Instituto Tecnológico de Durango

Analizador Léxico, Sintáctico y Semántico

Manual de Usuario

Carrera: **Ingeniería En Sistemas**

Materia: Lenguajes y Autómatas II

Estudiantes:

Jaime Andrés De La Cruz Cortés

Bernardo Salinas Jaquez

Fernanda Rodríguez Catarino

Grupo “7Y”

Facilitador**:** Rivera Saucedo Elda

Victoria de Durango, Durango, 05/10/19

Contenido

Presentación:

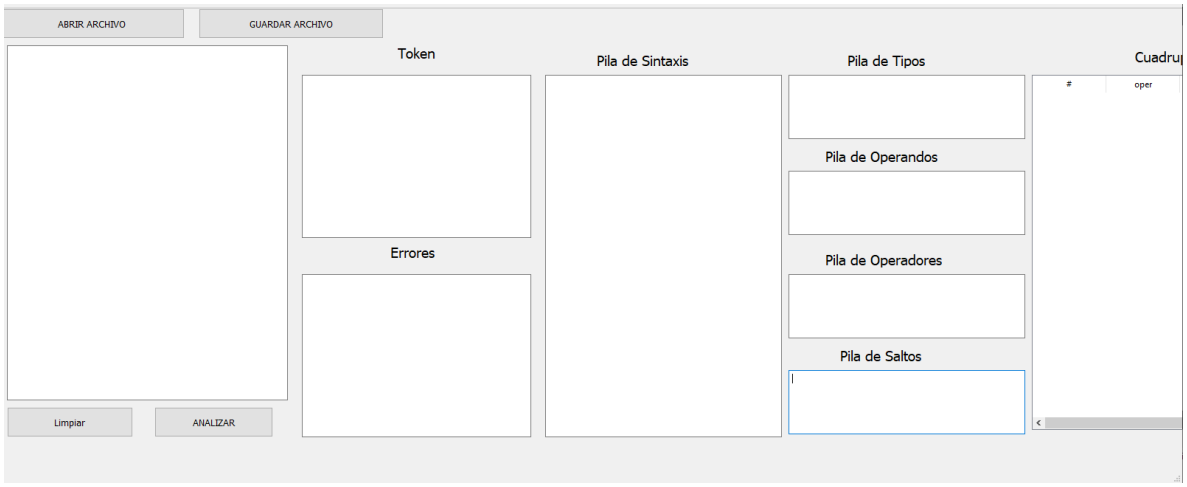
Dentro de este escrito se especifica el uso del sistema elaborado para todo aquel usuario que pretenda utilizarlo, con los fines que sean. Describiendo cada punto contenido en este; facilitando así el acceso, uso y manejo del mismo.

Objetivo General:

Crear un programa diseñado enteramente para la evaluación, sintáctica, léxica y semántica de un texto, así como su correspondiente generación de código para su análisis y comprensión.

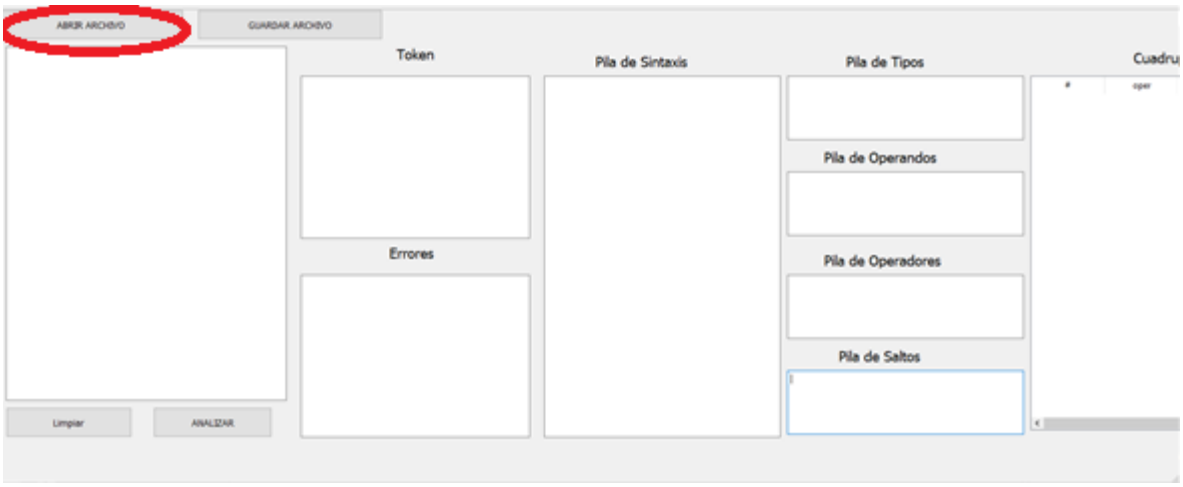
Como accesar al sistema:

Para accesar al sistema solo es necesario encontrarse en el único modulo grafico que se muestre en pantalla tal como este:



No se requieren acciones adicionales para su acceso más que correr el programa. En pantalla se muestran los siguientes atributos para que usted pueda identificar si se encuentra en el módulo correcto:

- Botón de acceso a archivos externos:



El botón de acceso a archivos funciona de la siguiente manera, al momento de dar clic se ejecuta la instrucción de la interfaz grafica que marca la ruta en la que se encuentra el archivo de la siguiente manera:

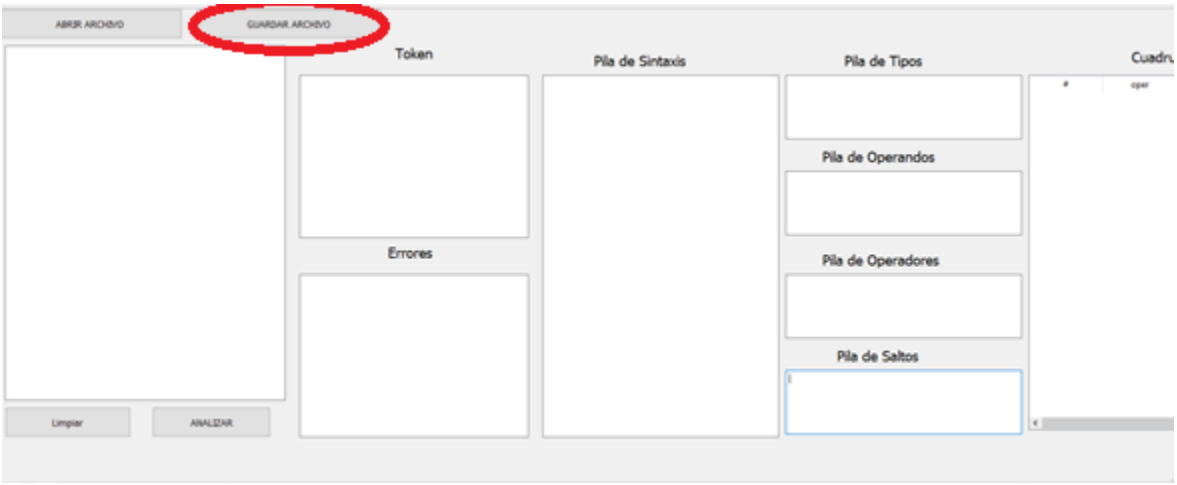
```
QString ruta=QFileDialog::getOpenFileName(
                this,tr("Abrir archivo"),
                "C://",
              "Eyement (*.eye)");
```

D esa manera se abre el archivo mediante:

```
if(ruta!=""){
    QFile inputFile(ruta);
    if (inputFile.open(QIODevice::ReadOnly))
    {
       QTextStream in(&inputFile);
       while (!in.atEnd())
       {
          QString line = in.readLine();
          ui->textoAnalizar->appendPlainText(line);
       }
         inputFile.close();
```

Y al mismo tiempo se guarda en su respectiva variable para ser agregado por medio de un método append al recuadro de análisis.

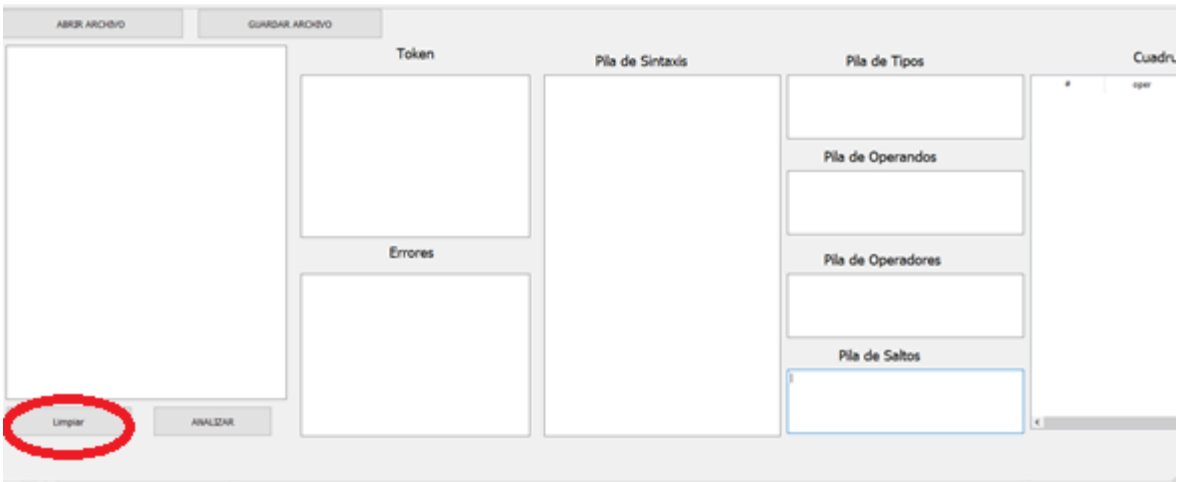
- Botón de guardado de archivos internos



El botón de guardar archivo funciona bajo mecánicas similares al botón de abrir archivo sin embargo en vez de buscar una ruta especifica para abrir el archivo, crea un archivo en la ruta y bajo el nombre que usted especifique

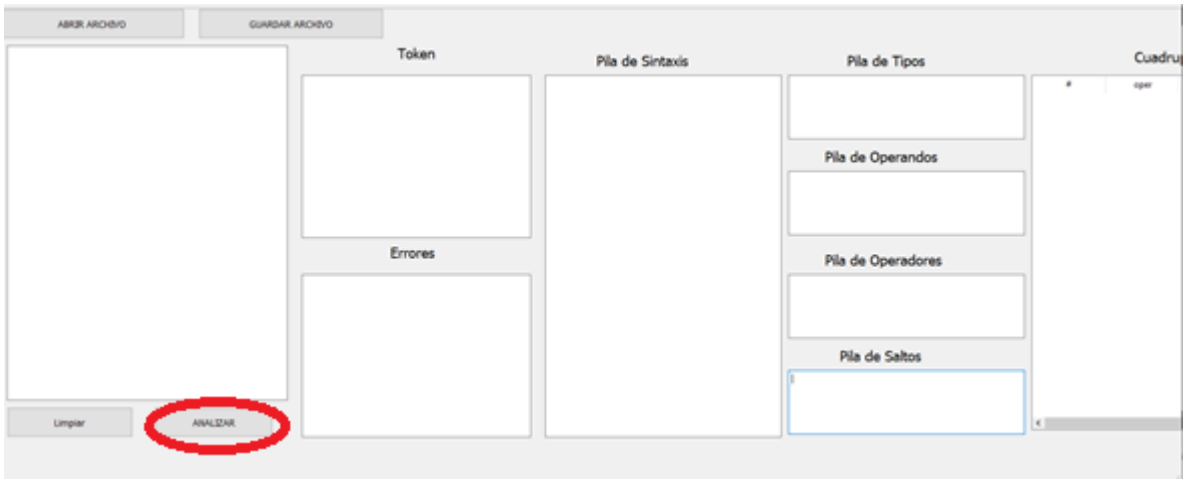
Al ser un analizador de texto lo mínimo que soporta son archivos de tamaño reducido como lo es 1 byte (sin ningún carácter guardado) hasta una cantidad que soporte tu sistema de almacenamiento.

- Botón de limpiar recuadro



El botón de limpiar funciona de tal manera que cualquier carácter guardado en la totalidad de los recuadros utilizados, sobrepone una plantilla en blanco de caracteres dejándolos todos en blanco.

- Botón de analizar



Dentro del botón analizar se encuentran en función el vaciado tanto de la tabla de cuádruplos, la pila y contador de cuádruplos, contadores de respuesta, tokens, errores, pasos de la pila, pila tanto de operandos, operadores y tokens y la puesta en blanco de los recuadros de textos.
Al termino de esta se procede a llamar a todos los métodos que constituyen la parte principal del programa y se imprimen en sus respectivos recuadros.

El botón de analizar en primera instancia depende del método

ConstruyeGramatica();
el cual consiste en declarar tanto las variables para el token, para el estado, columna y fila de la matriz de predicción.

Después se utiliza un while para confirmar que la pila de ejecución este vacía y si no está vacía se retira lo que este dentro.
En seguida se introduce un carácter de “$” en la pila de ejecución, que sirve para indicar el final del análisis sintáctico, también se inserta un “1” representando la producción numero 1 del analizador sintáctico.

Se estipula que las variables quieroToken y llena son verdaderas, que representa la necesidad del analizador sintáctico por un token y llena representa la necesidad de llenar una producción la cual se encuentra en tope de la pila.

Se imprime la pila y se evalúa si hubo algún error, si no se hará la siguiente acción siempre y cuando la pila de ejecución no esté vacía.

Primero se evalúa si el analizador requiere y un token el cual para saber si lo necesita se sustenta directamente con el método

Analiza();

Dentro de analiza se genera una variable de tipo vector llamada cadenaStd la cual va guardando el texto a analizar carácter por carácter dentro del vector.

Se inicializa la variable de estado en cero, se declara la variable correspondiente a la columna y se declara una variable char que guarde el carácter dependiendo de la posición en la que nos encontremos en el vector, y se inicializa la variable textoA en nulo.

Después de declarar una variable de tipo entero llamado numero para contar la posición correcta del vector, para en seguida de eso comenzar con un while que evalúe que la variable estado sea menos que treinta y dos, se inicializa la variable de la columna con el método:

Relaciona();

Dentro de este método se tomará como parámetro el carácter

Que estamos pidiendo en el método anterior y se declara variable entera llamada valor que se encargara de almacenar la variable que se obtenga de la columna en la matriz de predicción léxica. Después por medio de varios IF se evaluará el carácter que se envió y se comparara entre que valor de la columna de la matriz se encuentra.

Después de evaluar los IF entrara en un Switch que se encargara de evaluar el carácter que hayamos ingresado en la variable.

Una vez rellenada la variable de columna con el método anterior, se rellena la variable de estado con la matriz de predicción léxica donde la fila será igual a estado y la columna será igual a la variable columna.

Inmediatamente se harán comparaciones mediante un IF en donde el estado sea igual a un estado terminal, donde los estados mayores o iguales a 100 son terminales.

Una vez evaluados los estados terminales, se evaluará si se trata de operadores numéricos, palabras reservadas, operadores relacionales, comentarios, librerías o signos de agrupación.

Si el estado es igual a 100 se iniciará un método llamado

evaluaPR();

Dentro de este método se declara una variable contador-inicializada en cero, de inmediato se rellena la variable vector cadenaStd con el textoA convertido a string.

Después se inicializa un for donde se rellenará la variable de carácter con la cadenaStd anteriormente rellenada con el texto plano que se encontraba en el recuadro de analizar. Y ahí mismo dentro del for se evaluarán espacios, enters y tabulaciones para aumentar el contador declarado al principio del método.

Se declarará una variable de tipo String llamada temp donde al texto plano guardado dentro de la variable textoA por medio del método .mid (método del compilador) se encargará de eliminar cualquier espacio, enter o tabulador que se haya registrado.

De inmediato se evaluará por medio de IF's todas las palabras reservadas posibles retornando su número equivalente dentro de la matriz de predicción léxica.(el estado 101 específicamente representa que se recibió un identificador)

```
if(textoA=="import" || temp=="import"){
        return 135;
    }
    if(textoA=="class" || temp=="class"){
        return 134;
    }
    if(textoA=="begin" || temp=="begin"){
        return 133;
    }
    if(textoA=="end" || temp=="end"){
        return 132;
    }
    if(textoA=="def" || temp=="def"){
        return 136;
    }
    if(textoA=="as" || temp=="as"){
        return 137;
    }
    if(textoA=="integer" || temp=="integer"){
        return 138;
    }
    if(textoA=="float" || temp=="float"){
        return 139;
    }
    if(textoA=="char" || temp=="char"){
        return 140;
    }
    if(textoA=="string" || temp=="string"){
        return 141;
    }
    if(textoA=="boolean" || temp=="boolean"){
        return 142;
    }
    if(textoA=="if" || temp=="if"){
        return 143;
    }
    if(textoA=="endif" || temp=="endif"){
        return 144;
    }
    if(textoA=="else" || temp=="else"){
        return 145;
    }
    if(textoA=="endwhile" || temp=="endwhile"){
        return 146;
    }
    if(textoA=="while" || temp=="while"){
        return 147;
    }
    if(textoA=="endfor" || temp=="endfor"){
        return 148;
    }
    if(textoA=="for" || temp=="for"){
        return 149;
    }
    if(textoA=="enter" || temp=="enter"){
        return 150;
    }
```

```
if(textoA=="write" || temp=="write"){
    return 151;
}
if(textoA=="read" || temp=="read"){
    return 152;
}
if(textoA=="principal" || temp=="principal"){
    return 153;
}
if(textoA=="elseif" || temp=="elseif"){
    return 154;
}
if(textoA=="do" || temp=="do"){
    return 155;
}
if(textoA=="function" || temp=="function"){
    return 156;
}
if(textoA=="endfunction" || temp=="endfunction"){
    return 157;
}
if(textoA=="null" || temp=="null"){
    return 158;
}
if(textoA=="include" || temp=="include"){
    return 159;
}
return 101;
```

De regreso al método analiza(); después de terminar todas las evaluaciones la variable de carácter se rellenará con el vector cadenaStd en la dirección del vector en la que se haya quedado, para después aumentar el contador numero en 1 terminando el while.

Después se validará el texto plano dentro de textoA si está bien escrito el “&&” o “||” , a continuación, se utiliza la misma herramienta del for para eliminar espacios, tabuladores y enters.

Se evalúa el estado que quedo al final el cual fue obtenido de la matriz de predicción léxica, si el estado es mayor o igual a 100 y menor o igual a 199 se ira al método de

Token();

Evalúa el estado que fue mandado y concatena en la variable de impresión de tokens el token valido.

```
switch(e){
    case 100:
        //cout<<"Palabra reservada"<<endl;
        Tokens+=textoA+" Palabra reservada\n";
        break;
    case 101:
```

```
            //cout<<"Identificador"<<endl;
            Tokens+=textoA+" Identificador\n";
            break;
        case 102:
            //cout<<"Entero"<<endl;
            Tokens+=textoA+" Constante Entera\n";
            break;
        case 103:
            //cout<<"Constante decimal"<<endl;
            Tokens+=textoA+" Constante decimal\n";
            break;
        case 104:
            //cout<<"Constante con notación
cientifica"<<endl;
            Tokens+=textoA+" Constante con notacion
cientifica\n";
            break;
        case 105:
            //cout<<"Constante caracter"<<endl;
            Tokens+=textoA+" Constante caracter\n";
            break;
        case 106:
            //cout<<"Constante String"<<endl;
            Tokens+=textoA+" Constante String\n";
            break;
        case 107:
            //cout<<"Suma"<<endl;
            Tokens+=textoA+" Suma\n";
            break;
        case 108:
            //cout<<"Resta"<<endl;
            Tokens+=textoA+" Resta\n";
            break;
        case 109:
            //cout<<"Multiplicación"<<endl;
            Tokens+=textoA+" Multiplicacion\n";
            break;
        case 110:
            //cout<<"División"<<endl;
            Tokens+=textoA+" Division\n";
            break;
        case 111:
            //cout<<"Modulo"<<endl;
            Tokens+=textoA+" Modulo\n";
            break;
        case 112:
            //cout<<"Comentario"<<endl;
            Tokens+=textoA+" Comentario\n";
            break;
        case 113:
            //cout<<"Operador Y"<<endl;
            Tokens+=textoA+" Operador Y\n";
            break;
        case 114:
            //cout<<"Operador O"<<endl;
            Tokens+=textoA+" Operador O\n";
            break;
```

```
                case 115:
                    //cout<<"Operador NOT"<<endl;
                    Tokens+=textoA+" Operador NOT\n";
                    break;
                case 116:
                    //cout<<"Operador Diferente"<<endl;
                    Tokens+=textoA+" Operador Diferente\n";
                    break;
                case 117:
                    //cout<<"Igual"<<endl;
                    Tokens+=textoA+" Operador Igual\n";
                    break;
                case 118:
                    //cout<<"Mayor"<<endl;
                    Tokens+=textoA+" Operador Mayor\n";
                    break;
                case 119:
                    //cout<<"Mayor Igual"<<endl;
                    Tokens+=textoA+" Operador Mayor igual\n";
                    break;
                case 120:
                    //cout<<"Menor"<<endl;
                    Tokens+=textoA+" Operador Menor\n";
                    break;
                case 121:
                    //cout<<"Menor Igual"<<endl;
                    Tokens+=textoA+" Operador Menor Igual\n";
                    break;
                case 122:
                    //cout<<"Asignación"<<endl;
                    Tokens+=textoA+" Operador de
            Asignacion\n";
                    break;
                case 123:
                    //cout<<"Signo dos puntos"<<endl;
                    Tokens+=textoA+" Signo dos puntos\n";
                    break;
                case 124:
                    //cout<<"Signo de cierre"<<endl;
                    Tokens+=textoA+" Signo de cierre ;\n";
                    break;
                case 125:
                    //cout<<"Punto"<<endl;
                    Tokens+=textoA+" Punto\n";
                    break;
                case 126:
                    //cout<<"Parentesis abierto"<<endl;
                    Tokens+=textoA+" Parentesis abierto\n";
                    break;
                case 127:
                    //cout<<"Parentesis cerrado"<<endl;
                    Tokens+=textoA+" Paretesis cerrado\n";
                    break;
                case 128:
                    //cout<<"Corchete abierto"<<endl;
                    Tokens+=textoA+" Corchete abierto\n";
                    break;
```

```
    case 129:
        //cout<<"Corchete cerrado"<<endl;
        Tokens+=textoA+" Corchete cerrado\n";
        break;
    case 130:
        //cout<<"Es una coma"<<endl;
        Tokens+=textoA+" Es una coma\n";
        break;
    case 131:
        //cout<<#Es una libreria"<<endl;
        Tokens+=textoA+" Es una libreria\n";
        break;
    case 132:
        Tokens+=textoA+" Palabra reservada end\n";
        break;
    case 133:
        Tokens+=textoA+" Palabra reservada
begin\n";
        break;
    case 134:
        Tokens+=textoA+" Palabra reservada
class\n";
        break;
    case 135:
        Tokens+=textoA+" Palabra reservada
import\n";
        break;
    case 136:
        Tokens+=textoA+" Palabra reservada def\n";
        break;
    case 137:
        Tokens+=textoA+" Palabra reservada as\n";
        break;
    case 138:
        Tokens+=textoA+" Palabra reservada
integer\n";
        break;
    case 139:
        Tokens+=textoA+" Palabra reservada
float\n";
        break;
    case 140:
        Tokens+=textoA+" Palabra reservada
char\n";
        break;
    case 141:
        Tokens+=textoA+" Palabra reservada
string\n";
        break;
    case 142:
        Tokens+=textoA+" Palabra reservada
boolean\n";
        break;
    case 143:
        Tokens+=textoA+" Palabra reservada if\n";
        break;
    case 144:
```

```
            Tokens+=textoA+" Palabra reservada
endif\n";
            break;
        case 145:
            Tokens+=textoA+" Palabra reservada else";
            break;
        case 146:
            Tokens+=textoA+" Palabra reservada
endwhile\n";
            break;
        case 147:
            Tokens+=textoA+" Palabra reservada
while\n";
            break;
        case 148:
            Tokens+=textoA+" Palabra reservada
endfor\n";
            break;
        case 149:
            Tokens+=textoA+" Palabra reservada for\n";
            break;
        case 150:
            Tokens+=textoA+" Palabra reservada
enter\n";
            break;
        case 151:
            Tokens+=textoA+" Palabra reservada
write\n";
            break;
        case 152:
            Tokens+=textoA+" Palabra reservada
read\n";
            break;
        case 153:
            Tokens+=textoA+" Palabra reservada
principal\n";
            break;
        case 154:
            Tokens+=textoA+" Palabra reservada
elseif\n";
            break;
        case 155:
            Tokens+=textoA+" Palabra reservada do\n";
            break;
        case 156:
            Tokens+=textoA+" Palabra reservada
function\n";
            break;
        case 157:
            Tokens+=textoA+" Palabra reservada
endfunction\n";
            break;
        case 158:
            Tokens+=textoA+" Palabra reservada
null\n";
            break;
        case 159:
```

```
        Tokens+=textoA+" Palabra reservada
include\n";
        break;
                        }
```

Si el estado es diferente de la condición del IF que se expuso anteriormente se mandara a llamar el método

Errores();

Este metodo recibe el estado que se mando con anterioridad y se evaluara mediante un switch y a la vez se concatenara en la variable de impresión de errores el error.

```
switch(e){
    case 500:
        //cout<<"Error este simbolo no forma parte
del lenguaje"<<endl;
        errores+=textoA+" Error este simbolo no
forma parte del lenguaje\n";
        break;
    case 501:
        //cout<<"Error no puede declarar un
identificador de esa manera"<<endl;
        errores+=textoA+" Error no puede declarar
un identificador de esa manera\n";
        break;
    case 502:
        //cout<<"Error no puede escribir una
constante usando letras"<<endl;
        errores+=textoA+" Error no puede escribir
una constante usando letras\n";
        break;
    case 503:
        //cout<<"Error debe de escribir un numero
o un signo positivo o negativo despues de la E o
e"<<endl;
        errores+=textoA+" Error debe de escribir
un numero o un signo positivo o negativo despues
de la E o e\n";
        break;
    case 504:
        //cout<<"Error debe de escribir un numero
para completar la expresión"<<endl;
        errores+=textoA+" Error debe de escribir
un numero para completar la expresion\n";
        break;
    case 505:
        //cout<<"Error debe de escribir algo entre
las '"<<endl;
```

```
            errores+=textoA+" Error debe escribir un
caracter entre las '\n";
            break;
        case 506:
            //cout<<"Error indefinido"<<endl;
            errores+=textoA+" Error no cerro la '\n";
            break;
        case 507:
            //cout<<"No puede escribir algo entre la
expresión"<<endl;
            errores+=textoA+" No puede escribir algo
entre la expresion Y\n";
            break;
        case 508:
            //cout<<"No puede escribir algo entre la
expresión"<<endl;
            errores+=textoA+" No puede escribir algo
entre la expresion O\n";
            break;
        case 509:
            //cout<<"No puedes llamar a una libreria
de esa manera
            errores+=textoA+" No puede declara una
libreria de esa manera\n";
        case 510:
            errores+=textoA+" 510: Se esperaba la
palabra class o import al inicio\n";
            break;
        case 511:
            errores+=textoA+" 511: Se esperaba la
palabra class\n";
            break;
        case 512:
            errores+=textoA+" 512: Se esperaba la
palabra import o class\n";
            break;
        case 513:
            errores+=textoA+" 513: Se esperaba un
identificador, la palabra def o el inicio de un
estatuto\n";
            break;
        case 514:
            errores+=textoA+" 514: Se esperaba un
identificador\n";
            break;
        case 515:
            errores+=textoA+" 515: Se esperaba la
palabra as o ','\n";
            break;
        case 516:
            errores+=textoA+" 516: Se esperaba un tipo
de dato\n";
            break;
        case 517:
            errores+=textoA+" 517: Se esperaba un
identificador o el  inicio o bien de un
estatuto\n";
```

```
            break;
        case 518:
            errores+=textoA+" 518: Se esperaba la
palabra if\n";
            break;
        case 519:
            errores+=textoA+" 519: Se esperaba la
palabra else o endif\n";
            break;
        case 520:
            errores+=textoA+" 520: Se esperaba la
palabra while\n";
            break;
        case 521:
            errores+=textoA+" 521: Se esperaba la
palabra for\n";
            break;
        case 522:
            errores+=textoA+" 522: La expresión no
puede iniciar de esa manera\n";
            break;
        case 523:
            errores+=textoA+" 523: Se esperaba un
parentesis cerrado, el operador || o un signo de
puntuación\n";
            break;
        case 524:
            errores+=textoA+" 524: La expresión no
puede iniciar de esa manera\n";
            break;
        case 525:
            errores+=textoA+" 525: Se esperaba un
parentesis cerrado, el operador || o un signo de
puntuación\n";
            break;
        case 526:
            errores+=textoA+" 526: La expresión no
puede iniciar de esa manera\n";
            break;
        case 527:
            errores+=textoA+" 527: La expresión no
puede iniciar de esa manera\n";
            break;
        case 528:
            errores+=textoA+" 528: La expresión no
puede iniciar de esa manera\n";
            break;
        case 529:
            errores+=textoA+" 529: Se esperaba un
operador relacional o un signo de puntuación\n";
            break;
        case 530:
            errores+=textoA+" 530: La expresión no
puede iniciar de esa forma\n";
            break;
        case 531:
```

```
            errores+=textoA+" 531: Se esperaba un
parentesis cerrado, un operador o un signo de
puntuación\n";
            break;
        case 532:
            errores+=textoA+" 532: La expresión no
puede iniciar de esa forma\n";
            break;
        case 533:
            errores+=textoA+" 533: Se esperaba un
parentesis cerrado, un operador o un signo de
puntuación\n";
            break;
        case 534:
            errores+=textoA+" 534: Se esperaba un tipo
de constante\n";
            break;
        case 535:
            errores+=textoA+" 535: Se esperaba un
operador relacional\n";
            break;
        case 536:
            errores+=textoA+" 536: Se esperaba la
palabra enter\n";
            break;
        case 537:
            errores+=textoA+" 537: Se esperaba un
identificador\n";
            break;
        case 538:
            errores+=textoA+" 538: Se esperaba la
palabra write\n";
            break;
        case 539:
            errores+=textoA+" 539: La expresión no
puede iniciar de esa forma\n";
            break;
        case 540:
            errores+=textoA+" 540: Se esperaba un
identificador o una coma\n";
            break;
        case 541:
            errores+=textoA+" 541: Se esperaba la
palabra read\n";
            break;
        case 542:
            errores+=textoA+" 542: Se esperaba un
identificador\n";
            break;
        case 543:
            errores+=textoA+" 543: Se esperaba un
identificador o una coma\n";
            break;
        case 544:
            errores+="544: Error entre tipos\n";
            break;
        case 545:
```

```
            errores+=textoA+" 545: Variable no
declarada\n";
            break;
        default:
            errores+="Revisa tu sintaxis quizas
pusiste algo que no debe ir\n";
            break;
        }

                                    }
```

De vuelta al estado analiza solo resta retornar el estado actual.

De regreso al método ConstruyeGramatica(); el token será rellenado con lo que se obtuvo del método Analiza(); para después cambiar la variable quiero token a una condición de falso.

Ahora, dentro del IF si llena es igual a verdadero la variable equivalente a la columna de la matriz de predicción se rellenará con el método

RelacionaGramatica();

Primero se envía el token que se recibió anteriormente y mediante el uso de switch se evalúa el estado dependiendo de la columna de la matriz de predicción sintáctica y retorna la posición de la columna.

En la variable filaMP se rellena con el valor de lo que se encuentra en el tope de la pila de ejecución y se le resta 1 y en seguida se procede a eliminar lo que está en el tope de la pila en ese momento. Una vez teniendo los valores colMP y fila MP se rellena edoMP con la que se encuentre en la matriz de predicción sintáctica. En la fila, filaMP y en la columna, colMP. Después se llamará al método

LlenarPilaProducción();

Dentro de este método se introduce como parámetro la fila del método anterior, donde por medio de un for se aumentará el contador “i”, el cual dentro de la función if encontrada dentro del for evaluará las producciones donde la fila es equivalente al parámetro fila y la columna es equivalente a la variable “i” del for.

La variable llena cambiará su valor de condición a falso y se imprimirá la pila resultante con el método

ImprimePila();

En este ciclo se tomará el tamaño de la pila y dentro del ciclo se asignará al a variable String “ele” lo resultante de llamar al método evaluaElemento(); mandando como parámetro el valor actual de “i”

evaluaElemento();

Dentro de este método se tomará como parámetro la variable que se le envía a “ele”, donde se evaluara por medio de un if si el elemento es mayor o igual a 700 donde el 700 representa las acciones semánticas o de código intermedio dando como valor retornado un string nulo.

Después, dependiendo de lo que resulte, el elemento por medio de un switch evalúa el elemento que se envié, este puede evaluar producciones, vacíos y marcas de fondo falso.

```
if(elemento>=700){
        return "";
    }
    switch(elemento){
    case -1:
        return "ε";
    case -2:
        return "MFF";
    case 1:
        return "PROGRAM";
        break;
    case 2:
        return "A";
        break;
    case 3:
        return "DECLARA-LIB";
        break;
    case 4:
        return "DECLARA";
        break;
    case 5:
        return "B";
        break;
    case 6:
        return "C";
        break;
    case 7:
        return "TIPO";
        break;
    case 8:
        return "ESTATUTOS";
        break;
    case 9:
        return "EST_IF";
```

```
            break;
        case 10:
            return "D";
            break;
        case 11:
            return "EST_WHILE";
            break;
        case 12:
            return "EST_FOR";
            break;
        case 13:
            return "EXPR";
            break;
        case 14:
            return "E";
            break;
        case 15:
            return "EXPR2";
            break;
        case 16:
            return "F";
            break;
        case 17:
            return "EXPR3";
            break;
        case 18:
            return "G";
            break;
        case 19:
            return "EXPR4";
            break;
        case 20:
            return "H";
            break;
        case 21:
            return "EXPR5";
            break;
        case 22:
            return "I";
            break;
        case 23:
            return "TERM";
            break;
        case 24:
            return "J";
            break;
        case 25:
            return "FACT";
            break;
        case 26:
            return "OPREL";
            break;
        case 27:
            return "EST_ENTER";
            break;
        case 28:
            return "EST_ASIG";
```

```
            break;
        case 29:
            return "EST_WRITE";
            break;
        case 30:
            return "K";
            break;
        case 31:
            return "L";
            break;
        case 32:
            return "EST_READ";
            break;
        case 33:
            return "M";
            break;
        case 34:
            return "N";
            break;
        case 36:
            return "$";
            break;
        case 100:
            //cout<<"Palabra reservada"<<endl;

            break;
        case 101:
            //cout<<"Identificador"<<endl;
            return "Identificador";
            break;
        case 102:
            //cout<<"Entero"<<endl;
            return "Cteentera";
            break;
        case 103:
            //cout<<"Constante decimal"<<endl;
            return "Ctereal";
            break;
        case 104:
            //cout<<"Constante con notación
cientifica"<<endl;
            return "Ctenotacion cientifica";
            break;
        case 105:
            //cout<<"Constante caracter"<<endl;
            return "CteCaracter";
            break;
        case 106:
            //cout<<"Constante String"<<endl;
            return "CteString";
            break;
        case 107:
            //cout<<"Suma"<<endl;
            return "+";
            break;
        case 108:
            //cout<<"Resta"<<endl;
```

```
        return "-";
        break;
    case 109:
        //cout<<"Multiplicación"<<endl;
        return "*";
        break;
    case 110:
        //cout<<"División"<<endl;
        return "/";
        break;
    case 111:
        //cout<<"Modulo"<<endl;
        return "%";
        break;
    case 112:
        //cout<<"Comentario"<<endl;

        break;
    case 113:
        //cout<<"Operador Y"<<endl;
        return "&&";
        break;
    case 114:
        //cout<<"Operador O"<<endl;
        return "||";
        break;
    case 115:
        //cout<<"Operador NOT"<<endl;
        return "!";
        break;
    case 116:
        //cout<<"Operador Diferente"<<endl;
        return "!=";
        break;
    case 117:
        //cout<<"Igual"<<endl;
        return "==";
        break;
    case 118:
        //cout<<"Mayor"<<endl;
        return ">";
        break;
    case 119:
        //cout<<"Mayor Igual"<<endl;
        return ">=";
        break;
    case 120:
        //cout<<"Menor"<<endl;
        return "<";
        break;
    case 121:
        //cout<<"Menor Igual"<<endl;
        return "<=";
        break;
    case 122:
        //cout<<"Asignación"<<endl;
        return "=";
```

```
    break;
case 123:
    //cout<<"Signo dos puntos"<<endl;
    return ":";
    break;
case 124:
    //cout<<"Signo de cierre"<<endl;
    return ";";
    break;
case 125:
    //cout<<"Punto"<<endl;

    break;
case 126:
    //cout<<"Parentesis abierto"<<endl;
    return "(";
    break;
case 127:
    //cout<<"Par"<<endl;
    return ")";
    break;
case 128:
    //cout<<"Corchete abierto"<<endl;

    break;
case 129:
    //cout<<"Corchete cerrado"<<endl;

    break;
case 130:
    //cout<<"Es una coma"<<endl;
    return ",";
    break;
case 131:
    //cout<<#Es una libreria"<<endl;
    return "Identifiacdorlibreria";
    break;
case 132:
    return "end";
    break;
case 133:
    return "begin";
    break;
case 134:
    return "class";
    break;
case 135:
    return "import";
    break;
case 136:
    return "def";
    break;
case 137:
    return "as";
    break;
case 138:
    return "integer";
```

```
        break;
    case 139:
        return "float";
        break;
    case 140:
        return "char";
        break;
    case 141:
        return "string";
        break;
    case 142:
        return "boolean";
        break;
    case 143:
        return "if";
        break;
    case 144:
        return "endif";
        break;
    case 145:
        return "else";
        break;
    case 146:
        return "endwhile";
        break;
    case 147:
        return "while";
        break;
    case 148:
        return "endfor";
        break;
    case 149:
        return "for";
        break;
    case 150:
        return "enter";
        break;
    case 151:
        return "write";
        break;
    case 152:
        return "read";
        break;
    default:
        return "desconocido";
        }
```

Devuelta al método imprimePila(); la variable pasosPila concatenara lo que hay dentro de la pila de ejecución sintáctica.

En seguida después del for concatenara 2 enters y nos regresamos al método anterior.

Devuelta al método ConstruyeGramatica(); se evaluara por medio de IF's si el tope de la pila es mayor o igual a 100 donde 100 representa un estado de aceptación, si es así también evaluara si en el tope de la pila hay un token donde si el token existe y es igual a él, la pila de ejecución eliminara lo que haya en el tope de la pila y dentro de sí mismo evaluara si el texto es diferente a nulo donde si se cumple la condición cambiara el estado de "quieroToken" a verdadero.

En seguida de hacer la anterior evaluación se mandará a llamar al método que imprime la pila llamado ImprimePila(); para después ejecutar otra evaluación IF en la cual la condición indica que si el tope de la pila es mayor o igual al estado de aceptación 700 y al mismo tiempo es menor a 800 se mandara a llamar a un método con el parámetro que es el tope de la pila de ejecución el cual es

accionesSemanticayCodigoIntermedio();

Se comienza el método declarando una variable booleana la cual se llamará "existeOperando" inicializada en falso, la cual se encarga de verificar sobre que la pila de operandos no sea "null", Se declaran 4 variables las cuales pertenecer al código intermedio como lo son el operando, op1,op1 y una respuesta (que servirán para formar los cuádruplos).

Después se inicializa un switch con los parámetros del método en el cual si el case es igual a 700, se hace un push del texto plano dentro de la pila de operandos y otra en la pila de operandos "búsqueda" la cual se encarga de corregir la duplicidad en la generación de código, por medio de un FOR se recorre la pila de operandos búsqueda y se evalúa si ese operando ya existía anteriormente.

Si el operando ya existía se ejecuta en seguida un IF que lo evalúa, poniendo la variable que se encarga de la existencia de errores dentro de este método en falso, sacando lo que se encuentra en el tope de

pila de operandos y al mismo tiempo sacándolo también de la pila de operandos búsqueda, para enseguida concatenar en la variable de errores el texto plano y su debida advertencia sobre que “la variable ya esta definida” para que al final devuelva la variable de existencia de operandos a falso.

Se imprime la pila de operandos con su debido método y se sale del case.

En caso de que el case sea igual a 701 evaluara si la pila de operandos esta llena o con algún carácter dentro para realizar la siguiente operación mientras que la pila este llena, la cual seria introducir el estado dentro de la pila de tipos búsqueda la cual va insertando el tipo de datos de operandos y a la par de esto va a ir eliminando los operandos de la pila de operandos, se imprime la pila y rompe el case.

En caso de que el case sea igual a 702, se evaluara si el estado corresponde al numero de tipo mayor o igual a 102 y menor o igual a 106 los cuales corresponden a constantes enteras, flotantes, booleanas, etc. Y así evaluara con un IF anidado al estado por separado, agregando el respectivo tipo de constante a la pila de tipos e imprimiendo la pila con el método

imprimePilaTipos();

El cual funciona igual que imprimePilaOperandos(); pero con las variables orientadas a tipos.

Si no se cumple la condición se inicializará una variable “i” rellenada con lo que nos dé de resultado el método

buscaTipo();

Busca tipo se encarga de generar una variable de posición en “-1” la cual por si misma significa que no se ha encontrado nada, con un IF se evalúa si la pila de operandos búsqueda se

encuentra llena, si es así, se genera un for que recorre la pila entera evaluando por segunda ocasión en un IF anidado si el texto plano contiene el elemento de la pila en la posición “i” si es así la variable de posición se rellena con el valor de “i”.

Regresando al método de generación de código, dependiendo que nos retorne el método buscaTipo(); evaluaremos si la variable i permaneció en “-1”

Si es así se llama al método Errores(); dándole como parámetro el numero 545 equivalente a un error (variable no definida), se agrega a la pila de tipos el numero 138 (integer) y se estipula la variable de control de errores en falso, si la “i” es diferente a “-1” se agregará a la pila de tipos lo que se encuentre en la pila de tipos búsqueda en la posición “i” para después imprimir la pila de tipos.

Después de evaluar los anteriores IF se insertará en la pila de operandos lo que contenga el texto plano.

Se inicializa un IF en el cual evalúa por medio de variables globales si se encuentra algún READ o algún WRITE, donde si es así, se imprimirá la pila de operandos y se aumentará en 1 el contadorRAW que se encarga de contar los identificadores, constantes, etc. Que se encuentren dentro del READ o WRITE.

Se evalúa el case 703 donde se agrega a la pila de operadores lo que este en el estado actual y después se imprime la pila.

Se evalúa el 704, que manda a llamar al método

relacionaTiposOper(); (operadores)

Declara dos variables de tipo entero a las cuales se le va a asignar el tope de la pila de tipos y el tope -1 de la pila de tipos. En seguida entra un IF el cual evalúa si el tope de la pila de

operadores es mayor o igual a 107 y además menor o igual a 111 (+,-,*). Y es así se inicializa una variable de tipo entero llamada fila en 0. Después inicia un ciclo FOR iniciando desde 0 hasta 25, en este se hace un IF que valida que la matriz de tipos, tanto en su columna como en su fila, validando si es igual al operador uno y al operador dos, de ser así, la variable fila tomará el valor de "i", enseguida se declarará una variable llamada columna, la cual se rellenará con el método

relacionaMatrizTipos();

Se recibe como parámetro el estado, en el cual por medio de 5 IF's se evaluarán los estados desde el 107 al 115 retornando como valor en cada uno su respectivo valor en las columnas de la matriz de tipos.

Se declara otra variable llamada supuesto que se rellenará con lo que se encuentre en la matriz de tipos dependiendo de la fila y la columna que le indique el programa, enseguida se genera un IF en el cual se evalúa si el supuesto es menor a 500 (error) y se llamará al método

impimeYlimpiaPilas();

Se elimina el tope de la pila de tipos y se imprime dos veces.

Se agrega en la pila de tipos el valor de supuestos, se crea una variable Sting llamada oper que se rellenará con el método evaluaElemento(); con el parámetro que se encuentre en el tope de la pila de operadores, se elimina el tope de pila de operadores, se imprime la pila de operadores, se crea otra variable de tipo String llamada op2 que se rellena con el tope de la pila de operandos y se elimina el tope de la pila de

operandos para después imprimirlo y por último se crea una variable de tipo string llamada res, la cual concatena la letra “R” con una variable de tipo string que contiene el contador de las respuestas del cuádruplo. Luego se agrega en la pila de operandos lo que se encuentre en la variable res, para después imprimir en la pila de operandos.

En seguida se manda a llamar un método con parámetros que es el contador de cuádruplos, oper, op1, op2 y res, el cual se llama

formaCuadruplo();

Se crea una dirección de memoria, la cual va contener los atributos n, oper, op1, op2, res, y enseguida se insertará esa dirección de memoria en la lista enlazada “cuádruplos”.

Si la variable supuesta fue mayor a 500 lo que se hace es llamar al método Errores(); con el número 544 (error entre tipos). Al supuesto se le asignará op1. En seguida se eliminará el tope de la pila de tipos, se imprimirá la pila de tipos con el método imprimePilaTipos(); nuevamente vuelve a eliminar el tope de la pila de tipos y vuelve a imprimir la pila de tipos con el método de imprimePilaTipos(); se inserta el supuesto en la pila de tipos y se imprime en la pila de tipos con el método imprimePilaTipos(); se crea una variable Sting llamada oper que se rellenará con el método evaluaElemento(); con el parámetro que se encuentre en el tope de la pila de operadores, se elimina el tope de pila de operadores, se imprime la pila de operadores, se crea otra variable de tipo String llamada op2 que se rellena con el tope de la pila de operandos y se elimina el tope de la pila de operandos para después imprimirlo y por último se crea una variable de tipo string llamada res, la cual concatena la letra

“r” con una variable de tipo string que contiene el contador de las respuestas del cuádruplo. Luego se agrega en la pila de operandos lo que se encuentre en la variable res, para después imprimir en la pila de operandos.

En seguida se manda a llamar un método con parámetros que es el contador de cuádruplos, oper, op1, op2 y res, el cual se llama formaCuadruplo(); sinError pasa a ser falso pasando a ELSE IF el cual evalúa si el tope de la pila de operadores es mayor o igual a 116 y menor o igual a 121 lo cual es operadores relacionales. De ser así se inicializa una variable de tipo entero llamada fila en 0. Después inicia un ciclo FOR iniciando desde 0 hasta 25, en este se hace un IF que valida que la matriz de tipos, tanto en su columna como en su fila, validando si es igual al operador uno y al operador dos, de ser así, la variable fila tomará el valor de “i”, enseguida se declarará una variable llamada columna, la cual se rellenará con el método relacionaTipoMatriz(); Se declara otra variable llamada supuesto que se rellenará con lo que se encuentre en la matriz de tipos dependiendo de la fila y la columna que le indique el programa, enseguida se genera un IF en el cual se evalúa si el supuesto es menor a 500 (error) y se llamará al método Se agrega en la pila de tipos el valor de supuestos, se crea una variable Sting llamada oper que se rellenará con el método evaluaElemento(); con el parámetro que se encuentre en el tope de la pila de operadores, se elimina el tope de pila de operadores, se imprime la pila de operadores, se crea otra variable de tipo String llamada op2 que se rellena con el tope de la pila de operandos y se elimina el tope de la pila de operandos para después imprimirlo y por último se crea una variable de tipo string llamada res, la cual concatena la letra “r” con una variable de tipo string que contiene el contador de las

respuestas del cuádruplo. Luego se agrega en la pila de operandos lo que se encuentre en la variable res, para después imprimir en la pila de operandos.

En seguida se manda a llamar un método con parámetros que es el contador de cuádruplos, oper, op1, op2 y res, el cual se llama formaCuadruplo(); Si la variable supuesto fue mayor a 500 lo que se hace es llamar al método Errores(); con el número 544 (error entre tipos). Al supuesto se le asignará op1. En seguida se eliminará el tope de la pila de tipos, se imprimirá la pila de tipos con el método imprimePilaTipos(); nuevamente vuelve a eliminar el tope de la pila de tipos y vuelve a imprimir la pila de tipos con el método de imprimePilaTipos(); se inserta el supuesto en la pila de tipos y se imprime en la pila de tipos con el método imprimePilaTipos(); se crea una variable Sting llamada oper que se rellenará con el método evaluaElemento(); con el parámetro que se encuentre en el tope de la pila de operadores, se elimina el tope de pila de operadores, se imprime la pila de operadores, se crea otra variable de tipo String llamada op2 que se rellena con el tope de la pila de operandos y se elimina el tope de la pila de operandos para después imprimirlo y por último se crea una variable de tipo string llamada res, la cual concatena la letra “r” con una variable de tipo string que contiene el contador de las respuestas del cuádruplo. Luego se agrega en la pila de operandos lo que se encuentre en la variable res, para después imprimir en la pila de operandos.

En seguida se manda a llamar un método con parámetros que es el contador de cuádruplos, oper, op1, op2 y res, el cual se llama formaCuadruplo(); Se pasa al ELSE IF y evalúa que el tope de la pila de operadores sea mayor o igual a 113 y menor

o igual a 115. De ser así se inicializa una variable de tipo entero llamada fila en 0. Después inicia un ciclo FOR iniciando desde 0 hasta 25, en este se hace un IF que valida que la matriz de tipos, tanto en su columna como en su fila, validando si es igual al operador uno y al operador dos, de ser así, la variable fila tomará el valor de “i”, enseguida se declarará una variable llamada columna, la cual se rellenará con el método relacionaTipoMatriz(); Se declara otra variable llamada supuesto que se rellenará con lo que se encuentre en la matriz de tipos dependiendo de la fila y la columna que le indique el programa, enseguida se genera un IF en el cual se evalúa si el supuesto es menor a 500 (error) y se llamará al método Se agrega en la pila de tipos el valor de supuestos, se crea una variable Sting llamada oper que se rellenará con el método evaluaElemento(); con el parámetro que se encuentre en el tope de la pila de operadores, se elimina el tope de pila de operadores, se imprime la pila de operadores, se crea otra variable de tipo String llamada op2 que se rellena con el tope de la pila de operandos y se elimina el tope de la pila de operandos para después imprimirlo y por último se crea una variable de tipo string llamada res, la cual concatena la letra “r” con una variable de tipo string que contiene el contador de las respuestas del cuádruplo. Luego se agrega en la pila de operandos lo que se encuentre en la variable res, para después imprimir en la pila de operandos.

En seguida se manda a llamar un método con parámetros que es el contador de cuádruplos, oper, op1, op2 y res, el cual se llama formaCuadruplo(); Si la variable supuesto fue mayor a 500 lo que se hace es llamar al método Errores(); con el número 544 (error entre tipos). Al supuesto se le asignará op1. En seguida se eliminará el tope de la pila de tipos, se imprimirá la

pila de tipos con el método imprimePilaTipos(); nuevamente vuelve a eliminar el tope de la pila de tipos y vuelve a imprimir la pila de tipos con el método de imprimePilaTipos(); se inserta el supuesto en la pila de tipos y se imprime en la pila de tipos con el método imprimePilaTipos(); se crea una variable Sting llamada oper que se rellenará con el método evaluaElemento(); con el parámetro que se encuentre en el tope de la pila de operadores, se elimina el tope de pila de operadores, se imprime la pila de operadores, se crea otra variable de tipo String llamada op2 que se rellena con el tope de la pila de operandos y se elimina el tope de la pila de operandos para después imprimirlo y por último se crea una variable de tipo string llamada res, la cual concatena la letra “r” con una variable de tipo string que contiene el contador de las respuestas del cuádruplo. Luego se agrega en la pila de operandos lo que se encuentre en la variable res, para después imprimir en la pila de operandos.

En seguida se manda a llamar un método con parámetros que es el contador de cuádruplos, oper, op1, op2 y res, el cual se llama formaCuadruplo();

Volviendo al método accionesSemanticayCodigoIntermedio(); se evaluará si el case es igual a 705 agregara dentro de la pila de operadores un “-2” que equivale a una marca de fondo falso, imprimirá la pila de operadores y termina el case.

Si el case es igual a 706 se eliminará el tope de la pila de operadores y se imprimirá la pila de operadores.

Si el case es igual a 707 se generara la acción equivalente a un salto en falso, creando una variable llamada oper la cual concatenara las siglas “SF”, se rellenara la variable op1 con el tope de la pila de operandos y se eliminara lo que corresponda

al dicho tope, se mandara a llamar el método imprimePilaOperandos(); y se llamara al método formaCuadruplo(); con el parámetro del contador de cuádruplo mas uno y las variables oper y op1 u 2 campos nulos, además se agregara a la pila de saltos lo que guarde la variable contadora de cuádruplos en ese momento, después se imprimirá la pila de saltos y termina el case.

En caso de que el case sea 708, el equivalente a un salto incondicional, se generara un por que recorrerá los cuádruplos existentes, en los cuales evaluara si en la posición que vaya recorriendo los cuádruplos corresponden a la pila de saltos, si es así, la posición en la que se encuentre dentro del cuádruplo le asignara lo equivalente al contador de cuádruplos más dos, en seguida se eliminara lo que exista dentro del tope de la pila de saltos y se imprimirá la pila correspondiente, se concatena dentro de la variable oper las siglas “SI” y se manda a llamar el método formaCuadruplo(); con los parámetros de contador de cuádruplos más uno, oper y tres campos nulos, para que de esa forma se agregue a la pila de saltos lo que corresponda al contador de cuádruplos y se imprime la pila de saltos, termina el case.

En caso de que el case sea 709, comenzara con un FOR que recorrerá los cuádruplos uno por en los cuales evaluara si en la posición que vaya recorriendo los cuádruplos corresponden a la pila de saltos, si es así, la posición en la que se encuentre dentro del cuádruplo le asignara lo equivalente al contador de cuádruplos más uno, eliminara lo que exista en el tope de la pila de saltos e imprimirá la pila correspondiente.

En caso de que el case sea 717, por medio de un IF recorrerá el contador RAW(que es el que va contando los identificadores

y constantes), en el cual evaluara si la variable estaPresenteRead es verdadera concatenando en la variable oper la palabra “read” y haciendo lo mismo evaluando la variable estaPresenteWrite y concatenando la palabra “write” en la variable oper, una vez terminadas las evaluaciones se manda a llamar al método formaCuadruplo(); con los parámetros equivalentes al contador de cuádruplo, oper, 2 campos nulos y la posición de la pila de operandos en la posición “i”.

Enseguida, mientras que la pila de operandos este llena, eliminará la primera posición de la pila e imprimirá la pila de operandos.

Se rellenará el contadorRAW en “0” y se cambiaran las variables de control de READ y WRITE a falso, cierra el case.

Si el case es igual a 716, se agregará a la pila de operadores el estado de aceptación 150, se imprimirá la pila de operadores y se mandara a llamar el método formaCuadruplo con los parámetros de contador de cuádruplo y un “enter” concatenado, se eliminará lo que exista en el tope de la pila de operadores y se imprimirá la susodicha, termina el case.

Si el case es igual a 715, se agregará en la pila de saltos el contado de cuádruplos más uno y se imprimirá la pila de saltos, se cierra el case.

Si el case es igual a 714. Se concatenara dentro de la variable oper las siglas “SI”, se mandara a llamar al método formaCuadruplo(); con los parámetros, contador de cuádruplo más uno, oper, 2 en nulo, y una variable string que equivale a la pila de saltos en la posición del largo de la pila menos uno, se inicializara un FOR que recorra los cuádruplos uno por uno evaluando si los cuádruplos corresponden con lo que haya en

el tope de la pila de saltos y rellenando el cuádruplo en la posición en la que se encuentre con el contador de cuádruplos más uno, se elimina lo que exista dentro del tope de la pila de saltos, se imprime la pila y se vuelve a eliminar el tope para imprimir por ultima vez la pila de saltos, cierra el case.

Si el case es igual a 713, evaluara si dentro del tope de cuádruplos menos uno se encuentra un operador “<” o “<=”, concatenara en la variable oper las siglas “SF”, y rellenara la variable op1 con el tope de la pila de operandos, eliminando el tope de la pila de operandos, si no se cumple la condición se concatenara en la variable oper las siglas “SV”, se rellenara la variable op1 con el tope de la pila de operandos y se eliminara el tope de la pila de operandos, para después imprimir la pila de operandos, a continuación se manda a llamar al método formaCuadruplo(); con los parámetros de contador de cuádruplo más uno, oper, op1 y dos nulos, se agrega a la pila de saltos el contador del cuádruplo y se imprime la pila de saltos.

En caso de que el case equivalga a 712, se insertara en la pila de saltos el contador de cuádruplo mas dos y se imprimirá la pila-

En caso de que el case equivalga a 711, se crearan dos variables, una llamada op1 que se rellenara con el tope de la pila de tipos y otra llamada op2 que se rellenara con lo que este dentro de la pila de tipos en la posición tope menos dos, se evaluara por medio de un IF si el tope de la pila de operadores es igual a 122, si es así, se evaluara por medio de otro ID anidado si el op1 es igual al op2, si es así, se llamara al método imprimeYlimpiaPilas();, se creara una variable string llamada oper que se rellenara con el método evalúa elemento con los

parámetros de lo que equivalga al tope de la pila de operadores, se eliminara lo que exista en la pila de operadores y se imprimirá la pila, se creara otra variable llamada op2 inicializada en nulo y otra llamada res inicializada con lo que exista en el tope de la pila de operandos, para después eliminarlo y finalizar imprimiendo la pila, después se creara otra variable de tipo string llamada op1 la cual se rellenara con el tope de la pila de operandos, para después eliminarlo e imprimirlo, para terminar llamando al método formaCuadruplo con parámetros del contador de cuádruplos, oper, op1,op2,res Si la variable supuesto fue mayor a 500 lo que se hace es llamar al método Errores(); con el número 544 (error entre tipos). Al supuesto se le asignará op1. En seguida se eliminará el tope de la pila de tipos, se imprimirá la pila de tipos con el método imprimePilaTipos(); nuevamente vuelve a eliminar el tope de la pila de tipos y vuelve a imprimir la pila de tipos con el método de imprimePilaTipos(); se inserta el supuesto en la pila de tipos y se imprime en la pila de tipos con el método imprimePilaTipos(); se crea una variable Sting llamada oper que se rellenará con el método evaluaElemento(); con el parámetro que se encuentre en el tope de la pila de operadores, se elimina el tope de pila de operadores, se imprime la pila de operadores, se crea otra variable de tipo String llamada op2 que se rellena con el tope de la pila de operandos y se elimina el tope de la pila de operandos para después imprimirlo y por último se crea una variable de tipo string llamada res, la cual concatena la letra “r” con una variable de tipo string que contiene el contador de las respuestas del cuádruplo. Luego se agrega en la pila de operandos lo que se encuentre en la variable res, para después imprimir en la pila de operandos.

En seguida se manda a llamar un método con parámetros que es el contador de cuádruplos, oper, op1, op2 y res, el cual se llama formaCuadruplo();

Retomando el método anterior llamado ConstruyeGramatica(); proseguirá con eliminar el tope de la pila de ejecución. Se evaluará si el token es igual a 151, de ser así cambiara la variable estaPresenteWrite a verdadero. En seguida se evaluará si el token es igual a 152, si es así se cambiará el estado de la variable estaPresenteRead a verdadero.

Después se evaluará si el token se encuentra entre 101 y 106 o si es igual a 127. Si es así se evaluará si la pila de operadores está llena, de ser así se imprimirá la pila de operandos y llamará al método accionesSemanticayCodigoIntermedio y le enviará el valor 704. Si la condición sobre el tope de la pila de ejecución no es igual al token entonces se declaran dos 2 variables: una llamada TR que se rellenará con el método EvaluaElemento() con el token como parámetro y otra llamada TP que se rellenará con el método EvaluaElemento con el parámetro del tope de la pila de ejecución, después se concatenará en la variable de errores sobre que hay errores en la sintaxis porque se esperaba un “tp” y se recibió un “tr”

Después se rellenará la variable token con el tope de la pila de ejecución y la variable de control sin error pasará a ser falsa.

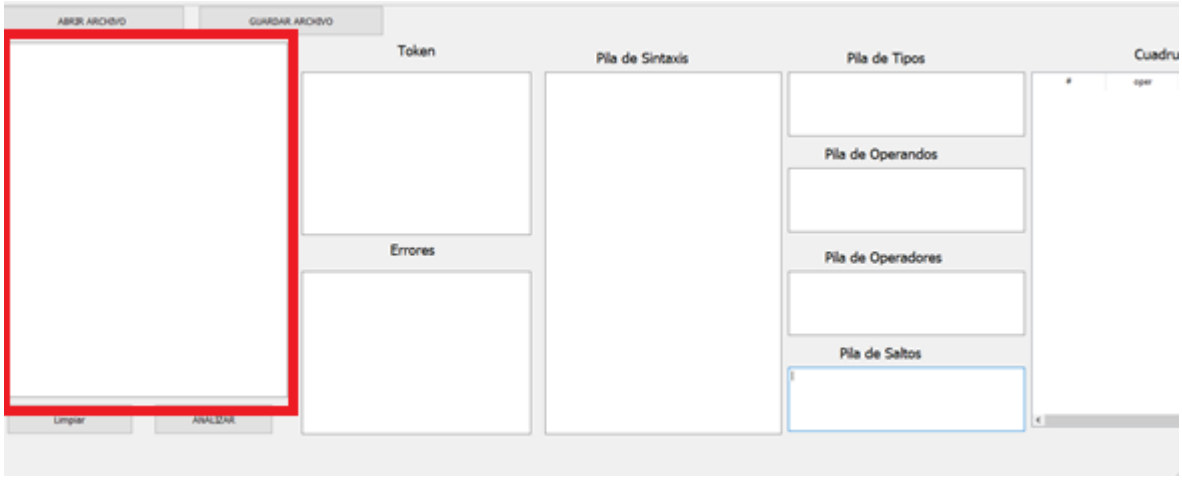
Si la condición anterior sobre el tipo de la pila de ejecución no es mayor que 100 se evaluará si el estado de la matriz de predicción es mayor que 500, si es así se mandará llamar al método errores con parámetro del estado de la matriz de predicción más 1.

La variable de control sin error pasará a ser falsa y se eliminará del tope de la pila de ejecución para después imprimir la pila. Si el estado de la matriz de predicción no es mayor que 500 entonces se evaluará que el tope de la pila de ejecución esté entre 0 y 34. Si es así la variable llena cambiará a ser verdadera.

En seguida se evaluará si el tope de la pila de ejecución es igual a -1, si es verdadero se eliminará el tope de la pila de ejecución y se imprimirá la pila, para después evaluar si el tope de la pila es igual a “$”. De ser así se eliminará el tope de la pila de ejecución y se imprimirá la pila evaluando inmediatamente la variable sin error y de ser así se imprimirá un mensaje que diga “se llegó al final de la pila y no hay errores”.

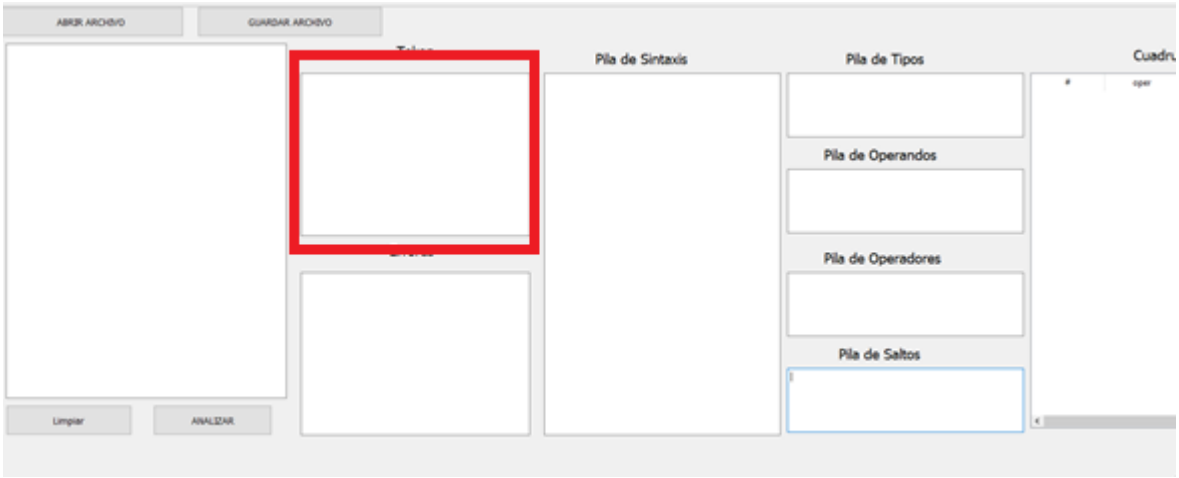
Retomando el método del botón “Analizar” se continuará con una condición que comparará la variable “errores”, si es diferente de nulo se mostrará un mensaje que diga “La sintaxis contiene errores léxicos, sintácticos o semánticos”. Se llenarán los siguientes cuadros de dialogo con las variables correspondientes y se imprimirán los cuádruplos en la tabla de cuádruplos con el método llenarCuadruplo();

- **Recuadro de análisis**



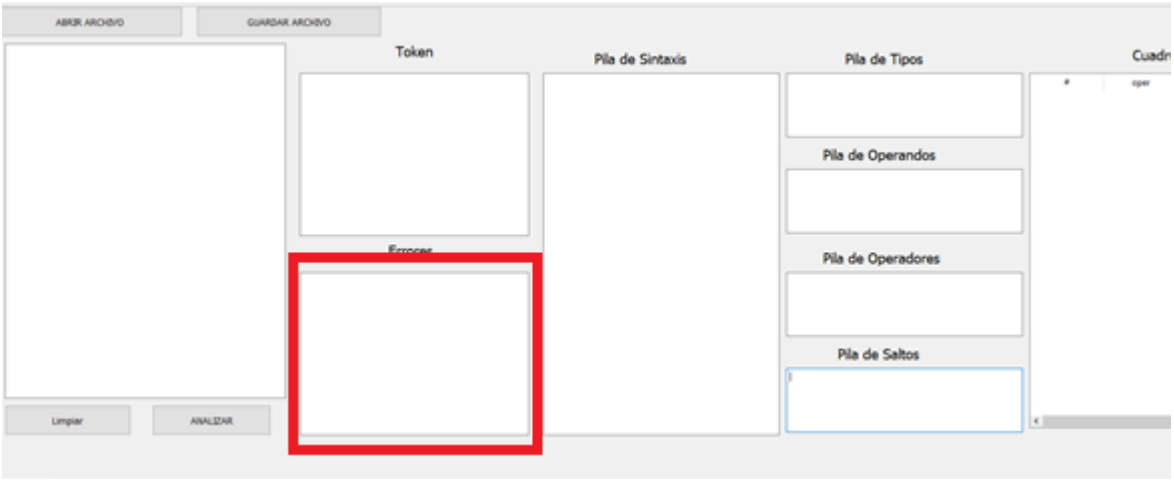
El recuadro de análisis lo único de que se encarga es de capturar el texto que desees introducir, y mediante los métodos del programa se extraerá a posteriori.

- **Recuadro de Tokens**



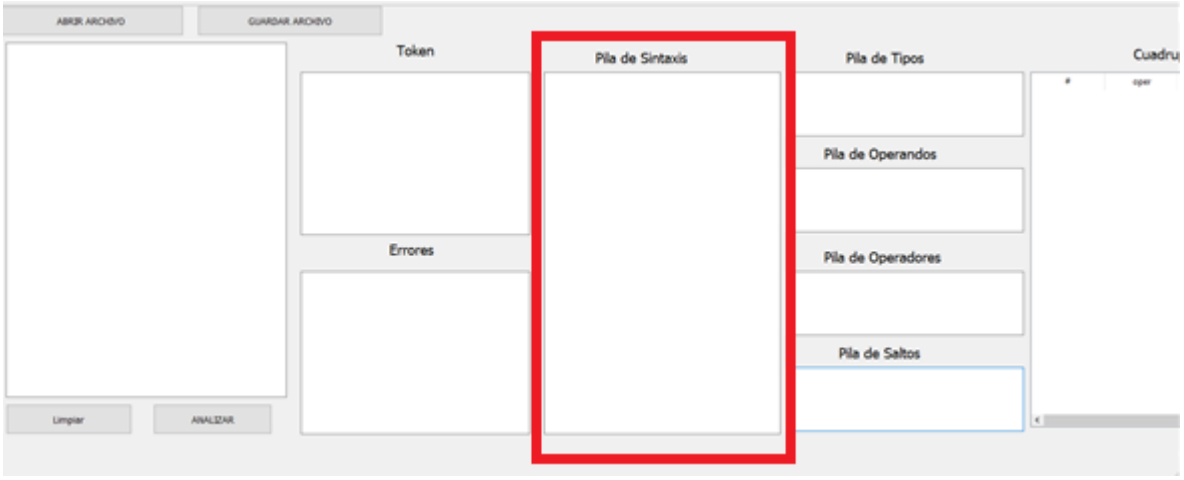
El recuadro de tokens lo único de que se encarga es de capturar el texto que desees introducir, y mediante los métodos del programa se extraerá a posteriori.

- **Recuadro de errores**



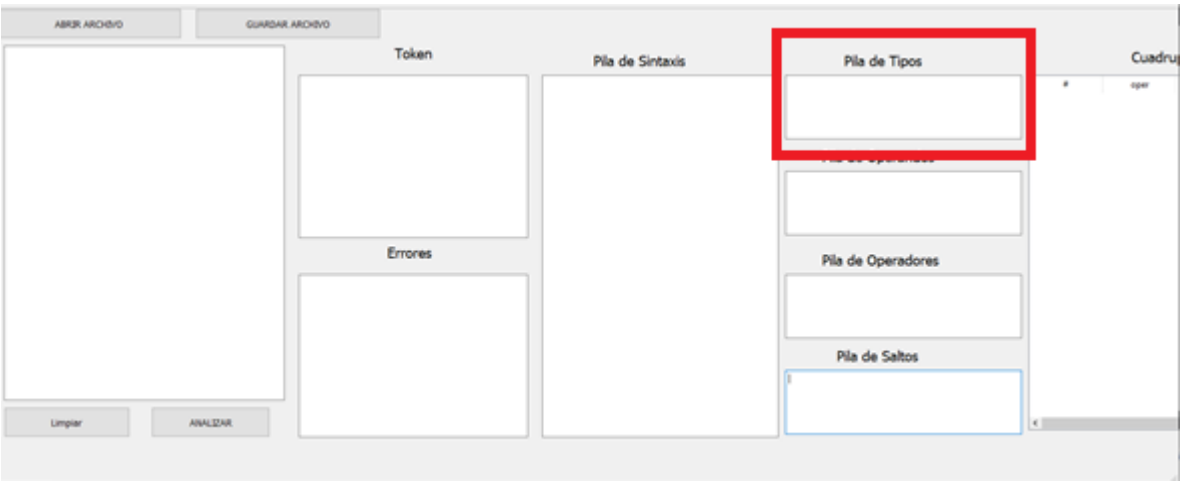
El recuadro de errores lo único de que se encarga es de capturar el texto que desees introducir, y mediante los métodos del programa se extraerá a posteriori.

- **Recuadro de pila de sintaxis**



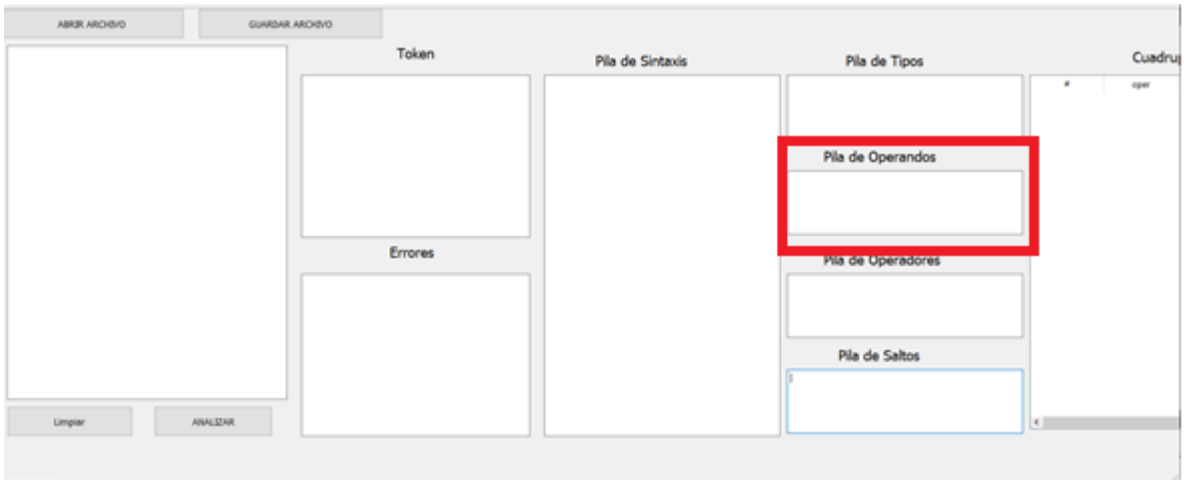
El recuadro de sintaxis lo único de que se encarga es de capturar el texto que desees introducir, y mediante los métodos del programa se extraerá a posteriori.

- **Recuadro de pila de tipos**



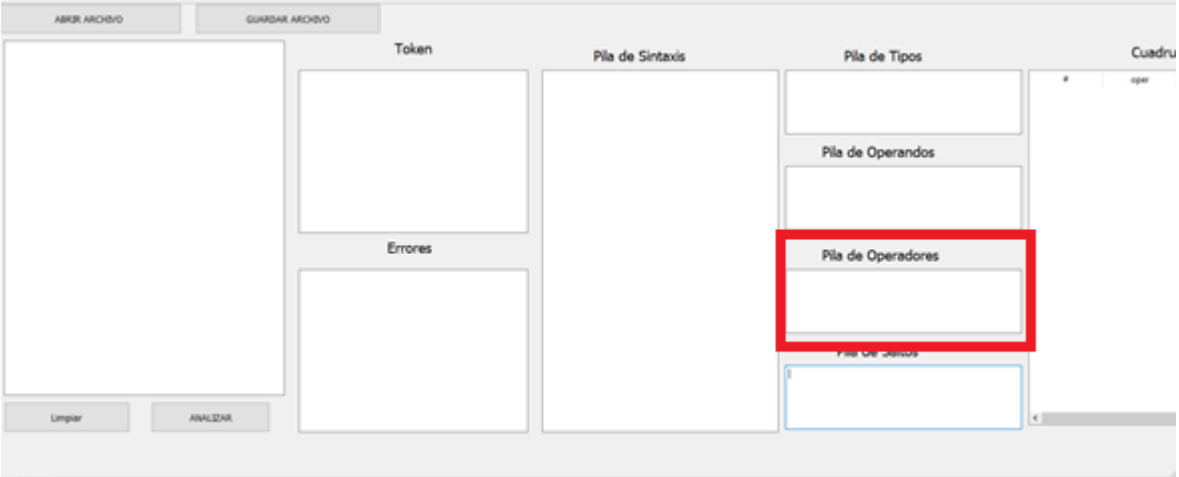
El recuadro de tipos lo único de que se encarga es de capturar el texto que desees introducir, y mediante los métodos del programa se extraerá a posteriorio.

- **Recuadro de Pila de Operandos**



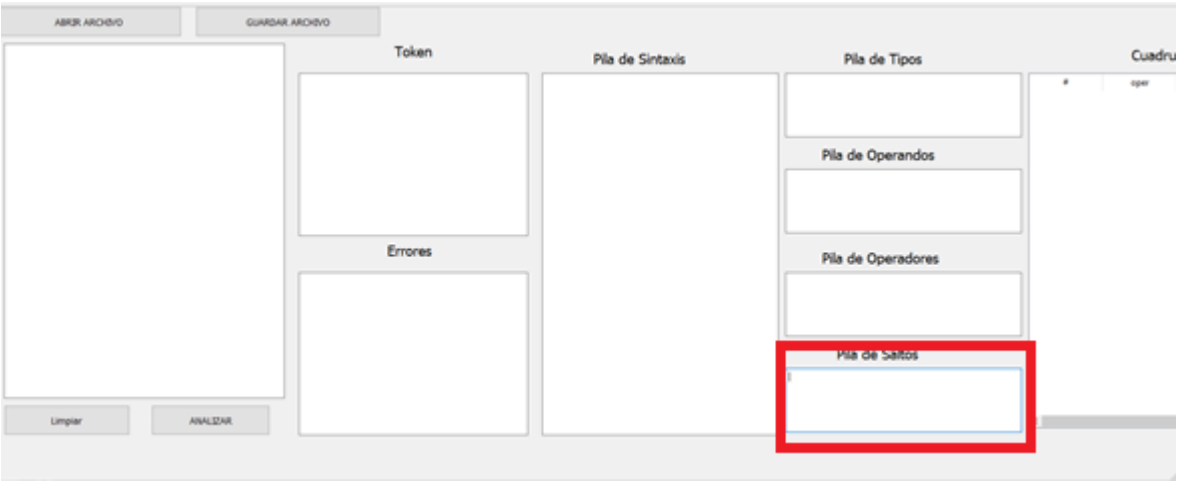
El recuadro de Operandos lo único de que se encarga es de capturar el texto que desees introducir, y mediante los métodos del programa se extraerá a posteriori.

- Recuadro de Pila de Operadores



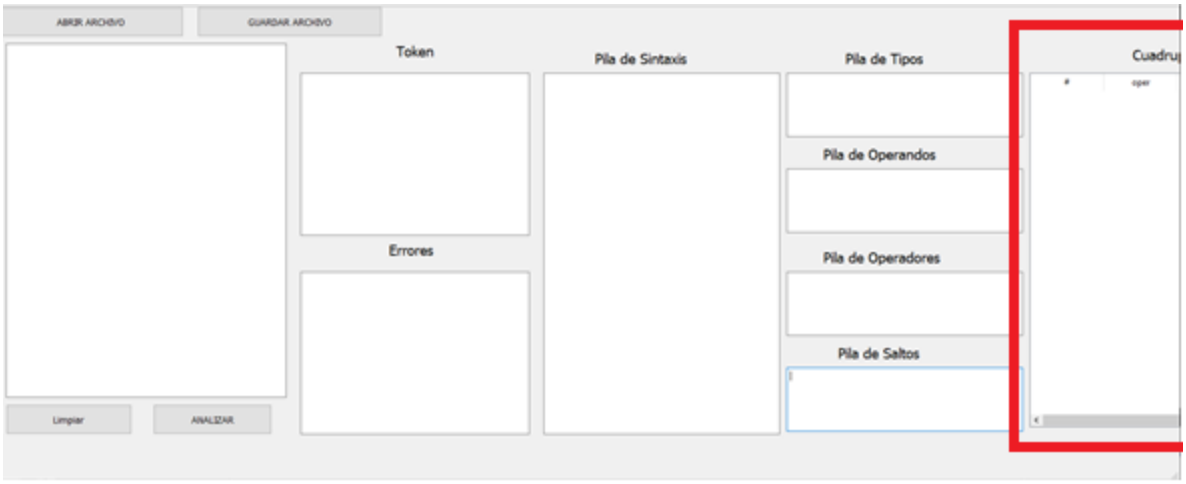
El recuadro de Operadores lo único de que se encarga es de capturar el texto que desees introducir, y mediante los métodos del programa se extraerá a posteriori.

- **Recuadro de Pila de Saltos**



El recuadro de saltos lo único de que se encarga es de capturar el texto que desees introducir, y mediante los métodos del programa se extraerá a posteriori.

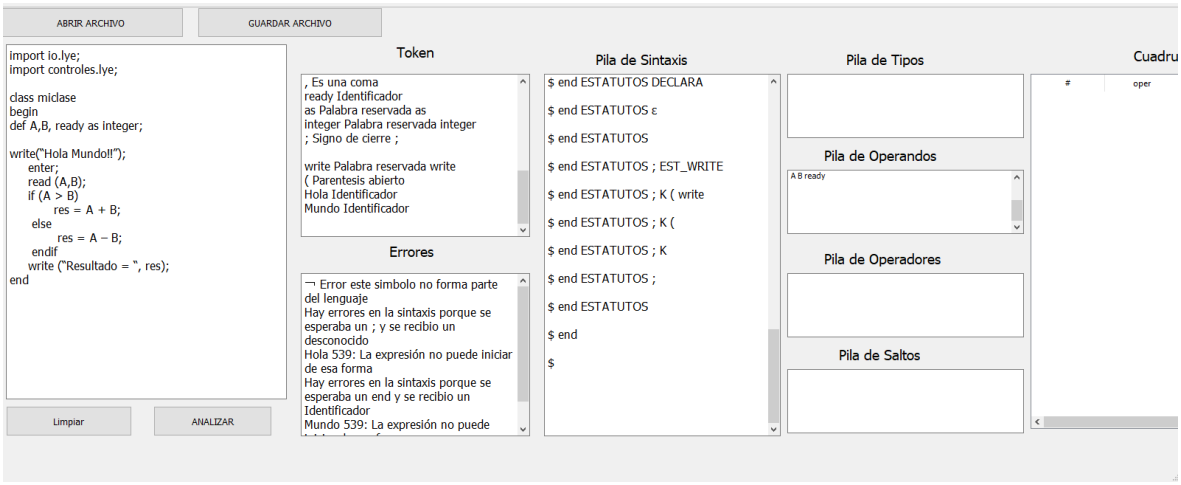
- **Recuadro de Cuádruplos Generados**



El recuadro de cuádruplos lo único de que se encarga es de capturar el texto que desees introducir, y mediante los métodos del programa se extraerá a posteriori.

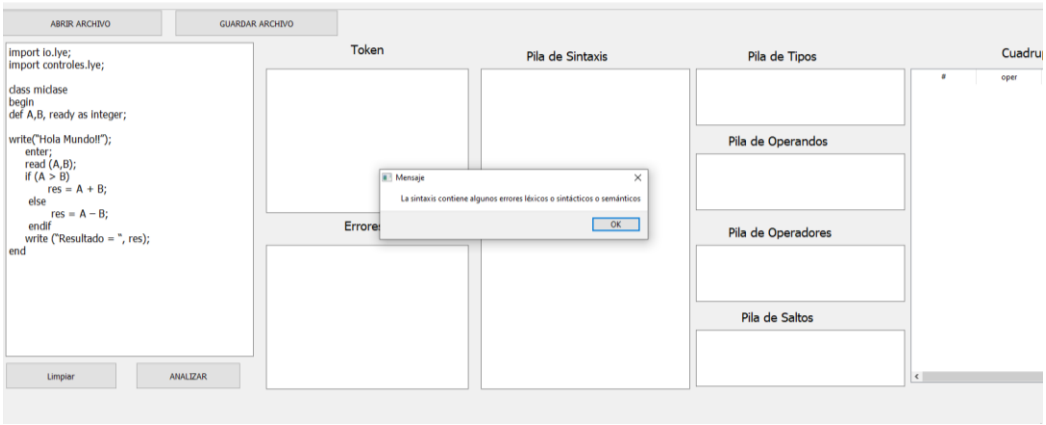
Captura de Información

El único campo por el cual podemos introducir es este:



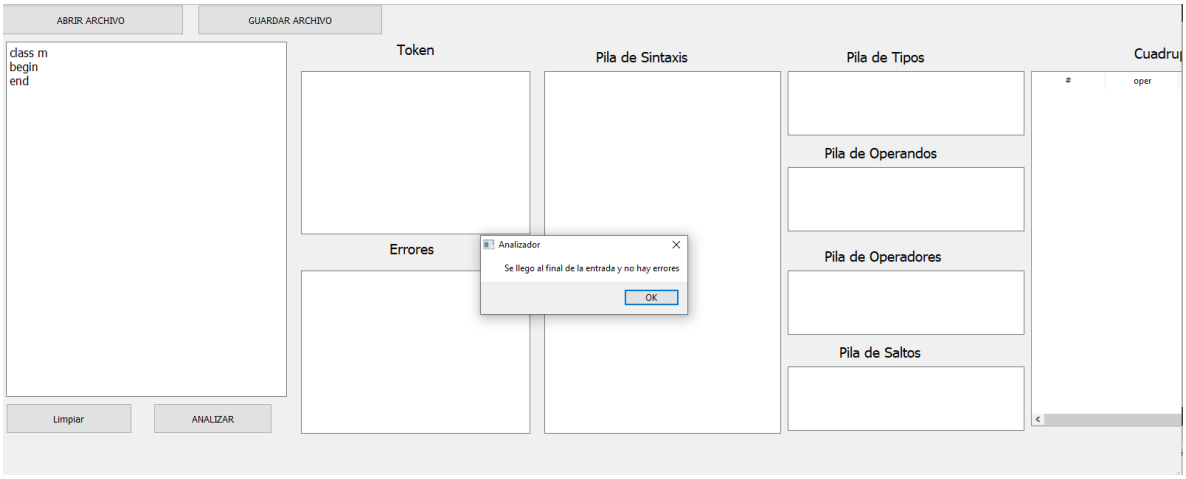
Ejemplo:

```
import io.lye;
import controles.lye;

class miclase
begin
def A,B, ready as integer;

write(“Hola Mundo!!”);
    enter;
    read (A,B);
    if (A > B)
        res = A + B;
     else
         res = A – B;
     endif
    write (“Resultado = “, res);
end
```



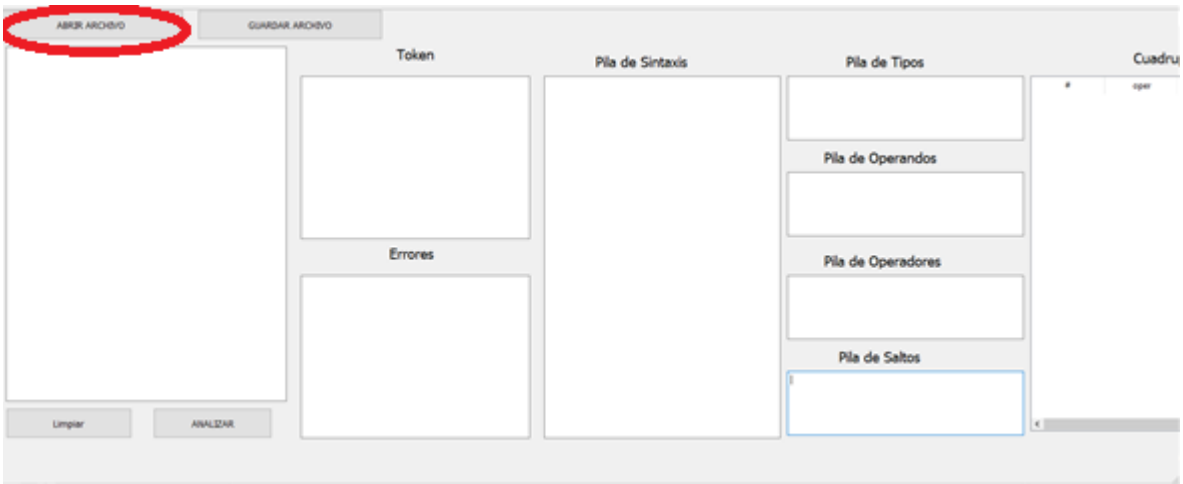
Si el enunciado introducido tiene errores léxicos, sintácticos o semánticos nos arrojara este cuadro de dialogo.

Si el enunciado introducido no contiene ningún erro se desplegará este cuadro de dialogo.

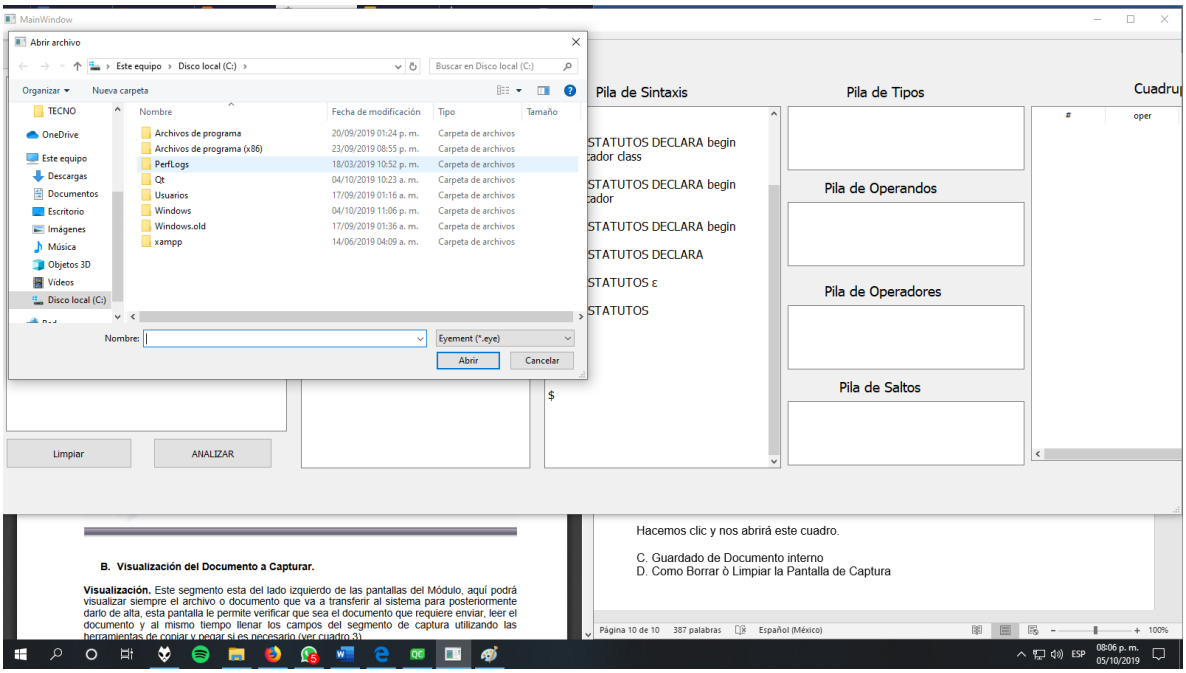


Visualización del Documento Externo

El programa solo abre archivos con terminación “.eye”, para hacerlo nos iremos al botón de abrir archivo.



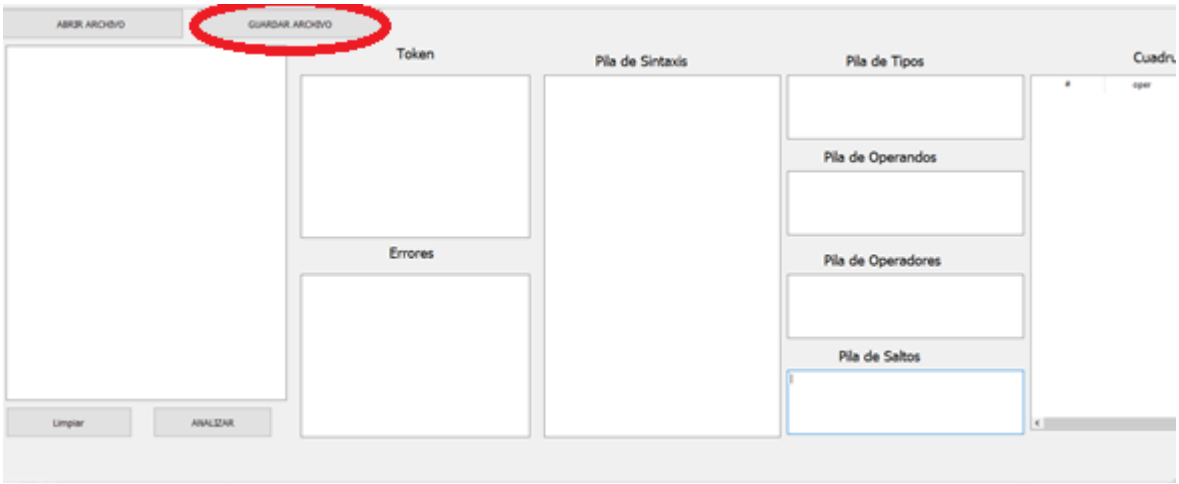
Hacemos clic y nos abrirá este cuadro.



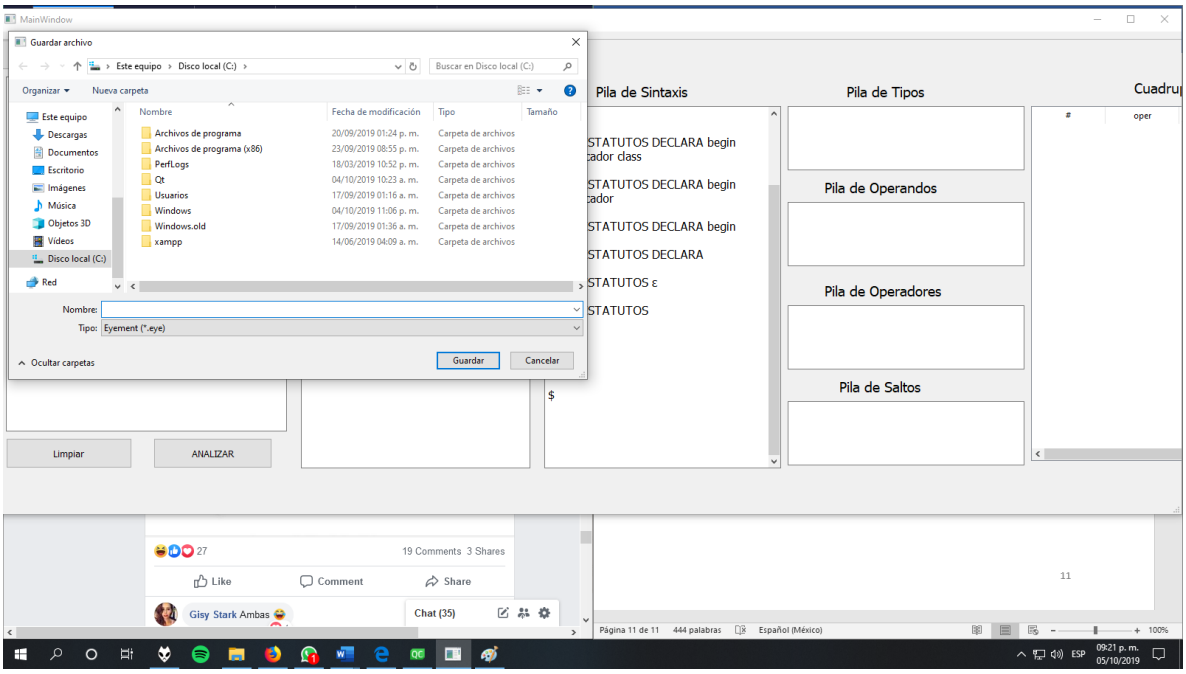
Buscamos nuestro archivo “.eye” y le daremos clic en el botón de “abrir del mismo recuadro para de esa manera se muestre en el recuadro de análisis.

Guardado de Documento interno

Para guardar un archivo escrito en el recuadro de análisis, daremos clic en este botón:



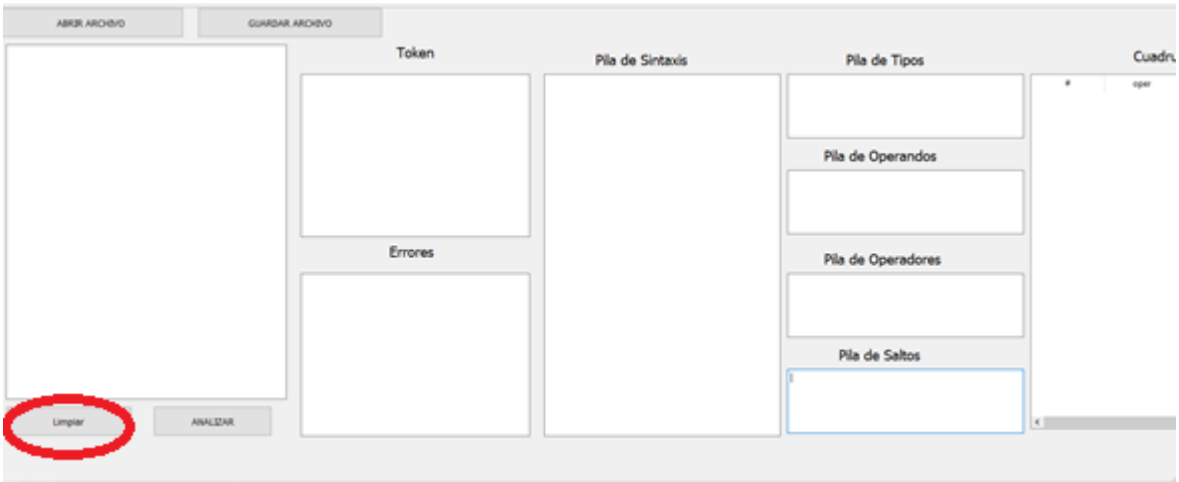
Nos saldrá un recuadro como este:



De igual forma buscaremos el directorio donde queramos guardarlo y daremos clic en el botón “guardar”.

Como Borrar ò Limpiar la Pantalla de Captura

Para borrar todo lo escrito dentro del recuadro de análisis se procederá a hacer clic en este botón:



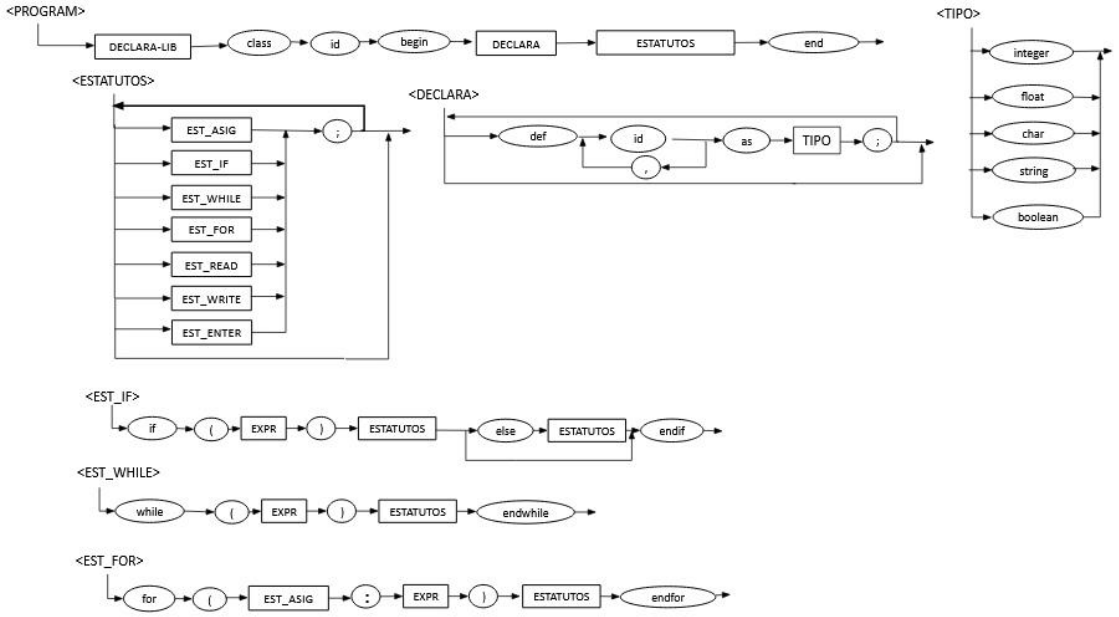
Una vez hecho clic el botón, el recuadro de análisis se pondrá en blanco y los demás recuadros también, permitiendo hacer un nuevo análisis de algún otro texto.

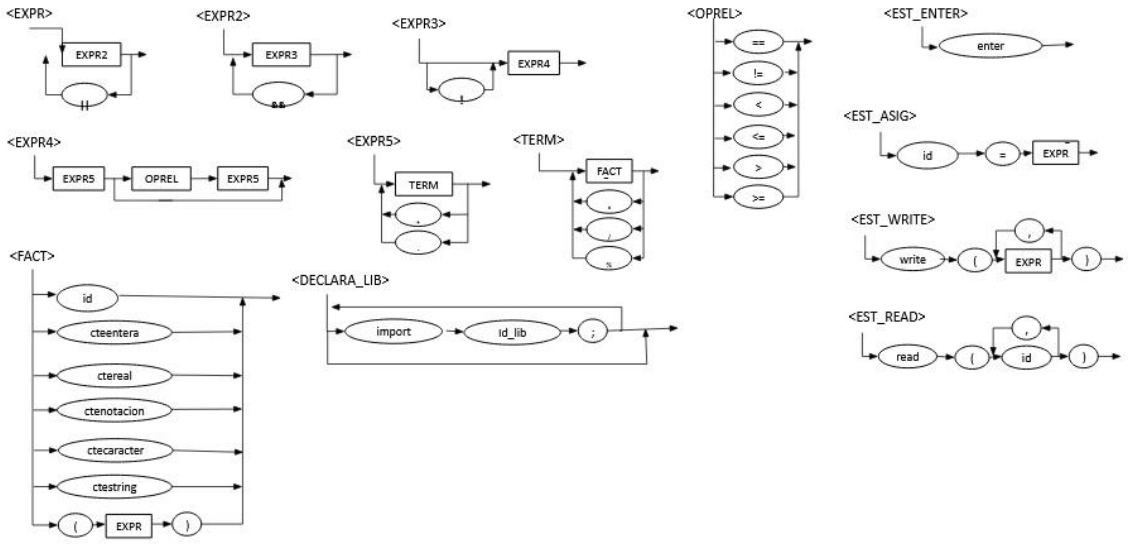
Analizador Léxico – Especificación de Tokens:

- Identificadores: Inicia con letra mayúscula o minúscula, seguida de letra, dígito o guion bajo (_), pero no tiene 2 guiones bajo seguidos, ni termina en guion bajo.
- Constantes numéricas: enteros, reales y de notación científica (como se vio en el aula)
- Constante carácter: ‘a’
- Constante string : “Este es un string”
- Operadores aritméticos: +, -, *, /, %
- Operadores lógicos: &&, ||, !
- Operadores Relacionales: ==, <, <=, >, >=, !=
- Asignación: =
- Signos de puntuación: , , ;, :, .
- Signos de agrupación: (,), [,]
- Comentarios: /* este es un comentario de bloque */

/********** Comentario *********/

Analizador Sintáctico – Estructura de Programación:





PRODUCCIONES:

PROGRAM -> DECLARA-LIB A

A -> class id begin DECLARA ESTATUTOS end

DECLARA-LIB -> A

DECLARA-LIB -> import id_lib ; DECLARA-LIB

DECLARA -> ε

DECLARA -> def B

B -> id C

C -> , B

C -> as TIPO ; DECLARA

TIPO -> integer

TIPO -> float

TIPO -> char

TIPO -> string

TIPO -> boolean

ESTATUTOS -> ε

ESTATUTOS -> EST_ASIG ; ESTATUTOS

ESTATUTOS -> EST_IF ; ESTATUTOS

ESTATUTOS -> EST_WHILE ; ESTATUTOS

ESTATUTOS -> EST_FOR ; ESTATUTOS

ESTATUTOS -> EST_READ ; ESTATUTOS

ESTATUTOS -> EST_WRITE ; ESTATUTOS

ESTATUTOS -> EST_ENTER ; ESTATUTOS

EST_IF -> if (EXPR) ESTATUTOS D

D -> endif

D -> else ESTATUTOS endif

EST_WHILE -> while (EXPR) ESTATUTOS endwhile

EST_FOR -> for (EST_ASIG : EXPR) ESTATUTOS endfor

EXPR -> EXPR2 E

E -> ε

E -> || EXPR

EXPR2 -> EXPR3 F

F -> ε

F -> && EXPR2

EXPR3 -> G

G -> EXPR4

G -> ! EXPR4

EXPR4 -> EXPR5 H

H -> ε

H -> OPREL EXPR5

EXPR5 -> TERM I

I -> ε

I -> + EXPR5

I -> - EXPR5

TERM -> FACT J

J -> ε

J -> * TERM

J -> / TERM

J -> % TERM

FACT -> id

FACT -> cteentera

FACT -> ctenotacion

FACT -> ctecaracter

FACT -> ctestring

FACT -> (EXPR)

OPREL -> ==

OPREL -> !=

OPREL -> <

OPREL -> <=

OPREL -> >

OPREL -> >=

EST_ENTER -> enter

EST_ASIG -> id = EXPR

EST_WRITE -> write (K

K -> EXPR L

L ->)

L -> , K

EST_READ -> read (M

M -> id N

N ->)

N -> , M

Analizador Semántico – Tabla de Símbolos:

I=integer F=float C=char S=String B=boolean X=Error

op1	op2	"+, -, *"	"/"	div mod	op relacional	not, and, or
I	I	I	F	I	B	X
I	F	F	F	X	B	X
I	C	X	X	X	X	X
I	S	X	X	X	X	X
I	B	X	X	X	X	X
F	I	F	F	X	B	X
F	F	F	F	X	B	X
F	C	X	X	X	X	X
F	S	X	X	X	X	X
F	B	X	X	X	X	X
C	I	X	X	X	X	X
C	F	X	X	X	X	X
C	C	X	X	X	B	X
C	S	X	X	X	B	X
C	B	X	X	X	X	X
S	I	X	X	X	X	X
S	F	X	X	X	X	X
S	C	X	X	X	B	X
S	S	X	X	X	B	X
S	B	X	X	X	X	X
B	I	X	X	X	X	X
B	F	X	X	X	X	X
B	C	X	X	X	X	X
B	S	X	X	X	X	X
B	B	X	X	X	B	B

Los elementos que reciben el lenguaje como ya lo vimos en el analizador léxico, pueden ser desde: Identificadores, Constantes Enteras, Constantes Decimales, Constantes con notación científica, Constantes carácter y constantes String.

G.C.I (acciones/generación de pilas/ generación de cuádruplos):

[link a método accionesSemanticayCodigoIntermedio();]

Código Fuente:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QTextStream>
#include <fstream>
#include <iostream>
#include <QFileDialog>
#include <QMessageBox>
#include <QStack>

using namespace std;
//Clase en donde se guarda la información de los cuadruplos
class cuadruplo{

    public:
        QString n;
        QString oper;
        QString op1;
        QString op2;
        QString res;
        cuadruplo(QString,QString,QString,QString,QString);
        QString getN();
        QString getOper();
        QString getOp1();
        QString getOp2();
        QString getRes();
        void setRes(QString r);

};
QString cuadruplo::getN(){
    return n;
}
QString cuadruplo::getOper(){
    return oper;
}
QString cuadruplo::getOp1(){
    return op1;
}
QString cuadruplo::getOp2(){
    return op2;
}
QString cuadruplo::getRes(){
    return res;
}
void cuadruplo::setRes(QString _r){
    res=_r;
}
cuadruplo::cuadruplo(QString _n,QString _oper,QString _op1,QString
_op2,QString _res){
    n=_n;
```

```
    oper=_oper;
    op1=_op1;
    op2=_op2;
    res=_res;
}


static QStack<int> pilaEjecucion;//Pila análisis sintactico
static QStack<QString> pilaOperandos;//Pila de operandos
static QStack<QString> pilaOperandosBusqueda;//Pila para facilitar la
busqueda de tipos en un futuro
static QStack<int> pilaTipos;//Pila de tipos para los identificadores
static QStack<int> pilaTiposBusqueda;//Pila para los tipos
static QStack<int> pilaOperadores;//Pila de operadores que se ingresan
static QStack<int> pilaSaltos;
static int edo;//Estado del analizado léxico
static bool sinError; //Varibale para medir si hay error
static QList<QSharedPointer<cuadruplo>> cuadruplos;
static int contadorRaW;//Variable que controla los elementos del read,
write
static bool estaPresenteRead=false;
static bool estaPresenteWrite=false;
static int contCuadruplo; //Variable que cuenta el valor de los
cuadruplos
static int contRes;//Variable que cuenta el valor de res
static int
matriz[23][32]={{4,125,500,500,107,108,500,0,0,0,2,1,109,13,10,12,16,17,1
8,19,21,20,128,129,126,127,123,124,500,111,130,101},

{2,100,100,100,100,100,100,100,100,100,100,2,1,100,100,100,100,100,100,100,10
0,100,100,100,100,100,100,100,100,100,3,100,100,100},

{2,22,101,101,101,101,101,101,101,101,101,2,2,101,101,101,101,101,101,101,101
,101,101,101,101,101,101,101,101,101,3,101,101,101},

{2,501,501,501,501,501,501,501,501,501,501,2,2,501,501,501,501,501,501,501,50
1,501,501,501,501,501,501,501,501,501,501,501,501,501},

{4,5,102,102,102,102,102,102,102,102,102,102,102,102,102,102,102,102,102,102,
102,102,102,102,102,102,102,102,102,102,102,102,102,102},

{6,502,502,502,502,502,502,502,502,502,502,502,502,502,502,502,502,502,502,50
2,502,502,502,502,502,502,502,502,502,502,502,502,502,502},

{6,103,7,7,103,103,103,103,103,103,103,103,103,103,103,103,103,103,103,103,10
3,103,103,103,103,103,103,103,103,103,103,103,103,103},

{9,503,503,503,8,8,503,503,503,503,503,503,503,503,503,503,503,503,503,503,50
3,503,503,503,503,503,503,503,503,503,503,503,503,503},

{9,504,504,504,504,504,504,504,504,504,504,504,504,504,504,504,504,504,504,50
4,504,504,504,504,504,504,504,504,504,504,504,504,504},

{9,104,104,104,104,104,104,104,104,104,104,104,104,104,104,104,104,104,104,10
4,104,104,104,104,104,104,104,104,104,104,104,104,104},
```

```
{11,11,11,11,11,11,11,11,11,11,11,11,11,11,505,11,11,11,11,11,11,11,11,11
,11,11,11,11,11,11,11,11},

{506,506,506,506,506,506,506,506,506,506,506,506,506,506,105,506,506,506,
506,506,506,506,506,506,506,506,506,506,506,506,506,506,506,506,506,506},

{12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,106,12,12,12,12,12,12,12,12
,12,12,12,12,12,12,12,12},

{110,110,110,110,110,110,110,110,110,110,110,110,14,110,110,110,110,110,1
10,110,110,110,110,110,110,110,110,110,110,110,110,110,110,110,110,110,110},

{14,14,14,14,14,14,14,14,14,14,14,14,15,14,14,14,14,14,14,14,14,14,14,14,
14,14,14,14,14,14,14,14,14},

{14,14,14,14,14,14,14,14,14,14,14,14,15,112,14,14,14,14,14,14,14,14,14,14
,14,14,14,14,14,14,14,14,14},

{507,507,507,507,507,507,507,507,507,507,507,507,507,507,507,507,507,113,
507,507,507,507,507,507,507,507,507,507,507,507,507,507,507,507,507,507},

{508,508,508,508,508,508,508,508,508,508,508,508,508,508,508,508,508,508,
114,508,508,508,508,508,508,508,508,508,508,508,508,508,508,508,508,508},

{115,115,115,115,115,115,115,115,115,115,115,115,115,115,115,115,115,115,
115,116,115,115,115,115,115,115,115,115,115,115,115,115,115,115,115,115},

{122,122,122,122,122,122,122,122,122,122,122,122,122,122,122,122,122,122,
122,117,122,122,122,122,122,122,122,122,122,122,122,122,122,122,122,122},

{118,118,118,118,118,118,118,118,118,118,118,118,118,118,118,118,118,118,
118,119,118,118,118,118,118,118,118,118,118,118,118,118,118,118,118,118},

{120,120,120,120,120,120,120,120,120,120,120,120,120,120,120,120,120,120,
120,121,120,120,120,120,120,120,120,120,120,120,120,120,120,120,120,120},

{509,509,509,509,509,509,509,509,509,509,509,22,509,509,509,509,509,509,5
09,509,509,509,509,509,509,509,509,509,509,509,509,509,509,509,131}};


static int
matrizPredictiva[34][49]={{1,510,1,510,510,510,510,510,510,510,510,510,510,51
0,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510
,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510,510}
,

{511,511,2,511,511,511,511,511,511,511,511,511,511,511,511,511,511,511,511,51
1,511,511,511,511,511,511,511,511,511,511,511,511,511,511,511,511,511,511,511
,511,511,511,511,511,511,511,511,511,511,511,511,511},

{4,512,3,512,512,512,512,512,512,512,512,512,512,512,512,512,512,512,512,512,
512,512,512,512,512,512,512,512,512,512,512,512,512,512,512,512,512,512,512,5
12,512,512,512,512,512,512,512,512,512,512,512,512},

{513,513,513,5,513,6,513,513,513,513,513,513,5,513,513,513,513,5,513,5,51
```

3,513,513,513,513,513,513,513,513,513,513,513,513,513,513,513,513,513,513
,513,513,5,5,5,513,513,513,5,513},

{514,514,514,7,514,514,514,514,514,514,514,514,514,514,514,514,514,514,51
4,514,514,514,514,514,514,514,514,514,514,514,514,514,514,514,514,514,514
,514,514,514,514,514,514,514,514,514,514,514,514,514,514,514,514,514,514},

{515,515,515,515,515,515,9,515,515,515,515,515,515,515,515,515,515,515,515,51
5,515,515,515,515,515,515,515,515,515,515,515,515,515,515,515,515,515,515,515
,515,515,515,515,515,515,515,8,515,515,515,515},

{516,516,516,516,516,516,516,10,11,12,13,14,516,516,516,516,516,516,516,5
16,516,516,516,516,516,516,516,516,516,516,516,516,516,516,516,516,516,516,51
6,516,516,516,516,516,516,516,516,516,516,516,516},

{517,517,517,16,517,517,517,517,517,517,517,517,17,517,517,15,15,18,15,19
,15,517,517,517,517,517,517,517,517,517,517,517,517,517,517,517,517,517,517,5
17,517,517,22,21,20,517,517,517,15,517},

{518,518,518,518,518,518,518,518,518,518,518,518,23,518,518,518,518,518,5
18,518,518,518,518,518,518,518,518,518,518,518,518,518,518,518,518,518,518,51
8,518,518,518,518,518,518,518,518,518,518,518,518},

{519,519,519,519,519,519,519,519,519,519,519,519,519,519,519,25,24,519,51
9,519,519,519,519,519,519,519,519,519,519,519,519,519,519,519,519,519,519,519
,519,519,519,519,519,519,519,519,519,519,519,519},

{520,520,520,520,520,520,520,520,520,520,520,520,520,520,520,520,520,26,5
20,520,520,520,520,520,520,520,520,520,520,520,520,520,520,520,520,520,520,52
0,520,520,520,520,520,520,520,520,520,520,520,520},

{521,521,521,521,521,521,521,521,521,521,521,521,521,521,521,521,521,521,
521,27,521,521,521,521,521,521,521,521,521,521,521,521,521,521,521,521,521,52
1,521,521,521,521,521,521,521,521,521,521,521,521},

{522,522,522,28,522,522,522,522,522,522,522,522,522,28,522,522,522,522,52
2,522,522,522,522,36,522,522,522,522,522,522,522,522,522,522,522,522,28,2
8,28,28,28,522,522,522,522,522,522,522,522},

{523,523,523,523,523,523,523,523,523,523,523,523,523,523,29,523,523,523,5
23,523,523,30,523,523,523,523,523,523,523,523,523,523,523,523,523,523,523,523
,523,523,523,523,523,523,523,29,29,29,523,523},

{524,524,524,31,524,524,524,524,524,524,524,524,524,31,524,524,524,524,52
4,524,524,524,524,36,524,524,524,524,524,524,524,524,524,524,524,524,31,3
1,31,31,31,524,524,524,524,524,524,524,524},

{525,525,525,525,525,525,525,525,525,525,525,525,525,525,525,32,525,525,525,5
25,525,525,525,32,33,525,525,525,525,525,525,525,525,525,525,525,525,525,525,
525,525,525,525,525,525,525,32,32,32,525,525},

{526,526,526,34,526,526,526,526,526,526,526,526,526,34,526,526,526,526,52
6,526,526,526,526,36,526,526,526,526,526,526,526,526,526,526,526,526,34,3
4,34,34,34,526,526,526,526,526,526,526,526},

{527,527,527,35,527,527,527,527,527,527,527,527,527,35,527,527,527,527,52

7,527,527,527,527,36,527,527,527,527,527,527,527,527,527,527,527,527,35,3
5,35,35,35,527,527,527,527,527,527,527,527},

{528,528,528,37,528,528,528,528,528,528,528,528,528,37,528,528,528,528,52
8,528,528,528,528,528,528,528,528,528,528,528,528,528,528,528,528,528,37,
37,37,37,37,528,528,528,528,528,528,528,528},

{529,529,529,529,529,529,529,529,529,529,529,529,529,529,38,529,529,529,5
29,529,529,38,38,529,529,39,39,39,39,39,39,529,529,529,529,529,529,529,52
9,529,529,529,529,529,38,38,38,529,529},

{530,530,530,40,530,530,530,530,530,530,530,530,530,40,530,530,530,530,53
0,530,530,530,530,530,530,530,530,530,530,530,530,530,530,530,530,530,530,40,
40,40,40,40,530,530,530,530,530,530,530,530},

{531,531,531,531,531,531,531,531,531,531,531,531,531,531,41,531,531,531,5
31,531,531,41,41,531,531,41,41,41,41,41,41,42,43,531,531,531,531,531,531,
531,531,531,531,531,531,41,41,41,531,531},

{532,532,532,44,532,532,532,532,532,532,532,532,532,44,532,532,532,532,53
2,532,532,532,532,532,532,532,532,532,532,532,532,532,532,532,532,532,532,44,
44,44,44,44,532,532,532,532,532,532,532,532},

{533,533,533,533,533,533,533,533,533,533,533,533,533,533,45,533,533,533,5
33,533,533,45,45,533,533,45,45,45,45,45,45,45,46,47,48,533,533,533,533
,533,533,533,533,45,45,45,533,533},

{534,534,534,49,534,534,534,534,534,534,534,534,534,55,534,534,534,534,53
4,534,534,534,534,534,534,534,534,534,534,534,534,534,534,534,534,534,534,50,
51,52,53,54,534,534,534,534,534,534,534,534},

{535,535,535,535,535,535,535,535,535,535,535,535,535,535,535,535,535,535,
535,535,535,535,535,535,535,535,535,56,57,58,59,60,61,535,535,535,535,535,535,535
,535,535,535,535,535,535,535,535,535,535,535,535},

{536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,
536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,536,5
36,536,536,536,536,62,536,536,536,536,536,536,536},

{537,537,537,63,537,537,537,537,537,537,537,537,537,537,537,537,537,537,5
37,537,537,537,537,537,537,537,537,537,537,537,537,537,537,537,537,537,537,537,53
7,537,537,537,537,537,537,537,537,537,537,537,537},

{538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,
538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,538,5
38,538,538,538,538,538,64,538,538,538,538,538,538},

{539,539,539,65,539,539,539,539,539,539,539,539,539,65,539,539,539,539,53
9,539,539,539,539,36,539,539,539,539,539,539,539,539,539,539,539,539,65,6
5,65,65,65,539,539,539,539,539,539,539,539},

{540,540,540,540,540,540,540,540,540,540,540,540,540,540,66,540,540,540,5
40,540,540,540,540,540,540,540,540,540,540,540,540,540,540,540,540,540,540,540,54
0,540,540,540,540,540,540,540,67,540,540,540,540},

{541,541,541,541,541,541,541,541,541,541,541,541,541,541,541,541,541,541,

```
541,541,541,541,541,541,541,541,541,541,541,541,541,541,541,541,541,541,5
41,541,541,541,541,541,541,68,541,541,541,541,541},

{542,542,542,69,542,542,542,542,542,542,542,542,542,542,542,542,542,542,5
42,542,542,542,542,542,542,542,542,542,542,542,542,542,542,542,542,542,54
2,542,542,542,542,542,542,542,542,542,542,542,542,542},

{543,543,543,543,543,543,543,543,543,543,543,543,543,543,70,543,543,543,5
43,543,543,543,543,543,543,543,543,543,543,543,543,543,543,543,543,543,54
3,543,543,543,543,543,543,543,71,543,543,543,543}};



static int producciones[71][15]={{2,3},//A DECLARA-LIB
                                 {132,8,4,133,101,134}, //end ESTATUTOS
DECLARA begin id class
                                 {-1},//ε
                                 {3,124,131,135},//DECLARA-LIB ; id_lib
import
                                 {-1},//ε
                                 {5,136},//B def
                                 {6,700,101},//C acción 1 semantico id
                                 {5,130},//B ,
                                 {4,124,701,7,137},//DECLARA ; acción 2
semantico TIPO as
                                 {138},//integer
                                 {139},//float
                                 {140},//char
                                 {141},//string
                                 {142},//boolean
                                 {-1},//ε
                                 {8,711,124,28},//ESTATUTOS ; EST_ASIG
                                 {8,124,9},//ESTATUTOS ; EST_IF
                                 {8,124,11},//ESTATUTOS ; EST_WHILE
                                 {8,124,12},//ESTATUTOS ; EST_FOR
                                 {8,124,32},//ESTATUTOS ; EST_READ
                                 {8,124,29},//ESTATUTOS ; EST_WRITE
                                 {8,124,27},//ESTATUTOS ; EST_ENTER
                                 {709,10,8,707,127,13,126,143},//D
ESTATUTOS ) EXPR ( if
                                 {144},//endif
                                 {144,8,708,145},//endif ESTATUTOS else

{714,146,8,707,127,13,715,126,147},//endwhile ESTATUTOS ) EXPR ( while

{714,148,8,713,127,13,712,123,28,126,149},//endfor ESTATUTOS ) EXPR :
EST_ASIG ( for
                                 {14,15},//E EXPR2
                                 {-1},//ε
                                 {13,703,114},//EXPR ||
                                 {16,17},// F EXPR3
                                 {-1},//ε
                                 {15,703,113},//EXPR2 &&
                                 {18},//G
                                 {19},//EXPR4
                                 {19,115},//EXPR4 !
                                 {20,21},//H EXPR5
```

```
                                {-1},//ε
                                {21,26},//EXPR5 OPREL
                                {22,23},//I TERM
                                {-1},//ε
                                {21,703,107},//EXPR5 (insertar pila de
operadores el operador recibido) +
                                {21,703,108},//EXPR5 (insertar pila de
operadores el operador recibido) -
                                {24,25},//J FACT
                                {-1},//ε
                                {23,703,109},//TERM (insertar pila de
operadores el operador recibido) *
                                {23,703,110},//TERM (insertar pila de
operadores el operador recibido) /
                                {23,703,111},//TERM  (insertar pila de
operadores el operador recibido) %
                                {702,101},//(Acción insertar en la pila
de tipos el identificador recibido) id
                                {702,102},//(Acción insertar en la pila
de tipos el identificador recibido) cteentera
                                {702,103},//(Acción insertar en la pila
de tipos el identificador recibido) ctereal
                                {702,104},//(Acción insertar en la pila
de tipos el identificador recibido) ctenotacioncientifica
                                {702,105},//(Acción insertar en la pila
de tipos el identificador recibido) ctecaracter
                                {702,106},//(Acción insertar en la pila
de tipos el identificador recibido) ctestring
                                {706,127,13,705,126},// ) EXPR (
                                {703,117},//==
                                {703,116},//!=
                                {703,120},//<
                                {703,121},//<=
                                {703,118},//>
                                {703,119},//>=
                                {716,150},//enter
                                {13,703,122,702,101},//EXPR (insertar
pila de operadores el operador recibido) = (Acción insertar en la pila de
tipos el identificador recibido) id
                                {717,30,126,151},//K ( write
                                {31,13},//L EXPR
                                {127},// )
                                {30,130},// K ,
                                {717,33,126,152},//M ( read
                                {34,702,101},//N id
                                {127},// )
                                {33,130}//M ,
                                };

static int matrizDeTipos[25][7]={{138,138,138,139,138,142,544},
                                 {138,139,139,139,544,142,544},
                                 {138,140,544,544,544,544,544},
                                 {138,141,544,544,544,544,544},
                                 {138,142,544,544,544,544,544},
                                 {139,138,139,139,544,142,544},
                                 {139,139,139,139,544,142,544},
                                 {139,140,544,544,544,544,544},
```

```
                                    {139,141,544,544,544,544,544},
                                    {139,142,544,544,544,544,544},
                                    {140,138,544,544,544,544,544},
                                    {140,139,544,544,544,544,544},
                                    {140,140,544,544,544,142,544},
                                    {140,141,544,544,544,544,544},
                                    {140,142,544,544,544,544,544},
                                    {141,138,544,544,544,544,544},
                                    {141,139,544,544,544,544,544},
                                    {141,140,544,544,544,142,544},
                                    {141,141,544,544,544,142,544},
                                    {141,142,544,544,544,544,544},
                                    {142,138,544,544,544,544,544},
                                    {142,139,544,544,544,544,544},
                                    {142,140,544,544,544,544,544},
                                    {142,141,544,544,544,544,544},
                                    {142,142,544,544,544,142,142}
                                   };

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->Token->setReadOnly(true);
    ui->Error->setReadOnly(true);
    ui->Token_2->setReadOnly(true);
    ui->tablaCuadruplos->setColumnCount(5);
    QStringList titulos;
    titulos<<"#"<<"oper"<<"op1"<<"op2"<<"res";
    ui->tablaCuadruplos->setHorizontalHeaderLabels(titulos);
}

MainWindow::~MainWindow()
{
    delete ui;
}
QString pilaOp;
QString pilaT;
QString pilaOper;
QString pilaSal;
QString evaluaElemento(int elemento){
    if(elemento>=700){
        return "";
    }
    switch(elemento){
    case -1:
        return "ε";
    case -2:
        return "MFF";
    case 1:
        return "PROGRAM";
        break;
    case 2:
        return "A";
        break;
    case 3:
```

```
        return "DECLARA-LIB";
        break;
    case 4:
        return "DECLARA";
        break;
    case 5:
        return "B";
        break;
    case 6:
        return "C";
        break;
    case 7:
        return "TIPO";
        break;
    case 8:
        return "ESTATUTOS";
        break;
    case 9:
        return "EST_IF";
        break;
    case 10:
        return "D";
        break;
    case 11:
        return "EST_WHILE";
        break;
    case 12:
        return "EST_FOR";
        break;
    case 13:
        return "EXPR";
        break;
    case 14:
        return "E";
        break;
    case 15:
        return "EXPR2";
        break;
    case 16:
        return "F";
        break;
    case 17:
        return "EXPR3";
        break;
    case 18:
        return "G";
        break;
    case 19:
        return "EXPR4";
        break;
    case 20:
        return "H";
        break;
    case 21:
        return "EXPR5";
        break;
    case 22:
```

```
    return "I";
    break;
case 23:
    return "TERM";
    break;
case 24:
    return "J";
    break;
case 25:
    return "FACT";
    break;
case 26:
    return "OPREL";
    break;
case 27:
    return "EST_ENTER";
    break;
case 28:
    return "EST_ASIG";
    break;
case 29:
    return "EST_WRITE";
    break;
case 30:
    return "K";
    break;
case 31:
    return "L";
    break;
case 32:
    return "EST_READ";
    break;
case 33:
    return "M";
    break;
case 34:
    return "N";
    break;
case 36:
    return "$";
    break;
case 100:
    //cout<<"Palabra reservada"<<endl;

    break;
case 101:
    //cout<<"Identificador"<<endl;
    return "Identificador";
    break;
case 102:
    //cout<<"Entero"<<endl;
    return "Cteentera";
    break;
case 103:
    //cout<<"Constante decimal"<<endl;
    return "Ctereal";
    break;
```

```
case 104:
    //cout<<"Constante con notación cientifica"<<endl;
    return "Ctenotacion cientifica";
    break;
case 105:
    //cout<<"Constante caracter"<<endl;
    return "CteCaracter";
    break;
case 106:
    //cout<<"Constante String"<<endl;
    return "CteString";
    break;
case 107:
    //cout<<"Suma"<<endl;
    return "+";
    break;
case 108:
    //cout<<"Resta"<<endl;
    return "-";
    break;
case 109:
    //cout<<"Multiplicación"<<endl;
    return "*";
    break;
case 110:
    //cout<<"División"<<endl;
    return "/";
    break;
case 111:
    //cout<<"Modulo"<<endl;
    return "%";
    break;
case 112:
    //cout<<"Comentario"<<endl;

    break;
case 113:
    //cout<<"Operador Y"<<endl;
    return "&&";
    break;
case 114:
    //cout<<"Operador O"<<endl;
    return "||";
    break;
case 115:
    //cout<<"Operador NOT"<<endl;
    return "!";
    break;
case 116:
    //cout<<"Operador Diferente"<<endl;
    return "!=";
    break;
case 117:
    //cout<<"Igual"<<endl;
    return "==";
    break;
case 118:
```

```
    //cout<<"Mayor"<<endl;
    return ">";
    break;
case 119:
    //cout<<"Mayor Igual"<<endl;
    return ">=";
    break;
case 120:
    //cout<<"Menor"<<endl;
    return "<";
    break;
case 121:
    //cout<<"Menor Igual"<<endl;
    return "<=";
    break;
case 122:
    //cout<<"Asignación"<<endl;
    return "=";
    break;
case 123:
    //cout<<"Signo dos puntos"<<endl;
    return ":";
    break;
case 124:
    //cout<<"Signo de cierre"<<endl;
    return ";";
    break;
case 125:
    //cout<<"Punto"<<endl;

    break;
case 126:
    //cout<<"Parentesis abierto"<<endl;
    return "(";
    break;
case 127:
    //cout<<"Par"<<endl;
    return ")";
    break;
case 128:
    //cout<<"Corchete abierto"<<endl;

    break;
case 129:
    //cout<<"Corchete cerrado"<<endl;

    break;
case 130:
    //cout<<"Es una coma"<<endl;
    return ",";
    break;
case 131:
    //cout<<#Es una libreria"<<endl;
    return "Identifiacdorlibreria";
    break;
case 132:
    return "end";
```

```
        break;
    case 133:
        return "begin";
        break;
    case 134:
        return "class";
        break;
    case 135:
        return "import";
        break;
    case 136:
        return "def";
        break;
    case 137:
        return "as";
        break;
    case 138:
        return "integer";
        break;
    case 139:
        return "float";
        break;
    case 140:
        return "char";
        break;
    case 141:
        return "string";
        break;
    case 142:
        return "boolean";
        break;
    case 143:
        return "if";
        break;
    case 144:
        return "endif";
        break;
    case 145:
        return "else";
        break;
    case 146:
        return "endwhile";
        break;
    case 147:
        return "while";
        break;
    case 148:
        return "endfor";
        break;
    case 149:
        return "for";
        break;
    case 150:
        return "enter";
        break;
    case 151:
        return "write";
```

```
            break;
        case 152:
            return "read";
            break;
        default:
            return "desconocido";
        }
}
void imprimePilaOperandos(){
    for(int i=0;i<pilaOperandos.size();i++){
        pilaOp+=pilaOperandos.at(i)+" ";
    }
    pilaOp+="\n\n";
}
void imprimePilaTipos(){
    for(int i=0;i<pilaTipos.size();i++){
        pilaT+=evaluaElemento(pilaTipos.at(i))+" ";
    }
    pilaT+="\n\n";
}
void imprimePilaOperadores(){
    for(int i=0;i<pilaOperadores.size();i++){
        pilaOper+=evaluaElemento(pilaOperadores.at(i))+" ";
    }
    pilaOper+="\n\n";
}
void imprimePilaSaltos(){
    for(int i=0;i<pilaSaltos.size();i++)
        pilaSal+=QString::number(pilaSaltos.at(i))+" ";
    pilaSal+="\n\n";
}

void formaCuadruplo(QString n,QString oper,QString op1,QString
op2,QString res){
    cuadruplo *obj=new cuadruplo(n,oper,op1,op2,res);
    cuadruplos.append(QSharedPointer<cuadruplo>(obj));
}

int Relaciona(char c){
   int valor;

   if(c>=48 && c<=57){
       return 0;
   }
   if(c>=65 && c<=90){
       return 10;
   }
   if(c>=97 && c<=122){
       return 11;
   }

   switch(c){

   case '.': valor=1;
       break;
   case 'E': valor=2;
       break;
```

```
    case 'e': valor=3;
        break;
    case '+': valor=4;
        break;
    case '-': valor=5;
        break;
    case 32:   valor=7;
        break;
    case '\n': valor=8;
                break;
    case '\t': valor=9;
        break;
    case '*': valor=12;
        break;
    case '/': valor=13;
        break;
        case 39: valor=14;
    break;
    case '"': valor=15;
        break;
    case '&': valor=16;
        break;
    case '|': valor=17;
        break;
    case '!': valor=18;
        break;
    case '=': valor=19;
        break;
    case '<':valor=20;
        break;
    case '>': valor=21;
        break;
    case '[': valor=22;
        break;
    case ']': valor=23;
        break;
    case '(': valor=24;
        break;
    case ')': valor=25;
        break;
    case ':': valor=26;
        break;
    case ';': valor=27;
        break;
    case '_': valor=28;
        break;
    case '%': valor=29;
        break;
    case ',': valor=30;
        break;

    default: valor=6;
    }
    return valor;
}
QString Tokens;
QString textoA;
```

```
QString texto;
void Token(int e){
    switch(e){
    case 100:
        //cout<<"Palabra reservada"<<endl;
        Tokens+=textoA+" Palabra reservada\n";
        break;
    case 101:
        //cout<<"Identificador"<<endl;
        Tokens+=textoA+" Identificador\n";
        break;
    case 102:
        //cout<<"Entero"<<endl;
        Tokens+=textoA+" Constante Entera\n";
        break;
    case 103:
        //cout<<"Constante decimal"<<endl;
        Tokens+=textoA+" Constante decimal\n";
        break;
    case 104:
        //cout<<"Constante con notación cientifica"<<endl;
        Tokens+=textoA+" Constante con notacion cientifica\n";
        break;
    case 105:
        //cout<<"Constante caracter"<<endl;
        Tokens+=textoA+" Constante caracter\n";
        break;
    case 106:
        //cout<<"Constante String"<<endl;
        Tokens+=textoA+" Constante String\n";
        break;
    case 107:
        //cout<<"Suma"<<endl;
        Tokens+=textoA+" Suma\n";
        break;
    case 108:
        //cout<<"Resta"<<endl;
        Tokens+=textoA+" Resta\n";
        break;
    case 109:
        //cout<<"Multiplicación"<<endl;
        Tokens+=textoA+" Multiplicacion\n";
        break;
    case 110:
        //cout<<"División"<<endl;
        Tokens+=textoA+" Division\n";
        break;
    case 111:
        //cout<<"Modulo"<<endl;
        Tokens+=textoA+" Modulo\n";
        break;
    case 112:
        //cout<<"Comentario"<<endl;
        Tokens+=textoA+" Comentario\n";
        break;
    case 113:
        //cout<<"Operador Y"<<endl;
```

```
    Tokens+=textoA+" Operador Y\n";
    break;
case 114:
    //cout<<"Operador O"<<endl;
    Tokens+=textoA+" Operador O\n";
    break;
case 115:
    //cout<<"Operador NOT"<<endl;
    Tokens+=textoA+" Operador NOT\n";
    break;
case 116:
    //cout<<"Operador Diferente"<<endl;
    Tokens+=textoA+" Operador Diferente\n";
    break;
case 117:
    //cout<<"Igual"<<endl;
    Tokens+=textoA+" Operador Igual\n";
    break;
case 118:
    //cout<<"Mayor"<<endl;
    Tokens+=textoA+" Operador Mayor\n";
    break;
case 119:
    //cout<<"Mayor Igual"<<endl;
    Tokens+=textoA+" Operador Mayor igual\n";
    break;
case 120:
    //cout<<"Menor"<<endl;
    Tokens+=textoA+" Operador Menor\n";
    break;
case 121:
    //cout<<"Menor Igual"<<endl;
    Tokens+=textoA+" Operador Menor Igual\n";
    break;
case 122:
    //cout<<"Asignación"<<endl;
    Tokens+=textoA+" Operador de Asignacion\n";
    break;
case 123:
    //cout<<"Signo dos puntos"<<endl;
    Tokens+=textoA+" Signo dos puntos\n";
    break;
case 124:
    //cout<<"Signo de cierre"<<endl;
    Tokens+=textoA+" Signo de cierre ;\n";
    break;
case 125:
    //cout<<"Punto"<<endl;
    Tokens+=textoA+" Punto\n";
    break;
case 126:
    //cout<<"Parentesis abierto"<<endl;
    Tokens+=textoA+" Parentesis abierto\n";
    break;
case 127:
    //cout<<"Parentesis cerrado"<<endl;
    Tokens+=textoA+" Paretesis cerrado\n";
```

```
    break;
case 128:
    //cout<<"Corchete abierto"<<endl;
    Tokens+=textoA+" Corchete abierto\n";
    break;
case 129:
    //cout<<"Corchete cerrado"<<endl;
    Tokens+=textoA+" Corchete cerrado\n";
    break;
case 130:
    //cout<<"Es una coma"<<endl;
    Tokens+=textoA+" Es una coma\n";
    break;
case 131:
    //cout<<#Es una libreria"<<endl;
    Tokens+=textoA+" Es una libreria\n";
    break;
case 132:
    Tokens+=textoA+" Palabra reservada end\n";
    break;
case 133:
    Tokens+=textoA+" Palabra reservada begin\n";
    break;
case 134:
    Tokens+=textoA+" Palabra reservada class\n";
    break;
case 135:
    Tokens+=textoA+" Palabra reservada import\n";
    break;
case 136:
    Tokens+=textoA+" Palabra reservada def\n";
    break;
case 137:
    Tokens+=textoA+" Palabra reservada as\n";
    break;
case 138:
    Tokens+=textoA+" Palabra reservada integer\n";
    break;
case 139:
    Tokens+=textoA+" Palabra reservada float\n";
    break;
case 140:
    Tokens+=textoA+" Palabra reservada char\n";
    break;
case 141:
    Tokens+=textoA+" Palabra reservada string\n";
    break;
case 142:
    Tokens+=textoA+" Palabra reservada boolean\n";
    break;
case 143:
    Tokens+=textoA+" Palabra reservada if\n";
    break;
case 144:
    Tokens+=textoA+" Palabra reservada endif\n";
    break;
case 145:
```

```
            Tokens+=textoA+" Palabra reservada else";
            break;
        case 146:
            Tokens+=textoA+" Palabra reservada endwhile\n";
            break;
        case 147:
            Tokens+=textoA+" Palabra reservada while\n";
            break;
        case 148:
            Tokens+=textoA+" Palabra reservada endfor\n";
            break;
        case 149:
            Tokens+=textoA+" Palabra reservada for\n";
            break;
        case 150:
            Tokens+=textoA+" Palabra reservada enter\n";
            break;
        case 151:
            Tokens+=textoA+" Palabra reservada write\n";
            break;
        case 152:
            Tokens+=textoA+" Palabra reservada read\n";
            break;
        case 153:
            Tokens+=textoA+" Palabra reservada principal\n";
            break;
        case 154:
            Tokens+=textoA+" Palabra reservada elseif\n";
            break;
        case 155:
            Tokens+=textoA+" Palabra reservada do\n";
            break;
        case 156:
            Tokens+=textoA+" Palabra reservada function\n";
            break;
        case 157:
            Tokens+=textoA+" Palabra reservada endfunction\n";
            break;
        case 158:
            Tokens+=textoA+" Palabra reservada null\n";
            break;
        case 159:
            Tokens+=textoA+" Palabra reservada include\n";
            break;
        }

}
QString errores;
void Errores(int e){
    switch(e){
    case 500:
        //cout<<"Error este simbolo no forma parte del lenguaje"<<endl;
        errores+=textoA+" Error este simbolo no forma parte del
lenguaje\n";
        break;
    case 501:
```

```
        //cout<<"Error no puede declarar un identificador de esa
manera"<<endl;
        errores+=textoA+" Error no puede declarar un identificador de esa
manera\n";
        break;
    case 502:
        //cout<<"Error no puede escribir una constante usando
letras"<<endl;
        errores+=textoA+" Error no puede escribir una constante usando
letras\n";
        break;
    case 503:
        //cout<<"Error debe de escribir un numero o un signo positivo o
negativo despues de la E o e"<<endl;
        errores+=textoA+" Error debe de escribir un numero o un signo
positivo o negativo despues de la E o e\n";
        break;
    case 504:
        //cout<<"Error debe de escribir un numero para completar la
expresión"<<endl;
        errores+=textoA+" Error debe de escribir un numero para completar
la expresion\n";
        break;
    case 505:
        //cout<<"Error debe de escribir algo entre las '"<<endl;
        errores+=textoA+" Error debe escribir un caracter entre las '\n";
        break;
    case 506:
        //cout<<"Error indefinido"<<endl;
        errores+=textoA+" Error no cerro la '\n";
        break;
    case 507:
        //cout<<"No puede escribir algo entre la expresión"<<endl;
        errores+=textoA+" No puede escribir algo entre la expresion Y\n";
        break;
    case 508:
        //cout<<"No puede escribir algo entre la expresión"<<endl;
        errores+=textoA+" No puede escribir algo entre la expresion O\n";
        break;
    case 509:
        //cout<<"No puedes llamar a una libreria de esa manera
        errores+=textoA+" No puede declara una libreria de esa manera\n";
    case 510:
        errores+=textoA+" 510: Se esperaba la palabra class o import al
inicio\n";
        break;
    case 511:
        errores+=textoA+" 511: Se esperaba la palabra class\n";
        break;
    case 512:
        errores+=textoA+" 512: Se esperaba la palabra import o class\n";
        break;
    case 513:
        errores+=textoA+" 513: Se esperaba un identificador, la palabra
def o el inicio de un estatuto\n";
        break;
    case 514:
```

```
            errores+=textoA+" 514: Se esperaba un identificador\n";
            break;
        case 515:
            errores+=textoA+" 515: Se esperaba la palabra as o ','\n";
            break;
        case 516:
            errores+=textoA+" 516: Se esperaba un tipo de dato\n";
            break;
        case 517:
            errores+=textoA+" 517: Se esperaba un identificador o el  inicio
o bien de un estatuto\n";
            break;
        case 518:
            errores+=textoA+" 518: Se esperaba la palabra if\n";
            break;
        case 519:
            errores+=textoA+" 519: Se esperaba la palabra else o endif\n";
            break;
        case 520:
            errores+=textoA+" 520: Se esperaba la palabra while\n";
            break;
        case 521:
            errores+=textoA+" 521: Se esperaba la palabra for\n";
            break;
        case 522:
            errores+=textoA+" 522: La expresión no puede iniciar de esa
manera\n";
            break;
        case 523:
            errores+=textoA+" 523: Se esperaba un parentesis cerrado, el
operador || o un signo de puntuación\n";
            break;
        case 524:
            errores+=textoA+" 524: La expresión no puede iniciar de esa
manera\n";
            break;
        case 525:
            errores+=textoA+" 525: Se esperaba un parentesis cerrado, el
operador || o un signo de puntuación\n";
            break;
        case 526:
            errores+=textoA+" 526: La expresión no puede iniciar de esa
manera\n";
            break;
        case 527:
            errores+=textoA+" 527: La expresión no puede iniciar de esa
manera\n";
            break;
        case 528:
            errores+=textoA+" 528: La expresión no puede iniciar de esa
manera\n";
            break;
        case 529:
            errores+=textoA+" 529: Se esperaba un operador relacional o un
signo de puntuación\n";
            break;
        case 530:
```

```
        errores+=textoA+" 530: La expresión no puede iniciar de esa
forma\n";
        break;
    case 531:
        errores+=textoA+" 531: Se esperaba un parentesis cerrado, un
operador o un signo de puntuación\n";
        break;
    case 532:
        errores+=textoA+" 532: La expresión no puede iniciar de esa
forma\n";
        break;
    case 533:
        errores+=textoA+" 533: Se esperaba un parentesis cerrado, un
operador o un signo de puntuación\n";
        break;
    case 534:
        errores+=textoA+" 534: Se esperaba un tipo de constante\n";
        break;
    case 535:
        errores+=textoA+" 535: Se esperaba un operador relacional\n";
        break;
    case 536:
        errores+=textoA+" 536: Se esperaba la palabra enter\n";
        break;
    case 537:
        errores+=textoA+" 537: Se esperaba un identificador\n";
        break;
    case 538:
        errores+=textoA+" 538: Se esperaba la palabra write\n";
        break;
    case 539:
        errores+=textoA+" 539: La expresión no puede iniciar de esa
forma\n";
        break;
    case 540:
        errores+=textoA+" 540: Se esperaba un identificador o una
coma\n";
        break;
    case 541:
        errores+=textoA+" 541: Se esperaba la palabra read\n";
        break;
    case 542:
        errores+=textoA+" 542: Se esperaba un identificador\n";
        break;
    case 543:
        errores+=textoA+" 543: Se esperaba un identificador o una
coma\n";
        break;
    case 544:
        errores+="544: Error entre tipos\n";
        break;
    case 545:
        errores+=textoA+" 545: Variable no declarada\n";
        break;
    default:
        errores+="Revisa tu sintaxis quizas pusiste algo que no debe
ir\n";
```

```
            break;
        }

    }

    int evaluaPR(){
        int conta=0;
        std::string cadenaStd = textoA.toStdString();
        for(int i=0;i<textoA.length();i++){
            char car=cadenaStd[i];
            if(car=='\n' || car=='\t' || car==32)
                conta++;
        }
        QString temp=textoA.mid(conta,textoA.length());

        if(textoA=="import" || temp=="import"){
            return 135;
        }
        if(textoA=="class" || temp=="class"){
            return 134;
        }
        if(textoA=="begin" || temp=="begin"){
            return 133;
        }
        if(textoA=="end" || temp=="end"){
            return 132;
        }
        if(textoA=="def" || temp=="def"){
            return 136;
        }
        if(textoA=="as" || temp=="as"){
            return 137;
        }
        if(textoA=="integer" || temp=="integer"){
            return 138;
        }
        if(textoA=="float" || temp=="float"){
            return 139;
        }
        if(textoA=="char" || temp=="char"){
            return 140;
        }
        if(textoA=="string" || temp=="string"){
            return 141;
        }
        if(textoA=="boolean" || temp=="boolean"){
            return 142;
        }
        if(textoA=="if" || temp=="if"){
            return 143;
        }
        if(textoA=="endif" || temp=="endif"){
            return 144;
        }
        if(textoA=="else" || temp=="else"){
            return 145;
        }
```

```
    if(textoA=="endwhile" || temp=="endwhile"){
        return 146;
    }
    if(textoA=="while" || temp=="while"){
        return 147;
    }
    if(textoA=="endfor" || temp=="endfor"){
        return 148;
    }
    if(textoA=="for" || temp=="for"){
        return 149;
    }
    if(textoA=="enter" || temp=="enter"){
        return 150;
    }
    if(textoA=="write" || temp=="write"){
        return 151;
    }
    if(textoA=="read" || temp=="read"){
        return 152;
    }
    if(textoA=="principal" || temp=="principal"){
        return 153;
    }
    if(textoA=="elseif" || temp=="elseif"){
        return 154;
    }
    if(textoA=="do" || temp=="do"){
        return 155;
    }
    if(textoA=="function" || temp=="function"){
        return 156;
    }
    if(textoA=="endfunction" || temp=="endfunction"){
        return 157;
    }
    if(textoA=="null" || temp=="null"){
        return 158;
    }
    if(textoA=="include" || temp=="include"){
        return 159;
    }
    return 101;
}

int Analiza(QString cadena){
    std::string cadenaStd = cadena.toStdString();
    edo=0;
    int col;
    char car=cadenaStd[0];

    textoA="";

    int numero=1;
    while(edo<=32){

        col=Relaciona(car);
```

```
edo=matriz[edo][col];

if(edo==107)
    textoA.append('+');
if(edo==108)
    textoA.append('-');
if(edo==109)
    textoA.append('*');
if(edo==125)
    textoA.append('.');
if(edo==128)
    textoA.append('[');
if(edo==129)
    textoA.append(']');
if(edo==126)
    textoA.append('(');
if(edo==127)
    textoA.append(')');
if(edo==124)
    textoA.append(';');
if(edo==123)
    textoA.append(':');
if(edo==111)
    textoA.append('%');
if(edo==130)
    textoA.append(',');

if(edo<100 || edo>=500){
    textoA.append(car);

}
if(edo==103 && (car=='e' || car=='E')){
    if(car=='e')
        textoA.append('e');
    if(car=='E')
        textoA.append('E');
    edo=7;
}



if(edo==119 && car=='='){
    textoA.append('=');
    edo=119;
}

if(edo==121 && car=='='){
    textoA.append('=');
    edo=121;
}

if(edo==116 && car=='='){
    textoA.append('=');
    edo=116;
}
```

```
if(edo==115 && car=='='){
    textoA.append('=');
    edo=116;
}

if(edo==117 && car=='='){
    textoA.append('=');
    edo=117;
}



if(edo==112 && car=='/'){
    textoA.append('/');
    edo=112;
}

if(edo==10 && car!=39){
    textoA.append(car);
    edo=11;
}else if(edo==10 && car!=39){
    textoA.append(car);
    edo=105;
}
if(edo==105 && car==39){
    textoA.append(car);
    edo=105;
}
if(edo==100){
    edo=evaluaPR();
}

if(edo==22 && car=='l'){
    edo=22;
}

if(edo==22 && car=='y'){
    edo=22;
}

if(edo==22 && car=='e'){
    edo=131;
}

if(edo==101 && car=='.'){
    textoA.append('.');
    edo=22;
}

if(edo==106 && car=='"'){
    textoA.append(car);
    edo=106;
}

car=cadenaStd[numero];

numero++;
```

```
    }
    if(textoA=="&&" || textoA.contains("&&")){
        edo=113;
    }else if (textoA.contains("&")) {
        edo=507;
    }
    if(textoA=="||" || textoA.contains("||")){
        edo=114;
    }else if (textoA.contains("|")) {
        edo=508;
    }

    //Elimina lo que hay enfrente de una cadena
    int conta=0;

    for(int i=0;i<textoA.length();i++){
        char car=cadenaStd[i];
        if(car==' ' || car=='\t' || car==32 )
            conta++;
    }
    int longitud=textoA.length();
    texto=texto.remove(0,longitud);
    if(conta!=0){
        textoA.remove(0,conta);
    }

    if(edo>=100 && edo<=199){
        Token(edo);
    }else{
        Errores(edo);
    }

return edo;
}

int RelacionaGramatica(int estado){
    switch(estado){
    case 100:
        //cout<<"Palabra reservada"<<endl;

        break;
    case 101:
        //cout<<"Identificador"<<endl;
        return 3;
        break;
    case 102:
        //cout<<"Entero"<<endl;
        return 36;
        break;
    case 103:
        //cout<<"Constante decimal"<<endl;
        return 37;
        break;
    case 104:
        //cout<<"Constante con notación cientifica"<<endl;
        return 38;
```

```
    break;
case 105:
    //cout<<"Constante caracter"<<endl;
    return 39;
    break;
case 106:
    //cout<<"Constante String"<<endl;
    return 40;
    break;
case 107:
    //cout<<"Suma"<<endl;
    return 31;
    break;
case 108:
    //cout<<"Resta"<<endl;
    return 32;
    break;
case 109:
    //cout<<"Multiplicación"<<endl;
    return 33;
    break;
case 110:
    //cout<<"División"<<endl;
    return 34;
    break;
case 111:
    //cout<<"Modulo"<<endl;
    return 35;
    break;
case 112:
    //cout<<"Comentario"<<endl;

    break;
case 113:
    //cout<<"Operador Y"<<endl;
    return 22;
    break;
case 114:
    //cout<<"Operador O"<<endl;
    return 21;
    break;
case 115:
    //cout<<"Operador NOT"<<endl;
    return 23;
    break;
case 116:
    //cout<<"Operador Diferente"<<endl;
    return 26;
    break;
case 117:
    //cout<<"Igual"<<endl;
    return 25;
    break;
case 118:
    //cout<<"Mayor"<<endl;
    return 29;
    break;
```

```
case 119:
    //cout<<"Mayor Igual"<<endl;
    return 30;
    break;
case 120:
    //cout<<"Menor"<<endl;
    return 27;
    break;
case 121:
    //cout<<"Menor Igual"<<endl;
    return 28;
    break;
case 122:
    //cout<<"Asignación"<<endl;
    return 24;
    break;
case 123:
    //cout<<"Signo dos puntos"<<endl;
    return 45;
    break;
case 124:
    //cout<<"Signo de cierre"<<endl;
    return 46;
    break;
case 125:
    //cout<<"Punto"<<endl;

    break;
case 126:
    //cout<<"Parentesis abierto"<<endl;
    return 13;
    break;
case 127:
    //cout<<"Parentesis cerrado"<<endl;
    return 14;
    break;
case 128:
    //cout<<"Corchete abierto"<<endl;

    break;
case 129:
    //cout<<"Corchete cerrado"<<endl;

    break;
case 130:
    //cout<<"Es una coma"<<endl;
    return 44;
    break;
case 131:
    //cout<<#Es una libreria"<<endl;
    return 1;
    break;
case 132:
    return 47;
    break;
case 133:
    return 4;
```

```
    break;
case 134:
    return 2;
    break;
case 135:
    return 0;
    break;
case 136:
    return 5;
    break;
case 137:
    return 6;
    break;
case 138:
    return 7;
    break;
case 139:
    return 8;
    break;
case 140:
    return 9;
    break;
case 141:
    return 10;
    break;
case 142:
    return 11;
    break;
case 143:
    return 12;
    break;
case 144:
    return 16;
    break;
case 145:
    return 15;
    break;
case 146:
    return 18;
    break;
case 147:
    return 17;
    break;
case 148:
    return 20;
    break;
case 149:
    return 19;
    break;
case 150:
    return 41;
    break;
case 151:
    return 42;
    break;
case 152:
    return 43;
```

```
        break;
    default:
        return 48;
    }
}

void LlenarPilaProduccion(int fila){
    for(int i=0;i<15;i++){
        if(producciones[fila][i]!=0){
        pilaEjecucion.push(producciones[fila][i]);
        }
    }
}


QString pasosPila;
void imprimePila(){
    for(int i=0;i<pilaEjecucion.size();i++){
        QString ele=evaluaElemento(pilaEjecucion.at(i));
        pasosPila+=ele+" ";
    }
    pasosPila+="\n\n";
}

int buscaTipo(){
    int pos=-1;
    if(!pilaOperandosBusqueda.empty()){
        for(int i=0;i<pilaOperandosBusqueda.size();i++){
            if(textoA.contains(pilaOperandosBusqueda.at(i))){
                pos=i;
                break;
            }
        }
    }
    return pos;
}
int relacionaMatrizTipos(int edo){
    if(edo>=107 && edo<=109)
        return 2;
    if(edo==110)
        return 3;
    if(edo==111)
        return 4;
    if(edo>=116 && edo<=121)
        return 5;
    if(edo>=113 && edo<=115)
        return 6;
}
void imprimeYLimpiaPilas(){
    pilaTipos.pop();
    imprimePilaTipos();
    pilaTipos.pop();
    imprimePilaTipos();
}

//LLenar la tabla con cuadruplos
void MainWindow::LlenarCuadruplo(){
```

```
    int con=0;
    while(con<cuadruplos.size()){
        ui->tablaCuadruplos->insertRow(ui->tablaCuadruplos->rowCount());
        int fila=ui->tablaCuadruplos->rowCount()-1;
        ui->tablaCuadruplos->setItem(fila,0,new
QTableWidgetItem(cuadruplos.at(con)->getN()));
        ui->tablaCuadruplos->setItem(fila,1,new
QTableWidgetItem(cuadruplos.at(con)->getOper()));
        ui->tablaCuadruplos->setItem(fila,2,new
QTableWidgetItem(cuadruplos.at(con)->getOp1()));
        ui->tablaCuadruplos->setItem(fila,3,new
QTableWidgetItem(cuadruplos.at(con)->getOp2()));
        ui->tablaCuadruplos->setItem(fila,4,new
QTableWidgetItem(cuadruplos.at(con)->getRes()));
        con++;
    }
}
void relacionaTiposOper(){
    int op1=pilaTipos.top();
    int op2=pilaTipos.at(pilaTipos.size()-2);
    if(pilaOperadores.top()>=107 && pilaOperadores.top()<=111){
        int fila=0;
        for(int i=0;i<25;i++){
            if(matrizDeTipos[i][0]==op1 && matrizDeTipos[i][1]==op2){
                fila=i;
                break;
            }
        }
        int columna=relacionaMatrizTipos(pilaOperadores.top());
        int supuesto=matrizDeTipos[fila][columna];
        if(supuesto<500){
            imprimeYLimpiaPilas();
            pilaTipos.push(supuesto);
            QString oper=evaluaElemento(pilaOperadores.top());
            pilaOperadores.pop();
            imprimePilaOperadores();
            QString op2=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString op1=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString res="R"+QString::number(++contRes);
            pilaOperandos.push(res);
            imprimePilaOperandos();

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,op2,res);

        }else{
            Errores(544);
            supuesto=op1;
            pilaTipos.pop();
            imprimePilaTipos();
            pilaTipos.pop();
            imprimePilaTipos();
            pilaTipos.push(supuesto);
            imprimePilaTipos();
```

```
            QString oper=evaluaElemento(pilaOperadores.top());
            pilaOperadores.pop();
            imprimePilaOperadores();
            QString op2=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString op1=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString res="R"+QString::number(++contRes);
            pilaOperandos.push(res);
            imprimePilaOperandos();

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,op2,res);

            sinError=false;
        }
    }else if(pilaOperadores.top()>=116 && pilaOperadores.top()<=121){
        int fila=0;
        for(int i=0;i<25;i++){
            if(matrizDeTipos[i][0]==op1 && matrizDeTipos[i][1]==op2){
                fila=i;
                break;
            }
        }
        int columna=relacionaMatrizTipos(pilaOperadores.top());
        int supuesto=matrizDeTipos[fila][columna];
        if(supuesto<500){
            imprimeYLimpiaPilas();
            pilaTipos.push(supuesto);
            QString oper=evaluaElemento(pilaOperadores.top());
            pilaOperadores.pop();
            imprimePilaOperadores();
            QString op2=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString op1=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString res="R"+QString::number(++contRes);
            pilaOperandos.push(res);
            imprimePilaOperandos();

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,op2,res);
        }else{
            Errores(544);
            supuesto=op1;
            pilaTipos.pop();
            imprimePilaTipos();
            pilaTipos.pop();
            imprimePilaTipos();
            pilaTipos.push(supuesto);
            imprimePilaTipos();
            QString oper=evaluaElemento(pilaOperadores.top());
            pilaOperadores.pop();
            imprimePilaOperadores();
            QString op2=pilaOperandos.top();
```

```
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString op1=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString res="R"+QString::number(++contRes);
            pilaOperandos.push(res);
            imprimePilaOperandos();

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,op2,res);
            sinError=false;
        }
    }else if(pilaOperadores.top()>=113 && pilaOperadores.top()<=115){
        int fila=0;
        for(int i=0;i<25;i++){
            if(matrizDeTipos[i][0]==op1 && matrizDeTipos[i][1]==op2){
                fila=i;
                break;
            }
        }
        int columna=relacionaMatrizTipos(pilaOperadores.top());
        int supuesto=matrizDeTipos[fila][columna];
        if(supuesto<500){
            imprimeYLimpiaPilas();
            pilaTipos.push(supuesto);
            QString oper=evaluaElemento(pilaOperadores.top());
            pilaOperadores.pop();
            imprimePilaOperadores();
            QString op2=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString op1=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString res="R"+QString::number(++contRes);
            pilaOperandos.push(res);
            imprimePilaOperandos();

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,op2,res);

        }else{
            Errores(544);
            supuesto=op1;
            pilaTipos.pop();
            imprimePilaTipos();
            pilaTipos.pop();
            imprimePilaTipos();
            pilaTipos.push(supuesto);
            imprimePilaTipos();
            QString oper=evaluaElemento(pilaOperadores.top());
            pilaOperadores.pop();
            imprimePilaOperadores();
            QString op2=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();
            QString op1=pilaOperandos.top();
            pilaOperandos.pop();
```

```
            imprimePilaOperandos();
            QString res="R"+QString::number(++contRes);
            pilaOperandos.push(res);
            imprimePilaOperandos();

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,op2,res);

            sinError=false;
        }
    }
}

void accionesSemanticayCodigoIntermedio(int accion){
    bool existeOperando=false;
    QString oper,op1,op2,res;
    switch(accion){
        case 700:
            pilaOperandos.push(textoA);
            pilaOperandosBusqueda.push(textoA);
            for(int i=0;i<pilaOperandosBusqueda.size();i++){
                if(pilaOperandosBusqueda.size()>1){

if(pilaOperandosBusqueda.at(i).contains(pilaOperandos.top()))
                        existeOperando=true;
                        break;
                }
            }
            if(existeOperando){
                sinError=false;
                pilaOperandos.pop();
                pilaOperandosBusqueda.pop();
                errores+=textoA+" Variable ya definida\n";
                existeOperando=false;
            }
            imprimePilaOperandos();
            break;
        case 701:
            while(!pilaOperandos.isEmpty()){
                pilaTiposBusqueda.push(edo);
                pilaOperandos.pop();
            }
            imprimePilaOperandos();
            break;
        case 702:
            if(edo>=102 && edo<=106){
                if(edo==102){
                    pilaTipos.push(138);
                    imprimePilaTipos();
                }
                if(edo==103 || edo==104){
                    pilaTipos.push(139);
                    imprimePilaTipos();
                }
                if(edo==105){
                    pilaTipos.push(140);
                    imprimePilaTipos();
                }
```

```
                if(edo==106){
                    pilaTipos.push(141);
                    imprimePilaTipos();
                }
            }else{
                int i=buscaTipo();
                if(i==-1){
                    Errores(545);
                    pilaTipos.push(138);
                    imprimePilaTipos();
                    sinError=false;
                }else{
                    pilaTipos.push(pilaTiposBusqueda.at(i));
                    imprimePilaTipos();
            }
            }
            pilaOperandos.push(textoA);
            if(estaPresenteRead || estaPresenteWrite){
                imprimePilaOperandos();
                contadorRaW++;
            }
            break;
        case 703:
            pilaOperadores.push(edo);
            imprimePilaOperadores();
            break;
        case 704:
            relacionaTiposOper();
            break;
        case 705:
            pilaOperadores.push(-2);
            imprimePilaOperadores();
            break;
        case 706:
            pilaOperadores.pop();
            imprimePilaOperadores();
            break;
        case 707://Accion salto en falso
            //Logica de la accion: pila_operandos.push()y
pila_saltos(contcuadruplos).
            oper="SF";
            op1=pilaOperandos.top();
            pilaOperandos.pop();
            imprimePilaOperandos();

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,"","");
            pilaSaltos.append(contCuadruplo);
            imprimePilaSaltos();
            break;
        case 708:
            //Se hace un salto incondicional,
tope_saltos(contcuadruplos+1) y pila:saltos(pos_actual)
            for(int i=0;i<cuadruplos.size();i++){
                if(cuadruplos.at(i)-
>getN()==QString::number(pilaSaltos.top())){
                    cuadruplos.at(i)-
>setRes(QString::number(contCuadruplo+2));
```

```
                    break;
                }
            }
            pilaSaltos.pop();
            imprimePilaSaltos();
            oper="SI";

formaCuadruplo(QString::number(++contCuadruplo),oper,"","","");
            pilaSaltos.append(contCuadruplo);
            imprimePilaSaltos();
            break;
         case 709:
            //Rellenar tope_saltos(ContCuadruplo+1)
            for(int i=0;i<cuadruplos.size();i++){
                if(cuadruplos.at(i)-
>getN()==QString::number(pilaSaltos.top())){
                    cuadruplos.at(i)-
>setRes(QString::number(contCuadruplo+1));
                    break;
                }
            }
            pilaSaltos.pop();
            imprimePilaSaltos();
            break;
         case 717://Concatena lo necesario para estatutos write y read
            for(int i=0;i<contadorRaW;i++){
                if(estaPresenteRead){
                   oper="read";
                }
                if(estaPresenteWrite){
                   oper="write";
                }

formaCuadruplo(QString::number(++contCuadruplo),oper,"","",pilaOperandos.
at(i));

            }

            while(!pilaOperandos.isEmpty()){
                pilaOperandos.remove(0);
                imprimePilaOperandos();
            }
            contadorRaW=0;
            estaPresenteRead=false;
            estaPresenteWrite=false;
            break;
        case 716://Agrega el enter, forma cuadruplo y elimina el tope de
la pila
            pilaOperadores.push(150);
            imprimePilaOperadores();

formaCuadruplo(QString::number(++contCuadruplo),"enter","","","");
            pilaOperadores.pop();
            imprimePilaOperadores();
            break;
        case 715://Guardar siguiente cuadruplo para el ciclo while
            pilaSaltos.push(contCuadruplo+1);
```

```
            imprimePilaSaltos();
            break;
        case 714://Llenar top-1 con contador de cuadruplos,llenar top de
la pila de saltos con contadorcuadruplos+1 y limpiar pilas
            oper="SI";

formaCuadruplo(QString::number(++contCuadruplo),oper,"","",QString::numbe
r(pilaSaltos.at(pilaSaltos.size()-2)));
            for(int i=0;i<cuadruplos.size();i++){
                if(cuadruplos.at(i)-
>getN()==QString::number(pilaSaltos.top())){
                    cuadruplos.at(i)-
>setRes(QString::number(contCuadruplo+1));
                    break;
                }
            }
            pilaSaltos.pop();
            imprimePilaSaltos();
            pilaSaltos.pop();
            imprimePilaSaltos();
            break;
        case 713:
            if(cuadruplos.at(cuadruplos.size()-1)->getOper()=="<" ||
cuadruplos.at(cuadruplos.size()-1)->getOper()=="<="){
                oper="SF";
                op1=pilaOperandos.top();
                pilaOperandos.pop();
                imprimePilaOperandos();
            }else{
                oper="SV";
                op1=pilaOperandos.top();
                pilaOperandos.pop();
                imprimePilaOperandos();
            }

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,"","");
            pilaSaltos.append(contCuadruplo);
            imprimePilaSaltos();
            break;
        case 712://Guardar siguiente cuadruplo para el ciclo for
            pilaSaltos.push(contCuadruplo+2);
            imprimePilaSaltos();
        case 711://Formar cuadruplo de asignación
            int op1=pilaTipos.top();
            int op2=pilaTipos.at(pilaTipos.size()-2);
            if(pilaOperadores.top()==122){
                    if(op1==op2){
                        imprimeYLimpiaPilas();
                        QString
oper=evaluaElemento(pilaOperadores.top());
                        pilaOperadores.pop();
                        imprimePilaOperadores();
                        QString op2="";
                        QString res=pilaOperandos.top();
                        pilaOperandos.pop();
                        imprimePilaOperandos();
                        QString op1=pilaOperandos.top();
```

```
                        pilaOperandos.pop();
                        imprimePilaOperandos();

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,op2,res);

                    }else{
                        Errores(544);
                        imprimeYLimpiaPilas();
                        QString
oper=evaluaElemento(pilaOperadores.top());
                        pilaOperadores.pop();
                        imprimePilaOperadores();
                        QString op2="";
                        QString res=pilaOperandos.top();
                        pilaOperandos.pop();
                        imprimePilaOperandos();
                        QString op1=pilaOperandos.top();
                        pilaOperandos.pop();
                        imprimePilaOperandos();

formaCuadruplo(QString::number(++contCuadruplo),oper,op1,op2,res);

                        sinError=false;
                    }
            }
            break;


    }
}

void ConstruyeGramatica(){
    int token=0,edoMP=0;
    int colMP=0,filaMP=0;
    while(!pilaEjecucion.empty()){
        pilaEjecucion.pop();
    }
    pilaEjecucion.push('$');
    pilaEjecucion.push(1);
    bool quieroToken=true,llena=true;
    imprimePila();
    sinError=true;
    while(!pilaEjecucion.empty()){

        if(quieroToken){
            token=Analiza(texto);
            quieroToken=false;
        }

        if(llena){
            colMP=RelacionaGramatica(token);
            filaMP=pilaEjecucion.top()-1;
            pilaEjecucion.pop();
            edoMP=matrizPredictiva[filaMP][colMP]-1;
            LlenarPilaProduccion(edoMP);
            llena=false;
            imprimePila();
```

```
        }
        if(pilaEjecucion.top()>=100){
            if(pilaEjecucion.top()==token){
                pilaEjecucion.pop();
                if(texto!=""){
                    quieroToken=true;
                }
                imprimePila();
                if(pilaEjecucion.top()>=700 && pilaEjecucion.top()<800){

accionesSemanticayCodigoIntermedio(pilaEjecucion.top());
                    pilaEjecucion.pop();
                }
                if(token==151){
                    estaPresenteWrite=true;
                }
                if(token==152){
                    estaPresenteRead=true;
                }
                if((token>=101 && token<=106) || token==127){
                    if(!pilaOperadores.empty()){
                        imprimePilaOperandos();
                        accionesSemanticayCodigoIntermedio(704);
                    }
                }


            }else{
                QString tr=evaluaElemento(token);
                QString tp=evaluaElemento(pilaEjecucion.top());
                errores+="Hay errores en la sintaxis porque se esperaba
un "+tp+" y se recibio un "+tr+"\n";
                token=pilaEjecucion.top();
                sinError=false;
            }
        }else{
            if(edoMP>500){
                Errores(edoMP+1);
                sinError=false;
                pilaEjecucion.pop();
                imprimePila();
            }else{
                if(pilaEjecucion.top()>0 && pilaEjecucion.top()<=34){
                    llena=true;
                }
            if(pilaEjecucion.top()==-1){
                pilaEjecucion.pop();
                imprimePila();
            }
            if(pilaEjecucion.top()=='$'){
                pilaEjecucion.pop();
                imprimePila();
                if(sinError){
                QMessageBox msgBox;
                msgBox.setText("Se llego al final de la entrada y no hay
errores");
                msgBox.exec();
```

```
                break;
                }
            }
            }

        }

    }
}

void MainWindow::on_pushButton_clicked()
{
    ui->tablaCuadruplos->setRowCount(0);
    cuadruplos.clear();
    contCuadruplo=0;
    contRes=0;
    Tokens="";
    errores="";
    pasosPila="";
    pilaOp="";
    pilaT="";
    pilaOper="";
    pilaSal="";
    ui->Token->setPlainText("");
    ui->Error->setPlainText("");
    ui->Token_2->setPlainText("");
    ui->pilaOperandos->setPlainText("");
    ui->pilaTipos->setPlainText("");
    ui->pilaOperadores->setPlainText("");
    ui->pilaSaltos->setPlainText("");
    pilaSaltos.clear();
    pilaTipos.clear();
    pilaOperandosBusqueda.clear();
    pilaTiposBusqueda.clear();
    pilaOperadores.clear();
    texto=ui->textoAnalizar->toPlainText();
    ConstruyeGramatica();
    if(errores!=""){
        QMessageBox::about(this,"Mensaje","La sintaxis contiene algunos
errores léxicos o sintácticos o semánticos");
    }
    ui->Token->appendPlainText(Tokens);
    ui->Error->appendPlainText(errores);
    ui->Token_2->appendPlainText(pasosPila);
    ui->pilaOperandos->appendPlainText(pilaOp);
    ui->pilaTipos->appendPlainText(pilaT);
    ui->pilaOperadores->appendPlainText(pilaOper);
    ui->pilaSaltos->appendPlainText(pilaSal);
    LlenarCuadruplo();
}

void MainWindow::on_pushButton_2_clicked()
{
    ui->textoAnalizar->setPlainText("");
   QString ruta=QFileDialog::getOpenFileName(
                this,tr("Abrir archivo"),
                "C://",
```

```
                "Eyement (*.eye)");
    if(ruta!=""){
    QFile inputFile(ruta);
    if (inputFile.open(QIODevice::ReadOnly))
    {
       QTextStream in(&inputFile);
       while (!in.atEnd())
       {
          QString line = in.readLine();
          ui->textoAnalizar->appendPlainText(line);
       }
       inputFile.close();
    }


}

}



void MainWindow::on_pushButton_3_clicked()
{
ui->textoAnalizar->setPlainText("");
    ui->tablaCuadruplos->setRowCount(0);
    cuadruplos.clear();
    contCuadruplo=0;
    contRes=0;
    Tokens="";
    errores="";
    pasosPila="";
    pilaOp="";
    pilaT="";
    pilaOper="";
    pilaSal="";
    ui->Token->setPlainText("");
    ui->Error->setPlainText("");
    ui->Token_2->setPlainText("");
    ui->pilaOperandos->setPlainText("");
    ui->pilaTipos->setPlainText("");
    ui->pilaOperadores->setPlainText("");
    ui->pilaSaltos->setPlainText("");
    pilaSaltos.clear();
    pilaTipos.clear();
    pilaOperandosBusqueda.clear();
    pilaTiposBusqueda.clear();
  pilaOperadores.clear();
}

void MainWindow::on_pushButton_4_clicked()
{

    QString fileName = QFileDialog::getSaveFileName(this,
            tr("Guardar archivo"), "",
            tr("Eyement (*.eye)"));
    if (fileName.isEmpty())
            return;
```

```
        else {
            QFile file(fileName);
            if (!file.open(QIODevice::WriteOnly)) {
                QMessageBox::information(this, tr("Unable to open file"),
                    file.errorString());
                return;
            }
    }
    QFile file(fileName);
    if ( file.open(QIODevice::ReadWrite) ){
        QTextStream stream( &file );
        stream <<ui->textoAnalizar->toPlainText()<< endl;
    }

}
```

Diccionario de datos

Constantes: enteras, reales sin exponente, de tipo carácter y de tipo cadena.(’\n’ y ’$’ (delimitadora de cadena)) .

Operadores aritméticos: +, -, *, /, %, ^

Operadores de comparación:<,>,#

Operadores lógicos o booleanos:&, |, !

Operadores de manejo de punteros:& y *

Símbolo de la sentencia de asignación: =

Paréntesis, corchetes, punto, llaves, comas, dos puntos y punto y coma.

Tipos de datos: INT, FLOAT, CHAR, STRING, BOOLEAN

Palabras reservadas: END, BEGIN, CLASS, IMPORT, DEF, AS, INTEGER, FLOAT, CHAR, STRING, BOOLEAN, IF, ENDIF, ELSE, ENDWHILE, WHILE, ENFFOR, FOR, ENTER, WRITE, READ, PRINCIPAL, ELSEIF, DO, FUNCTION, ENDFUNCTION, NULL e INCLUDE

Procedimientos estándar de entrada/salida: READ(parámetros), WRITE(parámetros), OPEN (cadena de caracteres o puntero a cadena), CLOSE, READ (parámetros) y WRITE(parámetros).

Comentarios: Comienzan por \ y acaban en fin de línea

3.3. Controles de auditoria implementados en el sistema.

Dentro de la auditoría se tomarán en cuenta lo siguiente:

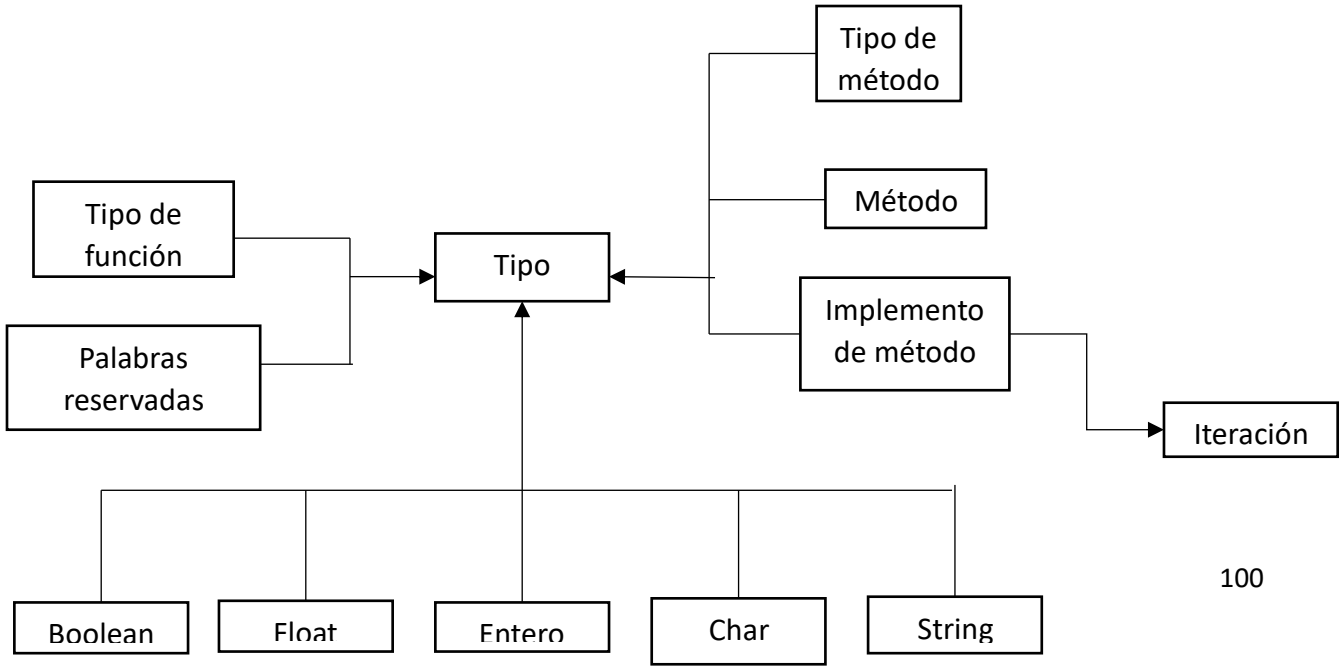
- Manuales propios de sus procesos y actuaciones.
- Detección de riesgos propios en relación con sus funciones y roles.
- Examen sobre la calidad de los controles existentes.
- Resultados de las operaciones realizadas en el analizador

Se busca corregir las debilidades en el programa

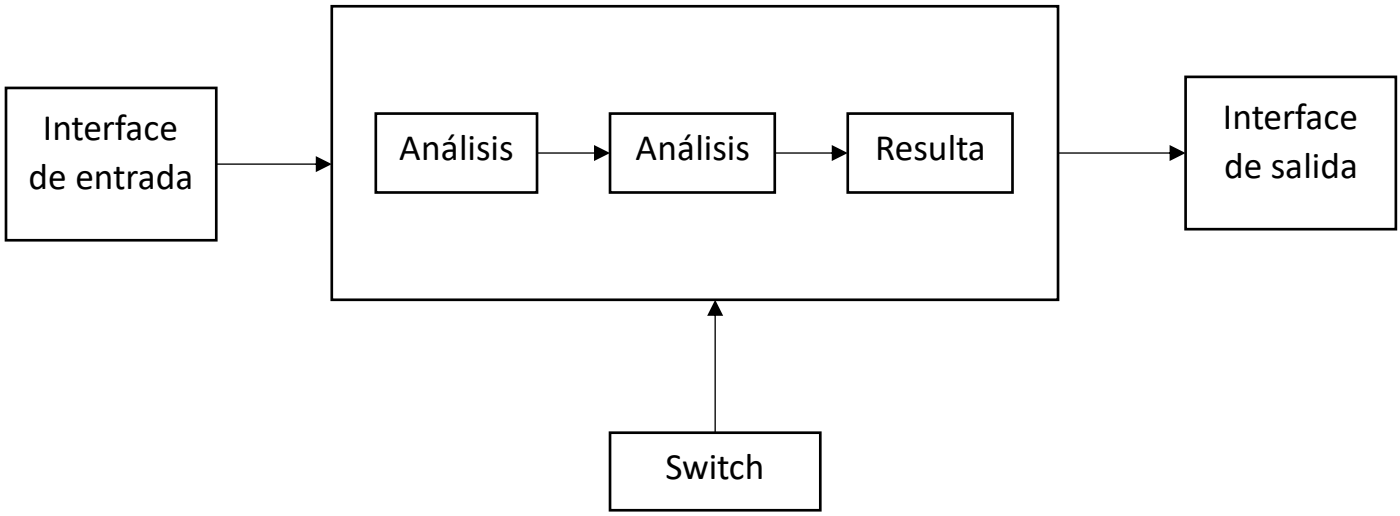
Requerimientos del sistema

1. El usuario introducirá una cadena al sistema por medio de un periférico no perteneciente al dispositivo lógico reconfigurable.
2. A esta cadena se le hará un análisis léxico con el cual se validará si los caracteres, por los cuales está conformada, pertenecen alfabeto definido.
    → Dependiendo del resultado del análisis anterior el sistema enviará una señal de éxito en caso de que se encuentren todos los caracteres de la cadena en el alfabeto y de error si ocurre lo contrario.
3. Si la señal enviada es aceptada el análisis sintáctico del sistema empezará.
4. El análisis sintáctico retomará los tokens del análisis léxico y los procesará con las reglas establecidas que tendrá el lenguaje.
    → Dependiendo del resultado del análisis anterior el sistema enviará un mensaje de éxito en caso de que la sintaxis de la cadena sea la correcta y un mensaje de error si no obedece las reglas establecidas.
5. Cuando el sistema termine de procesar la cadena volverá al estado inicial en espera de una nueva cadena.
6. No se necesita licencia de software para el lenguaje de programación que utilizamos para el analizador.

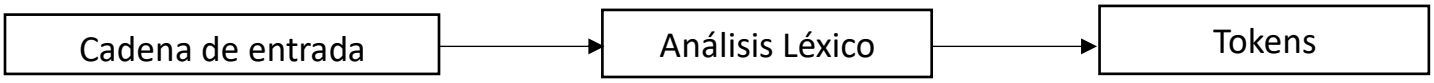
Modelo lógico de datos, diagrama entidad-relación.



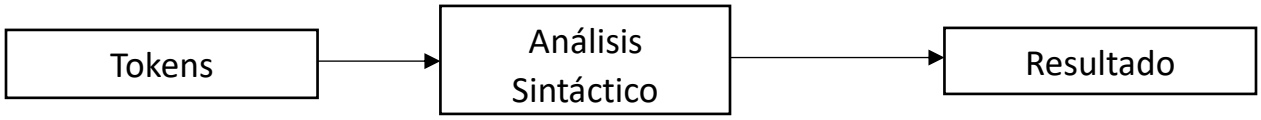
Diseño del sistema



Diseño de alto nivel



Diseño de analizador léxico



Diseño de analizador sintáctico

Matriz de procesos versus organización.

Q\T	import	id_lib	class	id	begin	Def	as	integer	float	char	string	boolean
PROGRAM	1	510	1	510	510	510	510	510	510	510	510	510
A	511	511	2	511	511	511	511	511	511	511	511	511
DELCARA_LIB	4	512	3	512	512	512	512	512	512	512	512	512
DECLARA	513	513	513	5	513	6	513	513	513	513	513	513
B	514	514	514	7	514	514	514	514	514	514	514	514
C	515	515	515	515	515	515	9	515	515	515	515	515
TIPO	516	516	516	516	516	516	516	10	11	12	13	14
ESTATUTOS	517	517	517	16	517	517	517	517	517	517	517	517

EST_IF	518	518	518	518	518	518	518	518	518	518	518	518
D	519	519	519	519	519	519	519	519	519	519	519	519
EST_WHILE	520	520	520	520	520	520	520	520	520	520	520	520
EST_FOR	521	521	521	521	521	521	521	521	521	521	521	521
EXPR	522	522	522	28	522	522	522	522	522	522	522	522
E	523	523	523	523	523	523	523	523	523	523	523	523
EXPR2	524	524	524	31	524	524	524	524	524	524	524	524
F	525	525	525	525	525	525	525	525	525	525	525	525
EXPR3	526	526	526	34	526	526	526	526	526	526	526	526
G	527	527	527	35	527	527	527	527	527	527	527	527
EXPR4	528	528	528	37	528	528	528	528	528	528	528	528
H	529	529	529	529	529	529	529	529	529	529	529	529
EXPR5	530	530	530	40	530	530	530	530	530	530	530	530
I	531	531	531	531	531	531	531	531	531	531	531	531
TERM	532	532	532	44	532	532	532	532	532	532	532	532
J	533	533	533	533	533	533	533	533	533	533	533	533
FACT	534	534	534	49	534	534	534	534	534	534	534	534
OPREL	535	535	535	535	535	535	535	535	535	535	535	535
EST_ENTER	536	536	536	536	536	536	536	536	536	536	536	536
EST_ASIG	537	537	537	63	537	537	537	537	537	537	537	537
EST_WRITE	538	538	538	538	538	538	538	538	538	538	538	538
K	539	539	539	65	539	539	539	539	539	539	539	539
L	540	540	540	540	540	540	540	540	540	540	540	540
EST_READ	541	541	541	541	541	541	541	541	541	541	541	541
M	542	542	542	69	542	542	542	542	542	542	542	542
N	543	543	543	543	543	543	543	543	543	543	543	543

Matriz de procesos parte 1: La Q representa la producción (lado izquierdo de la matriz) y el los números representan los estados a los que pasa según elemento (T).

if	(	)	else	endif	while	endwhile	for	endfor	||	&&	!	=	==	!=	<
510	510	510	510	510	510	510	510	510	510	510	510	510	510	510	510
511	511	511	511	511	511	511	511	511	511	511	511	511	511	511	511
512	512	512	512	512	512	512	512	512	512	512	512	512	512	512	512
5	513	513	513	513	5	513	5	513	513	513	513	513	513	513	513
514	514	514	514	514	514	514	514	514	514	514	514	514	514	514	514
515	515	515	515	515	515	515	515	515	515	515	515	515	515	515	515
516	516	516	516	516	516	516	516	516	516	516	516	516	516	516	516
17	517	517	15	15	18	15	19	15	517	517	517	517	517	517	517
23	518	518	518	518	518	518	518	518	518	518	518	518	518	518	518
519	519	519	25	24	519	519	519	519	519	519	519	519	519	519	519
520	520	520	520	520	26	520	520	520	520	520	520	520	520	520	520
521	521	521	521	521	521	521	27	521	521	521	521	521	521	521	521
522	28	522	522	522	522	522	522	522	522	522	36	522	522	522	522
523	523	29	523	523	523	523	523	523	30	523	523	523	523	523	523
524	55	524	524	524	524	524	524	524	524	524	36	524	524	524	524
525	525	32	525	525	525	525	525	525	32	33	525	525	525	525	525
526	34	526	526	526	526	526	526	526	526	526	36	526	526	526	526
527	35	527	527	527	527	527	527	527	527	527	36	527	527	527	527
528	37	528	528	528	528	528	528	528	528	528	528	528	528	528	528
529	529	38	529	529	529	529	529	529	38	38	529	529	39	39	39
530	40	530	530	530	530	530	530	530	530	530	530	530	530	530	530
531	531	41	531	531	531	531	531	531	41	41	531	531	41	41	41
532	44	532	532	532	532	532	532	532	532	532	532	532	532	532	532
533	533	45	533	533	533	533	533	533	45	45	533	533	45	45	45
534	55	534	534	534	534	534	534	534	534	534	534	534	534	534	534
535	535	535	535	535	535	535	535	535	535	535	535	535	56	57	58
536	536	536	536	536	536	536	536	536	536	536	536	536	536	536	536
537	537	537	537	537	537	537	537	537	537	537	537	537	537	537	537
538	538	538	538	538	538	538	538	538	538	538	538	538	538	538	538
539	65	539	539	539	539	539	539	539	539	539	36	539	539	539	539
540	540	66	540	540	540	540	540	540	540	540	540	540	540	540	540
541	541	541	541	541	541	541	541	541	541	541	541	541	541	541	541
542	542	542	542	542	542	542	542	542	542	542	542	542	542	542	542
543	543	70	543	543	543	543	543	543	543	543	543	543	543	543	543

Matriz de procesos parte 2

<=	>	>=	+	-	*	/	%	cteentera	ctereal	ctenotacion	ctecaracter	ctestring
510	510	510	510	510	510	510	510	510	510	510	510	510
511	511	511	511	511	511	511	511	511	511	511	511	511
512	512	512	512	512	512	512	512	512	512	512	512	512
513	513	513	513	513	513	513	513	513	513	513	513	513
514	514	514	514	514	514	514	514	514	514	514	514	514
515	515	515	515	515	515	515	515	515	515	515	515	515
516	516	516	516	516	516	516	516	516	516	516	516	516
517	517	517	517	517	517	517	517	517	517	517	517	517
518	518	518	518	518	518	518	518	518	518	518	518	518
519	519	519	519	519	519	519	519	519	519	519	519	519
520	520	520	520	520	520	520	520	520	520	520	520	520
521	521	521	521	521	521	521	521	521	521	521	521	521
522	522	522	522	522	522	522	522	28	28	28	28	28
523	523	523	523	523	523	523	523	523	523	523	523	523
524	524	524	524	524	524	524	524	31	31	31	31	31
525	525	525	525	525	525	525	525	525	525	525	525	525
526	526	526	526	526	526	526	526	34	34	34	34	34
527	527	527	527	527	527	527	527	35	35	35	35	35
528	528	528	528	528	528	528	528	37	37	37	37	37
39	39	39	529	529	529	529	529	529	529	529	529	529
530	530	530	530	530	530	530	530	40	40	40	40	40
41	41	41	42	43	531	531	531	531	531	531	531	531
532	532	532	532	532	532	532	532	44	44	44	44	44
45	45	45	45	45	46	47	48	533	533	533	533	533
534	534	534	534	534	534	534	534	50	51	52	53	54
59	60	61	535	535	535	535	535	535	535	535	535	535
536	536	536	536	536	536	536	536	536	536	536	536	536
537	537	537	537	537	537	537	537	537	537	537	537	537
538	538	538	538	538	538	538	538	538	538	538	538	538
539	539	539	539	539	539	539	539	65	65	65	65	65
540	540	540	540	540	540	540	540	540	540	540	540	540
541	541	541	541	541	541	541	541	541	541	541	541	541
542	542	542	542	542	542	542	542	542	542	542	542	542
543	543	543	543	543	543	543	543	543	543	543	543	543

Matriz de procesos parte 3

enter	write	read	,	:	;	end
510	510	510	510	510	510	510
511	511	511	511	511	511	511
512	512	512	512	512	512	512
5	5	5	513	513	513	5
514	514	514	514	514	514	514
515	515	515	8	515	515	515
516	516	516	516	516	516	516
22	21	20	517	517	517	15
518	518	518	518	518	518	518
519	519	519	519	519	519	519
520	520	520	520	520	520	520
521	521	521	521	521	521	521
522	522	522	522	522	522	522
523	523	523	29	29	29	523
524	524	524	524	524	524	524
525	525	525	32	32	32	525
526	526	526	526	526	526	526
527	527	527	527	527	527	527
528	528	528	528	528	528	528
529	529	529	38	38	38	529
530	530	530	530	530	530	530
531	531	531	41	41	41	531
532	532	532	532	532	532	532
533	533	533	45	45	45	533
534	534	534	534	534	534	534
535	535	535	535	535	535	535
62	536	536	536	536	536	536
537	537	537	537	537	537	537
538	64	538	538	538	538	538
539	539	539	539	539	539	539
540	540	540	67	540	540	540
541	541	68	541	541	541	541
542	542	542	542	542	542	542
543	543	543	71	543	543	543

Matriz de procesos parte 4

La siguiente lista son las producciones que contiene la matriz anterior en forma organizacional

PROGRAM -> DECLARA-LIB A

A -> class id begin DECLARA ESTATUTOS end

DECLARA-LIB -> A

DECLARA-LIB -> import id_lib ; DECLARA-LIB

DECLARA -> ε

DECLARA -> def B

B -> id C

C -> , B

C -> as TIPO ; DECLARA

TIPO -> integer

TIPO -> float

TIPO -> char

TIPO -> string

TIPO -> boolean

ESTATUTOS -> ε

ESTATUTOS -> EST_ASIG ; ESTATUTOS

ESTATUTOS -> EST_IF ; ESTATUTOS

ESTATUTOS -> EST_WHILE ; ESTATUTOS

ESTATUTOS -> EST_FOR ; ESTATUTOS

ESTATUTOS -> EST_READ ; ESTATUTOS

ESTATUTOS -> EST_WRITE ; ESTATUTOS

ESTATUTOS -> EST_ENTER ; ESTATUTOS

EST_IF -> if (EXPR) ESTATUTOS D

D -> endif

D -> else ESTATUTOS endif

EST_WHILE -> while (EXPR) ESTATUTOS endwhile

EST_FOR -> for (EST_ASIG : EXPR) ESTATUTOS endfor

EXPR -> EXPR2 E

E -> ε

E -> || EXPR

EXPR2 -> EXPR3 F

F -> ε

F -> && EXPR2

EXPR3 -> G

G -> EXPR4

G -> ! EXPR4

EXPR4 -> EXPR5 H

H -> ε

H -> OPREL EXPR5

EXPR5 -> TERM I

I -> ε

I -> + EXPR5

I -> - EXPR5

TERM -> FACT J

J -> ε

J -> * TERM

J -> / TERM

J -> % TERM

FACT -> id

FACT -> cteentera

FACT -> ctenotacion

FACT -> ctecaracter

FACT -> ctestring

FACT -> (EXPR)

OPREL -> ==

OPREL -> !=

OPREL -> <

OPREL -> <=

OPREL -> >

OPREL -> >=

EST_ENTER -> enter

EST_ASIG -> id = EXPR

EST_WRITE -> write (K

K -> EXPR L

L ->)

L -> , K

EST_READ -> read (M

M -> id N

N ->)

N -> , M

Matriz de programas versus entidades.

Q\Z	d	.	E	e	+	-	Teclado	/b	\n	\t	L	l	*	/	'
0	4	125	500	500	107	108	500	0	0	0	2	1	109	13	10
1	2	100	100	100	100	100	100	100	100	100	2	1	100	100	100
2	2	22	101	101	101	101	101	101	101	101	2	2	101	101	101
3	2	501	501	501	501	501	501	501	501	501	2	2	501	501	501
4	4	5	102	102	102	102	102	102	102	102	102	102	102	102	102
5	6	502	502	502	502	502	502	502	502	502	502	502	502	502	502
6	6	103	7	7	103	103	103	103	103	103	103	103	103	103	103
7	9	503	503	503	8	8	503	503	503	503	503	503	503	503	503
8	9	504	504	504	504	504	504	504	504	504	504	504	504	504	504
9	9	104	104	104	104	104	104	104	104	104	104	104	104	104	104
10	11	11	11	11	11	11	11	11	11	11	11	11	11	11	505
11	506	506	506	506	506	506	506	506	506	506	506	506	506	506	105
12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
13	110	110	110	110	110	110	110	110	110	110	110	110	14	110	110
14	14	14	14	14	14	14	14	14	14	14	14	14	15	14	14
15	14	14	14	14	14	14	14	14	14	14	14	14	15	112	14
16	507	507	507	507	507	507	507	507	507	507	507	507	507	507	507

17	508	508	508	508	508	508	508	508	508	508	508	508	508	508	508
18	115	115	115	115	115	115	115	115	115	115	115	115	115	115	115
19	122	122	122	122	122	122	122	122	122	122	122	122	122	122	122
20	118	118	118	118	118	118	118	118	118	118	118	118	118	118	118
21	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120
22	509	509	509	509	509	509	509	509	509	509	509	509	22	509	509	509

Matriz léxica parte 1

"	&	|	!	=	<	>	[	]	(	)	:	;	_	%	,	lye
12	16	17	18	19	21	20	128	129	126	127	123	124	500	111	130	101
100	100	100	100	100	100	100	100	100	100	100	100	100	3	100	100	100
101	101	101	101	101	101	101	101	101	101	101	101	101	3	101	101	101
501	501	501	501	501	501	501	501	501	501	501	501	501	501	501	501	501
102	102	102	102	102	102	102	102	102	102	102	102	102	102	102	102	102
502	502	502	502	502	502	502	502	502	502	502	502	502	502	502	502	502
103	103	103	103	103	103	103	103	103	103	103	103	103	103	103	103	103
503	503	503	503	503	503	503	503	503	503	503	503	503	503	503	503	503
504	504	504	504	504	504	504	504	504	504	504	504	504	504	504	504	504
104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104	104
11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
506	506	506	506	506	506	506	506	506	506	506	506	506	506	506	506	506
106	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
110	110	110	110	110	110	110	110	110	110	110	110	110	110	110	110	110
14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
507	113	507	507	507	507	507	507	507	507	507	507	507	507	507	507	507
508	508	114	508	508	508	508	508	508	508	508	508	508	508	508	508	508
115	115	115	115	116	115	115	115	115	115	115	115	115	115	115	115	115
122	122	122	122	117	122	122	122	122	122	122	122	122	122	122	122	122
118	118	118	118	119	118	118	118	118	118	118	118	118	118	118	118	118
120	120	120	120	121	120	120	120	120	120	120	120	120	120	120	120	120
509	509	509	509	509	509	509	509	509	509	509	509	509	509	509	509	131

Matriz léxica parte 2

Áreas de aplicación y/o alcance de los procedimientos. Esfera de acción que cubren los procedimientos

El analizador se usaría para traducir instrucciones de un lenguaje de alto nivel a instrucciones que la computadora puede interpretar y ejecutar. Para cada lenguaje de programación se requiere un compilador separado.

El compilador traduce todo el programa antes de ejecutarlo.

GLOSARIO

Token: a thing serving as a visible or tangible representation of a fact, quality, feeling, etc. (una representación visible o tangible de un hecho, cualidad, sentimiento, etc.)

Sintaxis: Disciplina lingüística que estudia el orden y la relación de las palabras o sintagmas en la oración, así como las funciones que cumplen.

Operandos: Los operandos pueden ser constantes, variables o llamadas a funciones, siempre que éstas devuelvan algún valor.

Operador: Símbolo utilizado en matemáticas para indicar la operación que se realiza entre los elementos que une o la relación que existe entre ellos.

ENLACES

https://docs.google.com/presentation/d/1DrmozwfKEMZjlLkfU9hsHylQGgy6emHMb2s2XBVPqmE/edit?usp=sharing

VII. ASESORÍA Y APOYO TÉCNICO

Bernardo Salinas Jaquez (618 297 7925)

Jaime Andres De La Cruz Cortés (618 320 2921)

Fernanda Rodríguez Catarino (618 113 2280)

DIRECTORIO

Diseño y Desarrollo Del Sistema

Bernardo Salinas Jaquez

Elaboración del Manual De Usuario y Técnico

Jaime Andres De La Cruz Cortes

Fernanda Rodríguez Catarino

Coordinador de Desarrollo

Rivera Saucedo Elda