Bitácora de errores

Fecha	Problema	Solución (programación)
Léxico		
19/09/15	Abrir el archivo que el usuario elija, esto implica ruta dinámica.	El usuario puede elegir cualquier archivo txt desde cualquier ubicación por lo que se manejo la clase jFileChooser de java que permite la implementación de una ventana con el fin de abrir un archivo situado en cualquier ubicación del sistema.
20/09/15	Tabla de símbolos compuesta por campos de diferentes tipos.	La tabla de símbolos está compuesta por campos de diferentes tipos (Object, int, String), se podrían aplicar arrays paralelos pero para evitar tediosa codificación se implementa una clase abstracta con los diferentes atributos y a partir de ésta se genero un arraylist.
21/09/15	Clasificación de lexemas.	Implementación de arrays con las palabras reservadas, signos y símbolos del lenguaje a modo de utilizarlos como diccionarios y utilizar estos valores para la clasificación de lexemas.
22/09/15	Asignación de dirección	Se busca que la tabla de símbolos contenga la dirección de un lexema que ya ha sido declarado y se está volviendo a usar para hacer una referencia más eficaz, se recorre toda la tabla buscando la primera aparición y se asigna al resto de apariciones.
08/10/15	Aplicación de autómatas	Se reemplazó la forma de asignar el tipo de identificador a un lexema, en lugar de comparar cadenas sobre un diccionario se diseñaron y aplicaron autómatas para su análisis. La clasificación se hizo por medio de sólo dos letras en lugar de toda una palabra, ej. “identificador” = “id”.
09/10/15	Campo de id numérico en la tabla de símbolos.	Se agregó un atributo numérico a la clase MyObject que es la generadora del arraylist de la tabla de símbolos, cada tipo de lexema tiene un identificador numérico equivalente que será usado en el analizador sintáctico para verificar las estructuras gramaticales, siendo más fácil comparar un valor entero que una cadena.
09/10/15	Campo de línea en la tabla de símbolos.	Se agregó otro atributo numérico a la clase MyObject para el control de la línea en la que se encuentra cada lexema, lo cual resultará útil tanto

		para el análisis sintáctico como semántico.
Sintáctico		
17/10/15	Implementación de análisis sintáctico.	Aplicación del análisis sintáctico por medio de un autómata que maneja los casos que se pueden presentar, esto recorriendo uno a uno los elementos de la tabla símbolos generada en el análisis léxico.
19/10/15	El autómata evalúa cadenas.	Si bien ya se había en el analizador léxico que cada tipo de token tuviera un id numérico único, es más sencillo evaluar este id en las producciones de la gramática en lugar de evaluar cadenas.
19/10/15	Se tiene un solo autómata para todas las producciones de la gramática.	Dividir este autómata en sub-autómatas de manera que cada uno se vuelve un método pero el estado 0 de cada uno se sigue manejando desde un método principal y ahí se manda al autómata correspondiente conforme se evalúan los tokens de la tabla de símbolos.
20/10/15	Autómata de la función IMPRIME no está bien generado ya que no sigue las producciones de la gramática como debe ser.	Restructuración del autómata y reestructuración del método del mismo generando los estados faltantes.
20/10/15	La función SINO puede ser utilizada sin declarar previamente un SI.	Implementación de un ArrayList para simular una pila, cada que se encuentra un SI se agrega a la pila, al encontrar un SINO se saca un elemento de la pila, si la pila está vacía y se declara un SINO se marca un error sintáctico.
20/10/15	En los autómatas se evalúa el siguiente valor de la tabla de símbolos, esto puede causar excepciones al salirse del rango del array TablaSímbolos.	Implementación de estructuras Try-Catch en las “producciones” de los métodos que simulan los autómatas.
22/10/15	No clasifica como problema, pero se para optimizar el proyecto se eliminó el uso de todos los ArrayList.	Implementación de un clase Pila y de una clase Nodo para la creación de pilas para sustituir los ArrayList
22/10/15	Fallas en algunos autómatas del analizador sintáctico.	Mejoramiento de los autómatas agregando redirigimiento a un estado ficticio en el estado 0 cuando no cumple ninguna de las condiciones el carácter 1 pero sí el carácter 2.
24/10/1	Demasiado cargado y	Optimización de código. Corrección en la división

5	complejo el código de los autómatas. Fallas en el análisis léxico en el caso de uso de la función mientras.	de los lexemas para este caso.
Análisis Semántico		
29/10/15 30/10/15 31/10/15	Implementación del análisis semántico.	Agregación de una clase para el análisis semántico donde se cuenta con varios métodos, cada uno de ellos para los problemas que representan los casos de uso semánticos.Este análisis es clasificado como un análisis de varias pasadas ya que la tabla de símbolos es corrida repetidas veces durante su ejecución.
03/11/15	Al comparar valor numérico y observar su tamaño, se encontraron problemas al tomar el número y al evaluar negativos.	Se verifica que el lexema de la tabla de símbolos que se vaya a comparar sea de id 2, ya que es los números. Al tomar el negativo se toma en cuenta que este se asigna con paréntesis.
03/11/15	En la parte de revisar variables declaradas pero no usadas, no se reiniciaba el contador que cuenta cuantas veces una variable está en el programa.	Después de revisar si había problema se reinicia el contador a 0 para evitar fallas internas.
03/11/15	Se requiere de una interfaz para manejar los analizadores, consultar la tabla de símbolos y editar el código.	Implementación de una clase interfazMain que contiene cajas de texto (JTextArea), botones (JButton) y menús (JMenuBar, JMenu, JMenuItem) para manejar todo el proceso de los análisis léxico, sintáctico y semántico, desde esta interfaz con sólo presionar botones. También es posible editar el código a procesar desde esta interfaz. Todo esto se lleva a cabo por medio de eventos que ejecutan los análisis generando instancias de las clases correspondientes a cada análisis.