TECNOLÓGICO NACIONAL DE MÉXICO EN CELAYA

DEPARTAMENTO DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

ACTIVIDAD 3

LENGUAJES Y AUTOMATAS II

DOCENTE DESIGNADO: ISC. RICARDO GONZÁLEZ GONZÁLEZ
EQUIPO 1. INTEGRANTES:
HERNANDEZ MARTINEZ ALINA ADRIANA
AVILA CARDENAS RICARDO
CID SOTO PABLO EMILIO
RICO GÓMEZ EDUARDO DANIEL

CELAYA, GTO A 15 MAYO 2020

POR MEDIO DE LA PRESENTE MANIFESTAMOS QUE:

LOS INTEGRANTES DEL EQUIPO NÚMERO **1**, ELABORAMOS LAS SIGUIENTES EVIDENCIAS ACADÉMICAS PARA CUMPLIR CON EL TEMARIO PROPUESTO EN LA MATERIA **LENGUAJES DE INTERFAZ** DE LA ESPECIALIDAD DE **INGENIERIA EN SISTEMAS COMPUTACIONALES QUE** CORRESPONDE A LA **EVALUACIÓN 3 DEL CICLO ESCOLAR ENERO-JUNIO 2020,** EN EL TECNOLÓGICO NACIONAL DE MÉXICO EN CELAYA.

HERNANDEZ MARTINEZ ALINA ADRIANA

CID SOTO PABLO EMILIO

RICO GÓMEZ EDUARDO DANIEL

AVILA CARDENAS RICARDO

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

DEPARTAMENTO DE SISTEMAS COMPUTACIONALES E INFORMÁTICA

ASUNTO: SOLICITUD DE ACTIVIDADES

Celaya, Guanajuato, 02/Mayo/2020

LENGUAJES Y AUTÓMATAS II
DOCENTE DESIGNADO: ISC. RICARDO GONZÁLEZ GONZÁLEZ
SEMESTRE ENERO-JUNIO 2020

ACTIVIDAD 3 (VALOR 100 PUNTOS)

LEA CUIDADOSAMENTE, Y REALICE LAS SIGUIENTE ACTIVIDADES, CONSIDERANDO LOS CRITERIOS DE CALIDAD PROPUESTOS EN LOS DOCUMENTOS DE LA GUÍA TUTORIAL, Y LA RÚBRICA DE EVALUACIÓN,

EL LECTOR DEBE TOMAR MUY EN CUENTA QUE ESTA ACTIVIDAD ES UN EXAMEN, Y NO DE UNA TAREA SENCILLA, PUES DEMANDA TIEMPO PARA INVESTIGAR, LEER, ANALIZAR, REDACTAR, ILUSTRAR Y PROPONER DE MANERA PROFESIONAL LOS TEMAS PROPUESTOS EN LA ESTRUCTURA TEMÁTICA DE ESTA MATERIA.

1. TOMANDO EN CUENTA TODAS LAS INDICACIONES Y EXPLICACIONES EN CLASE Y UNA VEZ DEFINIDA LA GRAMÁTICA Y EL LENGUAJE PROTOTIPO A IMPLEMENTAR, COMO EQUIPO INVESTIGUEN, DISEÑEN E IMPLEMENTEN LA PRIMERA FASE O ETAPA DE UN ANALIZADOR SINTÁCTICO.

2. LA ACTIVIDAD COMO EQUIPO DEBERÁ COMENZAR CON LA INVESTIGACIÓN Y FUNDAMENTACIÓN DE LOS SIGUIENTES TEMAS.

   A. INVESTIGAR. ¿ QUÉ ES UN ANÁLISIS SINTÁCTICO APLICADO A LA VALORACIÓN DE UN LENGUAJE FORMAL ?

   B. INVESTIGAR. ¿ EN QUÉ CONSISTE UN ANÁLISIS SINTÁCTICO Y QUÉ LO CARACTERIZA?

   C. IDENTIFICAR. ¿ QUÉ CASOS DE ESTUDIOS SON LOS IMPORTANTES A CONSIDERAR EN EL ANÁLISIS SINTÁCTICO ?

   D. INVESTIGAR Y PROPONER. ¿ QUÉ PROCESOS Y PROBLEMAS ATIENDE UN ANÁLISIS SINTÁCTICO ?

   E. INVESTIGAR. ¿ CÓMO IMPLEMENTAR UTILIZANDO UN ANÁLISIS SINTÁCTICO ?

NOTA : ESTOS TEMAS DEBEN SER EL RESULTADO DE UNA INVESTIGACIÓN Y ANÁLISIS COMO EQUIPO, Y NO UNA TRANSCRIPCIÓN DE TEMAS AISLADOS, PUES EN TODO MOMENTO LAS FUENTES DE CONSULTA DEBEN SER LA BASE DEL DESARROLLO DE ESTE PUNTO Y SUS APARTADOS.

3. COMO EQUIPO Y DERIVADO DEL ANÁLISIS ANTERIOR SE DEBEN PROPONER LOS ALGORITMOS Y ESTRUCTURAS DE DATOS NECESARIAS PARA LA IMPLEMENTACIÓN DE UN PROTOTIPO DE ANALIZADOR SINTÁCTICO. ES DECIR, MANEJO Y OPERACIONES CON ARCHIVOS, TABLA DE SÍMBOLOS, DISEÑO DE AUTÓMATAS, PILA DE ERRORES, ETC.

   CONCRETAMENTE EN ESTE PUNTO DEBERÁN COMO EQUIPO, REDACTAR Y DETALLAR TODOS LOS ELEMENTOS QUE SE USARÁN PARA LA IMPLEMENTACIÓN DEL ANALIZADOR SINTÁCTICO.

4. PARA EL INCISO C DEL PUNTO 2 ANTERIOR, SE DEBERÁN CREAR PROGRAMAS DE MÍNIMO 25 LÍNEAS (SIN CONSIDERAR LOS COMENTARIOS) CADA UNO, EN LOS CUALES SE CODIFIQUE EN EL LENGUAJE PROTOTIPO, INSTRUCCIONES CON LÓGICA QUE EJEMPLIFIQUEN LOS ERRORES QUE EN CADA CASO DE ESTUDIO SE PROPONGAN.

   TALES PROGRAMAS DEBEN ESTAR PERFECTAMENTE DOCUMENTADOS Y CO-RELACIONADOS CON LOS CASOS DE ESTUDIO CORRESPONDIENTES.

5. CARACTERÍSTICAS QUE LA ACTIVIDAD 3 DEBE POSEER PARA CONSIDERARSE COMPLETA.

   A. PLANTEAMIENTO Y FUNDAMENTACIÓN DE LOS CASOS DE USO DEL ANALIZADOR SINTÁCTICO, ASÍ COMO LOS ERRORES EN QUE DERIVARÁ CADA UNO DE ELLOS.

   B. PREPARACIÓN DE LOS PROGRAMAS SUFICIENTES, ESCRITOS EN EL LENGUAJE PROTOTIPO, QUE APOYEN LA COMPROBACIÓN DE CADA CASO DE ESTUDIO. TALES PROGRAMAS DEBERÁN ESTAR COMPLETAMENTE DOCUMENTADOS.

   C. GENERACIÓN DE UN CATÁLOGO DE ERRORES, CORRELACIONADO A LOS CASOS DE USO PROPUESTOS.

   D. DISEÑO DE UNA SOLUCIÓN MODELADA EN OBJETOS, APOYADA EN DIAGRAMAS UML QUE DESCRIBAN LA ARQUITECTURA PROPUESTA PARA EL PROTOTIPO DEL ANALIZADOR SINTÁCTICO.

   E. PROPUESTA Y MODELADO DE LAS PROPIEDADES Y COMPORTAMIENTOS DE UN ANALIZADOR SINTÁCTICO.

   F. MODELADO E IMPLEMENTACIÓN DE UN PROCESO DE :

      LECTURA DE CÓDIGO FUENTE => TOKENIZACIÓN => CATEGORIZACIÓN DE LOS TOKENS => CONSTRUCCIÓN Y LLENADO DE LA TABLA DE SÍMBOLOS => MANEJO Y DESPLIEGUE DE ERRORES => DESPLIEGUE DE LA TABLA DE SÍMBOLOS RESULTANTE => GENERACIÓN DE UNA INTERFACE GRÁFICA DE OPERACIÓN => ANÁLISIS SINTÁCTICO DE LOS CASOS DE ESTUDIO MÁS IMPORTANTES .

G. GENERACIÓN DE UN CALENDARIO DE ACTIVIDADES PLANIFICADAS VS. ACTIVIDADES REALIZADAS.

H. GENERACIÓN DE UNA BITÁCORA DE INCIDENCIAS.

I. TODAS LAS EVIDENCIAS GENERADAS Y REUNIDAS DEBERÁN INTEGRARSE A UN ARCHIVO PDF, NOMBRADO COMO SE INDICA MÁS ADELANTE.

   ESTAS EVIDENCIAS PODRÁN SE ELABORAR CON HERRAMIENTAS ELECTRÓNICAS, COMO PROCESADOR DE TEXTO, DE IMÁGENES, HOJAS DE CÁLCULO, ETC.

J. EL NÚCLEO DE CADA ALGORITMO CODIFICADO TAMBIÉN DEBERÁ FORMAR PARTE DEL ARCHIVO DE EVIDENCIAS.

6. DADAS LAS RESTRICCIONES DE MOVILIDAD A QUE OBLIGAN LAS ACTUALES CONDICIONES SANITARIAS GLOBALES, Y PARA EFECTOS DE LAS REVISIONES EN LOS AVANCES DE LOS PROYECTOS, CADA EQUIPO DEBERÁ ABRIR UNA CUENTA DE YOUTUBE A FIN DE SUBIR MUY PERIÓDICAMENTE VIDEOS CON LOS LOGROS ALCANZADOS.

   LOS VIDEOS DEBERÁN SER EXPLICADOS POR TODOS LOS INTEGRANTES DEL EQUIPO, TANTO EN EL FUNCIONAMIENTO INTERNO DEL PROYECTO (NIVEL ALGORITMOS) COMO SU FUNCIONAMIENTO EXTERNO (NIVEL GUI).

   CUALQUIER DUDA O SUGERENCIA DEBERÁ SER HECHA USANDO LAS CUENTAS DE CORREO INSTITUCIONAL.

   ESTA EVIDENCIA ES MUY IMPORTANTE Y EQUIVALDRÁ A LAS REVISIONES PRESENCIALES QUE HEMOS HECHO ANTES PARA VALIDAR SUS PROYECTOS.

   POR ÚLTIMO, LES RECOMIENDO AMPLIAMENTE QUE SI EL TIEMPO LES PERMITE, AVANCEN MÁS ALLÁ DE ESTA ACTIVIDAD 3, Y HAGAN LOS PREPARATIVOS PARA COMENZAR EL SIGUIENTE ANÁLISIS, QUE COMO BIEN SABEN, ES EL SEMÁNTICO.

   ESTO SERÍA DE MUCHA UTILIDAD, PUES FORMARÁ PARTE DE LA ACTIVIDAD 4, QUE SERÁ PUBLICADA PRONTO.

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

NOTA A CONSIDERAR.

CADA UNO DE LOS PUNTOS ANTERIORES DEBE SER DESARROLLADO CON LA PROFUNDIDAD ACORDE A UN NIVEL PROFESIONAL, Y APEGÁNDOSE COMPLETAMENTE A LAS DIRECTRICES DE LA GUÍA TUTORIAL.

NO CONCIBA ESTE TRABAJO, COMO UN SIMPLE RESUMEN O EJERCICIO DE TRANSCRIPCIÓN, PUES EL VALOR INDICADO AL INICIO DE ESTA ACTIVIDAD LE DARÁ A USTED UNA BUENA IDEA DE LO QUE SE ESPERA DE ELLA, EN CUANTO A CALIDAD Y EL APRENDIZAJE OBTENIDO, MISMO QUE SERÁ PUESTO A PRUEBA MEDIANTE UN EXAMEN ESCRITO O BIEN ORAL EN CLASE.

SI DECIDIÓ ELABORAR ESTA ACTIVIDAD EN EQUIPO, CADA INTEGRANTE DE ÉSTE DEBERÁ POSEER EL MISMO NIVEL DE CONOCIMIENTO, PUES TAN SOLO REPARTIR TEMAS ENTRE LOS INTEGRANTES DEL EQUIPO, SUPONDRÍA UN GRAVE ERROR DE INTERPRETACIÓN A LA INTENCIÓN DIDÁCTICA REAL DE ESTA ACTIVIDAD.

POR ÚLTIMO, ESTA ACTIVIDAD SOLO SE PODRÁ DESARROLLAR EN EQUIPO, SI SE REGISTRÓ EN UNO PREVIAMENTE, UTILIZANDO EL FORMATO ENTREGADO EN LA ACTIVIDAD INICIAL. DE LO CONTRARIO DEBERÁ ELABORAR Y ENTREGAR LA ACTIVIDAD DE FORMA INDIVIDUAL.

LA ENTREGA DE DICHO REGISTRO SE HARÁ VÍA CORREO ELECTRÓNICO ENVIANDO ÉSTE AL PROFESOR DESIGNADO, Y POSTERIORMENTE EN CLASE ENTREGANDO LA HOJA EN FÍSICO.

NOTA GENERAL DE LA ACTIVIDAD :

SE DEBE CONSIDERAR Y TOMAR EN CUENTA PARA EL CORRECTO CUMPLIMIENTO DE ESTA ACTIVIDAD, LO SOLICITADO EN LA GUÍA TUTORIAL, CONCRETAMENTE EN EL PUNTO 3 INCISO i (*Trabajo en equipo*).

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

FIRMAS EN LA ACTIVIDAD

DADA LAS ACTUALES RESTRICCIONES DE CONFINAMIENTO QUE VIVIMOS ACTUALMENTE, PARA EL CASO DE LAS FIRMAS DEL DOCUMENTO, SE SUGIERE REALIZAR LO SIGUIENTE.

EN LUGAR DE QUE SE FIRMEN TODAS LAS HOJAS DE LA EVIDENCIA, POR CADA UNO DE LOS INTEGRANTES DEL EQUIPO, SE HARÁ UNA HOJA QUE DEBERÁ PRECEDER A LA PORTADA DEL TRABAJO DONDE SE LEA LA SIGUIENTE LEYENDA.

"LOS INTEGRANTES DEL EQUIPO NÚMERO ___ , ELABORAMOS LAS SIGUIENTES EVIDENCIAS ACADÉMICAS PARA CUMPLIR CON EL TEMARIO PROPUESTO EN LA MATERIA _____
DE LA ESPECIALIDAD DE _____ QUE CORRESPONDE A LA EVALUACIÓN 3 DEL CICLO ESCOLAR ENERO-JUNIO 2020, EN EL TECNOLÓGICO NACIONAL DE MÉXICO EN CELAYA.

NOMBRES Y FIRMAS DE CADA UNO DE LOS INTEGRANTES DEL EQUIPO "

ESTA HOJA SE FIRMARÁ POR CADA UNO DE LOS INTEGRANTES Y SE ENVIARÁ VÍA ELECTRÓNICA ENTRE TODOS A FIN DE QUE EL JEFE DE EQUIPO PUEDA INTEGRAR TODAS LAS FIRMAS AL TRABAJO FINAL.

ADEMÁS DE ESTO, CADA INTEGRANTE DEL EQUIPO DEBERÁ FIRMAR LAS HOJAS DEL TRABAJO QUE APORTE PARA ESTA ACTIVIDAD 3.

OBSERVACIONES:

- ✓ LA REVISIÓN SERÁ EN DIVERSAS VERTIENTES. PUEDE SER AL MOMENTO DE SOLICITAR LA CARPETA DE EVIDENCIAS, O BIEN PUEDE SER AL SOLICITAR LA EXPOSICIÓN DE LA TAREA EN LA CLASE. TAMBIÉN PUEDE SER POR SOLICITUD EXPRESA DEL INTERESADO A PARTICIPAR EN CLASE EXPONIENDO BREVEMENTE SU ACTIVIDAD.
- ✓ AQUELLAS ACTIVIDADES EN FORMATO DIGITAL SE DEBERÁN TENER SIEMPRE, Y EN TODO MOMENTO A LA MANO EN UNA MEMORIA USB.
- ✓ ÉSTAS ACTIVIDADES PODRÁN SER SOLICITADAS EN LA CLASE, O BIEN PARA SU ENVÍO A UNA CUENTA DE CORREO.
- ✓ ESTAS ACTIVIDADES DEBEN ESTAR LISTAS E INTEGRADAS A LA CARPETA DE EVIDENCIAS (*FÍSICAMENTE*) A LA FECHA DE ENTREGA INDICADA AL FINAL DE ÉSTE DOCUMENTO.
- ✓ CADA HOJA QUE ENTREGUE DE SU ACTIVIDAD, DEBERÁ ESTAR FIRMADA AL MARGEN DERECHO, INCLUIDA LA PROPIA SOLICITUD DE LA ACTIVIDAD.
- ✓ UNA VEZ ELABORADA SU ACTIVIDAD, RECUERDE DIGITALIZARLA Y NOMBRARLA EN BASE A LA NOMENCLATURA QUE SE INDICA MÁS ADELANTE EN ESTE DOCUMENTO.
- ✓ SI SUS EVIDENCIAS ENVIADAS POR CORREO, NO CUMPLEN CON LA NOMENCLATURA SOLICITADA, NO SERÁN CONSIDERADAS COMO EVIDENCIAS PARA SU EVALUACIÓN.
- ✓ CON ESTA ACTIVIDAD, USTED DEBERÁ IR INTEGRANDO SUS CARPETAS FÍSICA Y ELECTRÓNICA DE EVIDENCIAS, Y AL FINAL DEL SEMESTRE EN UN DISCO COMPACTO HARÁ ENTREGA DE SU CARPETA ELECTRÓNICA DE EVIDENCIAS.
- ✓ PARA TENER DERECHO A LA REVISIÓN Y EVALUACIÓN DE SUS ACTIVIDADES, DEBE REGISTRAR SU ASISTENCIA A CLASE, EL DÍA SEÑALADO PARA LA ENTREGA DE LA MISMA.
- ✓ FALTAR A CLASE EL DÍA DE LA ENTREGA, ANULA LA REVISIÓN DE SUS EVIDENCIAS.
- ✓ POR ÚLTIMO, POR FAVOR GESTIONE APROPIADAMENTE SU TIEMPO, Y SEA PUNTUAL EN SU ENTREGA Y ASÍ EVITAR PROBLEMAS DE NULIDAD POR EXTEMPORANEIDAD.
- ✓ AÚN PARA TRABAJOS EN EQUIPO APLICAN TODAS LAS MISMAS OBSERVACIONES ANTERIORES.

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

LA NOMENCLATURA SOLICITADA PARA ENVIAR SU TRABAJO ES LA SIGUIENTE :

```
AAAA-MM-DD_MATERIA_DOCUMENTO_EQUIPO_NOCTROL_APELLIDOS_NOMBRE_SEM.PDF
```

(NOTA : *** TODO EN MAYÚSCULA ***)

DONDE :

AAAA	:	AÑO
MM	:	MES
DD	:	DÍA
MATERIA	:	LAII, PE, PG, LI
DOCUMENTO	:	A3-ACTIVIDAD 3, P1-PRACTICA 1, R1-REPORTE 1, T1-TAREA 1, PG1-PROGRAMA, ETC. (CAMBIANDO EL NÚMERO CONSECUTIVO POR EL QUE CORRESPONDA)
EQUIPO	:	NÚMERO DEL EQUIPO QUE CORRESPONDA SEGÚN INDICACIÓN DEL PROFESOR.
NOCTROL	:	SU NÚMERO DE CONTROL
APELLIDOS	:	SUS APELLIDOS
NOMBRE	:	SU NOMBRE
SEM	:	EL PERIODO SEMESTRAL EN CURSO: ENE-JUN / AGO-DIC

EJEMPLO :

SI EL TRABAJO SE HACE EN EQUIPO.

```
2020-05-02_LAII_A3_EQUIPO_99_9999999_PEREZ_PEREZ_JUAN_ENE-JUN20.PDF
```

SI EL TRABAJO SE HACE INDIVIDUALMENTE.

```
2020-05-02_LAII_A3_9999999_PEREZ_PEREZ_JUAN_ENE-JUN20.PDF
```

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

FECHA DE ENTREGA:

VÍA CORREO ELECTRÓNICO, EL VIERNES 15 DE MAYO DEL 2020, CON HORA LÍMITE DE ENTREGA HASTA LAS 14:00 HORAS (2 DE LA TARDE).

EN CASO DE QUE EL TRABAJO SE HAYA ELABORADO EN EQUIPO, EL JEFE DEL MISMO SERÁ EL ÚNICO RESPONSABLE DE ENVIAR LA ACTIVIDAD A LA SIGUIENTE CUENTA DE CORREO:

ricardo.gonzalez@itcelaya.edu.mx

MUY IMPORTANTE:

DESPUÉS DE LAS 14:00 HRS. EN PUNTO, LA ACTIVIDAD YA SERÁ CONSIDERADA COMO EXTEMPORÁNEA Y NO CONTARÁ COMO EVIDENCIA PARA SU EVALUACIÓN.

SE LE SUGIERE ENVIAR CON ANTICIPACIÓN SU ACTIVIDAD A FIN DE EVITAR CONFLICTOS POR NO ENTREGAR ÉSTA A TIEMPO.

RECUERDE ANEXAR A SU ARCHIVO .PDF DE EVIDENCIAS, ESTA SOLICITUD DE ACTIVIDADES CON TODAS LAS HOJAS FIRMADAS EN EL MARGEN DERECHO.

POR ÚLTIMO, CONSIDERE EL SIGUIENTE CALENDARIO OFICIAL, PARA GESTIONAR ADECUADAMENTE EL TIEMPO QUE SE LE OTORGA PARA DESARROLLAR ESTA ACTIVIDAD.

CALENDARIO 2019-2020 - EVALUACIÓN 3

A. INVESTIGAR. ¿QUÉ ES UN ANÁLISIS SINTÁCTICO APLICADO A LA VALORACIÓN DE UN LENGUAJE FORMAL?

ANÁLISIS SINTÁCTICO

Como mencionábamos en la actividad anterior, cualquier compilador debe de realizar dos tareas principales: análisis del programa a compilar y síntesis de un programa en lenguaje máquina que, cuando se ejecute, realizara correctamente las actividades descritas en el programa fuente. Ahora toca el exponer y estudiar la importancia, así como el marco teórico de un análisis sintáctico aplicado a la valoración de un lenguaje formal.

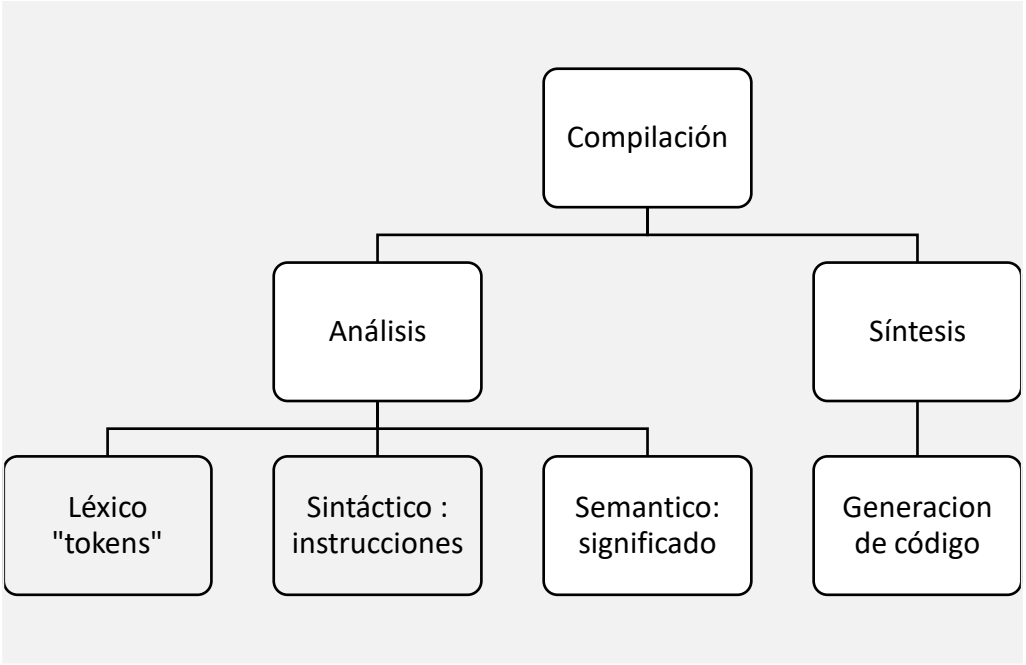


Figura 1. Procesos dentro de la compilación

El análisis gramatical es la tarea de determinar la sintaxis, o estructura, de un programa. Por esta razón también se le conoce como análisis sintáctico. La sintaxis de un lenguaje de programación por lo regular se determina mediante las reglas gramaticales de una gramática libre de contexto, de manera similar como se determina mediante expresiones regulares la estructura Léxica de los tokens reconocida por el analizador léxico.

En realidad, una gramática libre de contexto utiliza convenciones para nombrar y operaciones muy similares a las correspondientes en las expresiones regulares. Con la única diferencia de que las reglas de una gramática libre de contexto son recursivas. Por ejemplo. la estructura de una sentencia if debe permitir en general que otras sentencias if estén anidadas en ella, lo que no se permite en las expresiones regulares.

Este cambio aparentemente elemental para el poder de la representación tiene enormes consecuencias. La clase de estructuras reconocible por las gramáticas libres de contexto se incrementa de manera importante en relación con las de las expresiones regulares. Los algoritmos empleados para reconocer estas estructuras también difieren mucho de los algoritmos de análisis léxico, ya que deben utilizar llamadas recursivas o una pila de análisis sintáctico explícitamente administrada.

Las estructuras de datos utilizadas para representar la estructura sintáctica de un lenguaje ahora también deben ser recursivas en lugar de lineales (como lo son para lexemas y tokens). La estructura básica empleadas por lo regular alguna clase de árbol, que se conoce como árbol de análisis gramatical o árbol sintáctico. Sin embargo. al contrario de lo que sucede con los analizadores léxicos, donde sólo existe, en esencia un método algorítmico (representado por los autómatas finitos), el análisis sintáctico involucra el tener que elegir entre varios métodos diferentes, cada uno de los cuales tiene distintas propiedades y capacidades. De hecho, existen dos categorías generales de algoritmos:

- de análisis sintáctico descendente
- de análisis sintáctico ascendente (por la manera en que construyen el árbol de análisis gramatical o árbol sintáctico).

B. INVESTIGAR. ¿EN QUÉ CONSISTE UN ANÁLISIS SINTÁCTICO Y QUÉ LO CARACTERIZA?

EL PROCESO DEL ANÁLISIS SINTÁCTICO

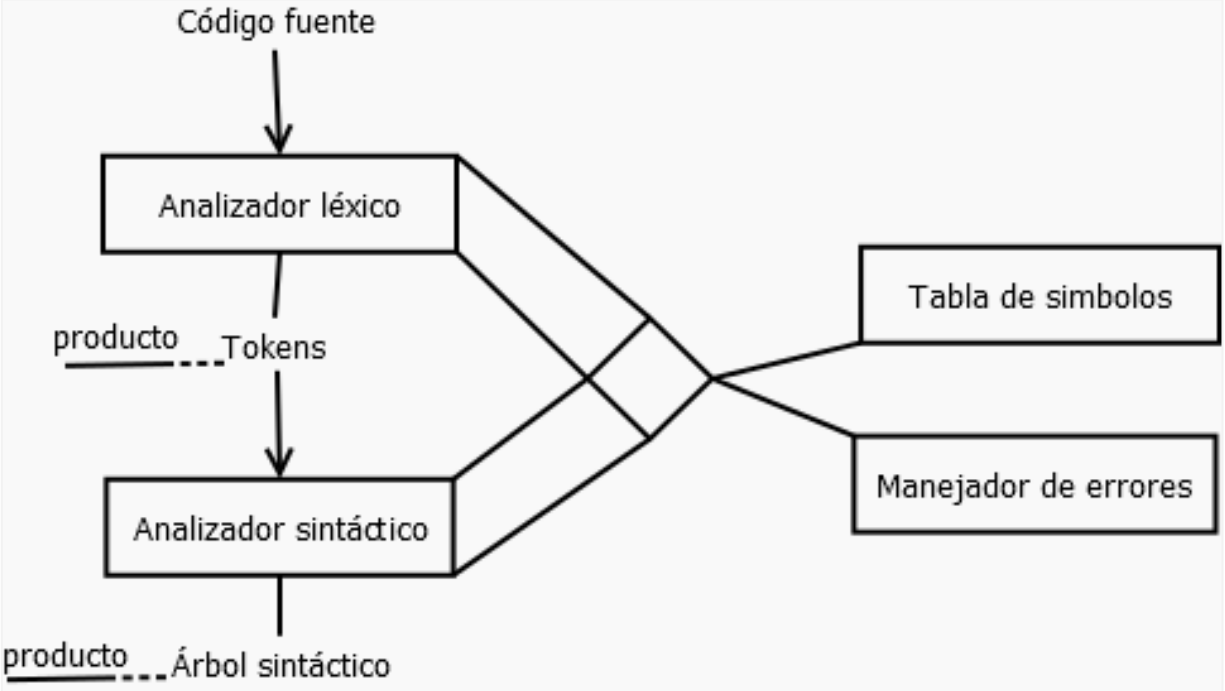


Imagen 1. Fases léxica y sintáctica de un compilador

La tarea del analizador sintáctico es determinar la estructura sintáctica de un programa a partir de los tokens producidos por el analizador léxico y, ya sea de manera explícita o implícita, construir un árbol de análisis gramatical o árbol sintáctico que represente esta estructura. De este modo, se puede ver el analizador sintáctico como una función que toma como su entrada la secuencia de tokens producidos por el analizador Léxico y que produce como su salida el Árbol sintáctico

La secuencia de tokens por lo regular no es un parámetro de entrada explícito, pero el analizador sintáctico llama a un procedimiento del analizador léxico, como getToken, para obtener el siguiente token desde la entrada a medida que lo necesite durante el proceso de análisis sintáctico.

Un problema más difícil de resolver para el analizador sintáctico que para el analizador Léxico es **el tratamiento de los errores**. En el analizador léxico, si hay un carácter que no puede ser parte de un token legal, entonces es suficientemente simple generar un token de error y consumir el carácter problemático. (En cierto sentido, al generar un token de error, el analizador léxico transfiere la dificultad hacia el analizador sintáctico.) Por otra parte, el analizador sintáctico no sólo debe mostrar el mensaje de error, sino que debe recuperarse del error y continuar el análisis sintáctico (para encontrar tantos errores como sea posible).

En ocasiones, un analizador sintáctico puede **efectuar reparación de errores**, en la cual interfiere una posible versión de código corregida a partir de la versión incorrecta que se le haya presentado (Esto por lo regular se hace solo en casos simples). Un aspecto particularmente importante de la recuperación de errores es la exhibición de mensajes de errores significativos y la reanudación del análisis sintáctico tan próximo al error real como sea posible. Esto no es fácil puesto que el analizador sintáctico puede no descubrir un error sino hasta mucho después de que el error real haya ocurrido.

CLASIFICACIÓN

La tarea esencial de un analizador es determinar si una determinada entrada puede ser derivada desde el símbolo inicial, usando las reglas de una gramática formal, y como hacer esto, existen esencialmente dos métodos:

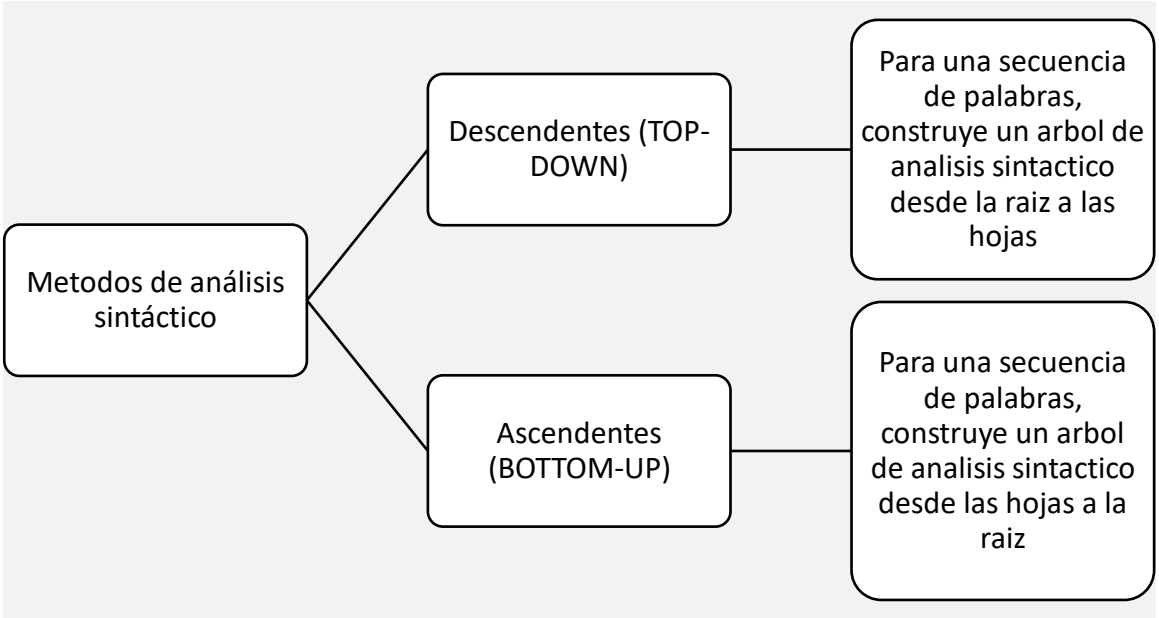


Figura 2. Métodos de análisis sintáctico

Un algoritmo de análisis sintáctico descendente analiza una cadena de tokens de entrada mediante la búsqueda de los pasos en una derivación por la izquierda. Un algoritmo así se denomina descendente debido a que el recorrido implicado del árbol de análisis gramatical es un recorrido de preorden y, de este modo, se presenta desde la raíz hacia las hojas Los analizadores sintácticos descendentes existen en dos formas: **analizadores sintácticos inversos o en reversa y analizadores sintácticos predictivos**. Un analizador sintáctico predictivo intenta predecir la siguiente construcción en la cadena de entrada utilizando uno o más tokens de búsqueda por adelantado, mientras que un analizador sintáctico inverso intentará las diferentes posibilidades para un análisis sintáctico de la entrada, respaldando una cantidad arbitraria en la entrada si una posibilidad falla. Aunque los analizadores sintácticos inversos son más poderosos que los predictivos. también son mucho más lentos, ya que requieren una cantidad de tiempo exponencial en general y, por consiguiente, son inadecuados para los compiladores prácticos.

Los algoritmos de análisis sintácticos ascendentes son en general más poderosos que los métodos descendentes. (Por ejemplo, la recursión por la izquierda no es un problema en el análisis sintáctico ascendente.) Como es de esperarse, las construcciones involucradas en estos algoritmos también son más complejas

Descendente	Ascendente
Se busca una derivación por la izquierda para una cadena de entrada	Se construye un árbol sintáctico para una cadena de entrada que comienza por las hojas y avanza hacia la raíz
No hay muchos analizadores sintácticos descendentes con retroceso - casi nunca se necesita el retroceso para analizar las construcciones de lenguajes de programación. El retroceso tampoco es muy eficiente.	Se "reduce" una cadena al símbolo inicial de la gramática
Una gramática recursiva por la izquierda puede causar que el analizador sintáctico entre en un lazo infinito (porque no consume ningún símbolo de entrada)	En cada paso de reducción, se sustituye una subcadena determinada que concuerda con el lado derecho de una producción por el símbolo del lado izquierdo de dicha producción (básicamente se busca el lado derecho que corresponde y se reemplaza por el lado izquierdo y se intenta de esa manera llegar a la raíz)
Al eliminar la recursividad y factorizar se puede obtener una gramática analizable por un analizador sintáctico predictivo por descenso que no necesite retroceso.	Se traza una derivación por la derecha en sentido inverso

Tabla1. Diferencias

C. IDENTIFICAR. ¿QUÉ CASOS DE ESTUDIOS SON LOS IMPORTANTES A CONSIDERAR EN EL ANÁLISIS SINTÁCTICO?

El punto en cuestión será mencionado en el punto E como una respuesta conjunta ya que están directamente correlacionados.

D. INVESTIGAR Y PROPONER. ¿QUÉ PROCESOS Y PROBLEMAS ATIENDE UN ANÁLISIS SINTÁCTICO?

El punto en cuestión será mencionado en el punto E como una respuesta conjunta ya que están directamente correlacionados.

E. INVESTIGAR. ¿CÓMO IMPLEMENTAR UTILIZANDO UN ANÁLISIS SINTÁCTICO?

MAEJO DE ERRORES SINTÁCTICOS

Si un compilador tuviera que procesar sólo programas correctos, su diseño e implantación se simplificarían mucho. los programadores a menudo escriben programas incorrectos, y un buen compilador debería ayudar al programador a identificar y localizar errores.

Considerar desde el principio el manejo de errores puede simplificar la estructura de un compilador y mejorar su respuesta a los errores.

lexicos	***Sintacticos***
producidos al escribir mal un identificador , una palabra clave o un operador	producidos por una expresion aritmetica o parentesis no equilibrados

Figura3. Errores

El manejo de errores de sintaxis es el más complicado desde el punto de vista de la creación de compiladores. Nos interesa que cuando el compilador encuentre un error, se recupere y siga buscando errores.

Por lo tanto, el manejador de errores de un analizador sintáctico debe tener como objetivos:

- Un analizador sintáctico debería intentar determinar que ha ocurrido un error Tan pronto como sea posible. Esperar demasiado tiempo antes de la declaración del error significa que la ubicación del error real puede haberse perdido.
- Después de que se ha presentado un error, el analizador sintáctico debe seleccionar un lugar probable para reanudar el análisis. Un analizador sintáctico siempre debería intentar analizar tanto código como sea posible, a fin de encontrar tantos errores reales como sea posible durante una traducción simple
- Un analizador sintáctico debería intentar evitar el problema de cascada de errores, en la cual un error genera una larga secuencia de mensajes de error falsos.
- Un analizador sintáctico debe evitar bucles infinitos en los errores en los que se genera una cascada sin fin de mensajes de error sin consumir ninguna entrada

En resumen:

Indicar los errores de forma clara y precisa. Aclarar el tipo de error y su localización
Recuperarse del error, para poder seguir examinando la entrada.
No ralentizar significativamente la compilación.

Algunos de estos objetivos entran en conflicto entre sí, de tal manera que un escritor de compiladores tiene que efectuar "convenios" durante la construcción de un manejador de errores. Por ejemplo, el evitar los problemas de cascada de errores y bucle infinito puede ocasionar que el analizador sintáctico omita algo de la entrada con lo que compromete el objetivo de procesar tanta información de la entrada como sea posible.

TIPO DE ERRORES	EJEMPLOS
Por declaración de datos	Valores iniciales no apropiados. Variables no declaradas
Error de computo	Cálculos con variables no numéricas

De comparación	Variables a comparar de diferente tipo de dato. Expresiones lógicas incorrectas
De control de proceso	Cierres u aperturas incorrectos de las estructuras de control. Existen decisiones incompletas en su expresión lógica. Falta un ;

Estos son algunos de los tipos de errores más comunes:

Tabla3. Tipos de errores más comunes

Para un mayor control de errores en el análisis léxico se propone una tabla de errores en la cual se clasificarán dependiendo el análisis que se esté ejecutando.

A continuación, le listan los id de error junto con el mensaje de descripción que corresponde a cada uno.

ID	ORIGEN
210	No se inició el programa
211	No se asignó un identificador
212	Se esperaba que se abriera una llave {
213	Se esperaba el cierre de llaves }
214	No se incluyó el método principal
215	Se esperaba abrir un paréntesis (
216	Se esperaba el cierre de paréntesis)
217	Se esperaba la asignación a la variable
218	No se asignó nada a la variable
219	Falto punto y coma
220	No hay operador relacional
221	Expresión no reconocida

Tabla4. Tabla de errores

FUNCIONES DEL ANALIZADOR SINTÁCTICO

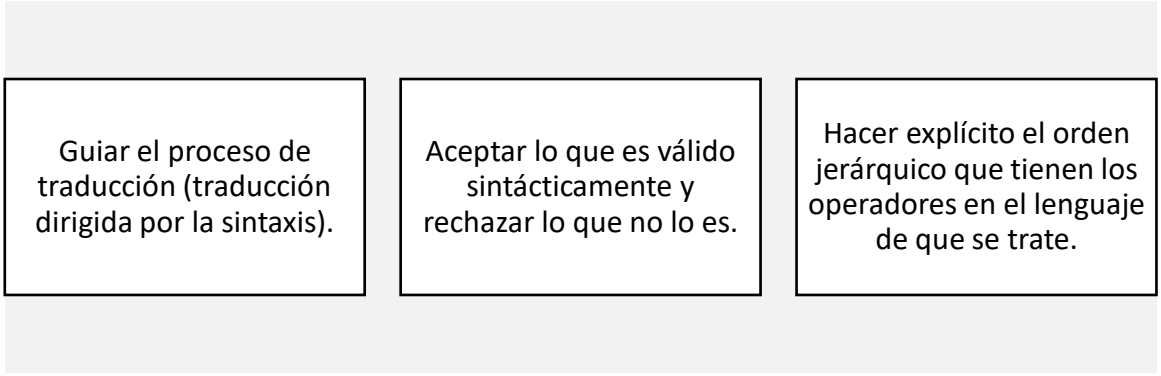


Figura5. Funciones analizador sintáctico

ACTUALIZACIÓN DE LA TABLA DE TOKENS

En la implementación nos dimos cuenta que nos hacían falta unos símbolos que eran necesarios para el analizador sintáctico así que actualizamos la tabla de tokens quedándonos de la siguiente manera.

TOKEN	LEXEMA
1	programa
2	principal
3	subrutina
4	entrada
5	salida
6	si
7	sino
8	mientras
20	+
21	-
22	*
23	/

25	=
30	<
31	>
32	==
33	>=
34	<=
35	!=
40	&
41	|
50	ent
51	rea
52	cad
55	Entero
56	Real
57	cadena
60	;
61	{
62	}
63	(
64	)
65	/*
66	*/
67	,
68	_
70	Identificador ($xy)

Tabla5. Tokens

CASOS DE ERRORES SINTÁCTICOS

DESCRIPCIÓN: SE INICIA ERRONEAMENTE EL PROGRAMA

este caso ocurre cuando el programador intenta iniciar el programa sin la palabra reservada

- ERROR: 210
- MENSAJE: “No se inició el programa”
- EJEMPLO:

```
$programa {
/*CONTENIDO*/
}
```

DESCRIPCIÓN: CUALQUIER SUBRUTINA DEBE DE TENER SU IDENTIFICADOR ASIGNADO,

este caso ocurre cuando el programador intenta declarar una subrutina sin su identificador

- ERROR: 211
- MENSAJE: “No se asignó un identificador”
- EJEMPLO:

```
Subrutina(){
/*CONTENIDO*/
}
```

DESCRIPCIÓN: LLAVES DE APERTURA MAL EQUILIBRADOS

Este caso ocurre cuando el programador olvida agregar la llave de apertura

- ERROR: 212
- MENSAJE: “Se esperaba que se abriera una llave {”
- EJEMPLO:

```
Subrutina $suma()
/*CONTENIDO*/
}
```

DESCRIPCIÓN: LLAVES DE CIERRE MAL EQUILIBRADOS

Este caso ocurre cuando el programador olvida agregar la llave de cierre

- ERROR: 213
- MENSAJE: “Se esperaba que se cerrara una llave }”
- EJEMPLO:

```
Subrutina $suma() {
/*CONTENIDO*/
```

DESCRIPCIÓN: NO SE INCLUYO EL PROCEDIMIENTO PRINCIPAL

Este caso ocurre cuando el programador olvida incluir el procedimiento principal

- ERROR: 214
- MENSAJE: “No se incluyó el método principal”
- EJEMPLO:

```
( ) {
entrada( );
}
```

DESCRIPCIÓN: PARENTESIS DE APERTURA MAL EQUILIBRADOS

Este caso ocurre cuando el programador olvida agregar los paréntesis de apertura

- ERROR: 215
- MENSAJE: “Se esperaba que se abriera una paréntesis (”
- EJEMPLO:

```
Subrutina $suma ) {
/*CONTENIDO*/ }
```

DESCRIPCIÓN: PARENTESIS DE CIERRE MAL EQUILIBRADOS

Este caso ocurre cuando el programador olvida agregar el paréntesis de cierre

- ERROR: 216
- MENSAJE: “Se esperaba que se cerrara una paréntesis)”

- EJEMPLO:

```
Subrutina $suma(  {
/*CONTENIDO*/ }
```

DESCRIPCIÓN: FALTA UN PUNTO Y COMA

Este caso ocurre cuando el programador olvida agregar el punto y coma al final de la expresión

- ERROR: 219
- MENSAJE: “Falto punto y coma”.
- EJEMPLO:

```
Entrada()
```

DESCRIPCIÓN: FALTO OPERADOR RELACIONAL DENTRO DE LA EXPRESIÓN

Este caso ocurre cuando el programador olvida ingresar el operador relacional

- ERROR: 220
- MENSAJE: “No hay operador relacional”.
- EJEMPLO:

```
si($cuatro  $cinco){
salida(432423);
}
```

DESCRIPCIÓN: EXISTE UNA EXPRESION NO RECONOCIDA POR EL LENGUAJE

Este caso ocurre cuando el programador ingresa alguna expresión que no es perteneciente al lenguaje definido

- ERROR: 221
- MENSAJE: “Expresión no reconocida”.
- EJEMPLO:

```
Entrada()
$id = (;##
```

PROGRAMAS PRUEBA

En la ejecución de los programas prueba aparecerán más errores de los mencionados, ya que un programa con errores sintácticos puede desencadenar más errores en líneas posteriores, aunque tengamos una cantidad significativa de errores en cascada, comúnmente se solucionan con solucionar el primer error.

Programa 1

```
/*En la siguiente línea nos genera un error 210 ya que nos hace falta la
palabra reservada programa*/
 $fdsafsa {
    ent $gfdsg,$fdaf;
    rea $fdsa,$gfsd;



    /*En la siguiente línea nos marcara el error 211 ya que no se asignó
un identificador a la subrutina*/
    subrutina (){
        entrada();
        mientras($tres>$cuatro){
            /*En la siguiente línea nos marcara un error 215 ya que no se
agregaron los paréntesis*/
            $gfds = entrada;
        }
    }



    subrutina $gfdgfds(){
        si($cuatro <= $cinco){
            salida(432423);
        }
        sino{
            /*En la siguiente línea nos marcara un error 216 ya que no se
cerraron los paréntesis*/
            entrada(;
        }
    }


    principal() {
        entrada();
    }
}
```

Programa 2

```
/*En la siguiente línea se mostrara un error 212 ya que no se abrieron
las llaves*/
programa $fdsafsa
    ent $gfdsg,$fdaf;
    rea $fdsa,$gfsd;



    subrutina $hola(){
        entrada();
        mientras($tres>$cuatro){
            $gfds = entrada();
        }
    }



    subrutina $gfdgfds(){
        si($cuatro <= $cinco){
            salida(432423);
        }
        sino{
            entrada();
        }
    /*En la siguiente línea se mostrara un error 213 ya que no se
abrieron las llaves*/


    principal() {
        /*En la siguiente línea marcara el error 219 ya que hace falta un
punto y coma*/
        entrada()



        /*En la siguiente línea marcara el error 221 ya que se quiere
asignar algo no reconocido por el lenguaje*/
        $id = (;
    }
}
```

Programa 3

```
programa $fdsafsa {
    ent $gfdsg,$fdaf;
    rea $fdsa,$gfsd;



    subrutina $hola(){
        entrada();
        mientras($tres>$cuatro){
            $gfds = entrada();
        }
    }



    subrutina $gfdgfds(){
        /*La siguiente línea dará el error 221 ya que no se puso un
condición en si*/
        si(){
            salida(432423);
        }
        sino{
            entrada();
        }
    }



    /*En la siguiente línea marcara el error 214 ya que no se incluyó el
procedimiento principal*/
    () {
        entrada();
        /*En la siguiente línea marcara el error 218 ya que se esperaba
realizar algo con la variable*/
        $fdafdsa
    }
}
```

MODELADO Y FASES DEL ANALIZADOR SINTÁCTICO

UNIFIED MODELING LANGUAGE

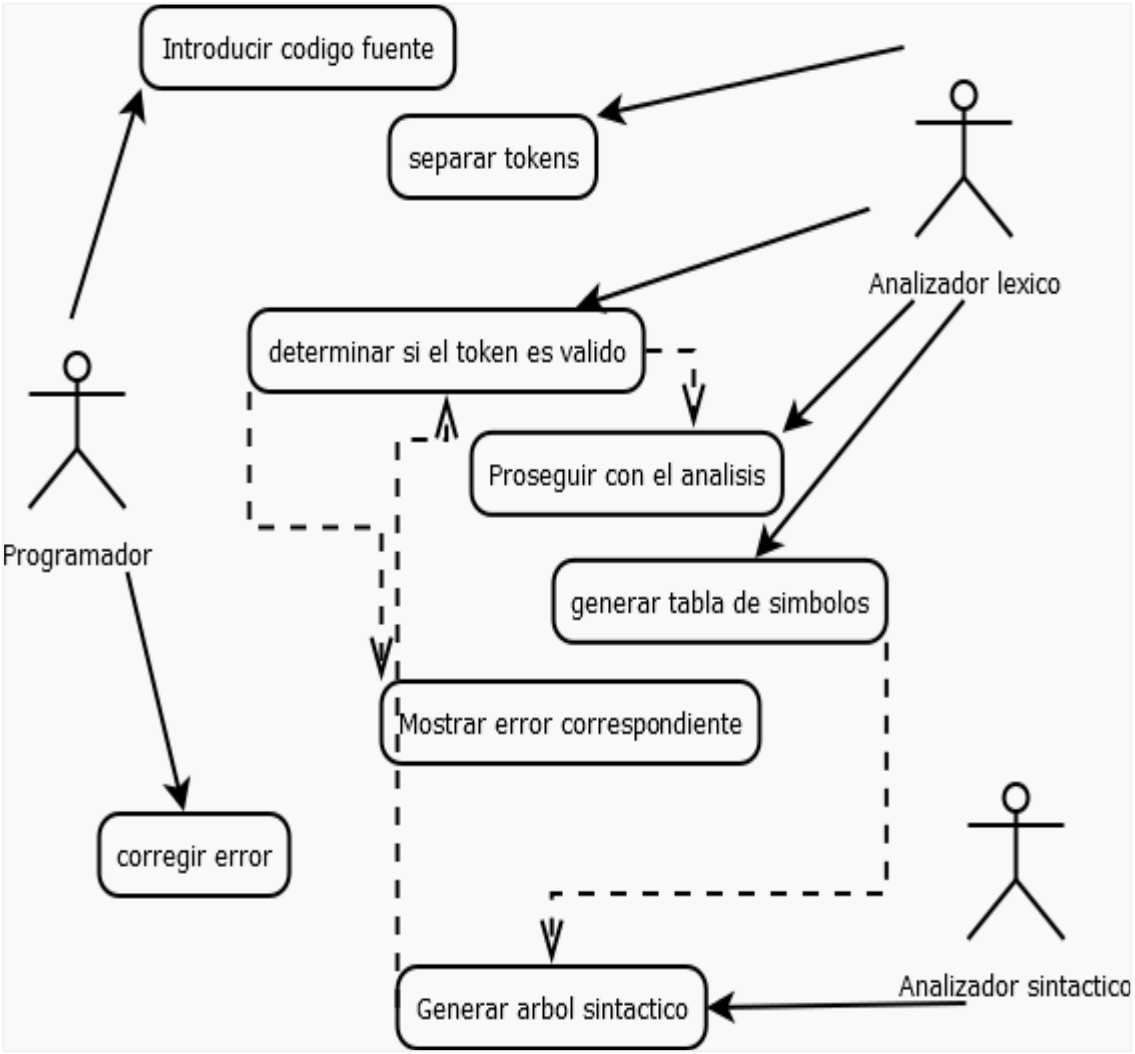


Figura3. Diagrama de casos de uso

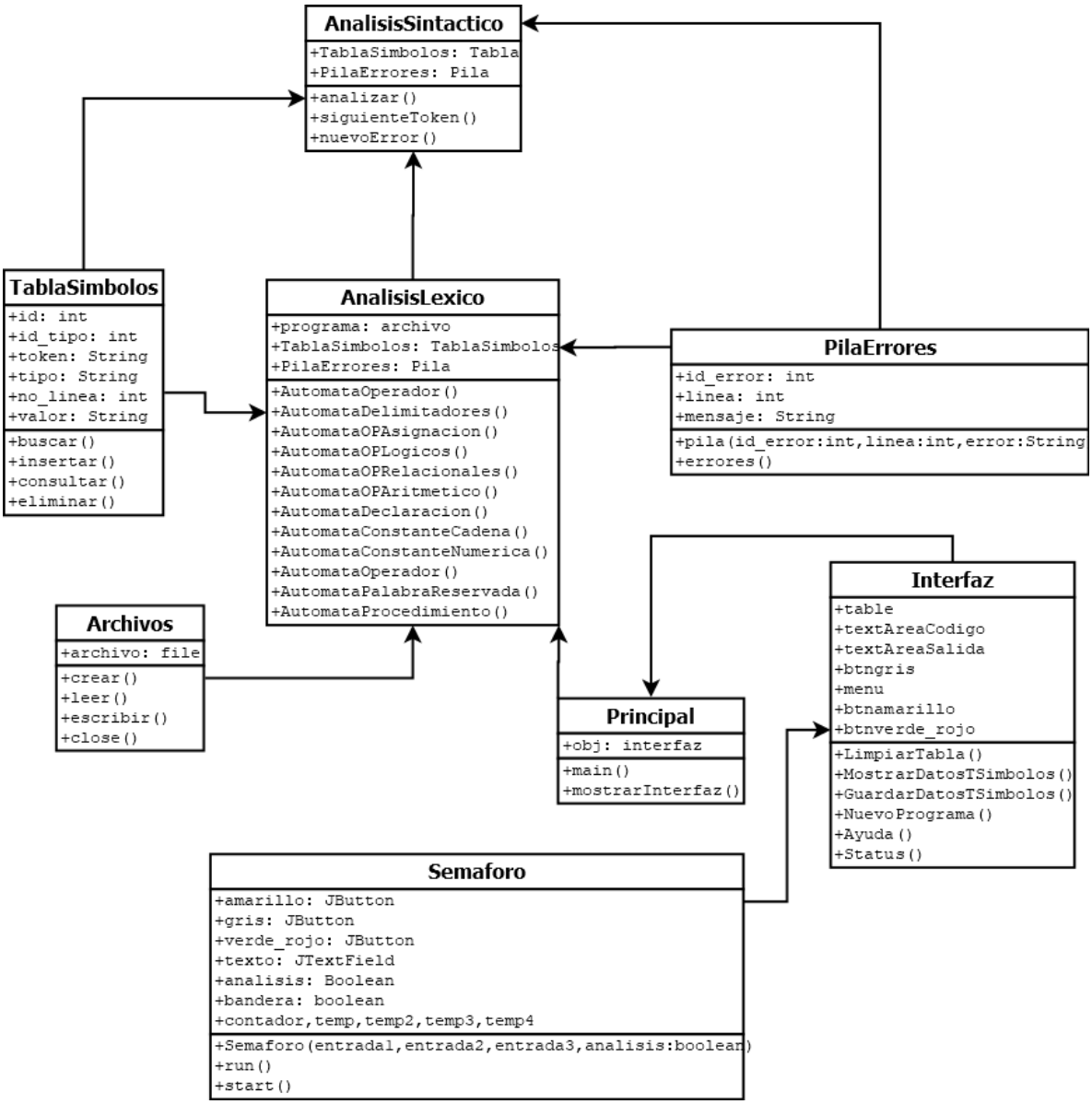


Figura4. Diagrama de clases

GRAMATICA TIPO 3 REGULAR DEL LENGUAJE PROTOTIPO

Debido a los problemas que tuvimos con algunos autómatas se decidio hacer nuevamente una actualización a la gramática así que a continuación se muestra la gramática utilizada junto con los diferentes autómatas referentes al analizador.

1. <programa>:: =programa <identificador> ´{´<declaraciones> * <procedimientos> * principal´ ()´ ´{´<instrucciones>*´}´ ´}’

2. <identificador>::= ‘$’<letra>[(<letra> | <digito>)*]

3. <declaraciones>:: =<tipoDato> { <identificador> [´,´<identificador>] } ‘;’

4. <tipoDato>:== entero | real | cadena

5. <letra> ::= a | b | c | … | x | y | z | A | B | C | … | X | Y | Z |

6. <digito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

7. <procedimientos>::= subrutina <identificador> ´()´ ´{´[<instrucciones>*]´}´

8. <instrucciones>::={ <estructuraDecision> | <estructuraCiclo> | <entrada> | <salida> | <asignacion>}

9. <asignación> ::= <identificador> ‘=’ (<entrada> | <expresión>)

10. <entrada> ::= entrada();

11. <salida>::=salida(<concatenación>);

12. <concatenación>::= { (<expresion> | <cadena>)’__’{<concatenación>} }

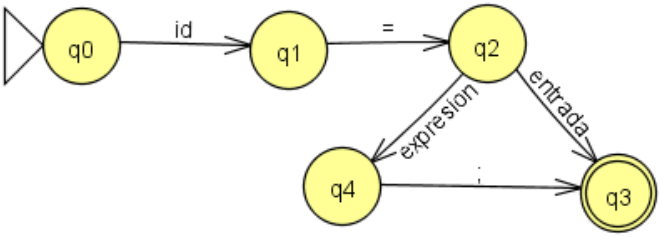
13. <expresion>::=(<identificador>| {<digito>}) [<OprAritmetico> <expresion>] | ´(´<expresión>´)´

14. <estructuraDecision>::= si ’(’ <condición> ´)´´{´ <instrucciones> ´}´ [sino ´{´<instrucciones>´}´]

15. <condición>::=<expresión> [(<OpRelacional>|<OpLogico>)<condicion>]

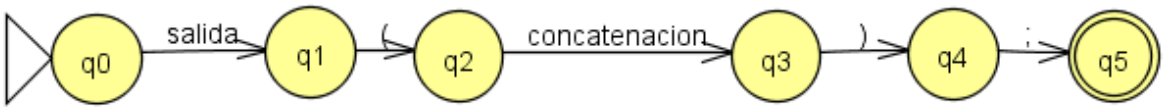
16. <estructuraCiclo>::= mientras´(´ <condición> ´)´ ´{´ <instrucciones> ´}´

Asignación



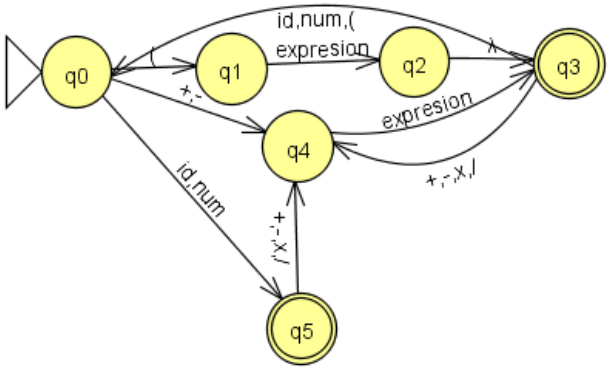
	id	=	entrada	expresión	;
→Q0	Q1				
Q1		Q2			
Q2			Q3	Q4	
*Q3					
Q4					Q3

Salida



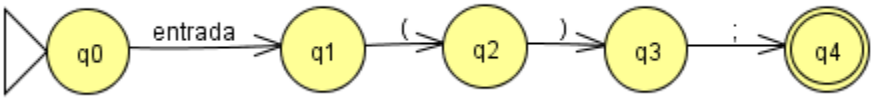
	salida	(	concatenacion	)	;
→Q0	Q1				
Q1		Q2			
Q2			Q3		
Q3				Q4	
Q4					Q5
*Q5					

Expresión



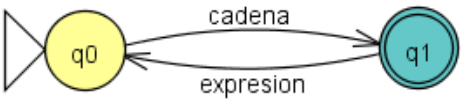
	expresión	(	+,-	*,/	id	num	
→Q0		Q1	Q4		Q5	Q5	
Q1	Q2	Q2	Q2				
Q2					Q3		Q3
*Q3		Q0	Q4	Q4	Q0	Q0	
Q4	Q3						
*Q5			Q4	Q4			

Entradas



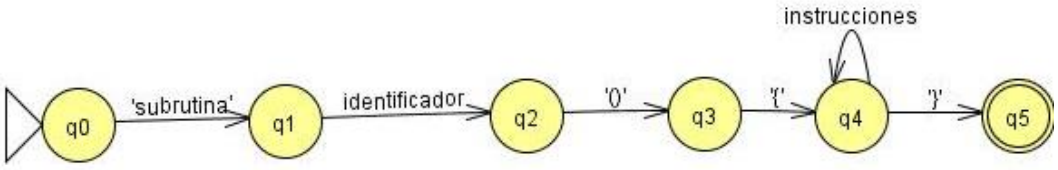
	entrada	(	)	;
→Q0	Q1			
Q1		Q2		
Q2			Q3	
Q3				Q4
*Q4				

Concatenación



	cadena	expresion
→Q0	Q1	
*Q1		Q0

Subrutina

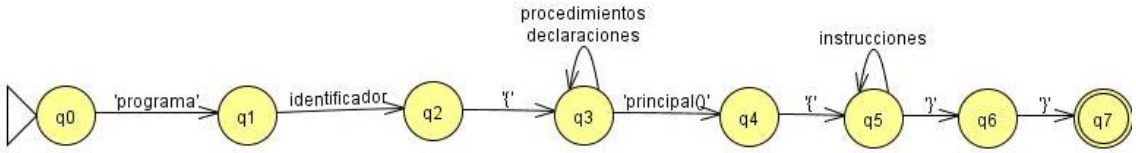


	subrutina	identificador	()	{	instruccion	}
→Q0	Q1					
Q1		Q2				
Q2			Q3			
Q3				Q4		
Q4					Q4	Q5
*Q5						

Estructura de decisión

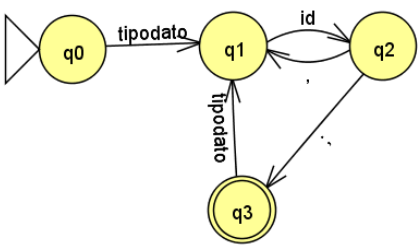
	si	(	condicion	)	{	instruccion	}	sino
→Q0	Q1							
Q1		Q2						
Q2			Q2					
Q3				Q4				
Q4					Q5			
Q5						Q6		
Q6							Q7	
*Q7								Q8
Q8					Q9			
Q9						Q10		
Q10							Q11	
*Q11								

Programa



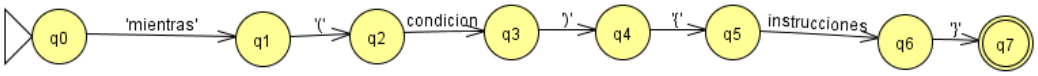
	programa	identificador	{	procedimientos	declaraciones	principal()	}	instrucciones
→Q0	Q1							
Q1		Q2						
Q2			Q3					
Q3				Q3	Q3	Q4		
Q4							Q5	
Q5							Q6	Q5
Q6							Q7	
*Q8								

Declaraciones



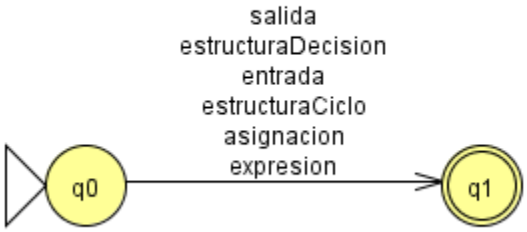
	tipodato	id	;	,
→Q0	Q1			
Q1		Q2		
Q2			Q3	Q1
*Q3	Q1			

Estructura de ciclo



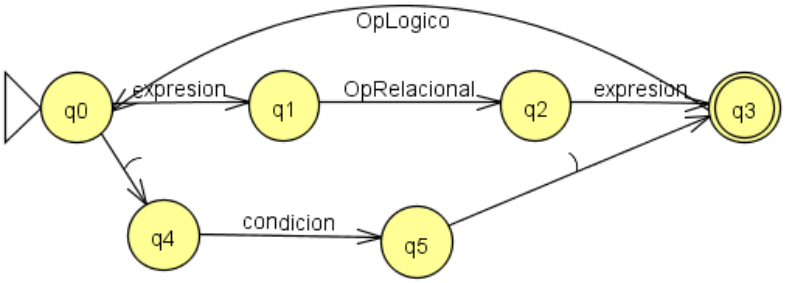
	mientras	(	condicion	)	{	instrucciones	}
→Q0	Q1						
Q1		Q2					
Q2			Q2				
Q3				Q4			
Q4					Q5		
Q5						Q6	
Q6							Q7
*Q7							

Instrucciones



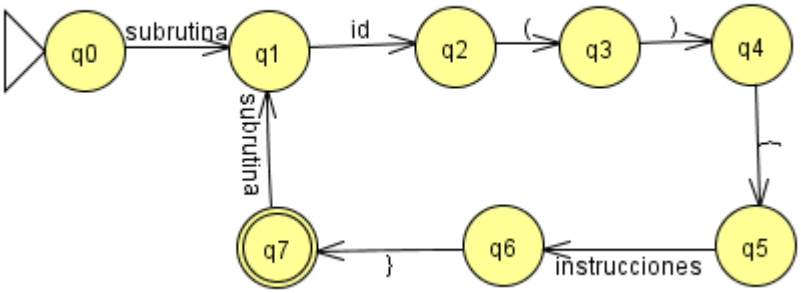
	Salida	estructuraDecision	Entrada	estructuraCiclo	Asignación	Expresión
→Q0	Q1	Q1	Q1	Q1	Q1	Q1
*Q1						

Condición



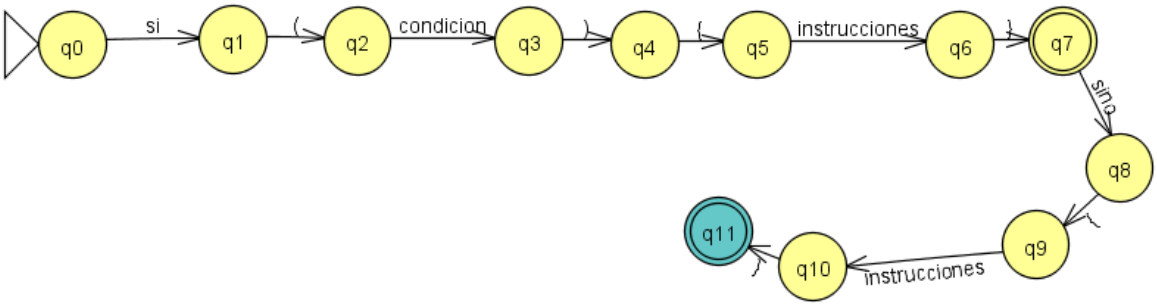
	expresion	OpRelacional	OpLogico	(	condicion	)
→Q0	Q1			Q4		
Q1		Q2				
Q2	Q3				Q3	
*Q3			Q0			
Q4					Q5	
Q5						Q3

Procedimientos



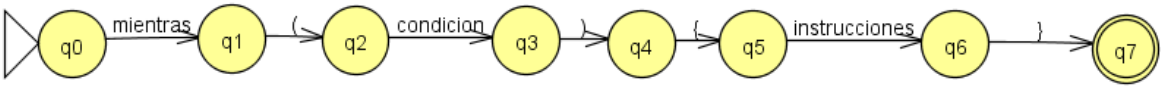
	subrutina	id	(	)	{	}	instrucciones
→Q0	Q1						
Q1		Q2					
Q2			Q3				
Q3				Q4			
Q4					Q5		
Q5							Q6
Q6						Q7	
*Q7	Q1						

si



	si	condicion	(	)	{	}	instrucciones	sino
→Q0	Q1							
Q1			Q2					
Q2		Q3						
Q3				Q4				
Q4					Q5			
Q5							Q6	
Q6						Q7		
*Q7								Q8
Q8					Q9			
Q9							Q10	
Q10						Q11		
*Q11								

mientras



	Mientras	condicion	(	)	{	}	instrucciones	sino
→Q0	Q1							
Q1			Q2					
Q2		Q3						
Q3				Q4				
Q4					Q5			
Q5							Q6	
Q6						Q7		
*Q7								

DOCUMENTACIÓN

A continuación, se detallará como fue implementado el Analizador Sintáctico dentro de nuestro Pseudocompilador que estamos desarrollando, se mostrara el diagrama de clase correspondiente a la parte mostrada, junto con una muestra del código en el cual se muestran los métodos que contiene la clase, los métodos se encuentran sin desplegar para mostrar la estructura en un espacio más compacto.

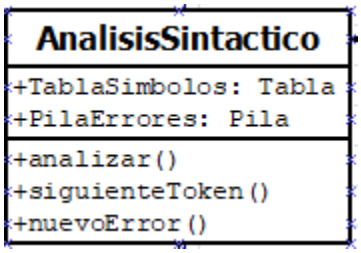
Se mencionarán las clases utilizadas e implementadas dentro del análisis sintáctico.

Se utilizo un método de árbol hacía arriba predictivo para el análisis, es decir se comprueba que los tokens tengan un orden y se ingrese el que sigue según la estructura de nuestro programa.

CLASE "ANALISISSINTACTICO":

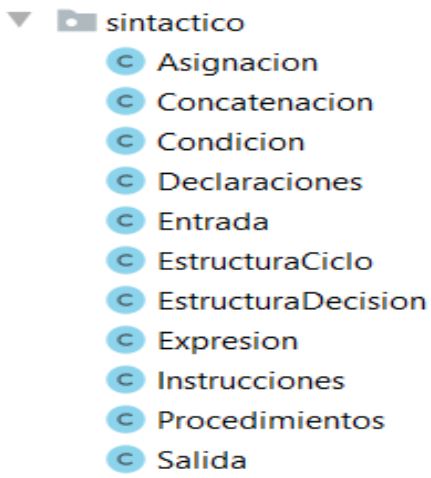
Primero vemos que nuestra clase de análisis sintáctico ya no trabaja con un archivo, a ella únicamente viene la tabla de símbolos previamente generada en nuestro analizador léxico ya que el análisis sintáctico corresponde a una correcta estructura gramatical de acuerdo a nuestro lenguaje, además de ello cuenta con la pila de errores donde se almacenaran y posteriormente se podrá acceder a los registros de los errores generados en la ejecución del analizador sintáctico.

Se tienen los métodos analizar, el cual de acuerdo a nuestra estructura gramatical llama a los autómatas designados para este análisis para que nos regrese si el orden es correcto o en caso de error de llame a nuevoError, donde se agrega un error a la pila de errores, finalizado ese análisis se pasa al siguiente token para ser analizado.



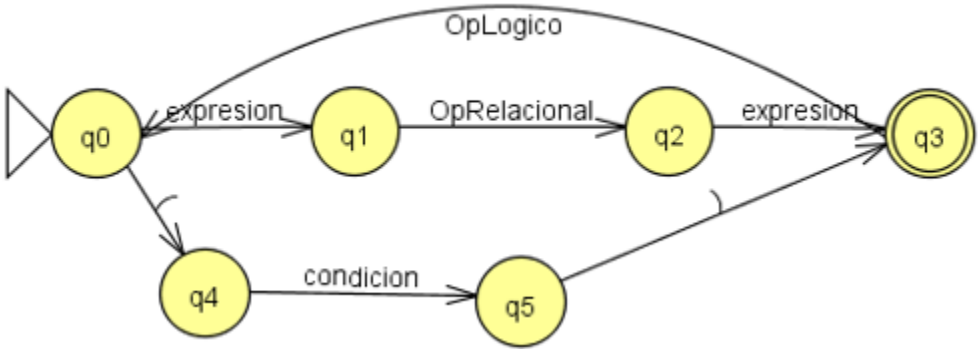
```java
import com.company.automatas.sintactico.*;
import com.company.pilaErrores.PilaErrores;
import com.company.tablaSimbolos.Simbolo;
import com.company.tablaSimbolos.TablaSimbolos;

public class AnalisisSintactico{
    public TablaSimbolos tablaSimbolos = new TablaSimbolos();
    public PilaErrores pilaErrores = new PilaErrores();
    public int actual = 0;

    public AnalisisSintactico(TablaSimbolos tablaSimbolos, PilaErrores pilaErrores) {...}

    public AnalisisSintactico() {}

    public void analizar(){...}

    public int siguienteToken(){...}

    public void nuevoError(int error){...}
}
```

Cada autómata se encuentra implementado en un objeto diferente para tener una mejor organización de los mismos.



De igual manera cada autómata cuenta con acceso a la tabla de símbolos y la pila de errores, esto porque es necesario que si se genera un nuevo error se pueda ingresar en el momento dado, además que se puedan seguir analizando los tokens conforme aparecen.

Las siguientes imágenes muestran la estructura del autómata, así como de la clase que lo conforma, esto se repite en cada autómata que se implementó, el siguiente ejemplo corresponde a la condición.

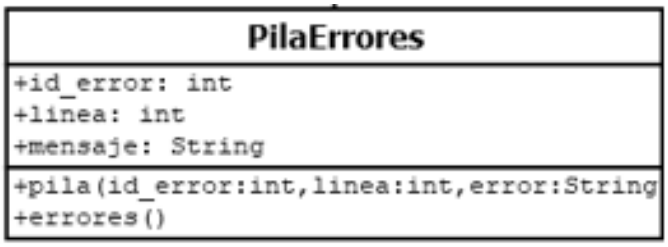



```
public class Condicion {
    public TablaSimbolos tablaSimbolos;
    public PilaErrores pilaErrores;
    public int posicion;

    public void nuevoError(int error){...}

    public Condicion(TablaSimbolos tablaSimbolos, PilaErrores pilaErrores,int posicion) {...}

    void q0(){...}

    void q1(){...}

    void q2(){...}

    void q3(){...}

    void q4(){...}

    void q5(){...}

    public int siguienteToken(){...}
}
```

CLASE "PILAERRORES"

La pila de errores al mostrar lo contenido, muestra todo lo almacenado dentro de ella, imprimiendo en pantalla el id del error, la línea en la cual se generó y el correspondiente mensaje de error.

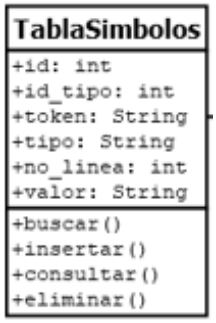
Cabe mencionar que la implementación dentro de la ejecución permite mostrar los errores específicos de cada analizador implementado, así como de todos los errores al mismo tiempo.




```
public class PilaErrores {
    public Stack<Error> pila = new Stack<~>();

    //MÉTODO PARA INGRESAR ALGÚN ERROR ENCONTRADO EN LA PILA DE ERRORES
    public void ingresa(int id,int linea) { pila.push(new Error(id,linea)); }

    //MÉTODO PARA MOSTRAR LO CONTENIDO EN LA PILA DE ERRORES
    public void mostrar(){...}

    //MÉTODO PARA IMPRIMIR EL CORRESPONDIENTE MENSAJE DE ERROR
    public String listaErrores(Error e){...}
}
```

CLASE "TABLASIMBOLOS"

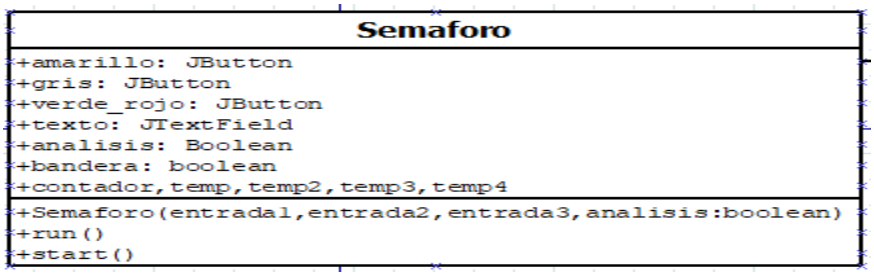
El uso de la tabla de símbolos cambia respecto al analizador léxico, en el anterior se generaba la tabla, en el analizador sintáctico la tabla de símbolos nos sirve a manera de consulta para verificar el correcto uso de la gramática estipulada.




```
public class TablaSimbolos {
    private ArrayList<Simbolo> tablaSimbolos = new ArrayList<>();

    //MÉTODO QUE MUESTRA LO CONTENIDO EN LA TABLA DE SIMBOLOS
    public void consultar(){...}

    //MÉTODO QUE PERMITE INGRESAR UN REGISTRO EN LA TABLA DE SIMBOLOS
    public void ingresar(Simbolo s){...}

    //MÉTODO QUE VACIA LA TABLA DE SIMBOLOS
    public void eliminar() { tablaSimbolos.clear(); }

    //MÉTODO QUE PERMITE BUSCAR UN REGISTRO DENTRO DE LA TABLA DE SIMBOLOS
    public Simbolo buscar(int id){...}
}
```

CLASE "PROCESOSEMAFORO"

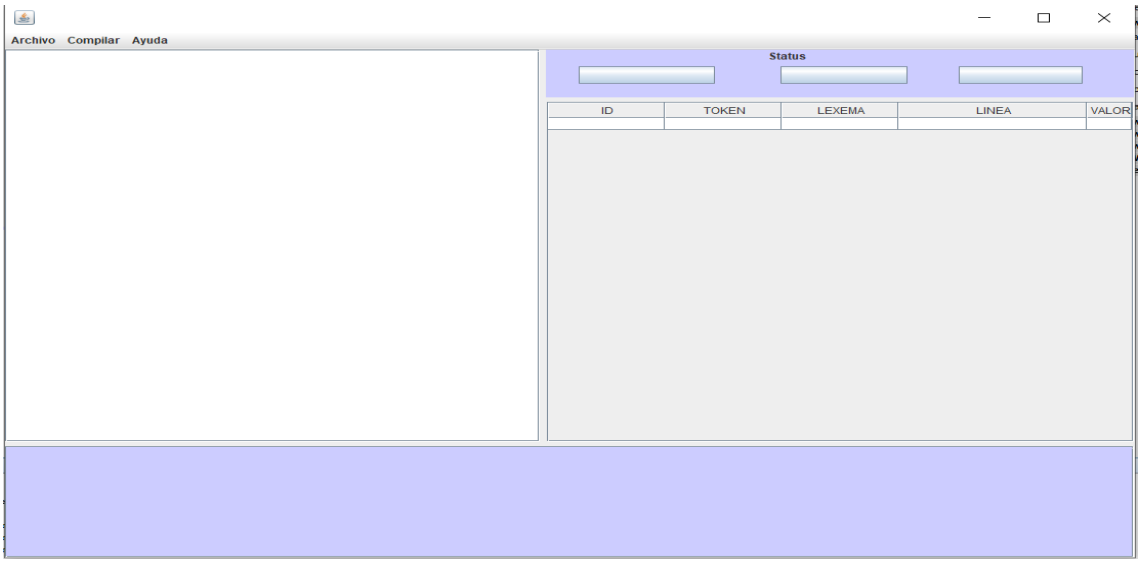
La clase en cuestión únicamente nos permite simular el comportamiento del semáforo, es decir hacer una "animación" hasta colorear los tres espacios designados del semáforo en lugar de que aparezcan los colores de manera repentina.



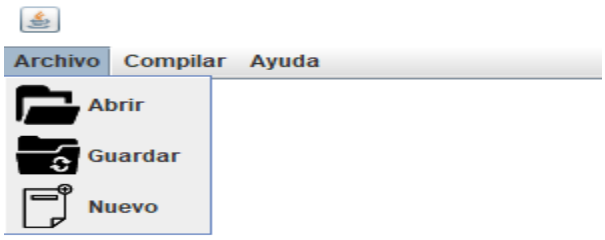
```
public class ProcesoSemaforo implements Runnable {

    JButton gris;
    JButton amarillo;
    JButton verde_rojo;
    JTextField texto;
    Boolean analisis,bandera;

    int contador=0,temp=0,temp2=0,temp3=0,temp4=0;

    public ProcesoSemaforo(JButton entrada1,JButton entrada2,JButton entrada3,JTextField entrada4,Boolean analisis){...}

    @Override
    public void run() {...}

    public void start() { new Thread( target: this).start(); }
}
```

INTERFAZ DE USUARIO

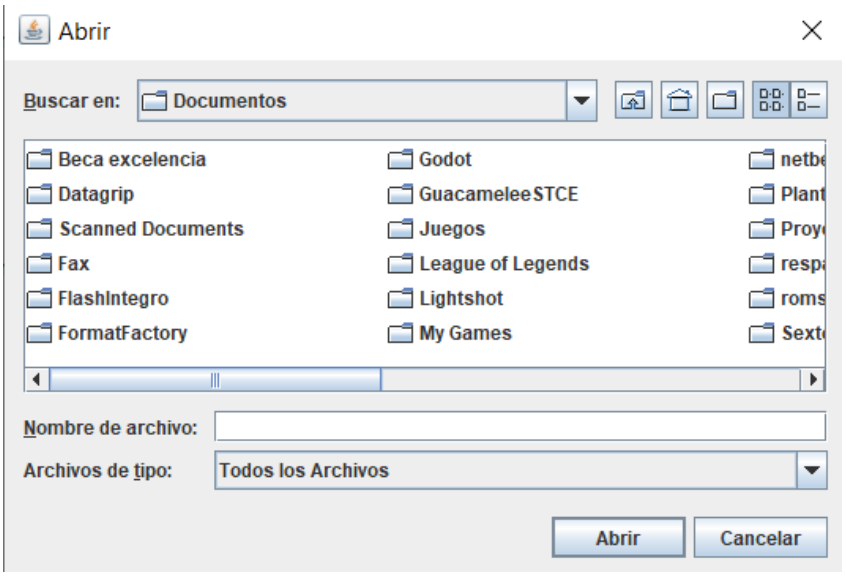
La vista principal de la interfaz de usuario es la siguiente:



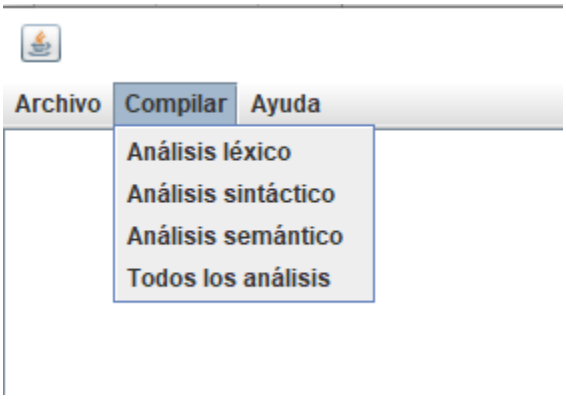
Para crear, guardar o abrir un archivo nos dirigimos a las siguientes opciones:



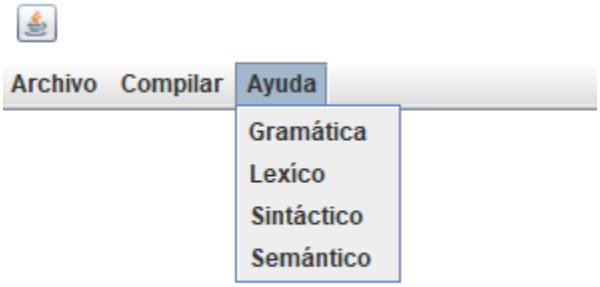
Se nos desplegaran ventanas como las siguientes al hacer uso de abrir y de guardar, al hacer uno nuevo se vacía nuestro espacio de código.



Al realizar la compilación, lo podemos hacer por partes, si decidimos hacer uso de un único análisis, si el código cuenta con un error, únicamente se mostrarán los errores referentes al análisis realizado, si se realizan todos los análisis se muestra todos los errores que surjan.

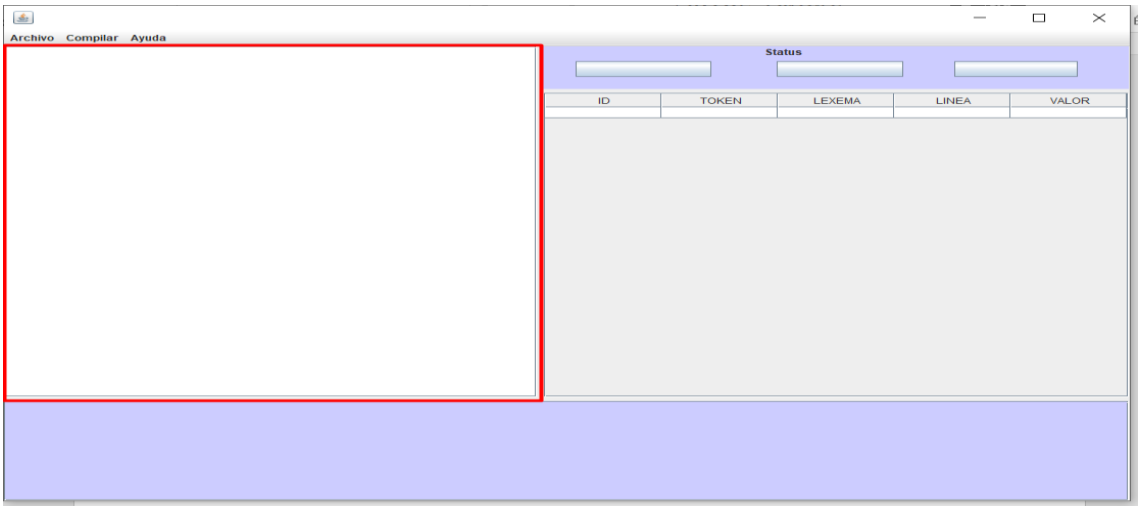


Mediante la opción ayuda se desplegarán los PDF referentes a la actividad realizada en cuestión, siendo gramática la actividad numero 1, léxico la actividad numero dos y de esta manera sucesivamente.



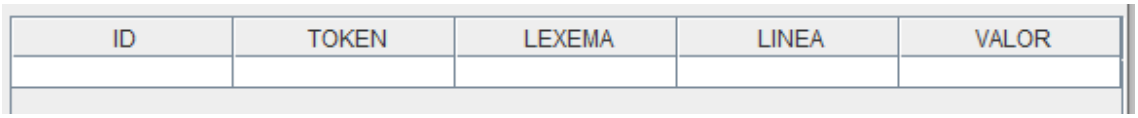
El PDF se desplegará en nuestro programa visualizador de documentos PDF preferido o en dado caso nos preguntará donde visualizarlo.

En el espacio marcado en color rojo es el espacio dedicado para escribir nuestro código de nuestros programas

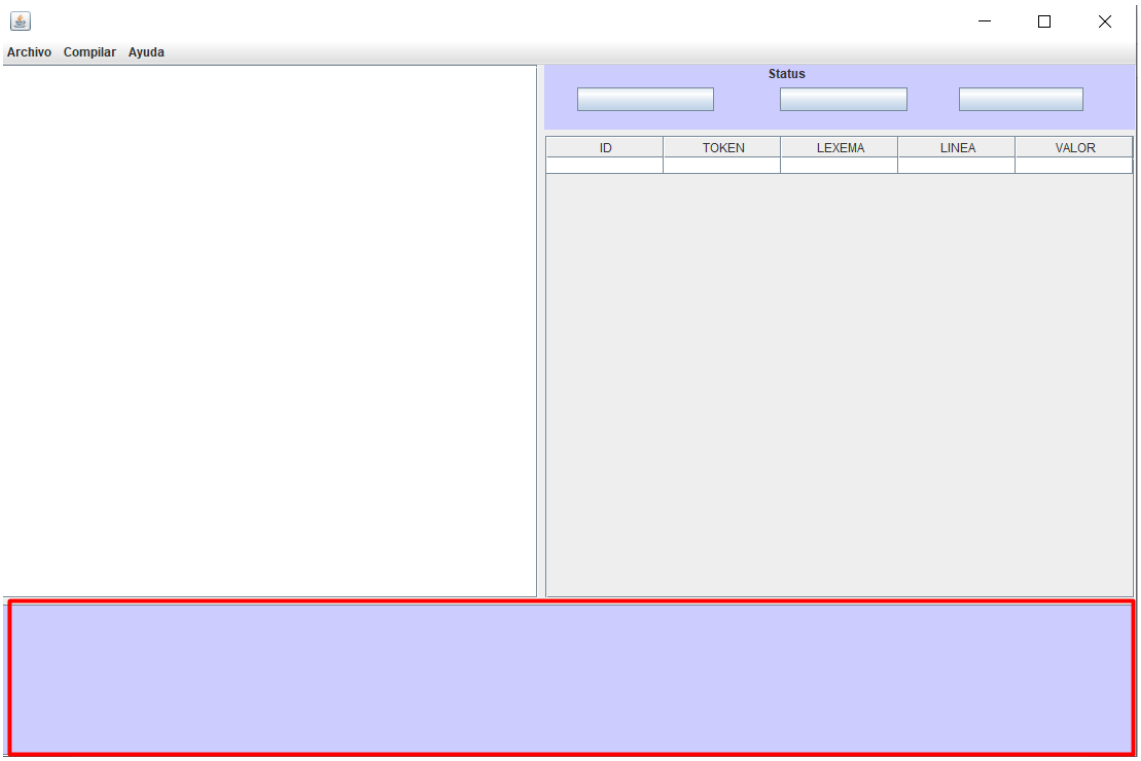


El espacio designado con la palabra “status”, es donde nuestro semáforo marca si fue compilado exitosamente mostrando un color verde al final o si genero algún error, mostrando el color rojo.

Donde vemos una table únicamente con las cabeceras se mostrará nuestra tabla de símbolos una vez se realice algún análisis.



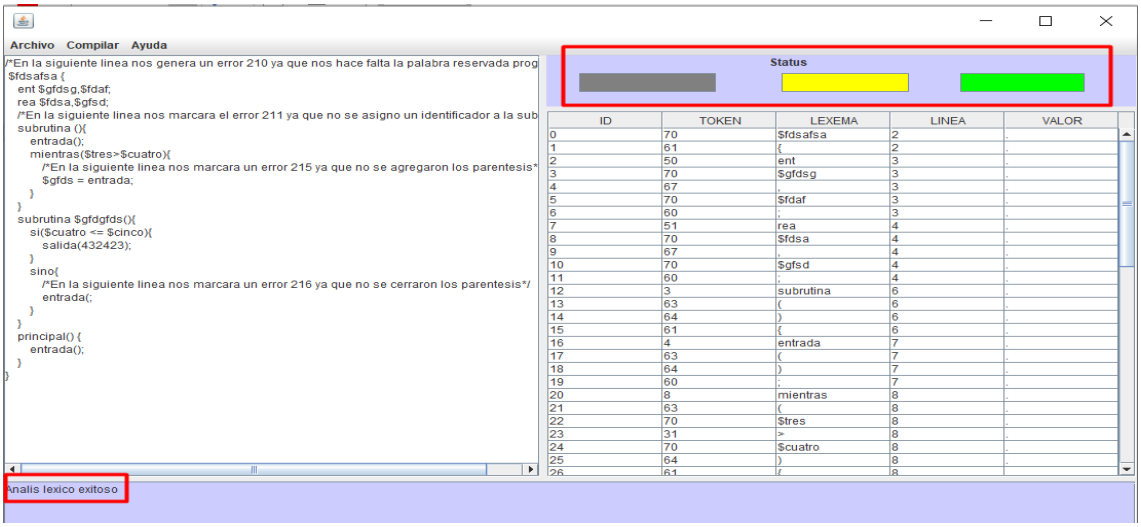
Por último, en el parte inferior marcado de rojo, es nuestra “consola” donde se muestran los mensajes que deban ser desplegados al usuario o en dado cado los errores que se generaron.



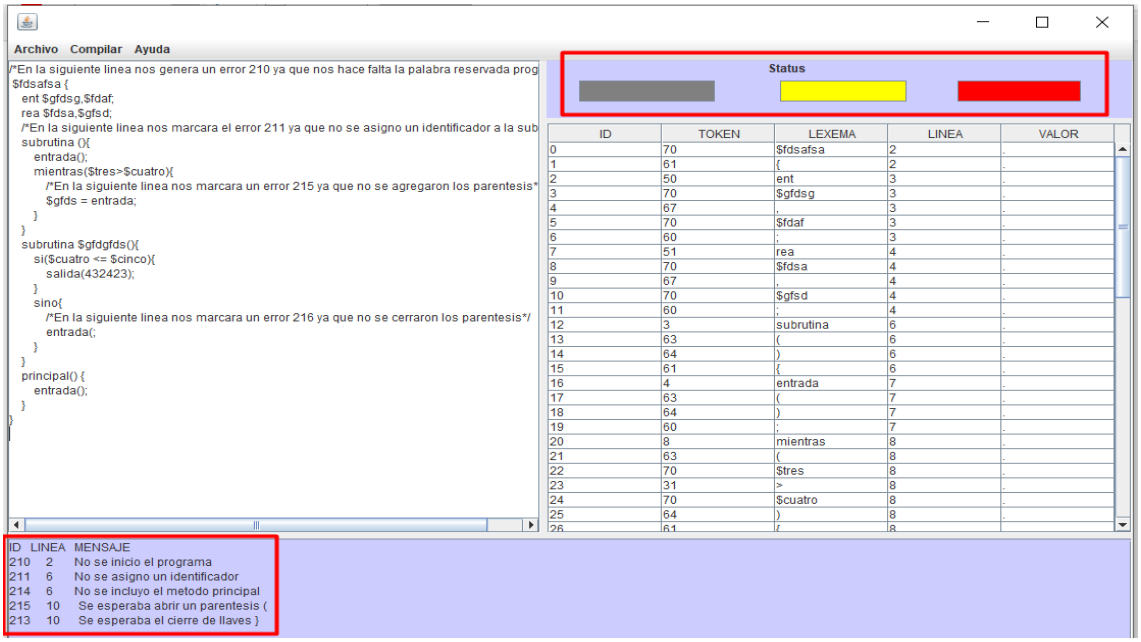
EJECUCIÓN DEL PROGRAMA

Para ejemplificar el comportamiento de nuestro Pseudocompilador veamos un ejemplo a continuación haciendo uso del programa prueba número 1 del presente documento.

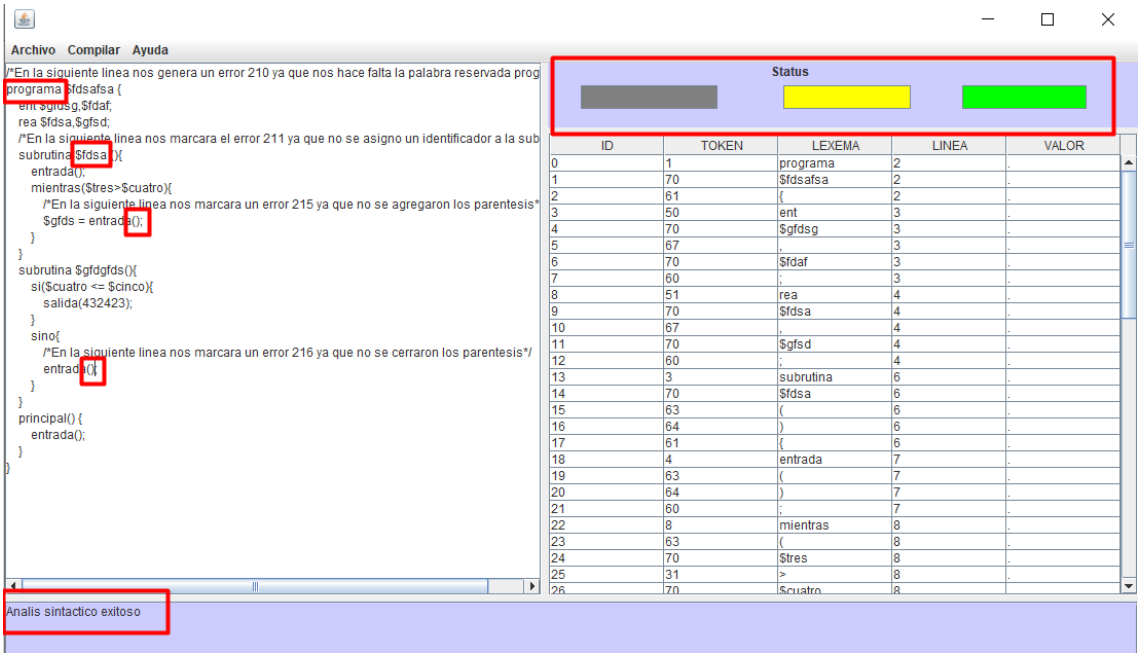
Si realizamos un análisis léxico nos encontramos con que todas las palabras se encuentran dentro del lenguaje, dado que es un análisis exitoso.



Al contrario de cuando hacemos un análisis sintáctico nos encontramos con que falla y sus respectivos errores.



Al corregir los errores nuestro compilador funciona correctamente.



En el proceso de corrección nos encontramos con nuevos errores, pero si solucionamos el principal lo normal es que todos los demás se solucionen correctamente.

BITACORA DE INCIDENCIAS			
Fecha	**Hora**	**Descripción de la incidencia**	**Solución**
Viernes 2 de mayo del 2020	7:00 PM	La forma de trabajo que se iba utilizar para realizar la actividad a distancia debido al confinamiento.	El equipo decidió organizarse por mensaje y ciertos días se haría reuniones a través de videollamadas para desarrollar correctamente esta y cumplir todos sus aspectos de manera adecuada.
Martes 5 de mayo del 2020	7:00 PM	Se solicito realizar algunos cambios a la GUI con base a un esquema enviado por el profesor.	El equipo se reunió para decidir qué contendría la GUI para ser más amigable al usuario y mejor organizada para observar todos los elementos importantes que debía mostrarse al usuario.
Viernes 8 de mayo del 2020	4:00 PM	Revisando la gramática que se tenía se pude concluir que algunos autómatas no eran correctos.	Se realizo una revisión de la gramática y de cada uno de los autómatas enfocados en el análisis sintáctico y se realizó la modificación de estos para llevar a cabo una correcta revisión.
Sábado 9 de mayo del 2020	8:00 AM	Se tenían dudas acerca de como sería la implementación del programa.	Se realizo una reunión donde se quedo de acuerdo a realizar las funcionalidades por un lado y la interfaz por otro para finalizando juntar las partes.

Lunes 11 de mayo de 2020	12:00 PM	Se tuvo problema el unir la GUI y el programa.	Se reviso y se procedió a realizar la unión de la GUI y el programa, realizando un pequeño cambio de IDE seleccionando NetBeans por su experiencia en su uso.