



Escuela de Computación

Segundo Proyecto

Analizador Sintactico

Jean Paul Rodríguez Flores

Carné: 2020059156

José Mauricio Muñoz Morera

Carné: 2020233164

Taylor Hernández Córdoba

Carné:2020196104

Compiladores e Intérpretes

Prof. Erika Marin

II semestre 2022

Tabla de contenidos

Introducción	3
Estrategia de Solución	4
Análisis de Resultados	5
Lecciones aprendidas	6
Casos de pruebas	7
Manual de usuario	9
Bitácora de trabajo	11
Bibliografía y fuentes digitales	12

Introducción

Esta documentación es sobre la elaboración de un analizador sintáctico de un compilador. Los analizadores sintácticos o comúnmente llamados parser son el segundo paso en el proceso de compilación de un lenguaje, este analiza la estructura de las expresiones en base a gramáticas. El análisis que se realiza es jerárquico es decir en base a árboles de derivación que se obtienen de las mismas gramáticas.

Este proyecto está elaborado en el lenguaje java y utiliza la librería cup, la cual brinda las funcionalidades de crear gramáticas a partir de los tokens obtenidos del analizador léxico. Devuelve los errores de sintaxis encontrados sin detenerse en el primero, esto realizando ciertas producciones para identificar los errores.

Respectivamente este parser, además de hacer la tarea de mostrar los errores, muestra una lista con todos los tokens válidos con su nombre, tipo de token, la cantidad de veces que aparece las diferentes líneas que lo hace y los errores de sintaxis encontrados definiendo la línea y la columna donde se ubican.

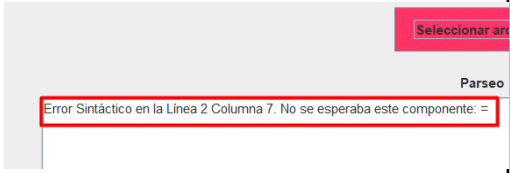
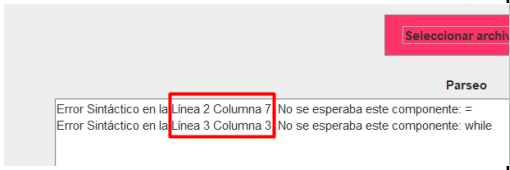
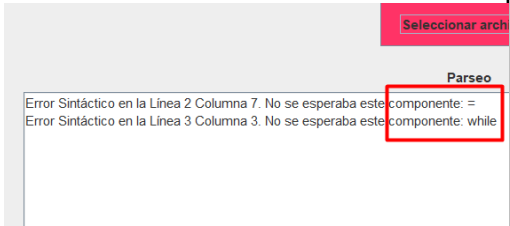
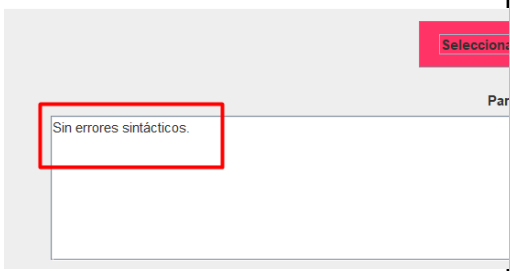
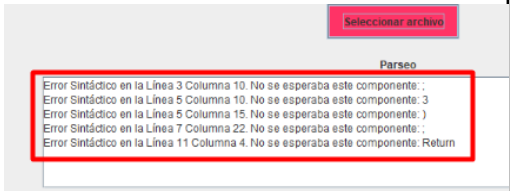
Estrategia de Solución

Se elaboró un analizador sintáctico en el lenguaje java, donde se puede utilizar la librería cup, la cual funciona propiamente como un parser, posee un archivo .flex donde se definen los tokens o terminales con un respectivo identificador que serán utilizados. Para la elaboración de este analizador se requiere de la previa realización del analizador léxico elaborado en el primer proyecto. Luego posee un archivo .cup que es donde se definen los tokens como terminales y los no terminales.

Luego se definen todas las gramáticas necesarias para los elementos solicitados de parsear para el lenguaje C con todas las estructuras. Contemplando los posibles errores que podrían ocurrir para aceptarlos, pero capturarlos a la vez, para que puedan ser todos detectados y no solo el primero a cómo funciona la librería cup.

Finalmente se elaboró una pantalla para mostrar la información de los errores encontrados, tanto los tokens del analizador léxico como lo hacía antiguamente, como los errores sintácticos encontrados. Junto con un objeto controlador que es el encargado de hacer todas estas funciones al leer un archivo .txt indicado por medio de un explorador de archivos también implementado para mayor facilidad.

Análisis de Resultados

Tarea	% Completitud	Justificación
Desplegar los errores en caso de que hayan.	100%	
Desplegar línea y columna en la que se encontró el error.	100%	
Desplegar el componente de error.	100%	
Desplegar un rótulo en caso que no se hayan encontrado errores.	100%	
No parar en el primer error	100%	

Lecciones aprendidas

- El estudiar la documentación de cualquier elemento, en este caso la librería, es sumamente importante para entender el debido funcionamiento de este y lograr obtener el mayor potencial de la herramienta.
- Los videotutoriales son de gran ayuda para conocer herramientas desde otra perspectiva o algunos funcionamientos que se desconocen.
- La práctica realizada en clases fue de gran ayuda, ya que la forma en las que se solucionan los errores es la misma.
- Hay que planear de antemano las producciones y estructuras a implementar, para que estén en orden y no sea muy complicado llegar a los errores.

Casos de pruebas

ejemplo #1

```
int Main(){  
  
    int x=;  
    int z=10;  
    while 3<30)){  
        x=x+5;  
        printf("hola ", x);  
    }  
    printf("fin");  
    int y = x + z  
    Return (y);  
}
```

ejemplo #2

```
int main()  
{  
    char resultado;  
  
    resultado=5+2;  
    printf("Resultado ", resultado);  
    resultado=5-2;  
    printf("Resultado ", resultado);  
    resultado=5*2;  
    printf("Resultado ", resultado);  
    resultado=5/2;  
    printf("Resultado ", resultado);  
  
    return(0);  
}
```

Resultado:

ejemplo #1:

Tokens			Errores		
Token	Tipo de Token	Línea	Error	Tipo de Error	Línea
int	Int	1, 3, 4, 10,			
Main	Identificador	1,			
(Operador_Parentesis_Izqu...	1, 7, 9, 11,			
)	Operador_Parentesis_Der...	1, 5 (2), 7, 9, 11,			
{	Operador_Corchete_Izquie...	1, 5,			
x	Identificador	3, 6 (2), 7, 10,			
=	Operador_Asignacion	3, 4, 6, 10,			
,	Operador_Puntoycoma	3, 4, 6, 7 (2), 9, 11,			
z	Identificador	4, 10,			
10	Literal_Entero	4,			
while	While	5,			
3	Literal_Entero	5,			
<	Operador_Menor	5,			
30	Literal_Entero	5,			
+	Operador_Suma	6, 10,			
z	Identificador	6,			

Seleccionar archivo

Parseo

Error Sintáctico en la Línea 3 Columna 10. No se esperaba este componente: ;
 Error Sintáctico en la Línea 5 Columna 10. No se esperaba este componente: 3
 Error Sintáctico en la Línea 5 Columna 15. No se esperaba este componente:)
 Error Sintáctico en la Línea 7 Columna 22. No se esperaba este componente: ;
 Error Sintáctico en la Línea 11 Columna 4. No se esperaba este componente: Return

ejemplo #2:

Tokens			Errores		
Token	Tipo de Token	Línea	Error	Tipo de Error	Línea
nt	Int	1,			
main	Identificador	1,			
(Operador_Parentesis_Izqu...	1, 5, 7, 9, 11, 13,			
)	Operador_Parentesis_Der...	1, 5, 7, 9, 11, 13,			
{	Operador_Corchete_Izquie...	2,			
char	Char	3,			
resultado	Identificador	3, 4, 5, 6, 7, 8, 9, 10, 11,			
,	Operador_Puntoycoma	3, 4, 5, 6, 7, 8, 9, 10, 11, 13,			
=	Operador_Asignacion	4, 6, 8, 10,			
5	Literal_Entero	4, 6, 8, 10,			
+	Operador_Suma	4,			
2	Literal_Entero	4, 6, 8, 10,			
printf	Identificador	5, 7, 9, 11,			
Resultado "	Literal_CADENA	5, 7, 9, 11,			
,	Operador_Coma	5, 7, 9, 11,			
z	Identificador	6,			

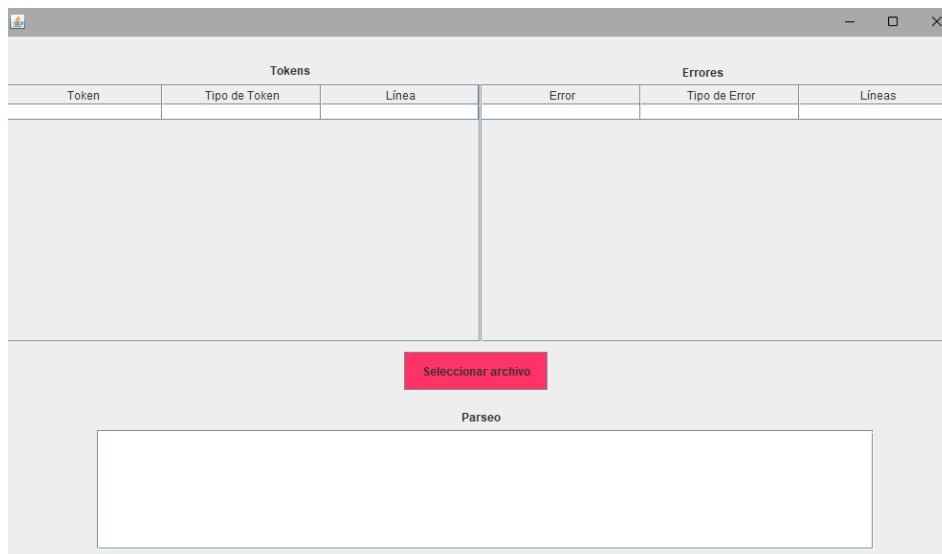
Seleccionar archivo

Parseo

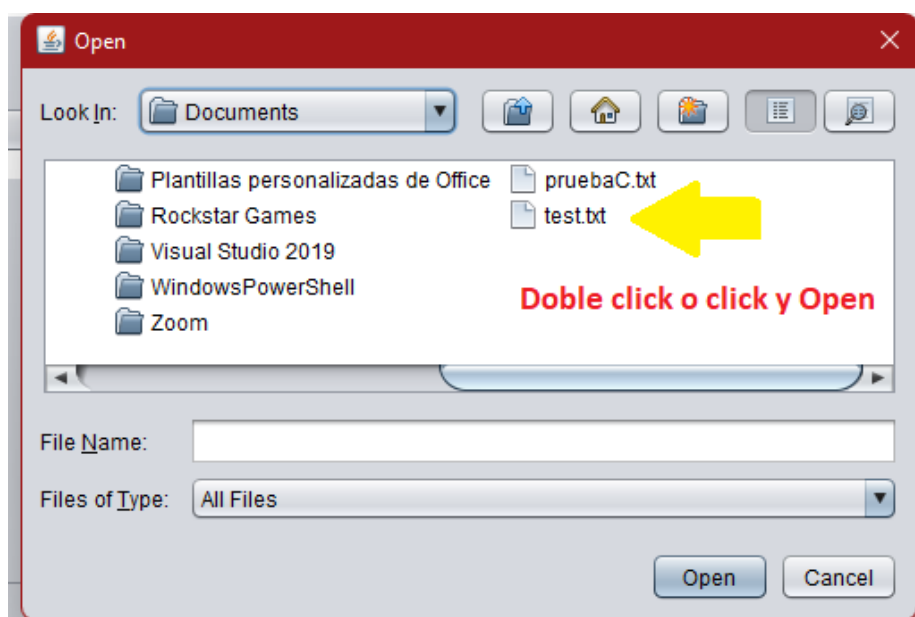
Sin errores sintácticos.

Manual de usuario

1. Iniciar el programa:



2. Seleccionar el archivo a compilar:



Bitácora de trabajo

Fecha	Actividades	Duración	Responsables
19/10/2022	<ul style="list-style-type: none"> • Análisis de la especificación del proyecto 	2h	<ul style="list-style-type: none"> • Jean Paul • Mauricio • Taylor
21/10/2022	<ul style="list-style-type: none"> • Investigación sobre la librería Cup 	2h	<ul style="list-style-type: none"> • Jean Paul • Mauricio • Taylor
25/10/2022	<ul style="list-style-type: none"> • Inicio del proyecto programado 	1h	<ul style="list-style-type: none"> • Jean Paul • Mauricio • Taylor
26/10/2022	<ul style="list-style-type: none"> • Creación de archivo Jflex específico para el Cup y los tokens que va a reconocer 	1h	<ul style="list-style-type: none"> • Jean Paul
27/10/2022	<ul style="list-style-type: none"> • Creación del archivo Cup para realizar el parser e inicio de generación de gramáticas 	1h	<ul style="list-style-type: none"> • Jean Paul
28/10/2022	<ul style="list-style-type: none"> • Continuación con las gramáticas 	1h	<ul style="list-style-type: none"> • Jean Paul • Taylor
30/10/2022	<ul style="list-style-type: none"> • Solución de errores shift/reduce y reduce/reduce 	1h	<ul style="list-style-type: none"> • Jean Paul
1/11/2022	<ul style="list-style-type: none"> • Modificación de la pantalla y el controlador para los nuevos errores 	2h	<ul style="list-style-type: none"> • Mauricio
2/11/2022	<ul style="list-style-type: none"> • Pruebas con archivos .c 	2h	<ul style="list-style-type: none"> • Jean Paul • Mauricio • Taylor
4/11/2022	<ul style="list-style-type: none"> • Modificaciones en la gramática • Solución de errores shift/reduce y reduce/reduce 	3h	<ul style="list-style-type: none"> • Jean Paul • Mauricio • Taylor
5/11/2022	<ul style="list-style-type: none"> • Realización de pruebas finales 	1h	<ul style="list-style-type: none"> • Jean Paul • Mauricio • Taylor
6/11/2022	<ul style="list-style-type: none"> • Realización de la documentación 	3h	<ul style="list-style-type: none"> • Jean Paul • Mauricio • Taylor

Bibliografía y fuentes digitales

- <http://www.cs.princeton.edu/~appel/modern/java/CUP/manual.html#intro>
- <http://www2.cs.tum.edu/projects/cup/>
- [CUP User's Manual \(princeton.edu\)](http://www.cs.princeton.edu/~appel/modern/java/CUP/manual.html#intro)
- [ANALIZADOR LÉXICO, SINTÁCTICO Y SEMÁNTICO DEL LENGUAJE PASCAL UTILIZANDO JFLEX Y CUP \(analizadoresjflexcup.blogspot.com\)](http://www2.cs.tum.edu/projects/cup/)
- <https://youtu.be/4Z6Tnit810Y>
- <https://ericknavarro.io/2019/04/26/02-Mi-primer-proyecto-utilizando-Jflex-y-Cup-Windows/>