCTION ID:a PAR1  PAR2 DOSPUNTOS tipos_dato_var:c FINCADENA BEGIN
```

```
instrucciones:d END FINCADENA

lista_param_f ::= lista_param_f:a FINCADENA lista_params_id:b DOSPUNTOS tipos_dato_var:c

               | lista_params_id:a DOSPUNTOS tipos_dato_var:b


lista_params_id ::= lista_params_id:a COMA tipo_id_f:b
                    | tipo_id_f:a

 tipo_id_f ::= VAR ID
             | ID

llamada_metodo ::= ID:a PAR1 parametros_llamada:b PAR2
                | ID:a PAR1 PAR2

llamada_funcion ::= ID:a PAR1 parametros_llamada:b PAR2
                | ID:a PAR1 PAR2


parametros_llamada ::=  parametros_llamada:a COMA expresion
            | expresion:a


declaraciones_tipos ::=declaraciones_tipos:a declaracion_tipo:b
                    | declaracion_tipo

declaracion_tipo ::=   listado_id:a IGUAL tipos_variable:b FINCADENA
                      |listado_id:a IGUAL expresion:b PRANGO expresion:c FINCADENA
                      | listado_id:a IGUAL ARRAY CORCHETE1 expresion:b PARRAY
expresion:c CORCHETE2 OF tipos_variable:d FINCADENA
                      | listado_id:a IGUAL ARRAY CORCHETE1 expresion:c CORCHETE2 OF
tipos_variable:d FINCADENA
                      | declaracion_record

declaracion_record ::= listado_id:a IGUAL RECORD lista_record:b END FINCADENA

lista_record ::= lista_record:a ID:b DOSPUNTOS tipo_record:c

                   | ID:a DOSPUNTOS tipo_record:b


tipo_record ::= tipos_variable:a FINCADENA
                   | expresion:b PRANGO expresion:c  FINCADENA
                   | ARRAY CORCHETE1 expresion:b PARRAY expresion:c CORCHETE2  OF
tipos_variable:d FINCADENA

                   | ARRAY CORCHETE1 expresion:c CORCHETE2 OF tipos_variable:d FINCADENA
```

```
declaracion_constantes ::=   declaracion_constantes:a ID:b IGUAL expresion:c FINCADENA
                        |ID:b IGUAL expresion:c FINCADENA

declaraciones_variables ::= declaraciones_variables:a declaracion_variables
                        | declaracion_variables:a

declaracion_variables ::= listado_id:a DOSPUNTOS tipos_dato_var:b FINCADENA
                        |listado_id:a DOSPUNTOS expresion:b PRANGO expresion:c FINCADENA
                        |listado_id:a DOSPUNTOS ARRAY CORCHETE1 expresion:b PARRAY
expresion:c CORCHETE2 OF tipos_dato_var:d FINCADENA
                         |listado_id:a DOSPUNTOS ARRAY CORCHETE1 expresion:c CORCHETE2
OF tipos_dato_var:d FINCADENA
                        | declaracion_var_record:a {

declaracion_var_record ::= listado_id:a IGUAL RECORD lista_record:b END FINCADENA



listado_id ::= listado_id:a COMA ID:e
            | ID:a

tipos_variable ::= INTEGER
                | REAL
                | STRING
                | CHAR
                | BOOL
                | VOID

tipos_dato_var ::= INTEGER
                | REAL
                | STRING
                | CHAR
                | BOOL
                | VOID
                | ID



writeln ::= IMPRIMIR PAR1 expresion_conca:a PAR2

readln ::= READLN PAR1 listado_id:a PAR2

expresion_conca ::= expresion_conca:a COMA expresion
                | expresion

asignacion_variable ::= ID:a DOSPUNTOS IGUAL expresion:
```

```
asignacion_record ::= ID:a PUNTO ID:b DOSPUNTOS IGUAL expresion:c

 asignacion_array ::= ID:a CORCHETE1 expresion:b CORCHETE2 DOSPUNTOS IGUAL expresion


sentencia_if ::= IF expresion:a  THEN instruccion_fin:b
                |IF expresion:a THEN BEGIN instrucciones:b END
                |IF  expresion:a THEN BEGIN instrucciones:b END ELSE
instruccion_else_fin:
                | IF expresion:a THEN instruccion_fin:b ELSE instruccion_else_fin:c
                |IF expresion:a THEN instruccion_fin:b ELSE BEGIN instrucciones:c END
                |IF expresion:a THEN BEGIN instrucciones:b END ELSE BEGIN
instrucciones:c END
                |IF expresion:a THEN instruccion_fin:b ELSE sentencia_if:c
                |IF expresion:a THEN BEGIN instrucciones:b END ELSE sentencia_if:c

;



sentencia_case ::= CASE  expresion:a  OF casos_case:b ELSE BEGIN instrucciones:c  END
FINCADENA END
                  | CASE expresion:a OF casos_case:b ELSE  instruccion:c  END

casos_case ::= casos_case:a caso_case:b
            | caso_case:a

caso_case ::=  expresiones_case:a DOSPUNTOS BEGIN instrucciones:b END FINCADENA
            | expresiones_case:a DOSPUNTOS instruccion:b

expresiones_case ::= expresiones_case:a COMA expresion:b
                    | expresion

sentencia_while ::= WHILE expresion:a  DO BEGIN instrucciones:b END
                | WHILE  expresion:a  DO instruccion_fin:b

sentencia_for ::= FOR ID:a DOSPUNTOS IGUAL expresion:b TO expresion:c DO
instruccion_fin:d
                |FOR ID:a DOSPUNTOS IGUAL expresion:b TO expresion:c DO BEGIN
instrucciones:d END


sentencia_repeat ::= REPEAT  instrucciones:b  UNTIL expresion

transferencia_break::=  BREAK

transferencia_continue::=  CONTINUE
```

```
expresion ::=  expresionAritmetica
            | expresionRelacional:
            | expresionLogica
            | llamada_funcion
            | ENTERO
            | DECIMAL
            | CADENA
            | CARACTER
            | BOOLEAN:
            | ID:a
            | ID:a PUNTO ID:b
            | ID:a CORCHETE1 expresion:b CORCHETE2

;



expresionAritmetica ::= MENOS expresion:a
            | expresion:a MAS expresion:b
            | expresion:a MENOS expresion:b
            | expresion:a MULT expresion:b
            | expresion:a DIV expresion:b
            | expresion:a DIVE expresion:b
            | expresion:a MOD expresion:b
            | PAR1 expresion:a PAR2



expresionRelacional ::= expresion:a IGUAL expresion:b
                    | expresion:a DIFERENTE expresion:b
                    | expresion:a MENOR expresion:b
                    | expresion:a MENORIGUAL expresion:b
                    | expresion:a MAYOR expresion:b
                    | expresion:a MAYORIGUAL expresion:b


expresionLogica ::= NOT expresion:a
                | expresion:a AND expresion:b
                | expresion:a OR expresion:b
                | expresion:a AND THEN expresion:b
                | expresion:a OR ELSE expresion:b
```