

I Semestre del año 2021

Compiladores e intérpretes

Proyecto 1 – Análisis Léxico y Sintáctico

Stasey Nerdrick Carvajal – 2017124886

Angie Cooper García - 2017120209

**Contenido**

[**Manual de usuario** 3](#_Toc42552643)

[Pre condiciones 3](#_Toc42552644)

[Pasos para la ejecución del programa 4](#_Toc42552645)

[**Pruebas de funcionalidad** 15](#_Toc42552646)

[**Descripción del problema** 22](#_Toc42552647)

[**Diseño del programa** 22](#_Toc42552648)

[Gramática 24](#_Toc42552649)

[**Librerías usadas** 29](#_Toc42552650)

[JFlex 29](#_Toc42552651)

[Cup 29](#_Toc42552652)

[**Análisis de resultados** 30](#_Toc42552653)

[Objetivos alcanzados 30](#_Toc42552654)

[**Bitácora** 31](#_Toc42552655)

[**Referencias** 32](#_Toc42552656)

# **Manual de usuario**

## Pre condiciones

Antes de iniciar con los pasos para el uso correcto del programa se debe tener en cuenta que es necesario contar con un ambiente de desarrollo de Java, en este caso se recomienda instalar el IDE de NetBeans **8.2** (\*Asegurarse de que la versión sea la 8.2\*).

En caso de que no se cuente con la herramienta de NetBeans se puede buscar en Youtube un tutorial para la instalación del mismo, en la sección de referencias puede encontrar un link directo al video “Descarga e Instala NetBeans IDE 8.2 y Java Development Kit para Windows 7/8/8.1/10” disponible en youtube que puede serle útil para la instalación de NetBeans.

Para la ejecución correcta del programa también es necesario tener las librerías JFlex y CUP, ambas son indispensables para que el programa se ejecute correctamente. Estas librerías vienen incluidas en el archivo del programa, en caso de que no sea así, los pasos a seguir serían los siguientes:

1. Descargar las librerías JFlex y CUP de sus páginas oficiales. A continuación se mostrarán los links de cada librería respectivamente:

<https://jflex.de/download.html>

<http://www2.cs.tum.edu/projects/cup/install.php>

1. Una vez descargadas ambas librerías lo siguiente es incluirlas en el proyecto. En la sección de referencias se podrá encontrar un video llamado “Como agregar una librería a NetBeans” que se encuentra disponible en la plataforma de Youtube. Cabe destacar que en este video crean un proyecto en NetBeans desde cero, por lo que los pasos deben aplicarse sobre el proyecto del programa (en los siguientes pasos se indicará como abrir un programa en Netbeans), en este proyecto buscarán la carpeta “Libraries” y a partir de ahí seguirán los pasos del video.

## Pasos para la ejecución del programa

Captura de pantalla de un celular

Descripción generada automáticamentePrimeramente se debe descargar el archivo “PP1\_CooperAngie\_NerdrickStasey.zip”, una vez descargado el archivo, el siguiente paso es descomprimirlo, para descomprimir el archivo se debe dar click derecho sobre el mismo y elegir la opción “Extraer en PP1\_CooperAngie\_NerdrickStasey\” (ver la figura 1).

Figura 1

Una vez descomprimido el archivo zip, se creará una carpeta llamada “PP1\_CooperAngie\_NerdrickStasey”, dentro deberán haber dos carpetas llamadas “Documentación” y “Programa”, además, también se encontrará un archivo de texto llamado “info.txt”. Dentro de la carpeta “Programa” se debe verificar que se encuentren las siguientes carpetas:

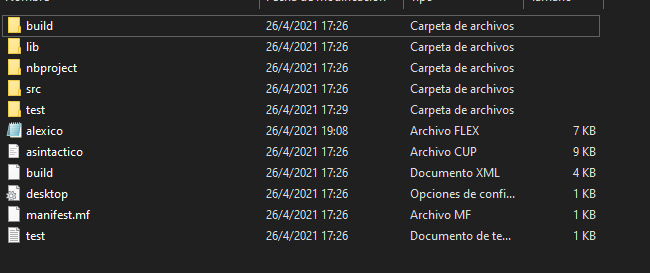
* Dentro de la carpeta “Compilador” se deben encontrar los siguientes archivos (ver figura 2).

Figura 2

Verificar que dentro de la carpeta “lib” se encuentren las librerías de JFlex y CUP, en caso de que no estén, ir a la sección de pre condiciones e incluirlas en el proyecto como se indica en la sección.

Una vez que se verificó que estuvieran los archivos indicados anteriormente, el siguiente paso es abrir la aplicación de NetBeans, en caso de no tenerla instalada ir a la sección de pre condiciones y seguir los pasos indicados para su instalación.

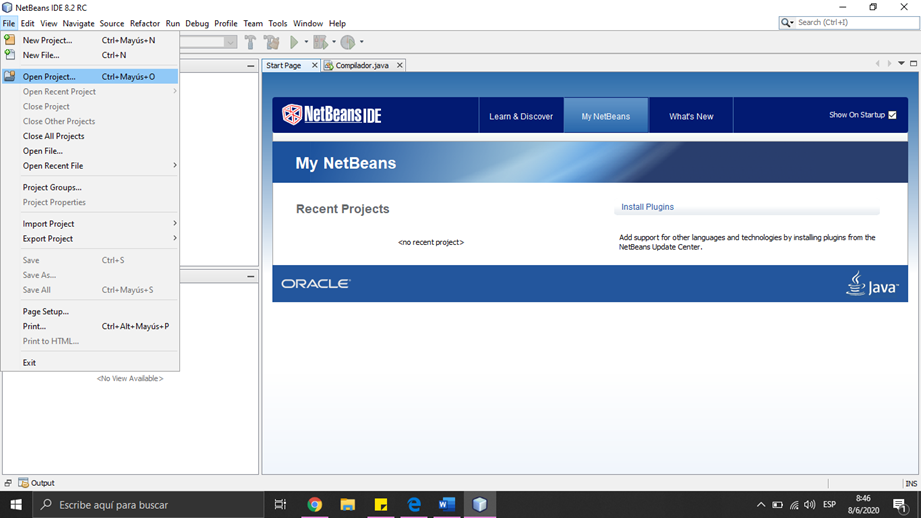
Una vez abierta la aplicación de NetBeans el próximo paso es abrir el proyecto (ver figura 3).

Figura 3

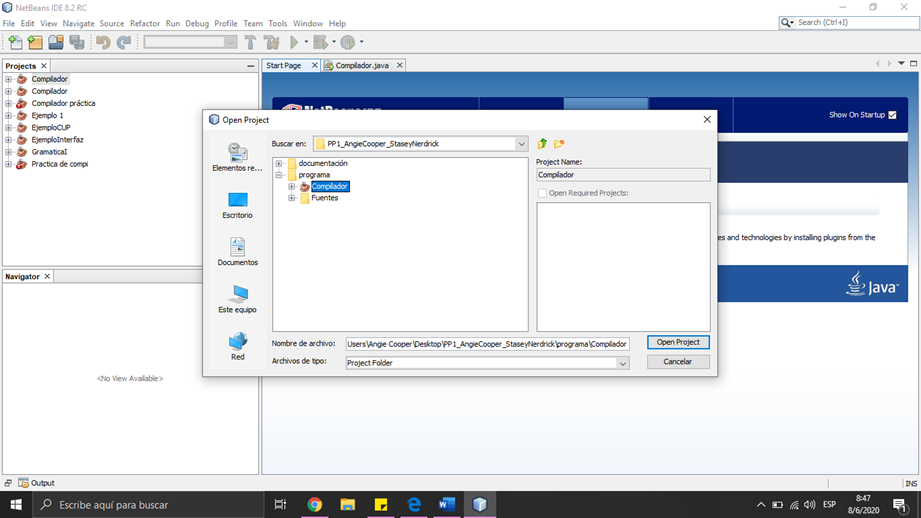
Después buscamos la ubicación en la que se encuentra la carpeta “PP1\_CooperAngie\_NerdrickStasey”, ingresamos a la carpeta llamada “Programa” y elegimos el archivo llamado “Compilador”, tiene un ícono al lado de una taza de café color naranja (ver figura 4).

Figura 4

Una vez abierto el proyecto, lo siguiente es elegir el archivo que contiene el main. Para esto elegimos la carpeta “Source Packages”, esta se puede ver a mano derecha debajo del proyecto “Compilador”, dentro de la carpeta “Source Packages” damos doble click sobre el archivo “Compilador.java”, una vez que se abre el archivo ejecutamos el programa haciendo click sobre la la flecha color ver que aparece en la barra superior de la aplicación (ver figuras 5 y 6).

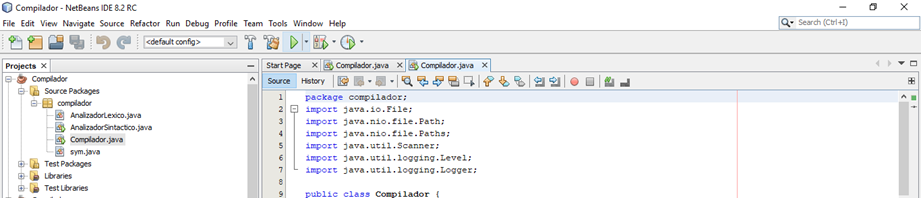
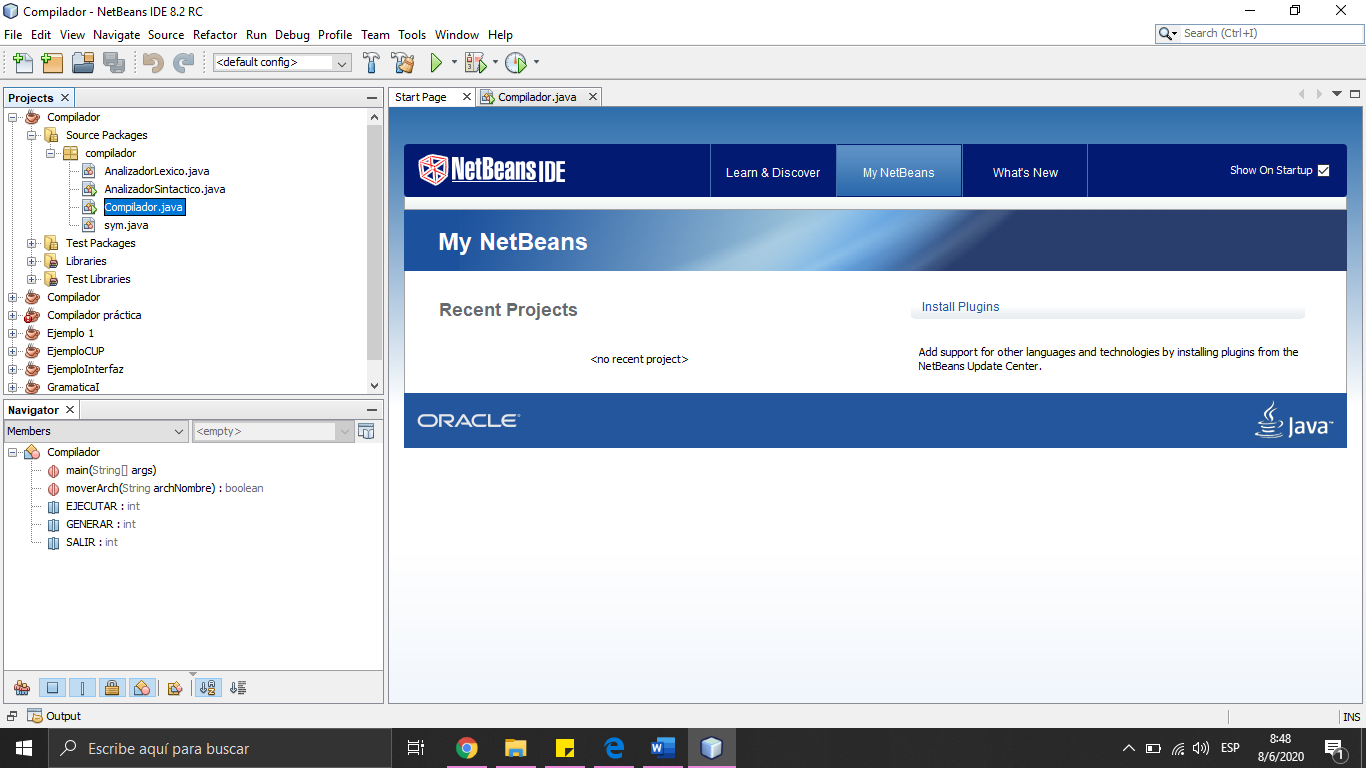


Figura 5

Figura 6

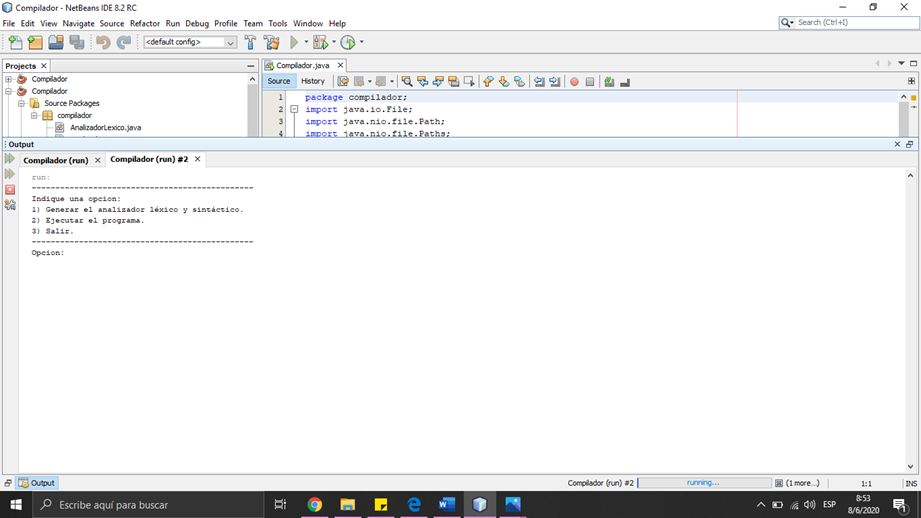
Después de ejecutar el archivo “Compilador.java”, se mostrará el siguiente menú (ver figura 7).

Figura 7

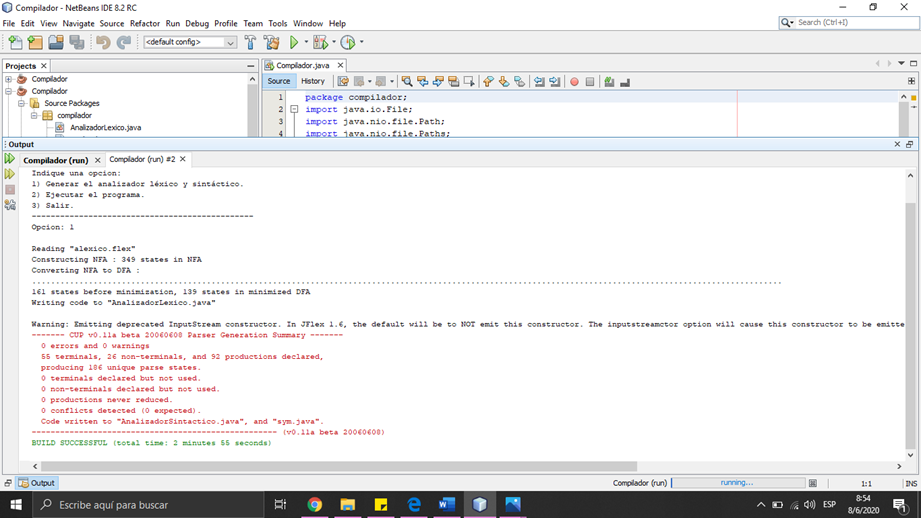
Al elegir la opción 1 se vuelve a generar el analizador léxico y sintáctico. Esto se hace en caso de que se le apliquen modificaciones a los archivos “alexico.flex” y “asintactico.cup” que se encuentran dentro de la carpeta “Programa” del proyecto; el fin de volver a generar los analizadores es que se reflejen las modificaciones que se le hayan hecho a los mismos. A continuación se muestra un ejemplo de la consola al elegir la opción 1 (ver figura 8).

Figura 8

Al elegir la opción 2 el programa lee como entrada lo que hay en el archivo de texto llamado “test.txt”, este archivo se encuentra dentro de la carpeta “Compilador” que esta a su vez está en la carpeta llamada “Programa”. En el archivo “test.txt” se coloca el código fuente a utilizar para que sea analizado por el analizador léxico y sintáctico.

Por ejemplo, si el archivo “test.txt” tiene escrito el siguiente código dentro de el (ver figura 9).

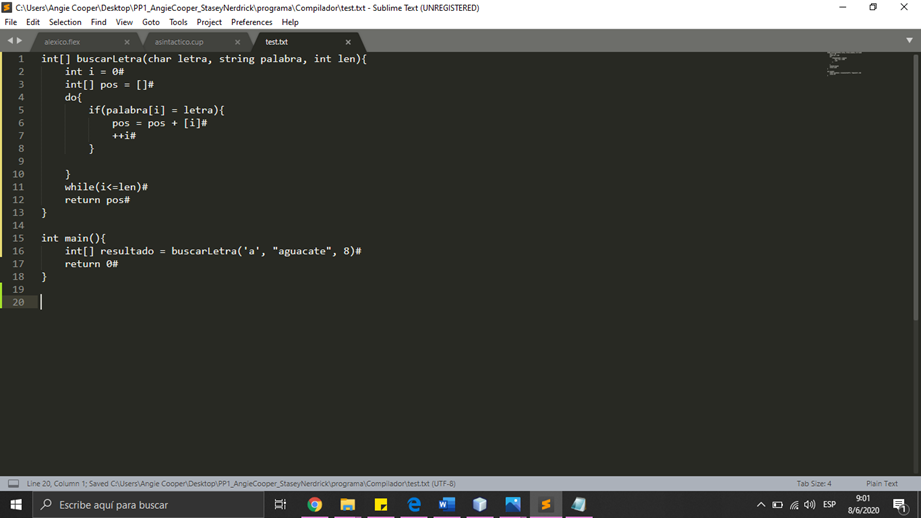
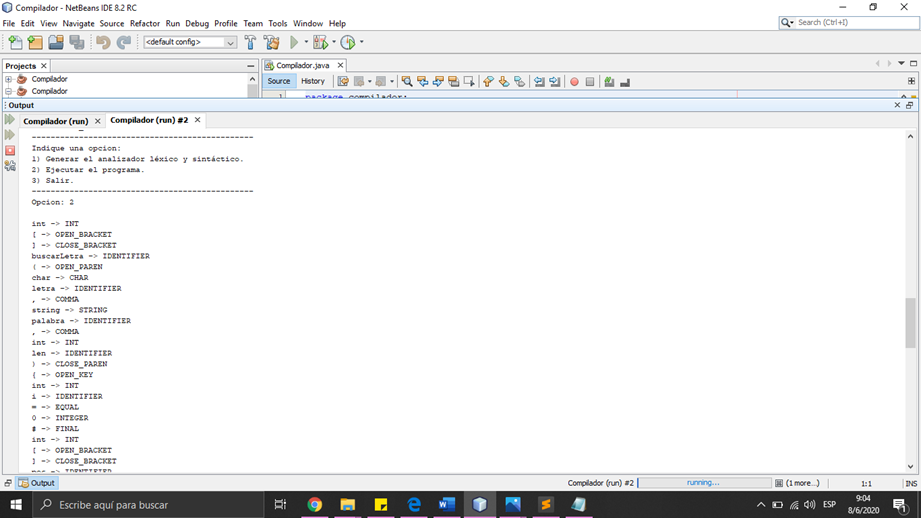
Al elegir la opción 2 del menú, lo que se muestra a continuación es lo que imprimirá la consola (ver figura 10).

Figura 10

Figura 9

En la consolá se mostrará lo que el programa leyó del código fuente junto al token correspondiente. Luego de mostrar la información leída el programa vuelve al menú inicial(ver figura 11).

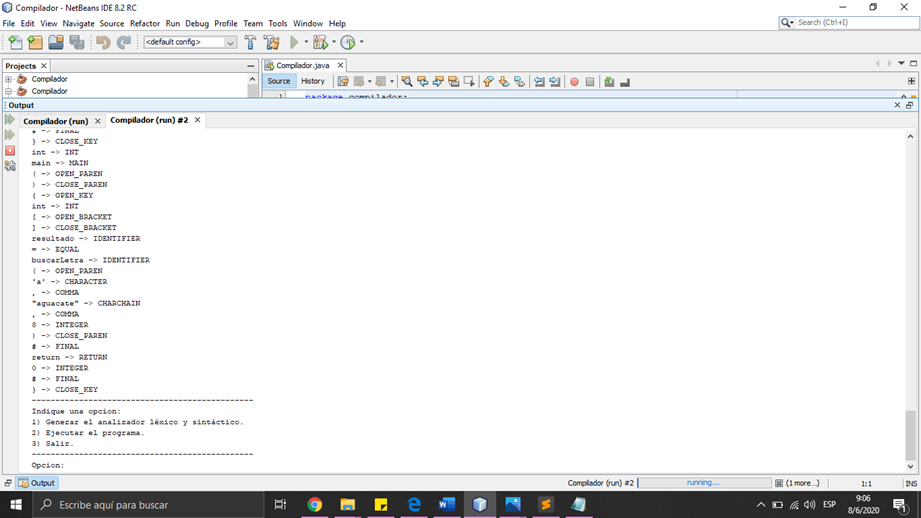


Figura 11

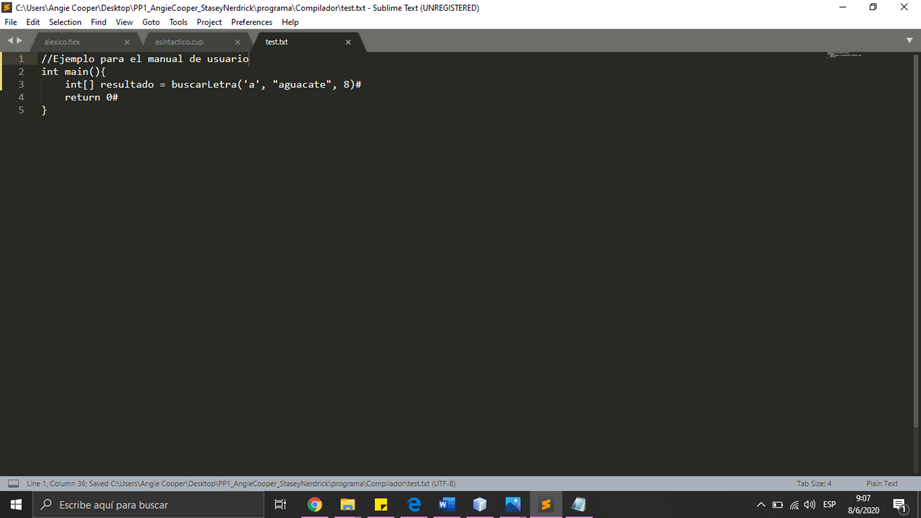
A continuación se mostrará un ejemplo con un código más pequeño dentro del archivo “test.txt” (ver figura 12 y 13).

Figura 12

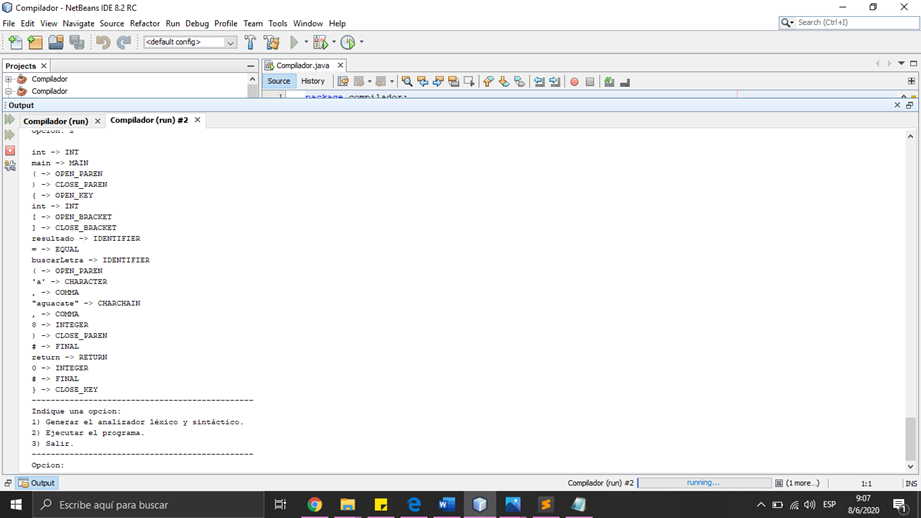
Otro ejemplo, es cuando hay errores en el código fuente, el programa indica el número de línea y la columna en la que se encuentra. A continuación se muestra como ejemplo un código en el que hace falta el final de línea, que es el símbolo “#” (ver figura 15).

Figura 13

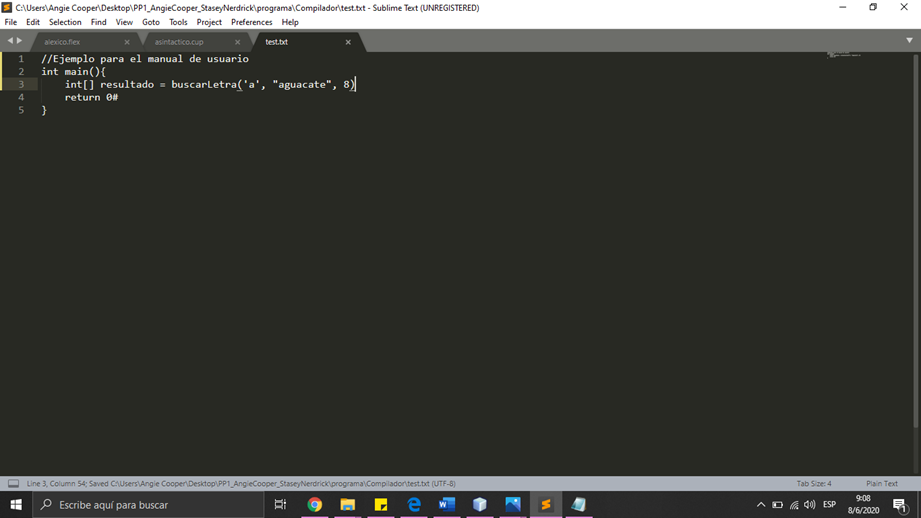
Una vez elegida la opción 2 del menú, el programa muestra lo siguiente según lo leído en el archivo “test.txt” (ver figura 16).

Figura 15

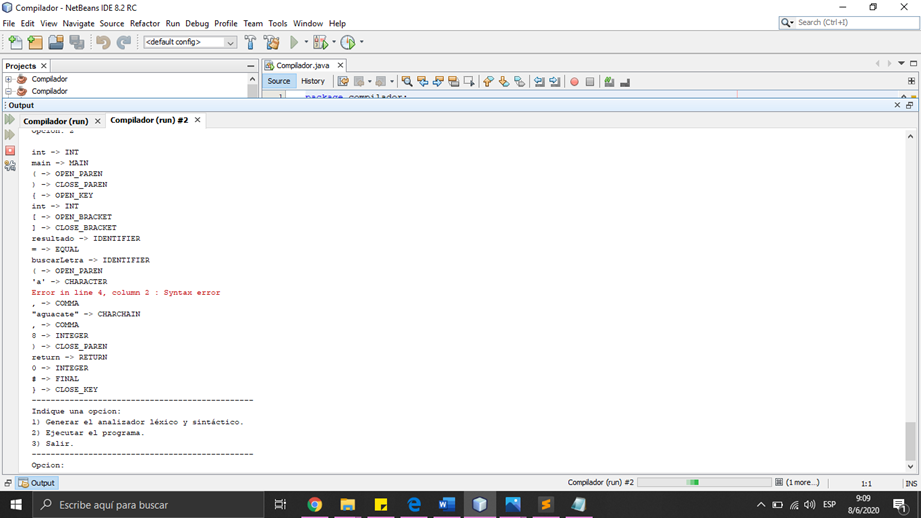
En la figura 16 se puede observar como el programa indica el error encontrado, especificando la linea y la columna en la que se encuentra el error sintáctico.

Figura 16

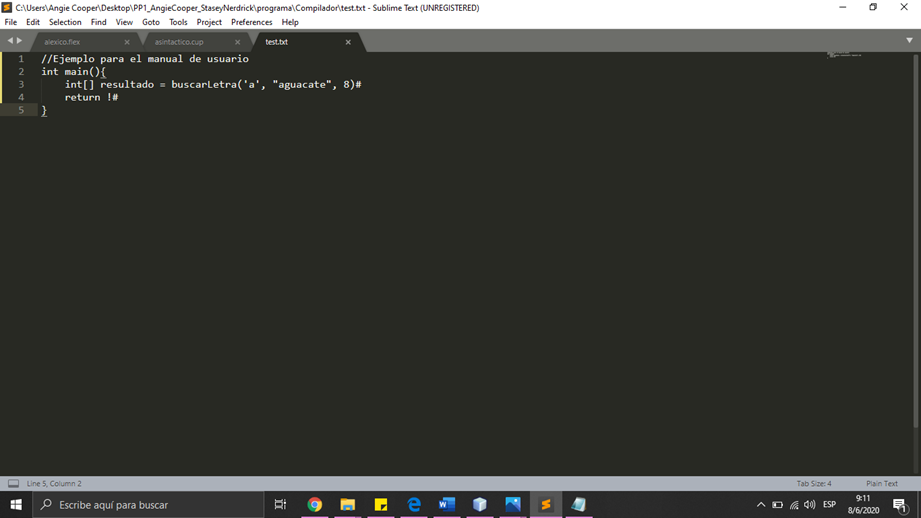
El programa también indica errores léxicos. Por ejemplo, a continuación se muestra un código en el que se hace uso de un lexema que no existe en el código (ver figura 17).

Figura 17

El programa lee el código e imprime en pantalla el error léxico encontrado, indicando también la linea y columna específica en la que se encuentra el error (ver figura 18).

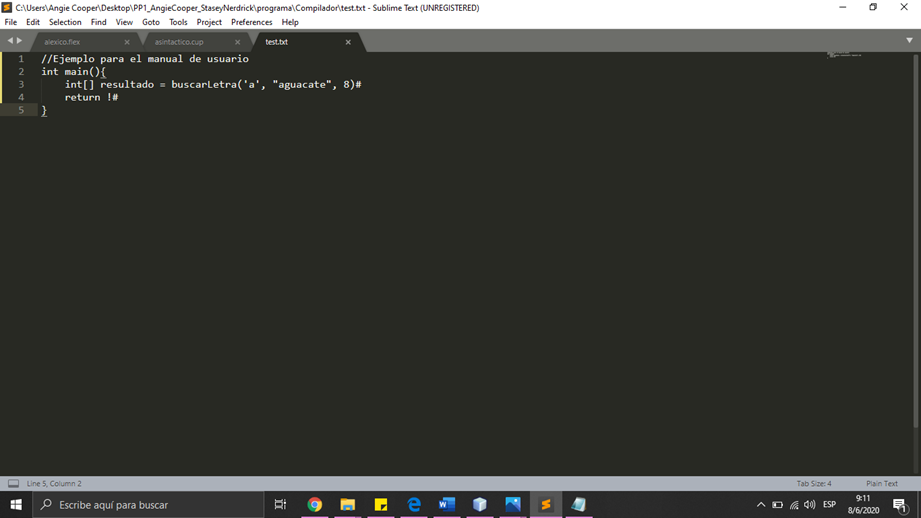


Figura 18

# **Pruebas de funcionalidad**

Esta es una muestra de lo que se encuentra en test.txt, el archivo fuente del compilador.

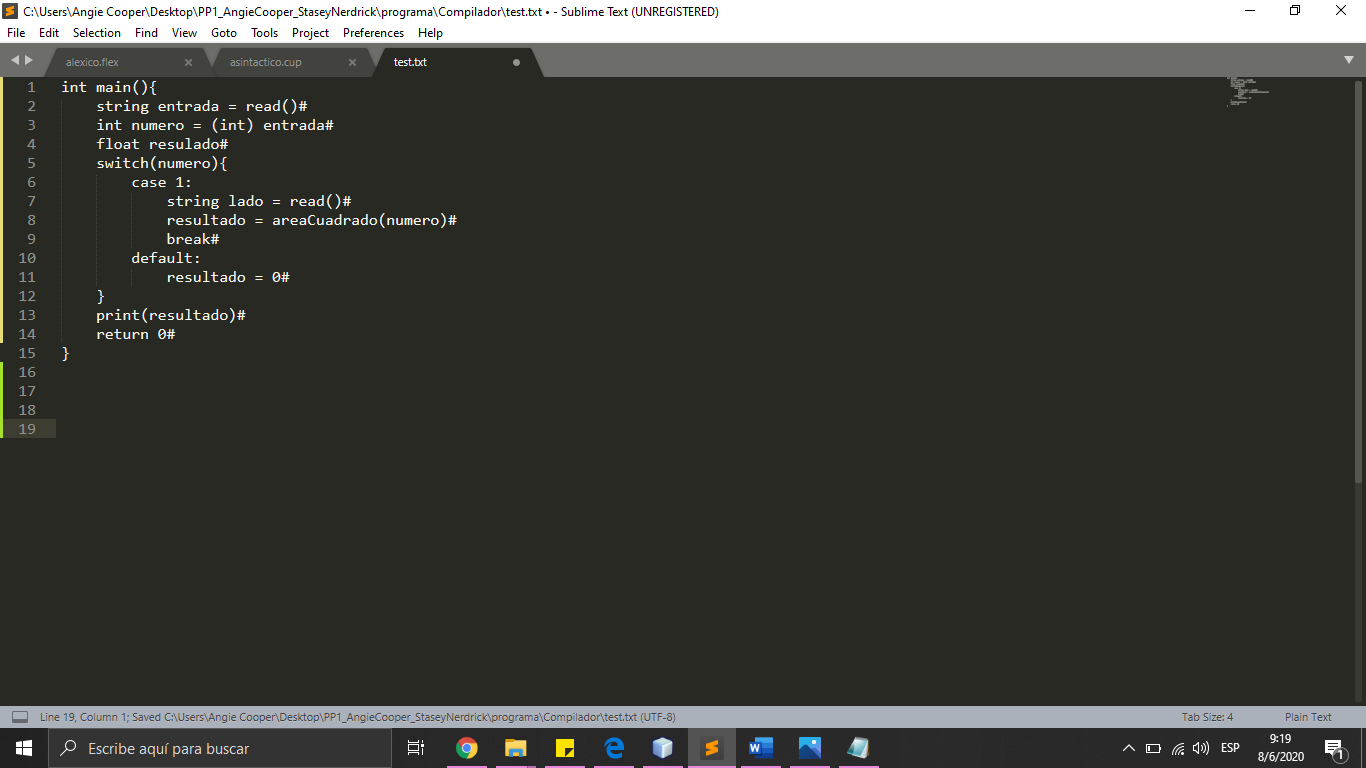
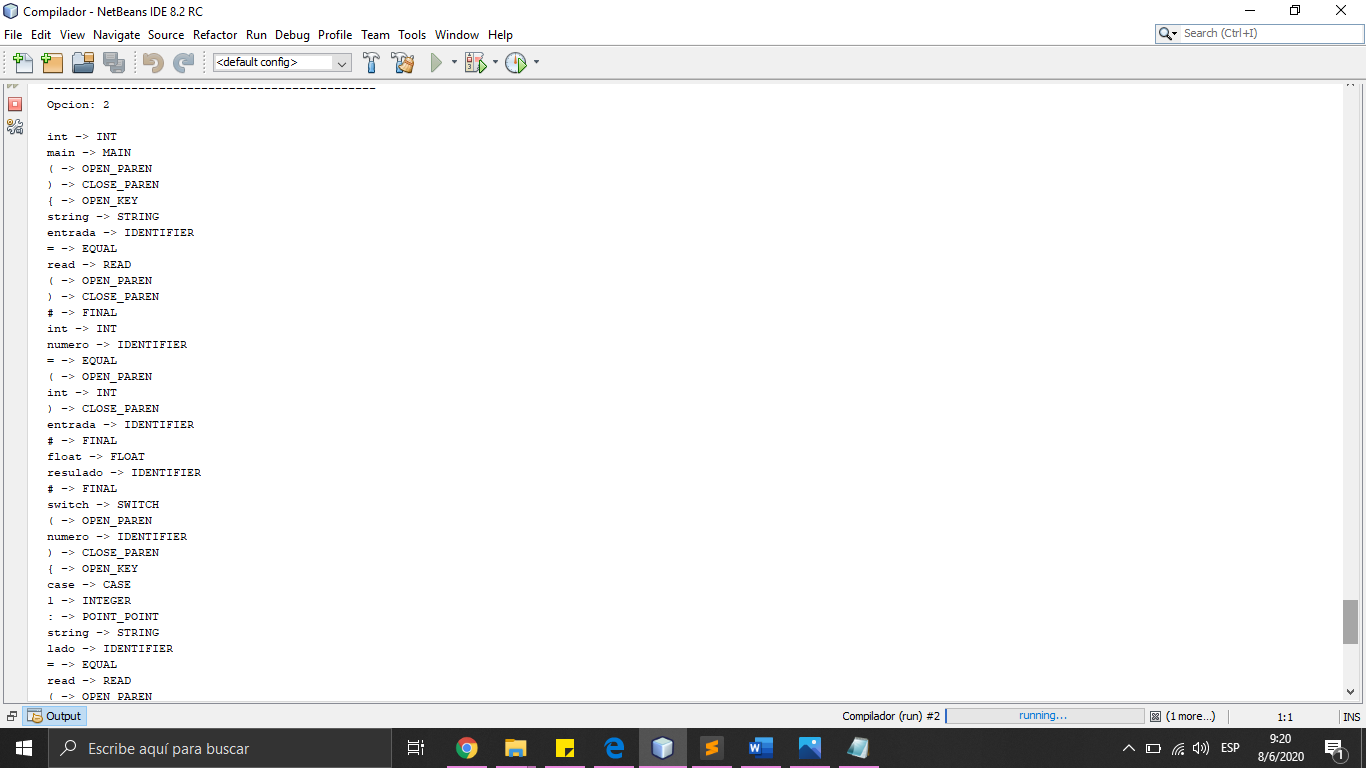


Figura 19

Al ejecutar el programa, da este resultado:



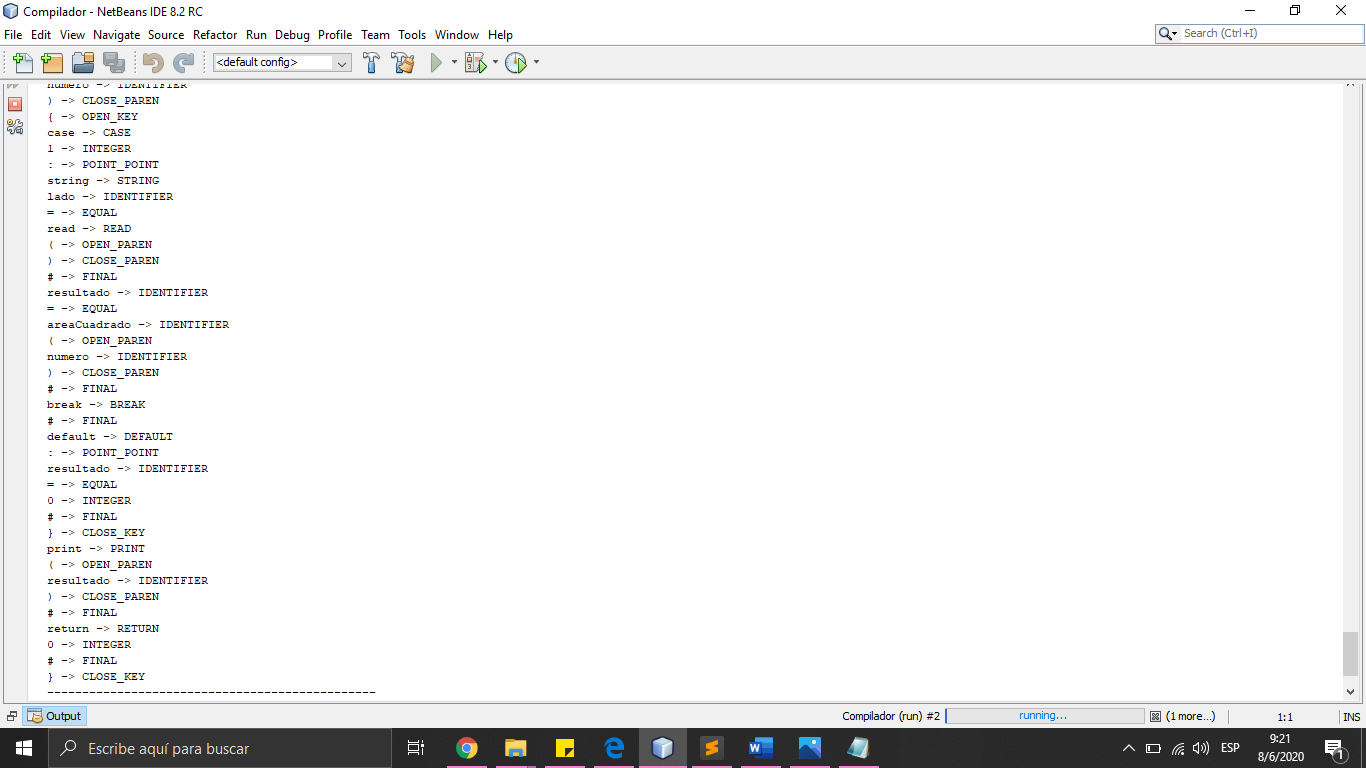
Figura 20

Figura 21

Como vemos, no se generan problemas y se lee el código correctamente.

Ejemplo cuando se detecta un error léxico.

Se tiene este código fuente, donde se pueden notar errores léxicos como:

* Se utiliza el símbolo “!”, cuando este no existe en el programa.
* Se abre una comilla (“) y no se cierra.

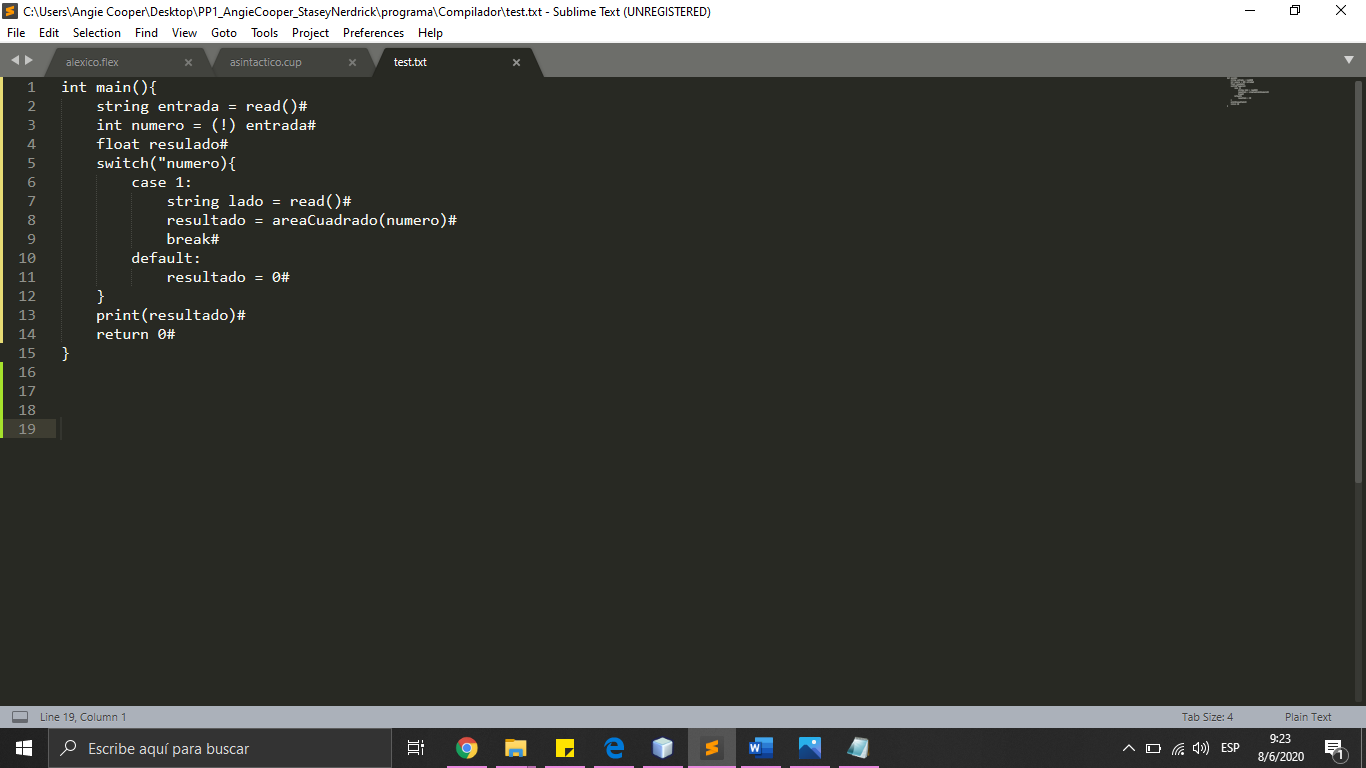


Figura 22

Debido a esto, el programa imprime que hubo errores; para estos indica el número de línea y el número de columna en los que se encuentran (algunos errores léxicos generan errores sintácticos porque se identifican mal los tokens debido al error).

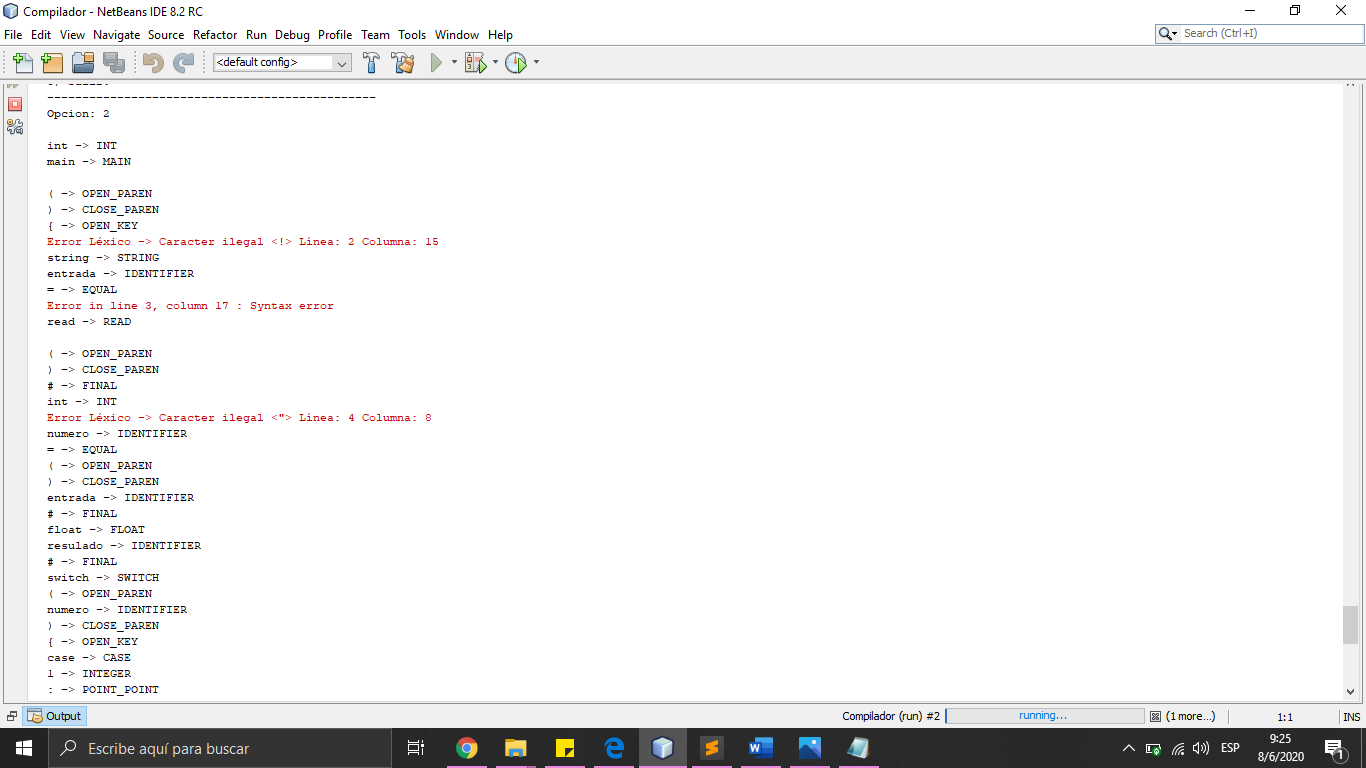


Figura 23

Ejemplo cuando se detecta un error sintáctico.

Se tiene este código fuente, donde se pueden notar errores sintácticos como:

* No se cierra el paréntesis de la función reservada read.
* No se agrega el final de línea (#) después de utilizar por segunda vez la función reservada read.

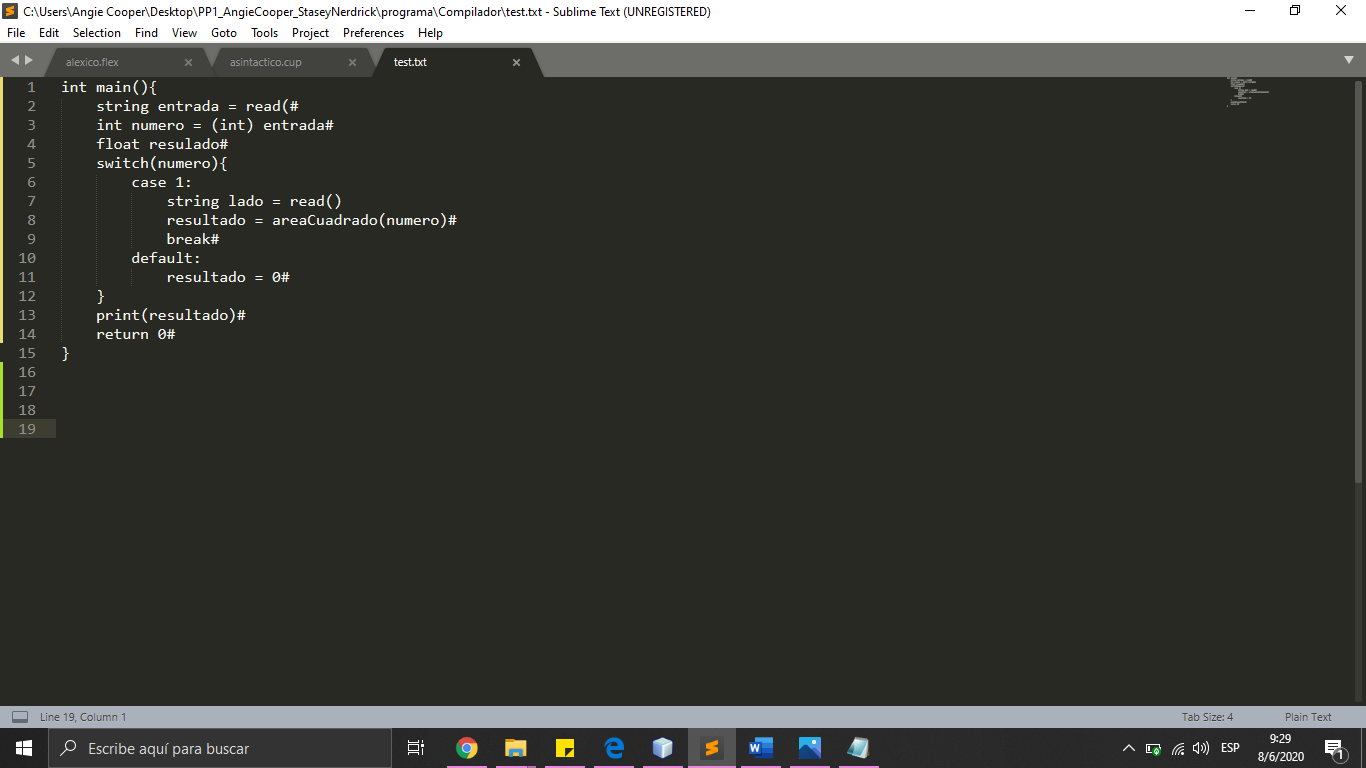


Figura 24

Debido a esto, el programa imprime que hubo errores; para estos indica el número de línea y el número de columna en los que se encuentran.

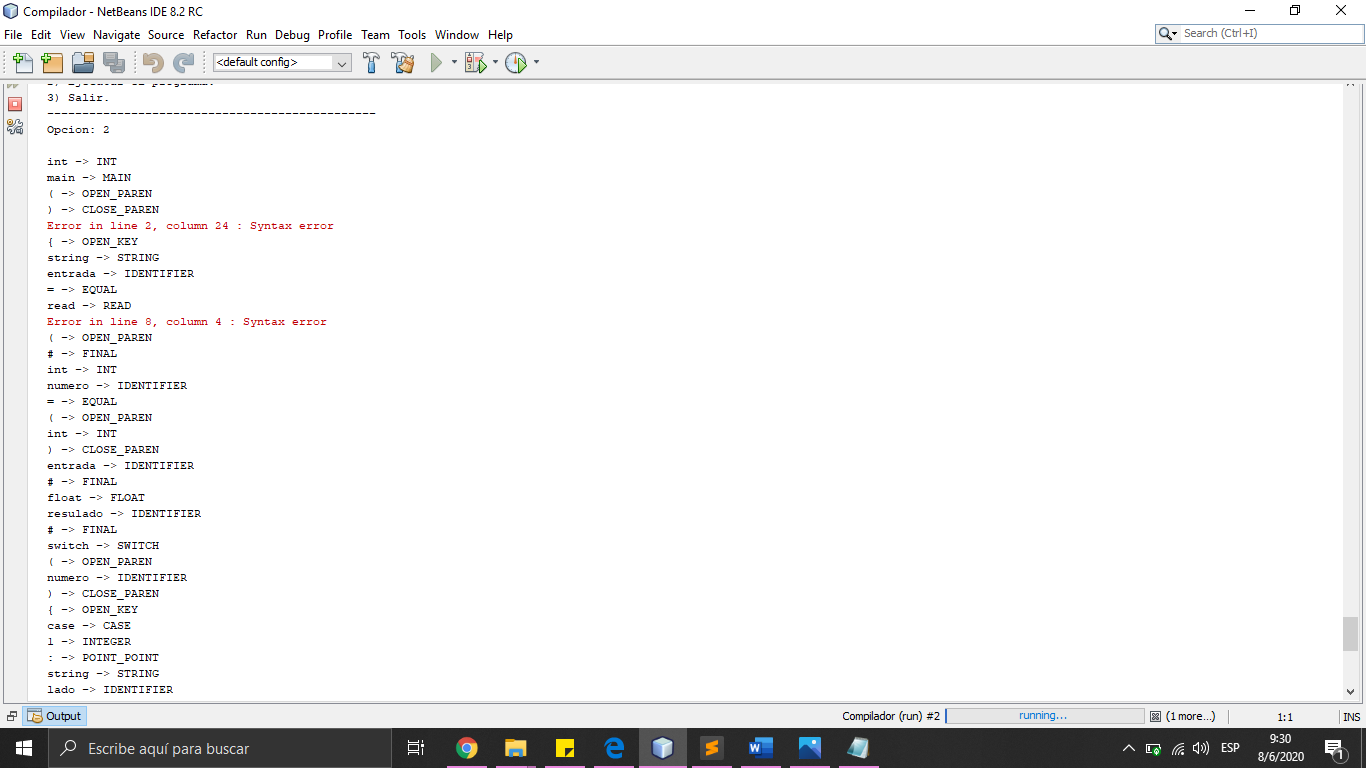


Figura 25

Ejemplo mostrando los comentarios.

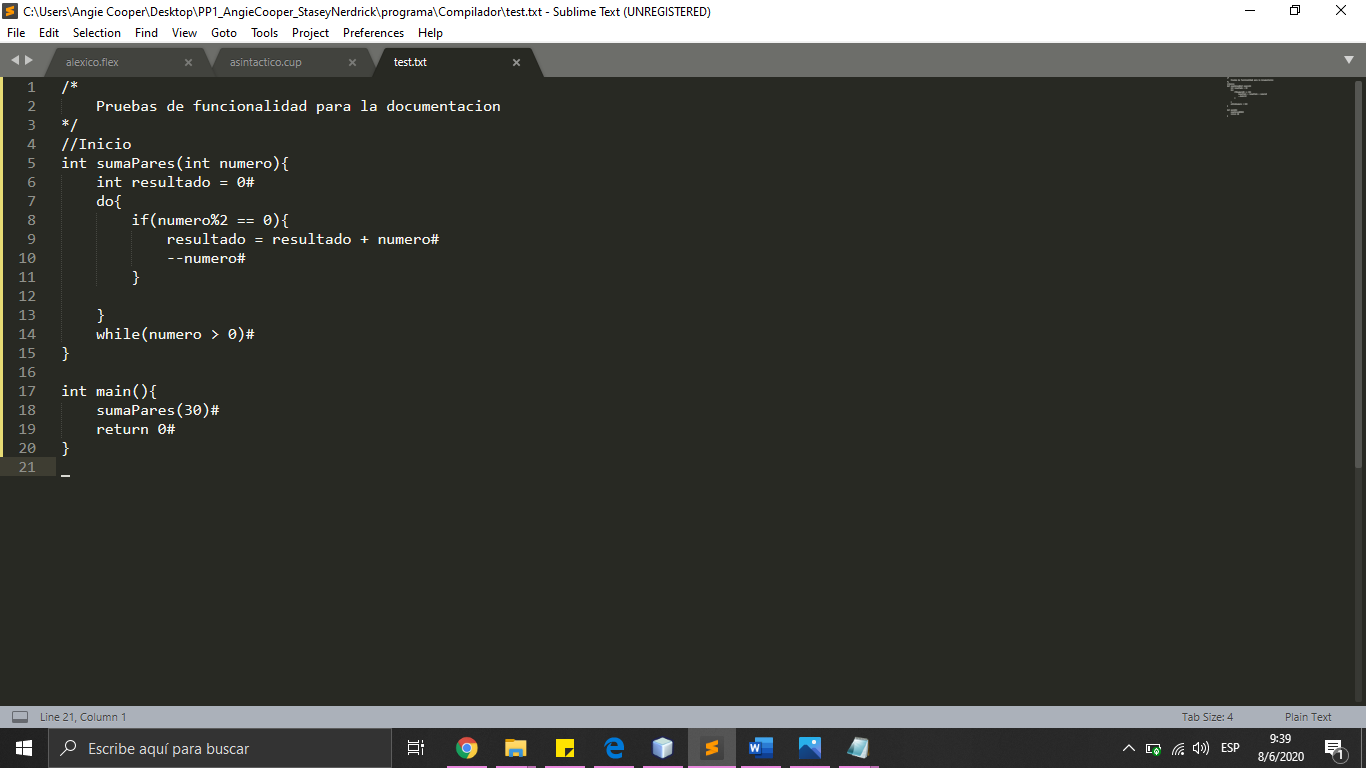


Figura 26

Al ejecutar el programa da como resultado:

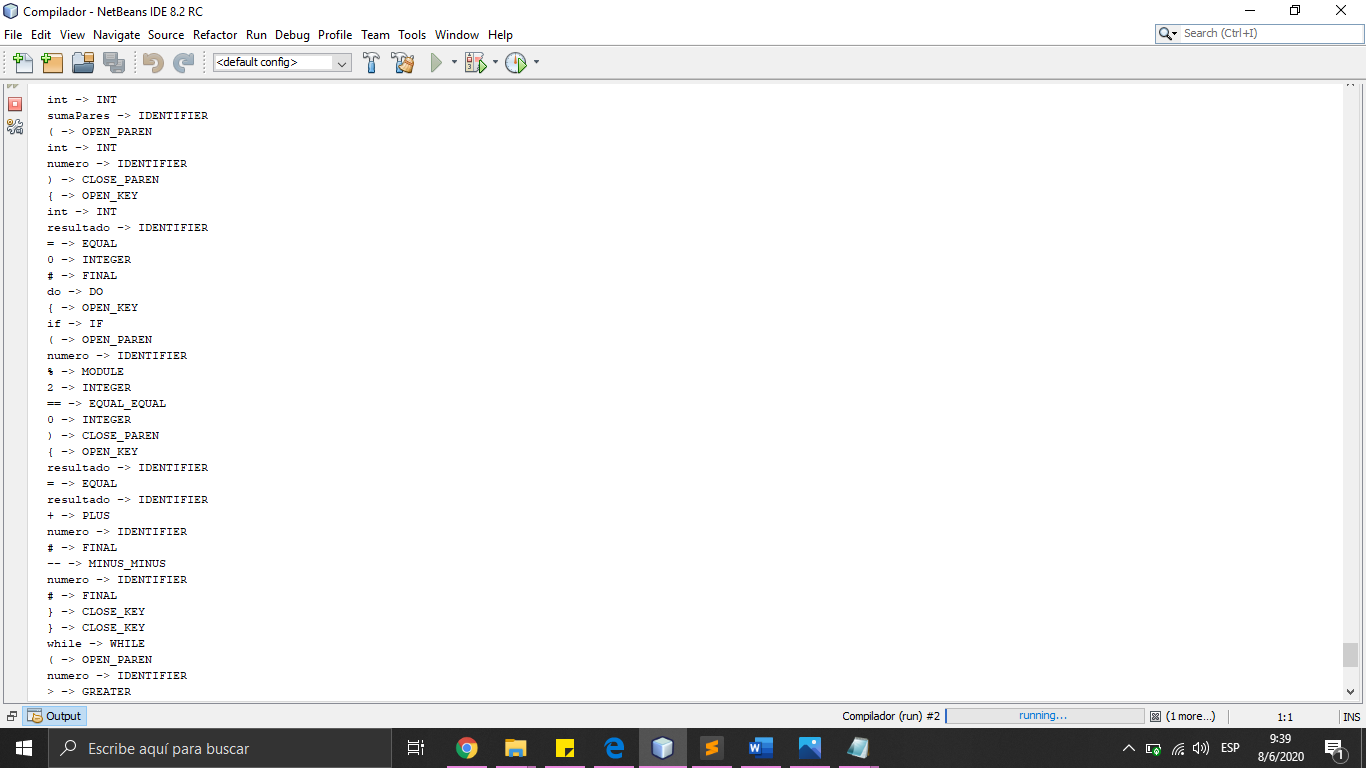


Figura 27

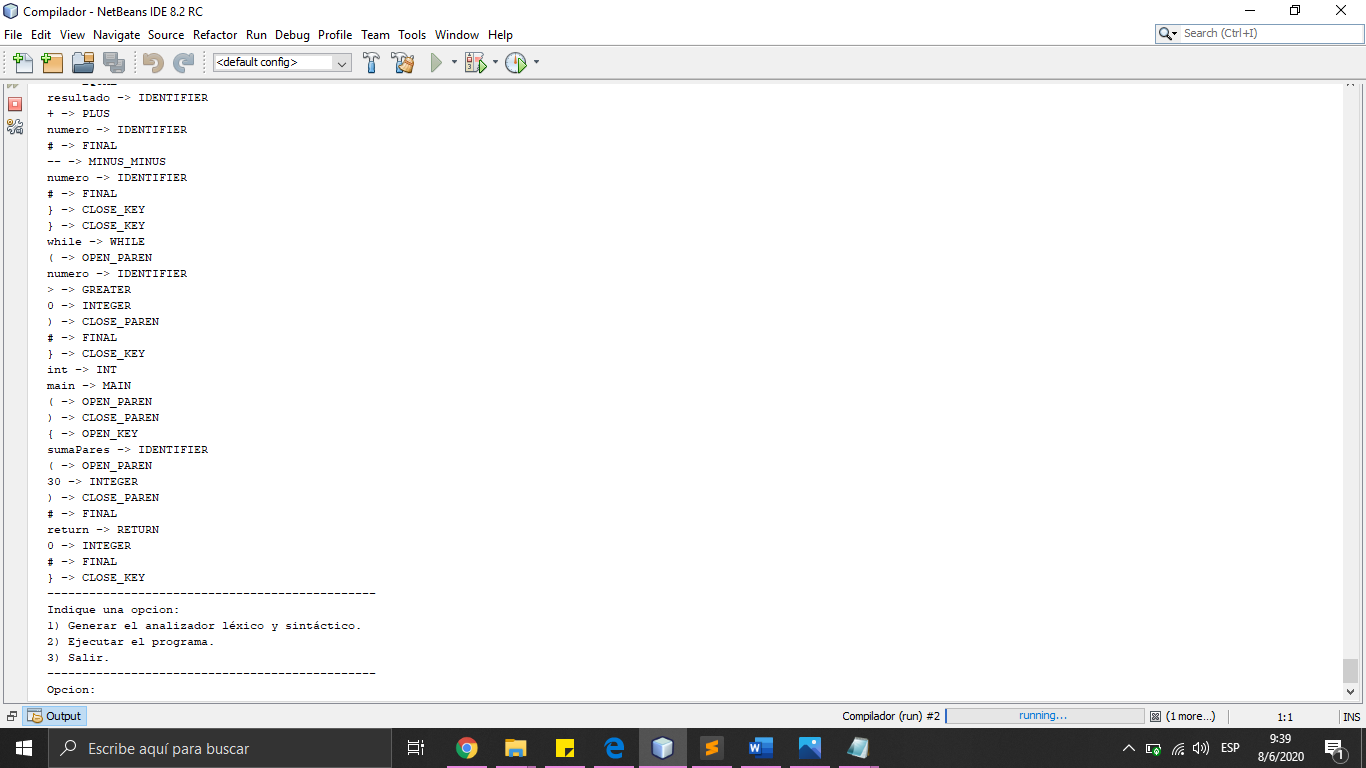


Figura 28

A partir de este ejemplo se puede evidenciar que los comentarios se ignoran, y por este motivo no se imprimen los tokens de estos.

# **Descripción del problema**

Un grupo de desarrolladores desea crear un lenguaje imperativo que le permita realizar operaciones básicas para la configuración de chips, debido a la gran demanda que presenta esta industria ultimamente, y la necesidad de que estos chips sean configurados por lenguajes cada vez más ligeros y potentes.

Debido a lo mencionado anteriormente, el grupo de desarrolladores quiere desarrollar su propio lenguaje para el funcionamiento de sistemas empotrados, por lo que este proyecto comprenderá el desarrollo de la fase de análisis léxico y análisis sintáctico para la gramática que se desarrollará también en este proyecto.

# **Diseño del programa**

Para el desarrollo de la gramática utilizada en el analizador sintáctico se tomaron como referencia las gramáticas de los lenguajes Pascal y C, ambos cumplen con la característica de ser lenguajes imperativos. Con la ayuda de ambas gramáticas nos guiamos para el orden y las caracteristicas que debían tener nuestra gramática para poder hacer uso de ella en el analizador sintáctico y que este funcionara de manera correcta.

Para los comentarios de código que permite la gramática, se utilizó como referencia un ejemplo del manual de usuario de JFlex, a continuación se muestra una imagen del código analizado para la sección de los comentarios de código y en la sección de referencias se puede encontrar el link directo al manual de usuario de JFlex de donde se extrajo el código:

Una captura de pantalla de una computadora

Descripción generada automáticamente

El desarrollo del analizador léxico y del analizador sintáctico se realizó con ayuda de tutoriales acerca del uso de JFlex y Cup y como integrar los mismos en Java para poder desarrollar de manera funcional ambos analizadores, como base para esta parte del proyecto se utilizaron principalmente los videos “Creación de un analizador sintáctico con JFlex y CUP en Java” y “Creación de un analizador léxico con JFlex” disponibles en Youtube (En la sección de referencias se encuentran los links directos para cada uno de estos videos).

Para la producción de errores nos basamos en ejemplos del uso de la producción “error” que es propia de la librería Cup, también, nos guiamos con el pdf “Error Recovery in CUP” (en la sección de referencias se puede encontrar el link directo al pdf).

## Gramática

A continuación se muestra la gramática desarrollada que se utilizó en el analizador sintáctico:

init\_program ::= create\_function main\_function

| main\_function

;

main\_function ::= INT MAIN OPEN\_PAREN CLOSE\_PAREN block

;

create\_function ::= create\_function type IDENTIFIER OPEN\_PAREN parameters CLOSE\_PAREN block

| type IDENTIFIER OPEN\_PAREN parameters CLOSE\_PAREN block

;

parameters ::= empty

| parameters COMMA type IDENTIFIER

| type IDENTIFIER

;

block ::= OPEN\_KEY sentences CLOSE\_KEY

| error CLOSE\_KEY

;

sentences ::= empty

| operation FINAL sentences

| create\_var FINAL sentences

| assing\_var FINAL sentences

| modifyArrayElement FINAL sentences

| createArray FINAL sentences

| if\_structure

| FOR OPEN\_PAREN for\_structure CLOSE\_PAREN block sentences

| RETURN operation FINAL

| BREAK FINAL

| PRINT OPEN\_PAREN operation CLOSE\_PAREN FINAL sentences

| READ OPEN\_PAREN IDENTIFIER CLOSE\_PAREN FINAL sentences

| error sentences

;

for\_structure ::= create\_var FINAL operation FINAL operation FINAL ;

if\_structure ::= IF negation OPEN\_PAREN operation CLOSE\_PAREN OPEN\_KEY sentences CLOSE\_KEY sentences

elif\_structure

else\_structure

;

else\_structure ::= empty

|ELSE OPEN\_KEY sentences CLOSE\_KEY sentences

;

elif\_structure ::= empty

| ELIF negation OPEN\_PAREN operation CLOSE\_PAREN OPEN\_KEY sentences CLOSE\_KEY sentences

| elif\_structure ELIF negation OPEN\_PAREN operation CLOSE\_PAREN OPEN\_KEY sentences CLOSE\_KEY sentences

;

operation ::= operation relational\_operator additive\_operation

| additive\_operation

;

relational\_operator ::= GREATER

| GREATER\_EQUAL

| MINOR

| MINOR\_EQUAL

| EQUAL\_EQUAL

| DIFFERENT

;

additive\_operation ::= additive\_operation additive\_operator multiplicative\_operation

| multiplicative\_operation

;

additive\_operator ::= MINUS

| PLUS

| OR

;

multiplicative\_operation ::= multiplicative\_operation multiplicative\_operator unary\_operation

| unary\_operation

;

multiplicative\_operator ::= DIVIDE

| MULTI

| POWER

| MODULE

| AND

;

unary\_operation ::= unary\_operator unary\_operation

| basic\_operation

;

unary\_operator ::= MINUS\_MINUS

| PLUS\_PLUS

| MINUS

;

basic\_operation ::= INTEGER

| DECIMAL

| CHARCHAIN

| CHARACTER

| IDENTIFIER

| TRUE

| FALSE

| negation OPEN\_PAREN operation CLOSE\_PAREN

| IDENTIFIER OPEN\_BRACKET INTEGER CLOSE\_BRACKET

| OPEN\_BRACKET arrayList CLOSE\_BRACKET

| call\_function

| NULL

;

parameter\_list ::= parameter\_list COMMA operation

| operation

;

call\_function ::= IDENTIFIER OPEN\_PAREN parameter\_list CLOSE\_PAREN

| IDENTIFIER OPEN\_PAREN CLOSE\_PAREN

;

arrayList ::= empty

| operation

| operation COMMA arrayList

;

empty ::=

;

modifyArrayElement ::= IDENTIFIER OPEN\_BRACKET INTEGER CLOSE\_BRACKET EQUAL operation ;

createArray ::= type IDENTIFIER OPEN\_BRACKET INTEGER CLOSE\_BRACKET

| type IDENTIFIER OPEN\_BRACKET CLOSE\_BRACKET EQUAL OPEN\_BRACKET arrayList CLOSE\_BRACKET

;

create\_var ::= type IDENTIFIER EQUAL operation

| type IDENTIFIER

;

assing\_var ::= IDENTIFIER EQUAL operation;

negation ::= empty

|NOT

;

type ::= INT

| FLOAT

| CHAR

| STRING

| BOOL

;

# **Librerías usadas**

## JFlex

Por medio de la librería JFlex de Java se desarrolló el analizador léxico. Esta librería nos permitió desarrollar el scanner que se solicitó en el proyecto para el reconocimiento de los tokens en el código fuente.

## Cup

Por medio de la librería Cup de Java se desarrolló el analizador sintáctico. Esta librería nos permitió desarrollar el parser solicitado en el proyecto para la comprobación sintáctica del código fuente según la gramática desarrollada.

# **Análisis de resultados**

## Objetivos alcanzados

Para este proyecto logramos desarrollar un analizador léxico funcional con ayuda de la herramienta JFlex de Java. También, logramos desarrollar el analizador sintáctico con ayuda de la herramiento Cup de java, este analizador utiliza la gramática que desarrollamos para este proyecto donde utilizamos como referencia la gramática de otros lenguajes imperativos. En ambos analizadores incluimos la recuperación y el reporte de errores del código fuente que se está analizando.

# **Bitácora**

<https://github.com/lisa3001/PP1---Compiladores-E-Int-rpretes>

# **Referencias**

Arturo De Casso. (18 de abril del 2014). *Creación de un analizador sintáctico con JFlex y CUP en Java* [Archivo de video]. Youtube. <https://www.youtube.com/watch?v=XQHivIfKvMk&feature=youtu.be>

Arturo De Casso. (16 de abril del 2014). *Creación de un analizador léxico con JFlex* [Archivo de video]. Youtube. <https://www.youtube.com/watch?v=mHFBLm4GXnk&t=1s>

*Error Recovery In CUP*. (s.f). <https://www.cs.auckland.ac.nz/courses/compsci330s1c/lectures/330ChaptersPDF/Chapt6.pdf>

Klein, G., Rowe, S., y Décamps, R. (28 de febrero del 2018). *JFlex User’s Manual.*JFlex. <https://jflex.de/manual.html>

Walter Rosero. (9 de mayo del 2018). *Descarga e Instala NetBeans IDE 8.2 y Java Development Kit para Windows 7/8/8.1/10* [Archivo de video]*.* <https://www.youtube.com/watch?v=UogvQ_kBQzc>

Edgar Garcia. (14 de mayo del 2016). *Como agregar una librería a NetBeans* [Archivo de video]. <https://www.youtube.com/watch?v=zkCoQhQBIFc>