Manual Técnico

Analizador Léxico

La clase analizador léxico es la que se encarga de poder realizar todo el análisis léxico del archivo de entrada o el texto introducido en la aplicación, dentro de esta clase se encuentra los siguientes métodos

- **public list<token> AnalizadorLex(string entrada**): es el método encargado de la verificación de los lexemas del cadena de entrada pertenezca a alguno de los tokens, se maneja los estado de transición del autómata. Retorna List<Token>.
- **Public void AgregarToken(Toke.tipos tipo**): método que se encarga de agregar los lexemas valido a una lista de tokens y asi mismo aquellos no validos a una lista de errores.
- **Public bool palabrasReservadas(string entrada):** método que se encarga de verificar las palabras reservadas que existen dentro del lenguaje. Retorna un valor booleano.

AnalizadorSintactico

La clase analizador sintáctico es la que se encarga de poder verificar el orden en que son enviado los tokens y que estos sean correctos.

Constructor:

Public AnalizadorSintactico(List<Token> token) : constructor que recibe como parámetro la lista de token obtenida después de realizar el análisis léxico.

Métodos:

- **public void Match(Toke.Tipos tipos):** el método match se encarga de verificar si el token enviado, es el token que se esperaba en la lista de tokens.

Los Siguientes métodos hacen referencia a los No terminales de la gramática tipo 2 que estará al final.

- public void Inicio()
- public void CuerpoPrincipal();
- public vod RepetirCuerpoPrinciapl()
- public void Bloque_Nivel()
- public void CuerpoNivel();
- public void RepetirCuerpoNivel()
- public void BloqueDimensiones()
- public void BloquePared()
- public void RepPared()
- public void NumId()
- public void Espacio()
- public void DosPuntos()
- public void Espacios()
- public void DeclaracionVariable()

- public void RepetirId()
- public void AsignarVariable

Clase Token

La clase token es utilizada para el manejo de todos los token definidos dentro del lenguaje posee los siguientes métodos:

- **public String getLexema():** devuelve el lexema guardado dentro de un objeto token.
- **Public int getFila():** devuelve el número de fila en que se encuentra un token especifico.
- **public int getColumna():** devuelve el número de la columna en que se termina un token especifico.
- **public string getTipo():** devuelve el tipo de token al que pertenece un objeto token

TipoToke:

CORCHETE_IZQ,
CORCHETE_DER,
DOS_PUNTOS,
LLAVE_IZQ,
LLAVE_DER,
PUNTO_COMA,
PUNTO,
COMA,
PARENTESIS_IZQ,
PARENTESIS_DER,
IGUAL,
SUMA,
RESTA,
MULTIPLICACION,
DIVISION,
PALABRA_RESERVADAS,
VARIABLE,
IDENTIFICADOR,
NUMERO,
LETRA,
PRINCIPAL,
INTERVALO,
NIVEL,
DIMENSIONES,
INICIO_PERSONAJE,
UBICACION_SALIDA,
PARED,
CASILLA,
PERSONAJE,
PASO,
CAMINATA,
ENEMIGO,
Varias_casillas,
OTRO

Clase ManejoErrores:

Esta clase es utilizada para poder manejar los errores de tipo léxico y sintáctico encontrados en análisis del programa.

Constructor:

Public ManejoErrores(string error, string tipo, string descripcion, int fila,int columan)

Métodos:

- public string getError(): se encarga de obtener el error que se ha guardado.
- Public string getTipo(): se encarga de obtener el tipo de error (léxico o sintáctico)
- Public string getDescripcion(): obtiene un pequeña descripción del error.
- Pubic int getFila(): obtiene la fila del error.
- Public in getColumna: obtiene la columna en donde está el error.

Clase Cordenadas:
La clase cordenadas se encarga de guardar la el recorrido del personaje

Constructor:
Public cordenadas(int x,int y): recibe como parámetro la posición tanto para x y para y dentro del juego.

Métodos:

- Public int getX(): obtiene la posición para x del personaje.
- Public int getY() obtiene la posición para y del personaje.

Clase Enemigo:
La clase enemigo es utilizada para el manejo de los enemigos del juego.

Métodos:

- **Public void caminataHorizontal(int x1, int x2, int y1):** método que se encarga de guardar la posición del enemigo cuando hace una caminata de forma horizontal
- **Public void caminataVerticall(int x1, int y1, int y2):** método que se encarga de guardar la posición del enemigo cuando hace una caminata de forma vertical

Clase Variables
Clase que se encarga del manejo de las variables dentro del juego.

Constructor:

- Public Variables(string identificador): recibe como parámetro el identificador que tendrá la variable.

Métodos:

- **Public string getId():** obtiene el identificador de un objeto variable.
- **Public int getValo():** obtiene el valor de un objeto variable.
- **Public void setValor(int valor):** recibe como parámetro un valor entero para establecerlo como nuevo dentro de un objeto variable.

Clase Juego
La clase que se encarga de la extracción de los datos para poder formar el juego.

Métodos:

- Public List<Variable> variables(List<Token> tokens): es usado para poder extraer cada variable definida dentro primer bloque pared y primer bloque personaje. Retornar un valor List<Variable>
- Public void GuardarDatos(List<Token> tokens, List<Varibles> variables) método usado para guardar los datos para el personaje, enemigo, nivel, pared, etc.

Obtiene el primer bloque personaje y el primer bloque enemigo. Obtiene el último bloque intervalo, dimensiones, inicio, fin

Formulario Form1

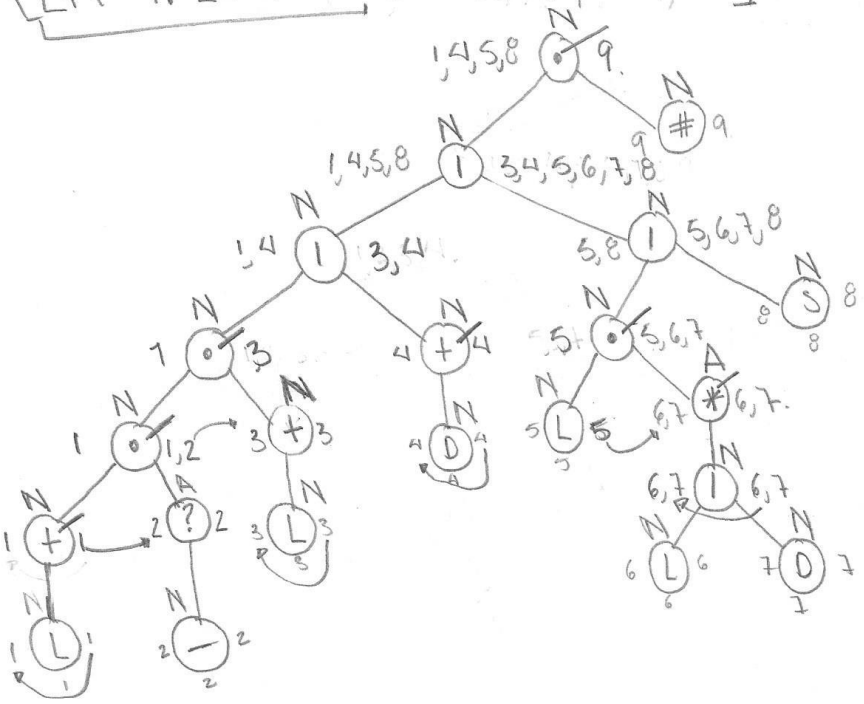
Contiene la interfaz gráfica de la aplicación.

menuStrip1 contiene las herramientas de la cinta superior de la interfaz
Explorador OpenFileDialog utilizado para permitirle al usuario poder abrir un archivo desde cualquier dirección
Guardar SaveFileDialog usado para que el usuario pueda guardar un archivo ya existente verificando que sea válido o un archivo nuevo.
Guardar Como SaveFileDialog usado para guardar una archivo ya existen con otro nombre y nueva dirección
Texto RichTextBox que le permite al usuario realizar la edición del texto
Métodos:
 public void Pintar(String entrada) método encargado de pintar aquellos token validos dentro del editor de texto, recibe como parámetro la lista de tokens que se genero

Eventos:

- **private void analizarToolStripMenuItem_Click(object sender, EventArgs e)** evento que se encarga de correr el elemento explorador y así mismo extrae el contenido del elemento areaTra para poder analizarlo
- **private void abrirToolStripMenuItem_Click(object sender, EventArgs e)** Evento que se encarga de llamar al elemento explorador
- **private void guardarToolStripMenuItem_Click(object sender, EventArgs e)** Evento que se encarga de llamar al elemento Guardar
- **private void guardarComoToolStripMenuItem_Click(object sender, EventArgs e)** Evento que se encarga de llamar al elemento GuardarComo.
- **private void acercaDeToolStripMenuItem_Click(object sender, EventArgs e)** Evento que abre una venta, donde se muestran los datos del creador de la aplicación.
- private void tableroJuegoToolStripMenuItem_Click(object sender, EventArgs e) Cargar la interfaz gráfica del tablero.

$(L+(-)?L+ \mid (D)+ \mid [L(L \mid D)* \mid S])\#$

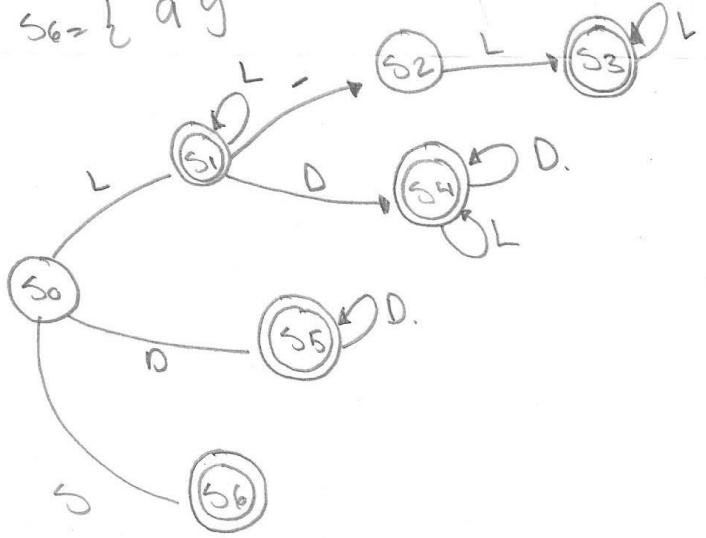


Σ	#	Follow Pass
L	1	1, 2, 3
—	2	3, 9
L	3	3, 9
D	4	4, 9
L	5	6, 7, 9
L	6	6, 7, 9
D	7	6, 7, 9
S	8	9
#	9	

$S = \{\ [\ ]\ :\ \{\ \}\ ;$
$\quad ..\ ,\ (\ )$
$\quad = \ +\ \ *\ \ |\ \ -$

Tabla Transicion

Estado	L	-	D	S
$S_0 = \{^{L}1, ^{D}4, ^{L}5, ^{S}8\}$	S1		S5	S6
(*) $S_1 = \{^{L}1, ^{-}2, ^{L}3, ^{L}6, ^{D}7, ^{\#}9\}$	S1	S2	S4	
$S_2 = \{^{L}3\}$	S3			
* $S_3 = \{^{L}3, ^{\#}9\}$	S3			
* $S_4 = \{^{L}6, ^{D}7, ^{\#}9\}$	S4		S4	
* $S_5 = \{4, 9\}$			S5	
* $S_6 = \{^{\#}9\}$				



GRAMATICA TIPO 2

<Inicio>::=[Tk.Prinicipal]:{<CuerpoPrincipal>}

<CuerpoPrincipal>::=[Tk.intervalo]:<bloque_inter><repetir>
|[Tk.nivel]: <bloque_nivel><repetir>
|[Tk.enemigo]: <bloque_enemigo><repetir>
|[Tk.personaje]: <bloque_personaje><repetir>

<repetir>::=[Tk.intervalo]: <bloque_inter><repetir>
|[Tk.nivel]: <bloque_nivel><repetir>
|[Tk.enemigo]: <bloque_enemigo><repetir>
|[Tk.personaje]: <bloque_personaje><repetir>
|EP

<bloque_inter>::= (numero);

<bloque_nivel>::={<CuerpoNivel>}

<CuerpoNivel>::= [Tk.Dimensiones]:<bloque_dimensiones><CN>
|[Tk.Inicio]:<bloque_dimensiones><CN>
|[Tk.Salida]:<bloque_dimensiones><CN>
|[Tk.pared]: {<bloque_pared>}<CN>

<^CN>::= [Tk.Dimensiones]:<bloque_dimensiones><^CN>
|[Tk.Inicio]:<bloque_dimensiones><^CN>
|[Tk.Salida]:<bloque_dimensiones><^CN>
|[Tk.pared]: {<bloque_pared>}<^CN>
|Ep

<bloque_dimensiones>::= (numero,numero);

<bloque_pared>::= [Tk.Casilla]:<espacio> <Rep_Pared>
|[Tk.VariasCasillas]: <espacios> <Rep_Pared>
|[Tk.Variable]:= <Dec_varialbe> <Rep_Pared>
|identificador:= <Asig_Variable> <Rep_Pared>

<Rep_Pared>::= [Tk.Casilla]: <espacio> <Rep_Pared>
|[Tk.VariasCasillas]: <espacios> <Rep_Pared>
|[Tk.Variable]: <Dec_varialbe> <Rep_Pared>
|identificador:= <Asig_Variable> <Rep_Pared>
|Ep

<Num_id>::= numero| identificador //verificar error
<espacio>::=(<Num_id>,<Num_id>);

<espacios>::=(<Num_id>..<Num_id>,<Num_id>);
|(<Num_id>,<Num_id>..<Num_id>);
|(<Num_id>..<Num_id>,<Num_id>..<Num_id>);

<Dec_varialbe>::= identificador<Rep_id>;
<Rep_id>::=,identificador<Rep_id>
|Ep

<Asig_Variable>--> <Num_id>+<Num_id>;
|<Num_id>-<Num_id>;
|<Num_id>*<Num_id>;
|<Num_id>/<Num_id>;
|<Num_id>;

<bloque_enemigo>::= {<CuerpoEnemigo>}

<CuerpoEnemigo>::= [Tk.Caminata]:(<Caminar>);<Otra_Caminata>
|Ep

<Otra_Caminata>::= [Tk.Caminata]:(<Caminar>);<Otra_Caminata>
| Ep

<Caminar>::= numero..numero,numero
|numero,numero..numero

<bloque_personaje>::= [Tk.Personaje]:{<CuerpoPersonaje>}

<CuerpoPersonaje>::= [Tk_Paso]:<paso><RepetirCuerpo>
|[Tk_Caminata]:<Moverse> <RepetirCuerpo>
|[Variable]:<Dec_Variable> <RepetirCuerpo>
|identificador:=<Asig_Variables> <RepetirCuerpo>

<RepetirCuerpo>::= [Tk_Paso]:<Paso><RepetirCuerpo>
|[Tk_Caminata]:<Moverse> <RepetirCuerpo>
|[Variable]:=<Dec_Variable> <RepetirCuerpo>
|identificador:=<Asig_Variables> <RepetirCuerpo>
| Ep

<Paso>::= (<Num_id>,<Num_id>);

<Moverse>::= (<Num_id>..<Num_id>,<Num_id>);
|(<Num_id>,<Num_id>..<Num_id>);