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

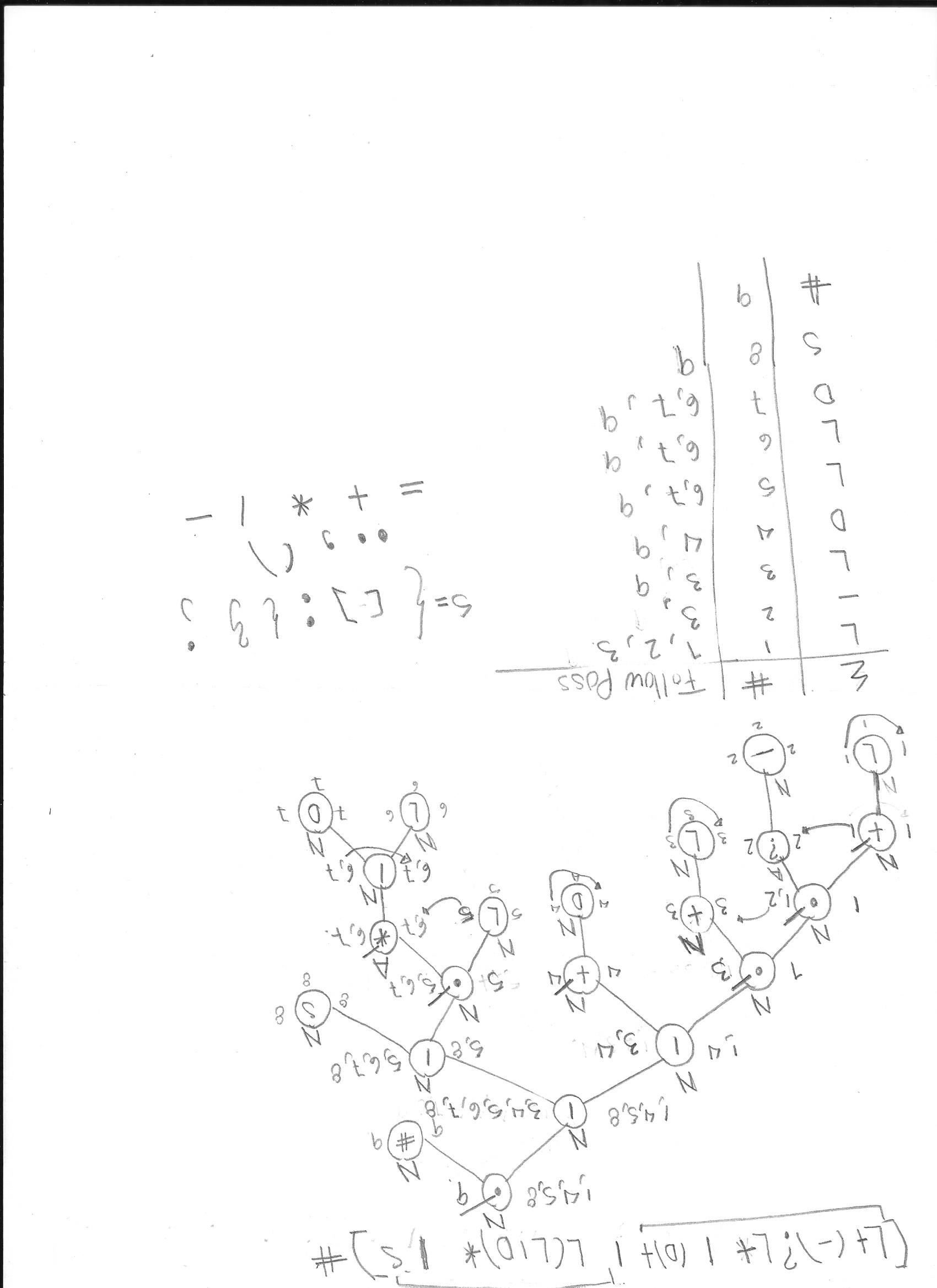
**Texto** RichTextBox que le permite al usuario realizar la edición del texto

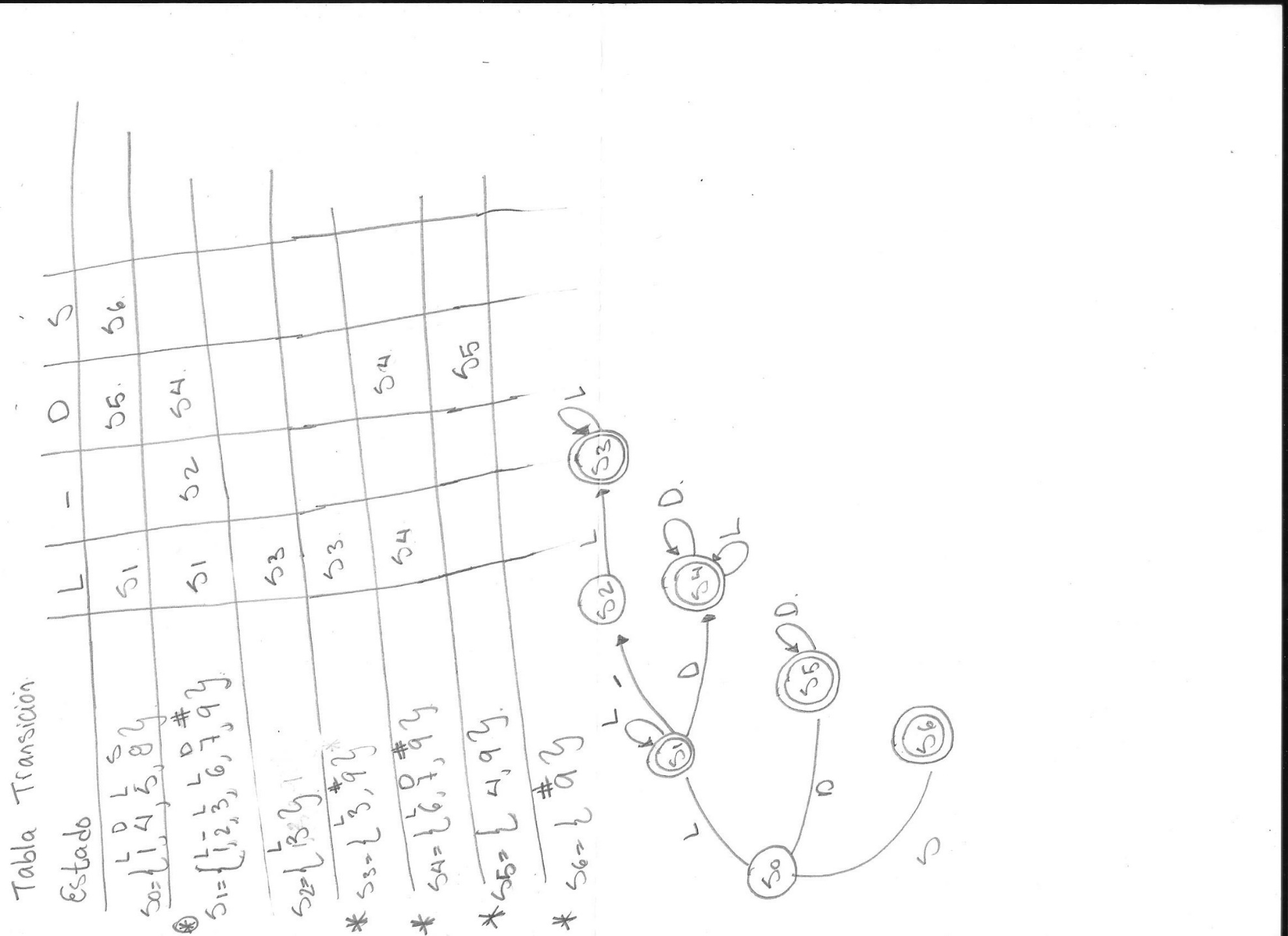
Métodos:

 **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.





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>);