Manual de Usuario de Comilador

Lenguajes y Autómatas II Autor: Hernández Osorio José Daniel

Para el uso del compilador se tiene primero que conocer la tabla de tokens con el cual fue diseñado el compilador, esta tabla de token se localiza en la última parte del manual del compilador, en ella se puede observar el uso de las palabras reservadas del lenguaje, así como las estructuras de los tokens de mismo. Así como usuario podrá realizar, programas que ejecuten las operaciones de suma, resta, división y multiplicación, y observando las fases de un compilador antes de la ejecute un programa.

En la figura 1, se muestra el inicio del compilador el cual es con una, de las ventanas que simula ser una ventana de carga del programa, el motivo del funcionamiento de la venta es el de solo dar una presentación agradaba del programa, seguido de esta ventana se muestra la ventana principal del programa la cual se muestra en la figura 2.



Figura 1. Ventana de carga.

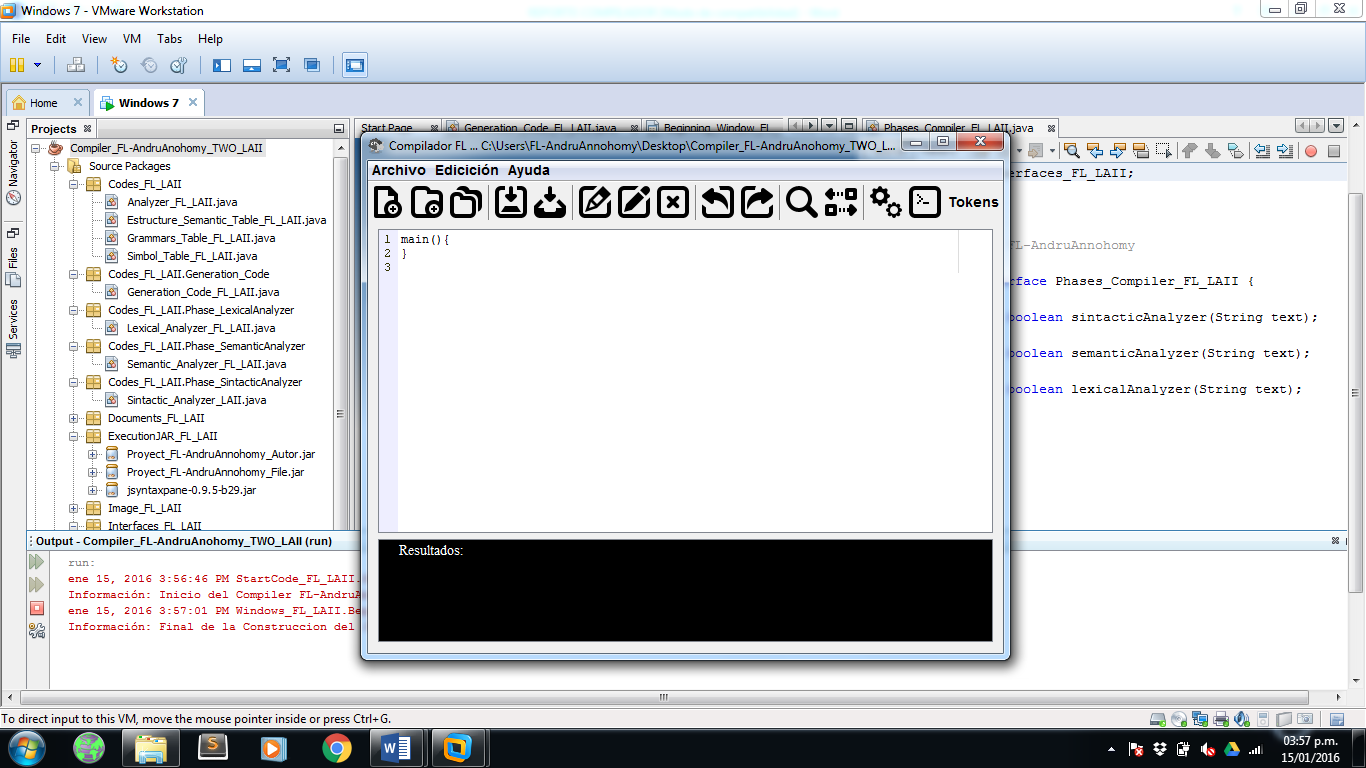
En la figura 2, se observan distintas opciones las cuales contemplan, diversas tareas del compilador, las cuales consisten en lo siguiente:

1. Botón de Nuevo.
2. Botón de Nuevo Proyecto.
3. Botón Abrir.
4. Botón de Guardar.
5. Botón de Guardar Como.
6. Botón Copiar.
7. Botón Pegar.
8. Botón Cortar.
9. Botón de Deshacer
10. Botón de Rehacer.
11. Botón de Buscar.
12. Botón de Remplazar.
13. Botón Compilar.
14. Botón Ejecutar.
15. Botón de Tokens.
16. Área de Edición de Programas.
17. Área de Resultados.

8

10

12



9

7

6

5

4

3

2

1

11

13

15

14

17

16

Figura 2. Ventana de principal del compilador.

Botón Nuevo: consiste en la creación de un nuevo archivo con extensión .fl para la creación de un programa, la función principal de este botón es la de borrar los datos y guardarlos en el archivo anterior y generar el nuevo, dar click en el botón se podrá observar que se borra todo, pero de deja un método principal llamado main como el inicio del nuevo programa, como se muestra en la figura 2. El botón nuevo proyecto tiene la tarea de crear una nueva carpeta donde se guardaran los archivos del programa.

El botón abrir es aquel que permitirá la apertura o lectura de un archivo ya creado previamente en el programa, así que al dar click el botón él nos mostrar la ventana de la figura 3, en la cual podremos elegir que archivo se abrirá para su edición o ejecución del mismo, este a su vez dando como resultado la lectura del archivo y mostrarnos los datos como se muestra en la figura 4.

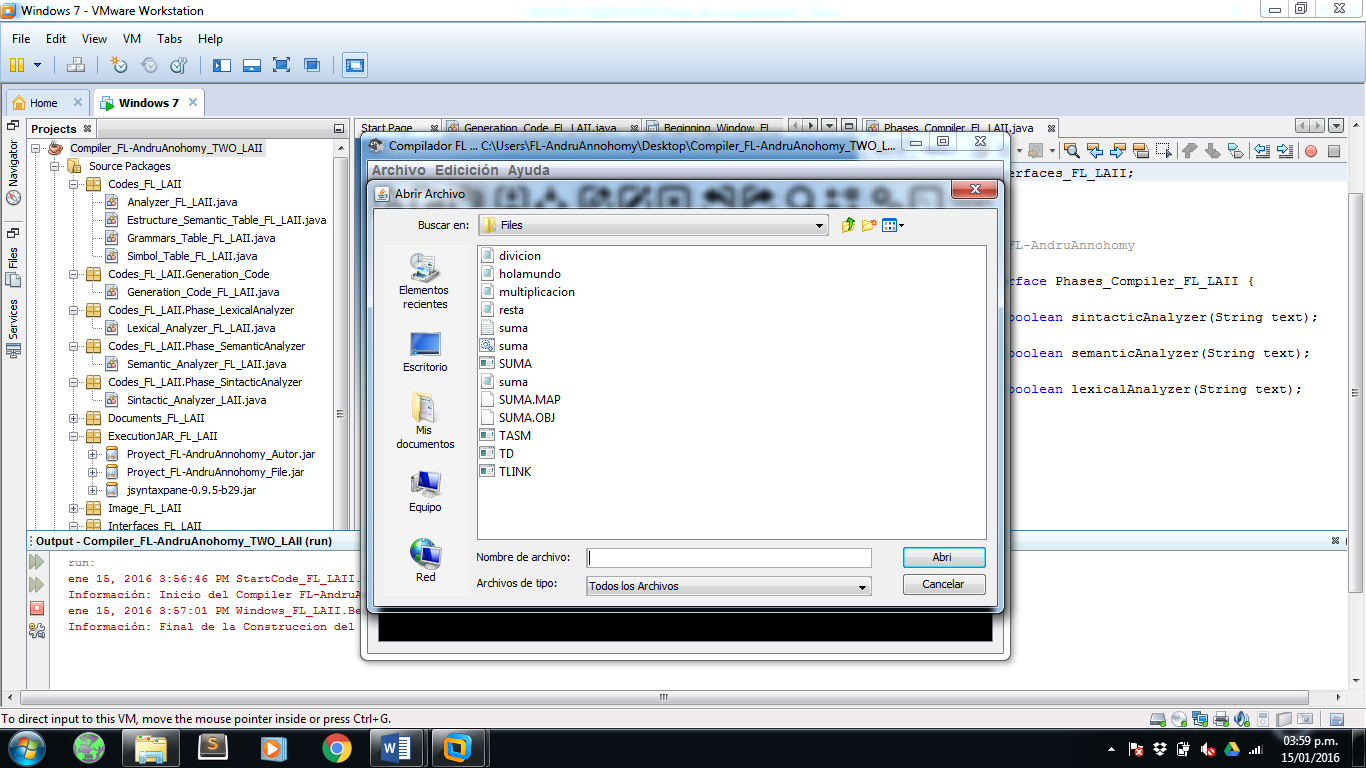


Figura 3. Abrir un archivo.

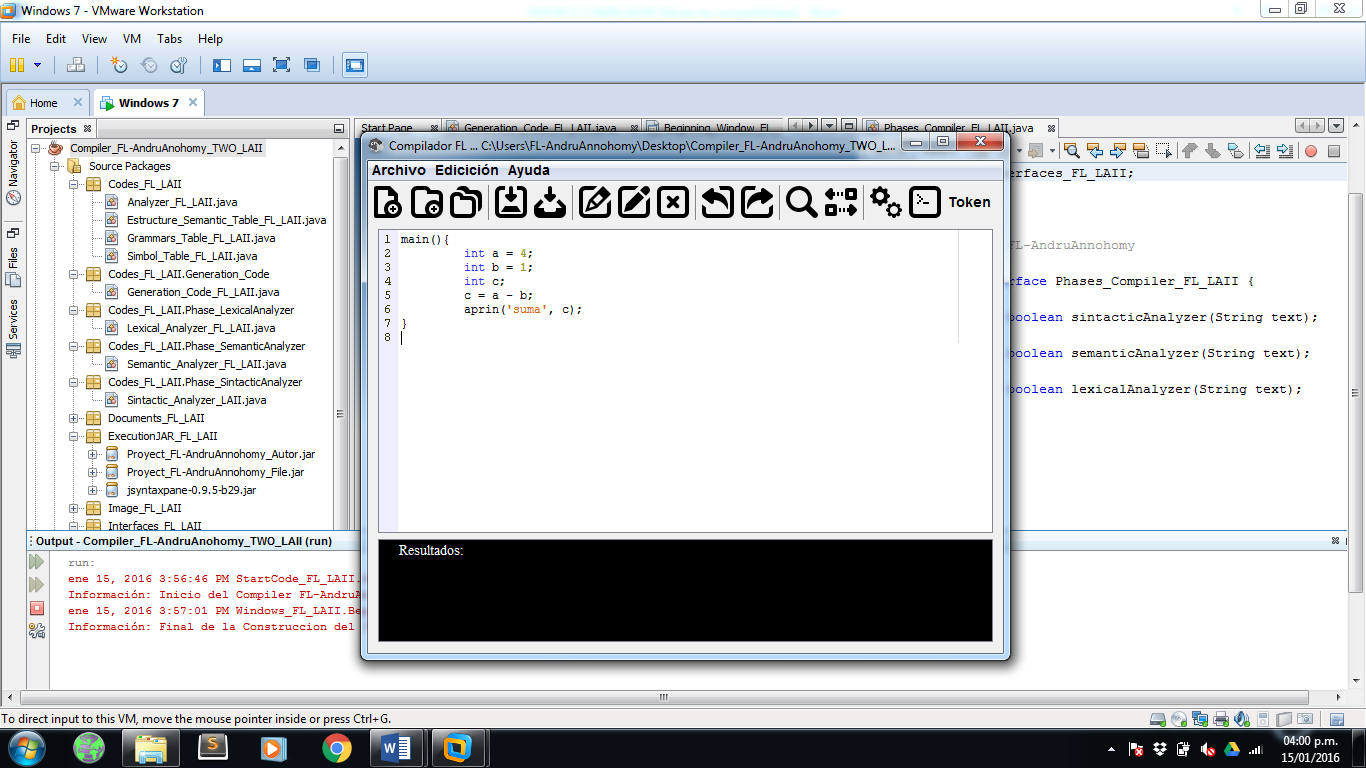


Figura 4. Ventana principal con programa abierto.

El botón guardar y guardar como, nos sirve para el guardado de los datos ingresados en el área principal o en su momento para guardar las modificaciones del mismo archivo en el que se esté trabajando. En caso de ser un nuevo archivo siempre se nos pedirá que guarde el archivo antes de realizar cualquier actividad, en esta acción se está mostrando la ventana de la figura 5, donde se tendrá que ingresar el nombre del archivo.

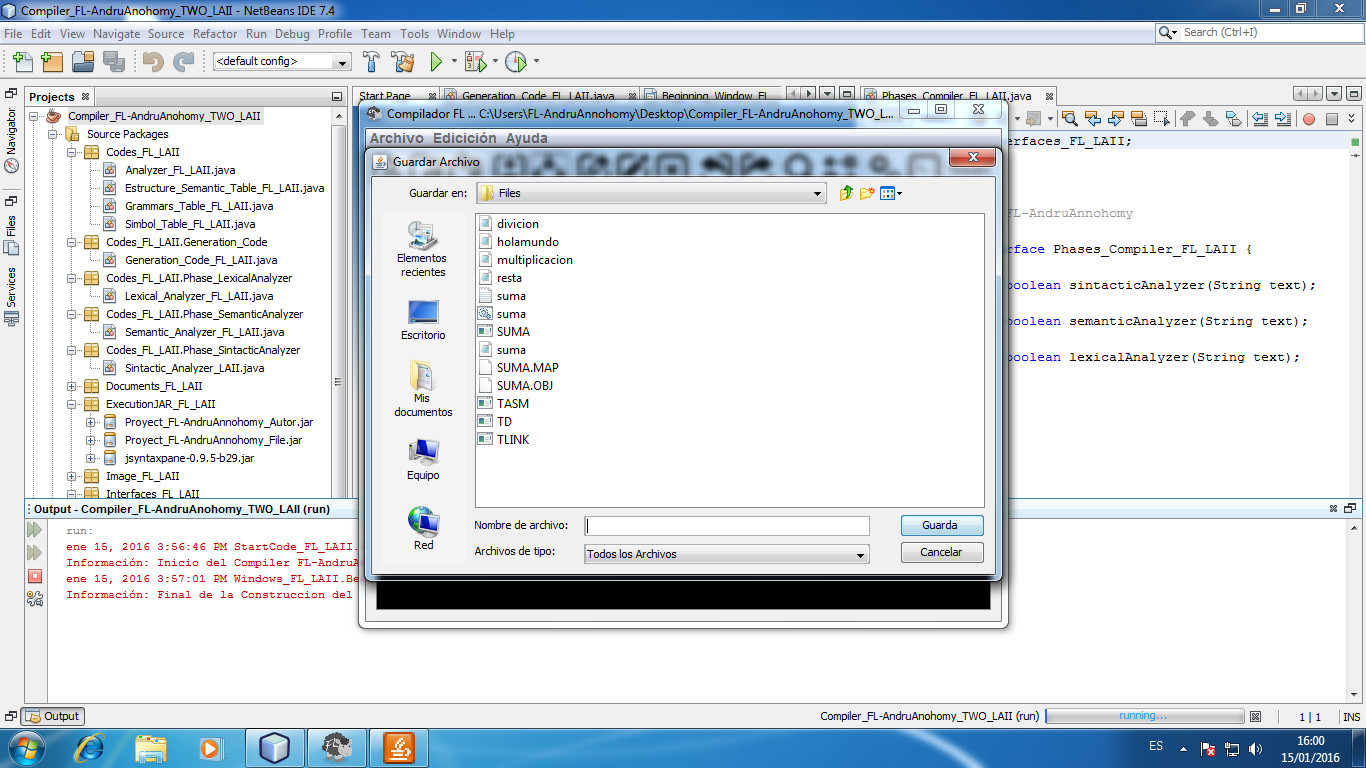


Figura 5. Guardar un archivo.

El botón copiar, pegar y cortar como su nombre de cada uno lo dice tiene la tarea de copiar, pegar y cortar texto dentro del área de edición del compilador. Estas funciones también funcionan dentro del compilador con el meta-comando que corresponde a cada función. También contamos con los botones de deshacer y rehacer los cuales se encargan de recuperar los datos borrados si es que no se quería realizar esta función así como el de deshacer una acción realizada, estos dos botones también cuenta con el meta-comados correspondiente a cada acción.

Botón buscar, nos permite buscar un carácter dentro del texto que se encuentra en el área de edición con el motivo de poder modificarlo o eliminarlos con más rapidez, esta función se observada en la figura 6, donde se puede ver que se despliega un ventana emergente del área de edición donde el usuario podrá ingresar los datos que en un determinado momento este buscado.

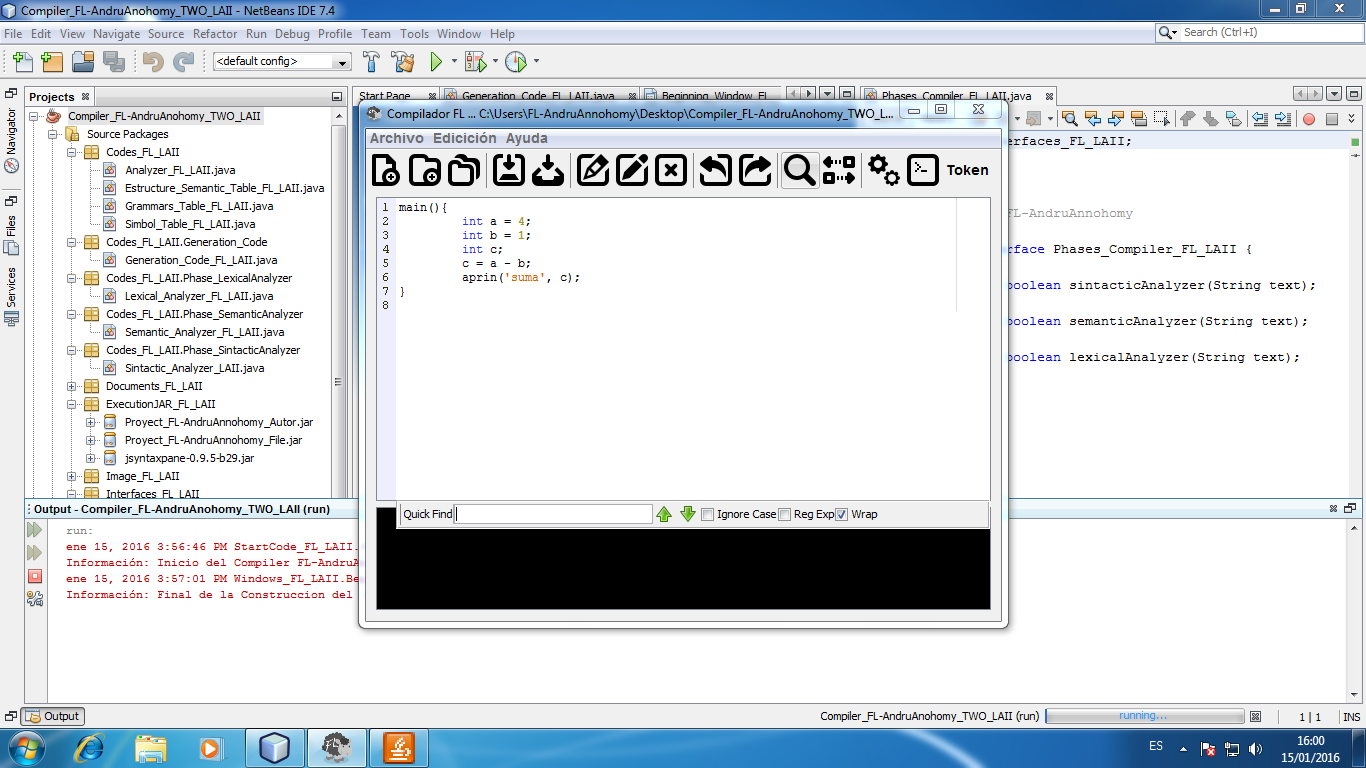


Figura 6. Ventana de búsqueda de carácter.

Botón de remplazar, como su nombre lo dice es para remplazar caracteres o cadenas de caracteres los cuales se sustituyen por unos nuevos. Esta acción se puede observar en la figura 7, donde se muestra na ventana emergente la cual es para la realización de la sustitución de una cadena o carácter por otro, también la figura se muestra que esta acción no solo puede sustituir una sola cadena, sino todas las cadenas que sean iguales.

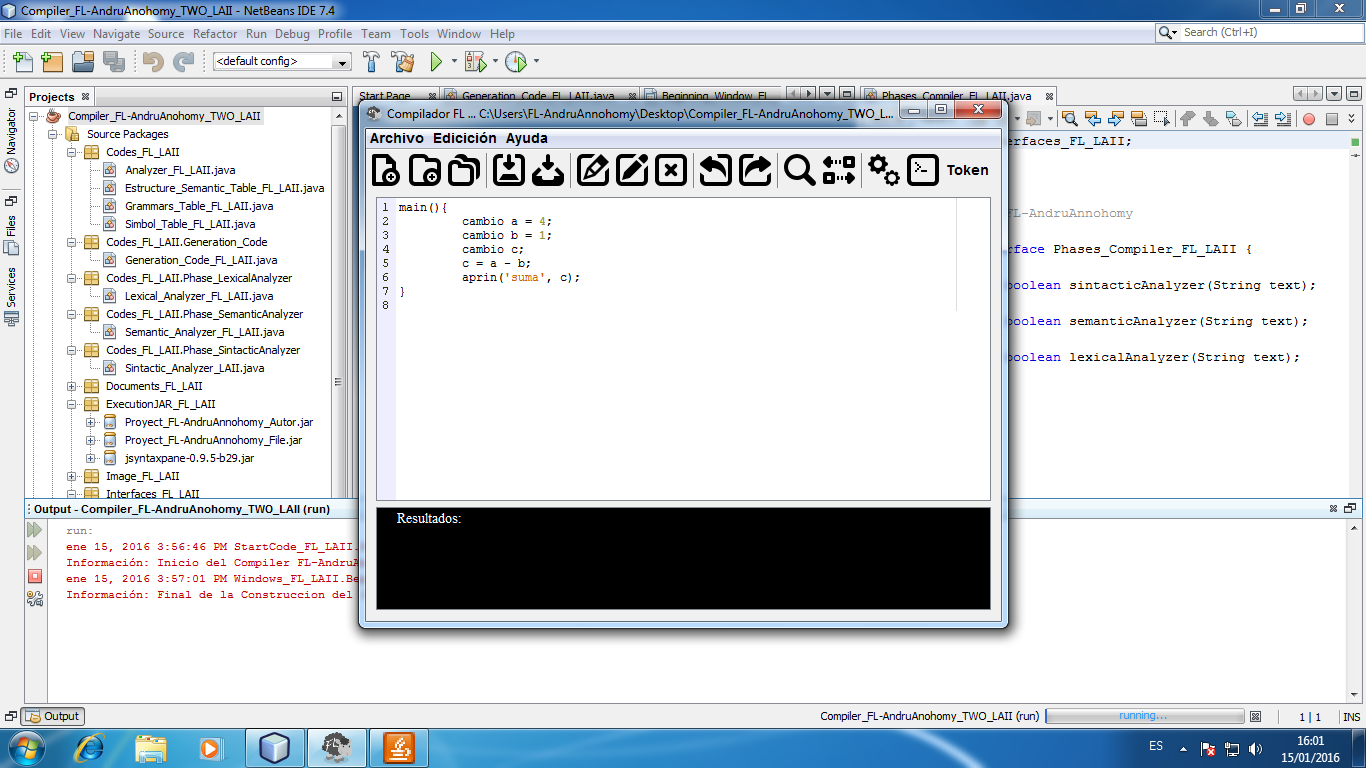
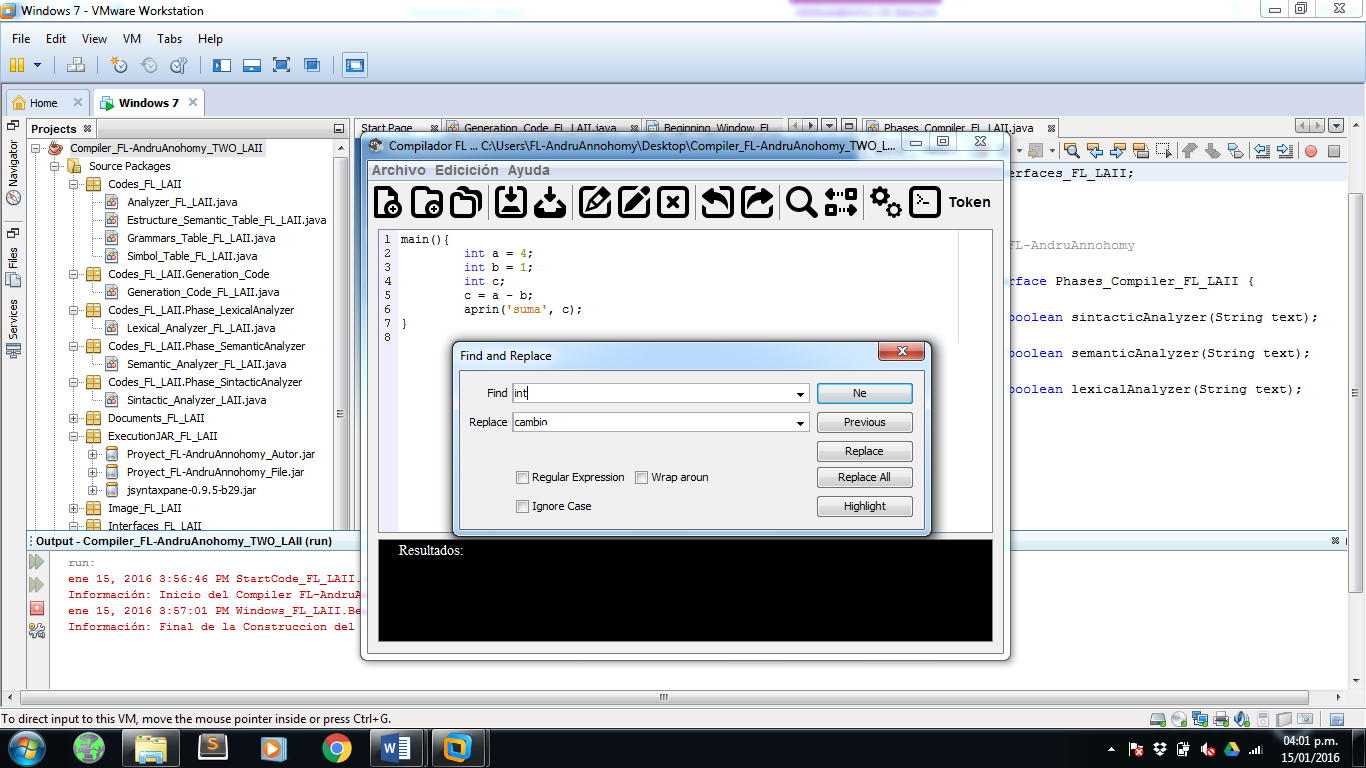


Figura 7. Acción de remplazar cadena o carácter.

El botón de compilar, es una de las funciones principales del proyecto, ya que en esta función se realizan distintos tipos de eventos o análisis del texto en el compilador, con la finalidad de analizar los caracteres y las estructuras gramaticales de los datos ingresados en el área de edición con el fin de evitar errores en al momento de la ejecución del programa, en estas fases del compilador existen diversas funciones o tareas que le permiten a cada análisis verificar y alertar en caso de error para que no se ejecute el programa y nos dé como resultado un error mayor, los análisis que se realizan al dar click en el botón son los siguientes:

* Análisis léxico, de color verde.
* Análisis sintáctico, de color azul.
* Análisis semántico, de color naranja.

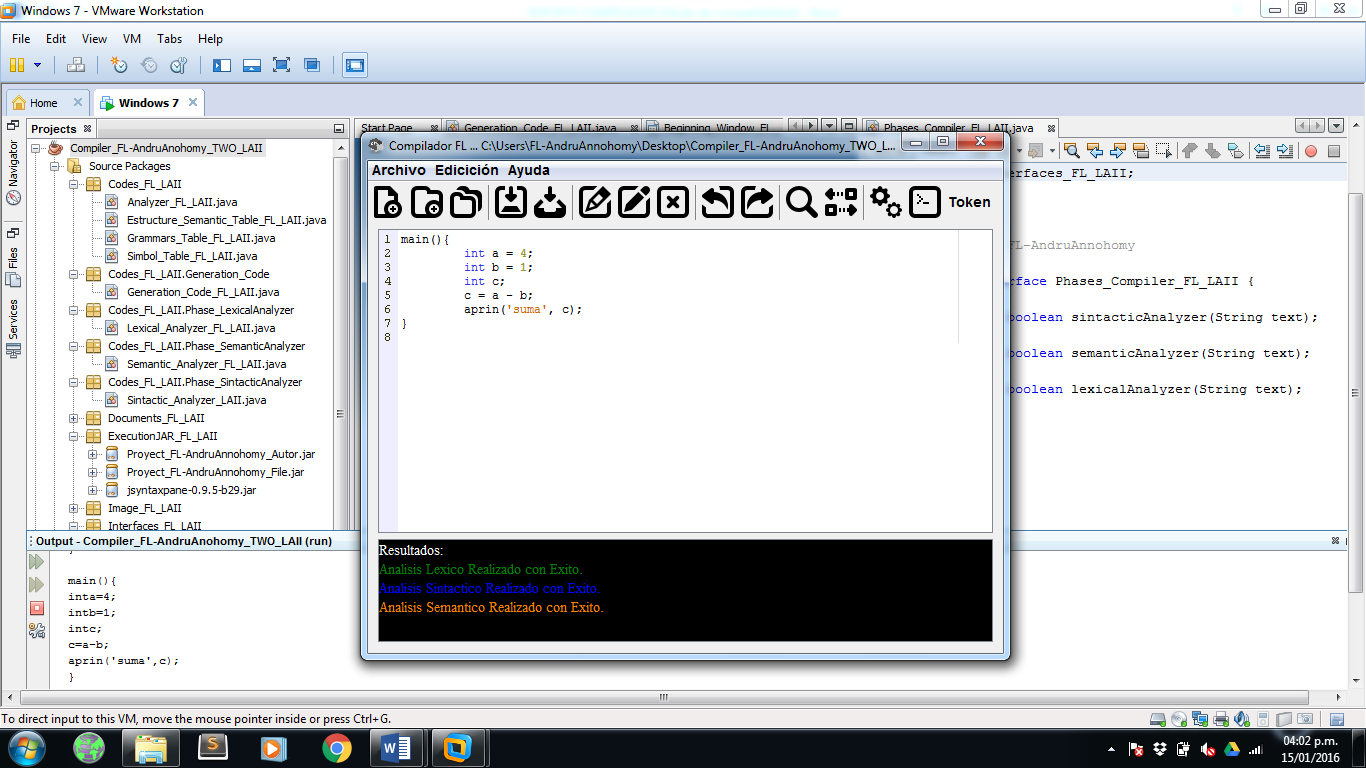


Figura 8. Resultados del botón compilar.

El botón token, es la función que nos permite observar los resultados del análisis léxico y sintáctico los cuales nos dan como resultado la identificación de tokens, palabras reservadas y gramáticas del lenguaje. Esta función nos permite verificar que se haya realizado con éxito las 2 etapas del compilador y que los datos que se hayan arrojado sean los ideales para pasar al análisis semántico. Este tipo de función se observa en la figura 9, la cual nos muestra una lista de todo lo mencionado antes.

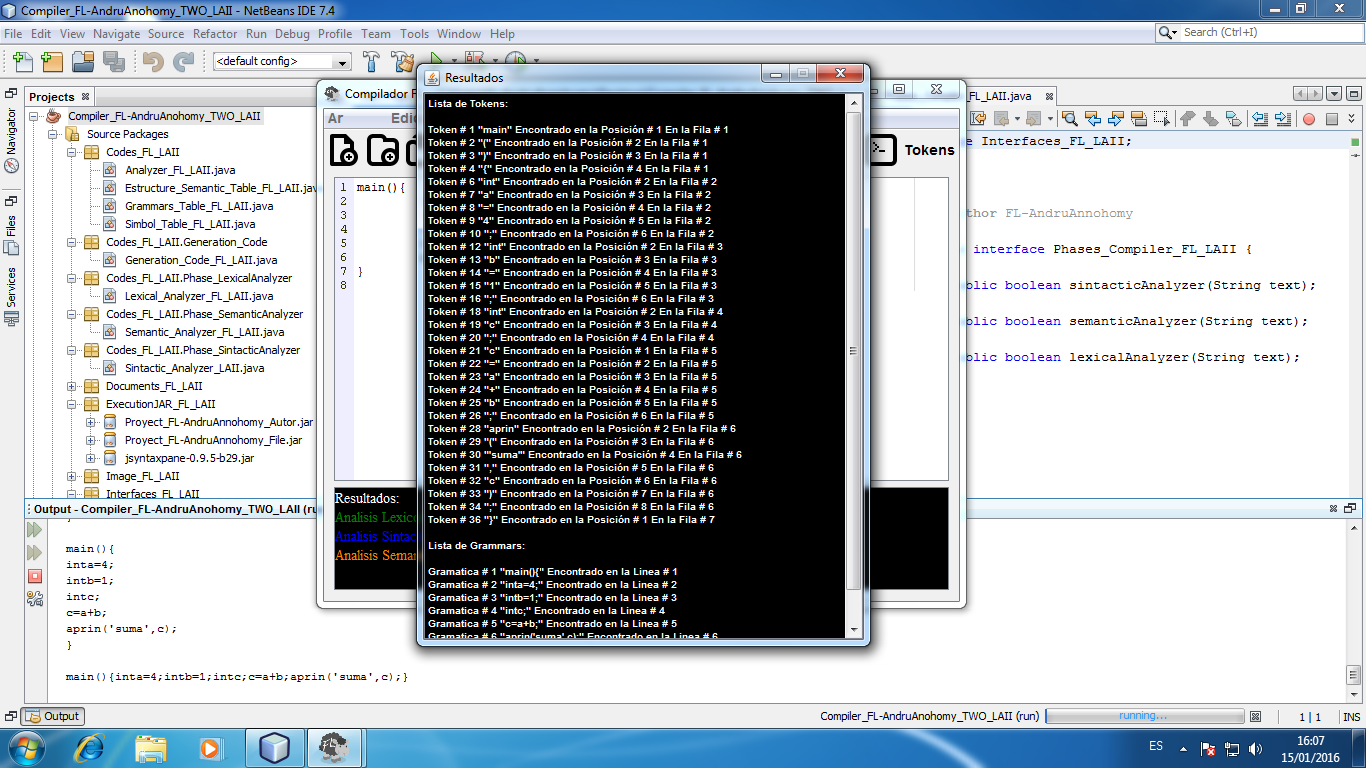


Figura 9. Resultados del análisis léxico y sintáctico.

El botón de ejecutar, se realiza la ejecución del programa con la respectiva operación, esta operación de ejecución pasa por procesos, primero pasa por el generador de código el cual interpreta el código de área de edición a un lenguaje ensamblador, después el proceso siguiente es el de generar el código objeto, el cual se realiza por medio del ligado y el ensamblado y por último la ejecución del programa ya en tipo .exe. Estos tipos de archivos que se generan en el proceso de la ejecución del programa, se observan en la figura 11 y el procesos de la ejecución del programa se observa el a figura 10.

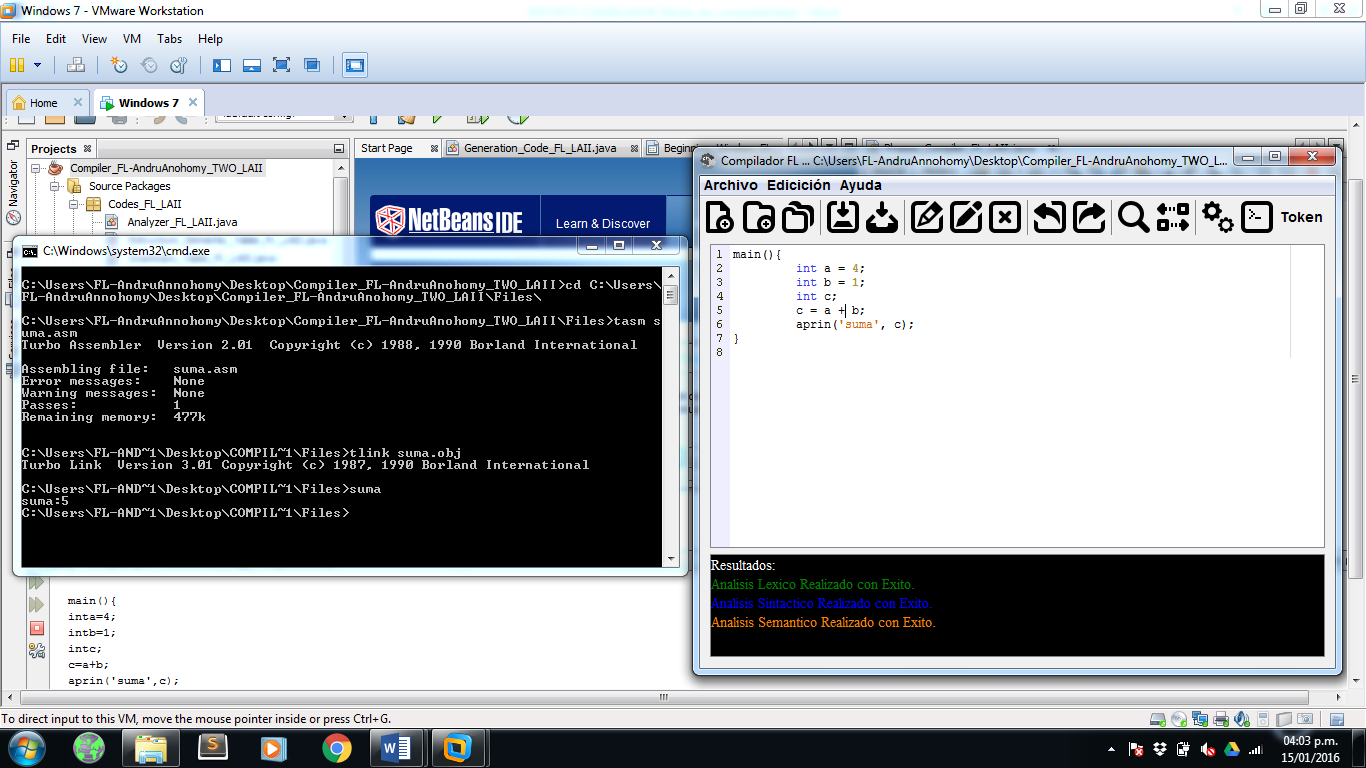


Figura 10. Ejecución de un programa.

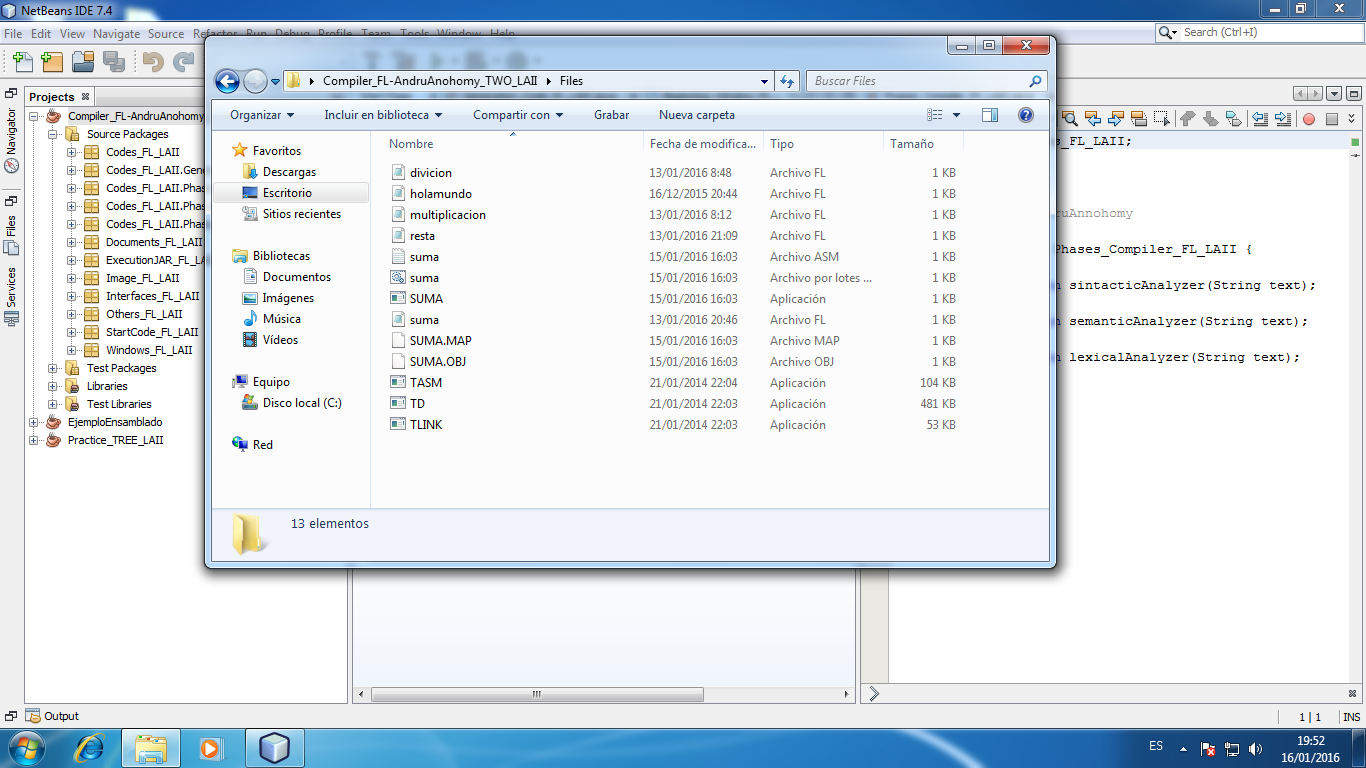


Figura 11. Archivos creados al ejecutar el programa.

En la parte superior del menú de iconos existe un menú, el cual es muy similar al de los demás programas donde se repinen las funciones del programa, como se muestra en las figuras 12, 13 y 14. Estas funciones son similares a las anteriores y solo consta con tres funciones adicionales las cuales son: salir, que es terminar la ejecución del compilador, autor, muestra el nombre del autor y ayuda que muestra un archivo pdf con un manual el cual sirve de apoyo para el uso del programa.

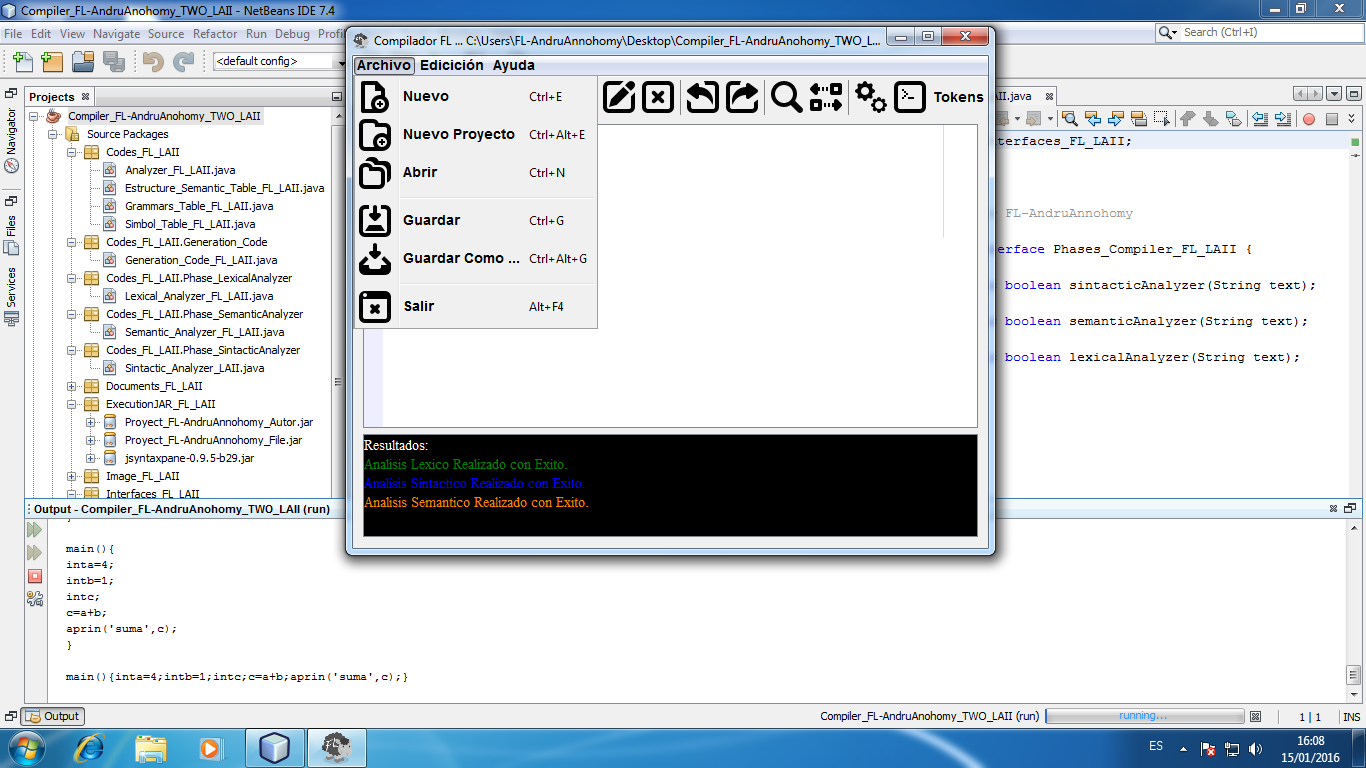


Figura 12. Menú archivo.

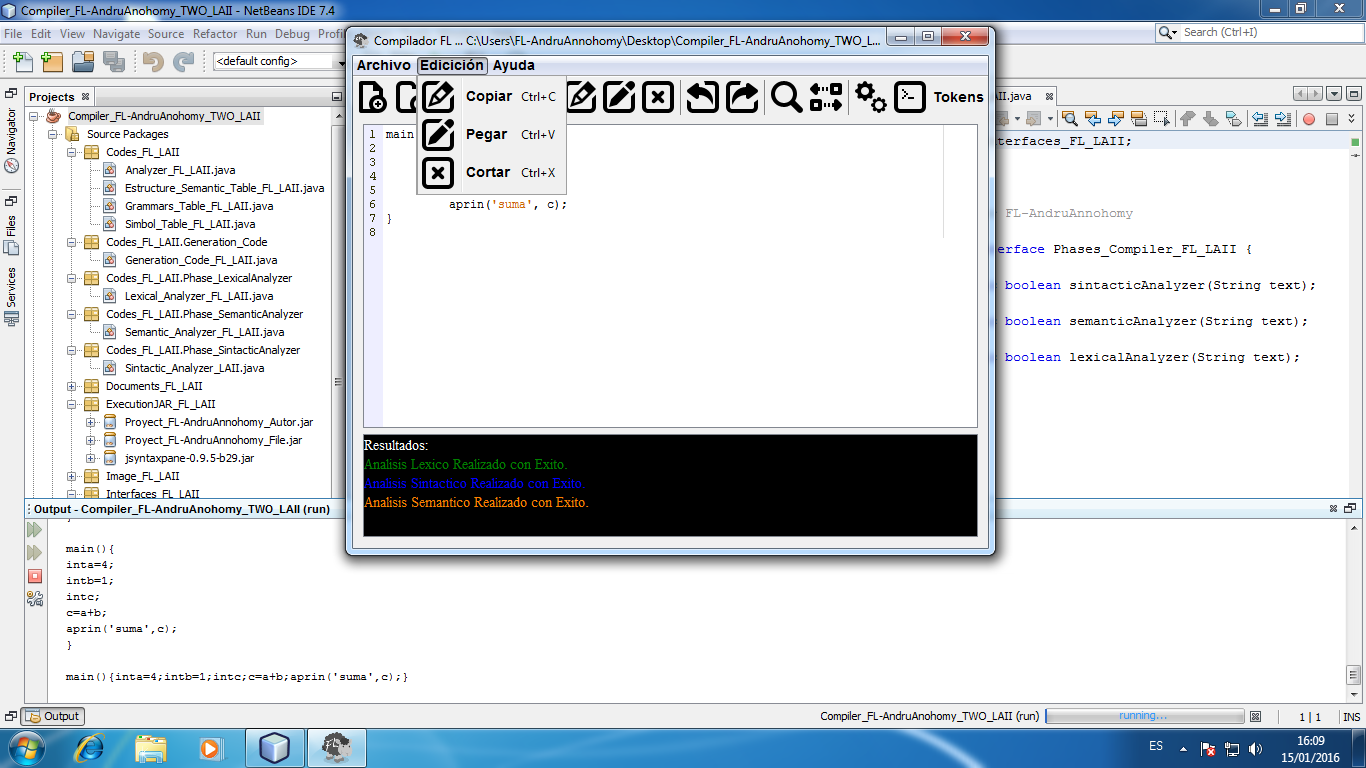


Figura 13. Menú edición.



Figura 14. Menú ayuda.

En el análisis léxico existe un error el cual permite al usuario verificar que se haya ingresado un carácter que no pertenezca al lenguaje. Esto se muestra en la figura 15 en el área de resultados, donde su puede ver el resultado del error junto con la línea donde el análisis encontró el carácter que, no pertenece al lenguaje. Este tipo de error solo es funcional, para la revisión de los caracteres del programa.

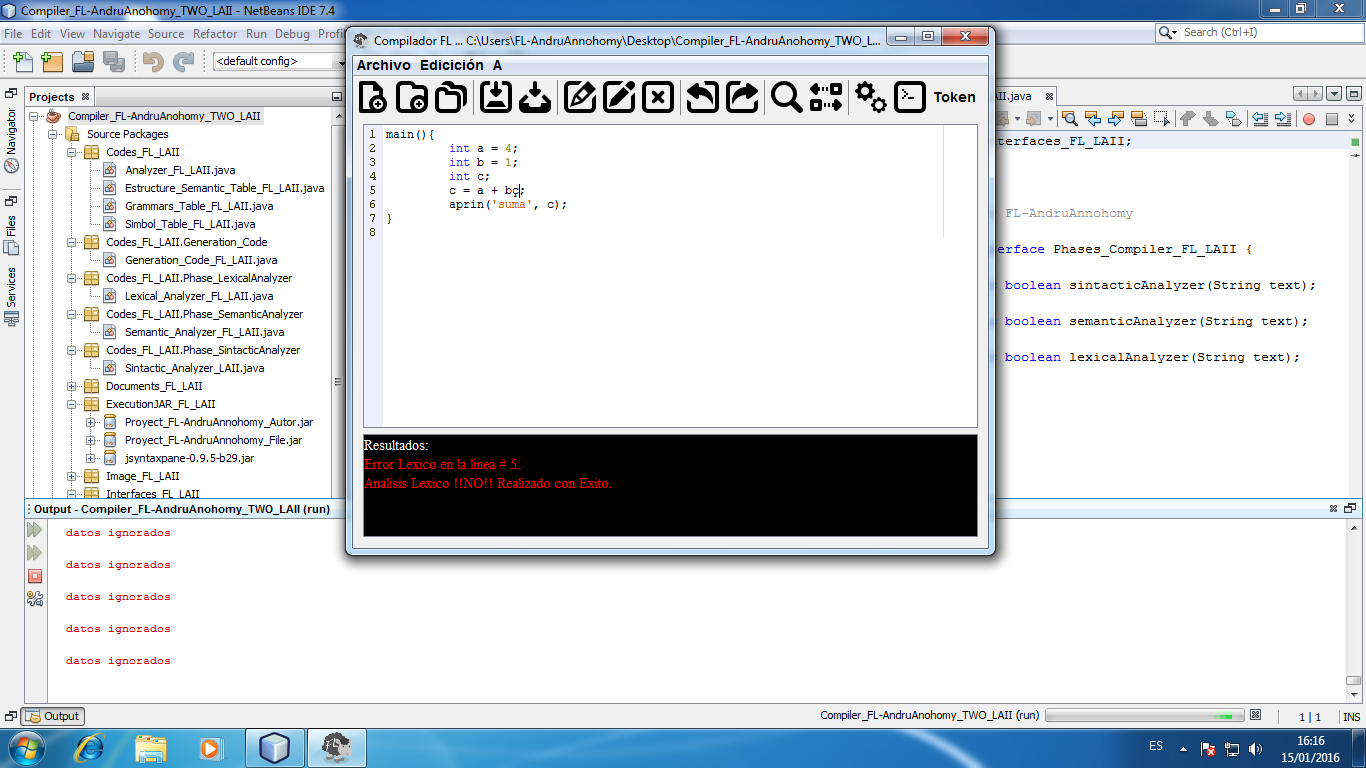


Figura 15. Error léxico.

El análisis sintáctico, se puede tener presente cuando se arroje un error con el de las figuras 16 y 17, donde se puede observar que el error puede estar presente cuando una de las estructuras no haya sido escrita bien o que, se nos haya omitido algo. Este tipo de error es fiable para el usuario, con el fin de poder lo guiar en el uso del compilador y que el funcionamiento del mismo sea el óptimo, así como la ejecución del programa.

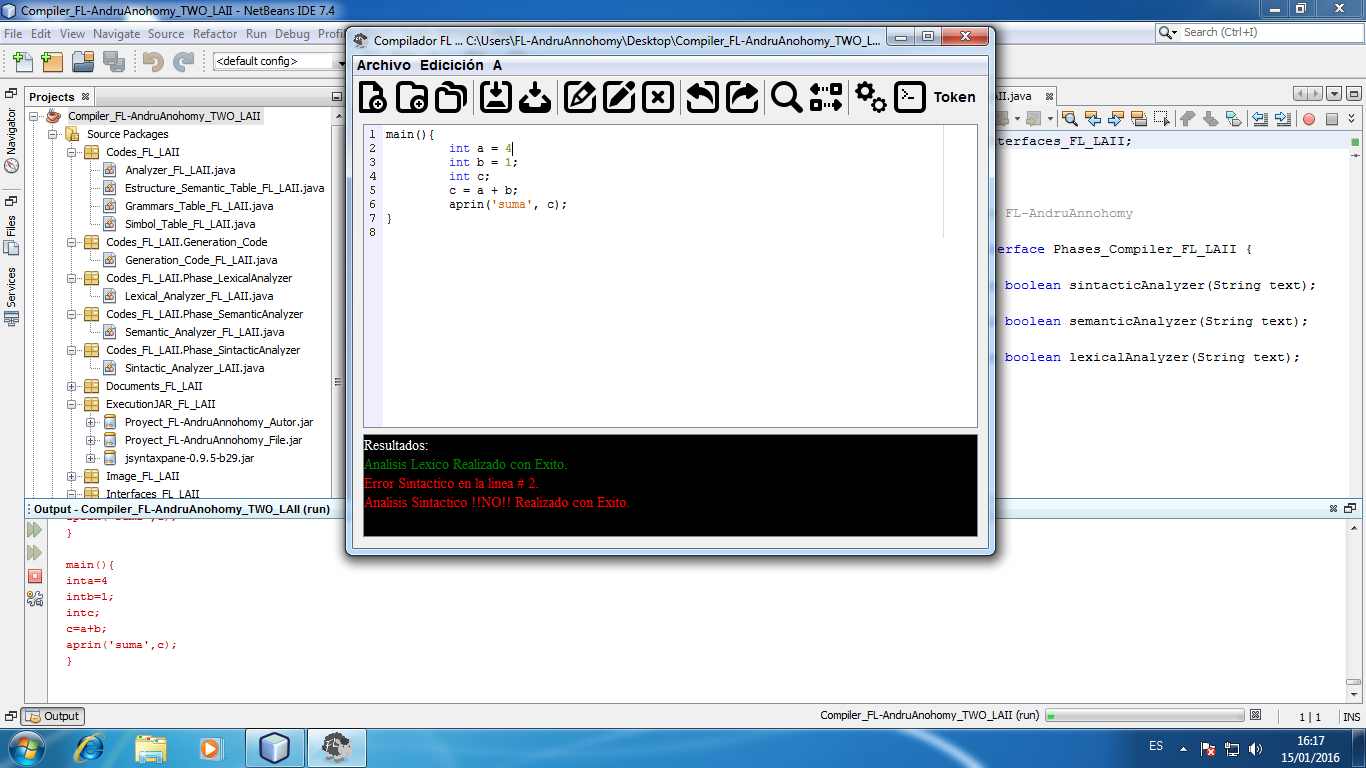


Figura 16. Error sintáctico por falta de punto y coma.

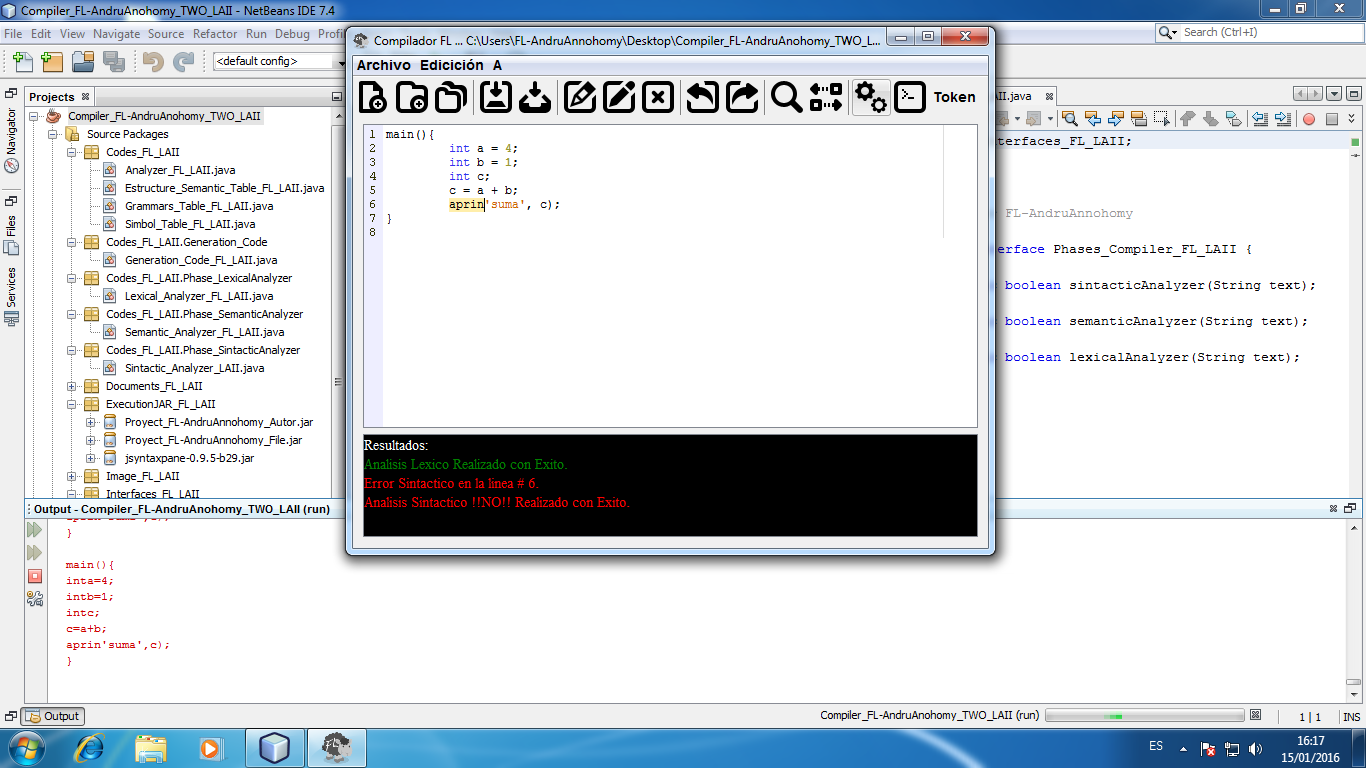


Figura 17. Error sintáctico por falta de paréntesis.

El análisis semántico, es el más relevante en la ejecución del programa motivo por el que, es aquel que nos revisa la coherencia de las estructuras de los programas que se ingresan, todo con el fin de que cuando se ejecute una instrucción sea lo más coherente y que no se confunda el compilador en el momento de la conversión de las estructuras al lenguaje ensamblador y sea posible la creación del código objeto.

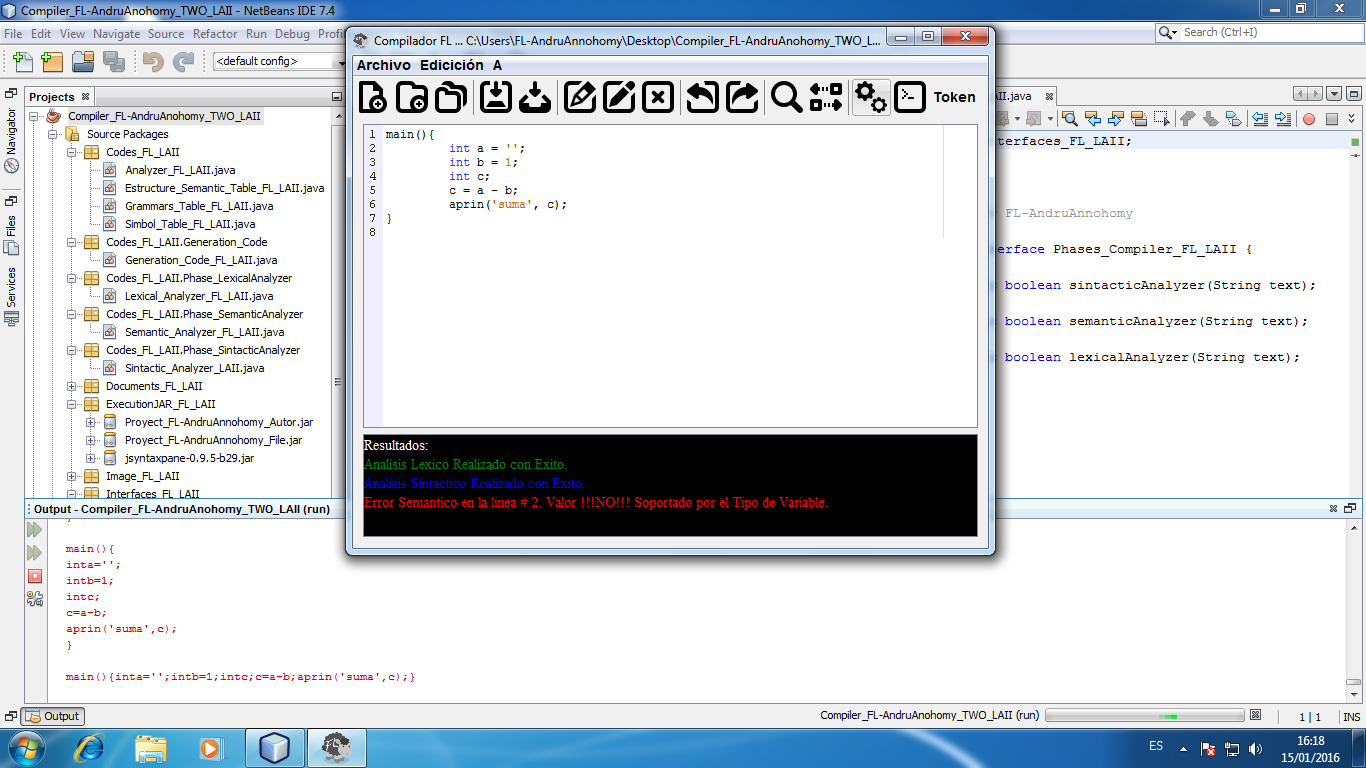


Figura 18. Error sintáctico valor no soportado.

Algunos de los errores que estaremos encontrando son los siguientes, los cuales se muestran en las figuras 18 a 24:

* Error por valor no soportado, este ocurre cuando el usuario quiere poner un valor que no esté soportado por la variable, un ejemplo en la figura 18 y 20.
* Error por operación con cadena, esto currare cuando se quiere realizar una operación con una cadena debió a que las cadenas no pueden realizar operación como suma, resta, multiplicación y división, un ejemplo en la figura 19.



Figura 19. Error semántico operación con cadena.

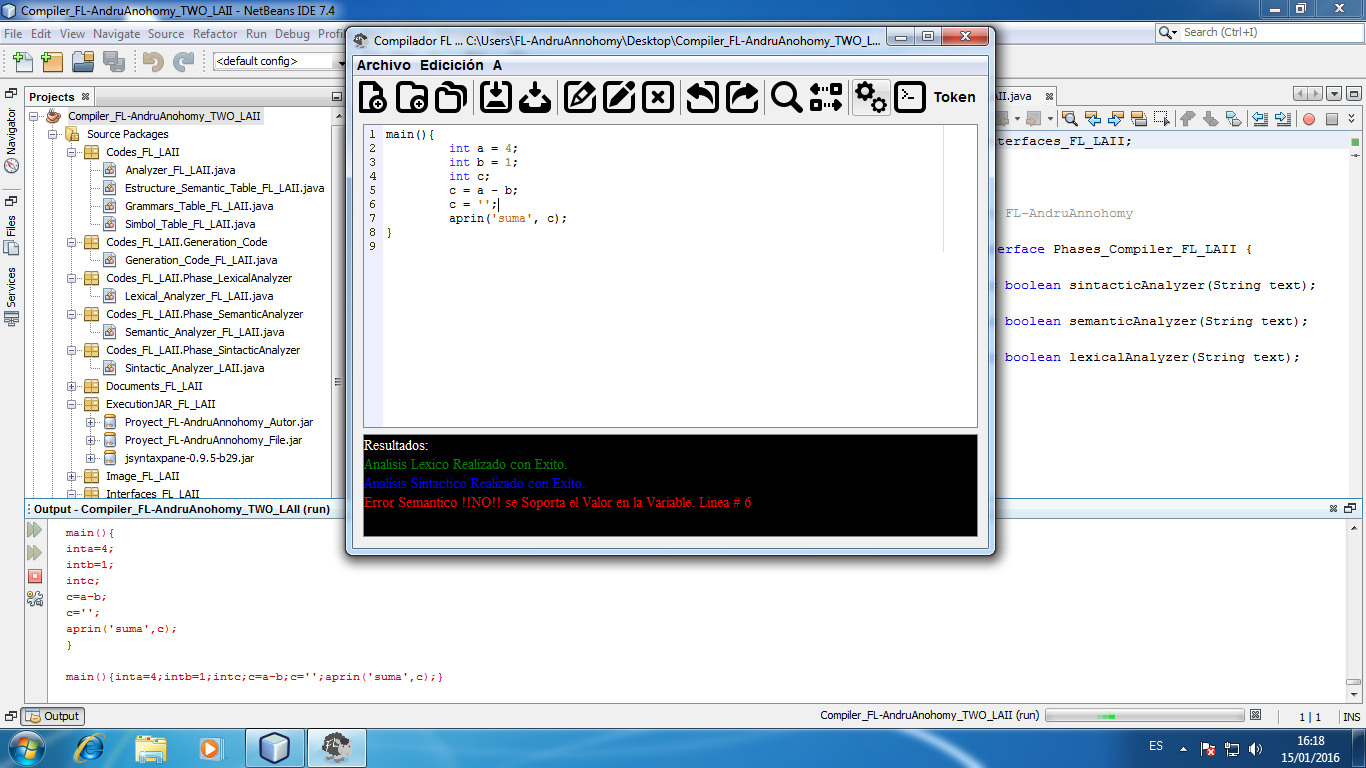


Figura 20. Error semántico por valor no soportado.

* Error orden de las estructuras, esto se debe a que las estructuras deben de tener un orden de ejecución, y el programa de verificar que se realiza la declaración de las sentencias en un orden específico, así también estas deben están dentro del método main de cada programa, un ejemplo en la figura 21 y 22.

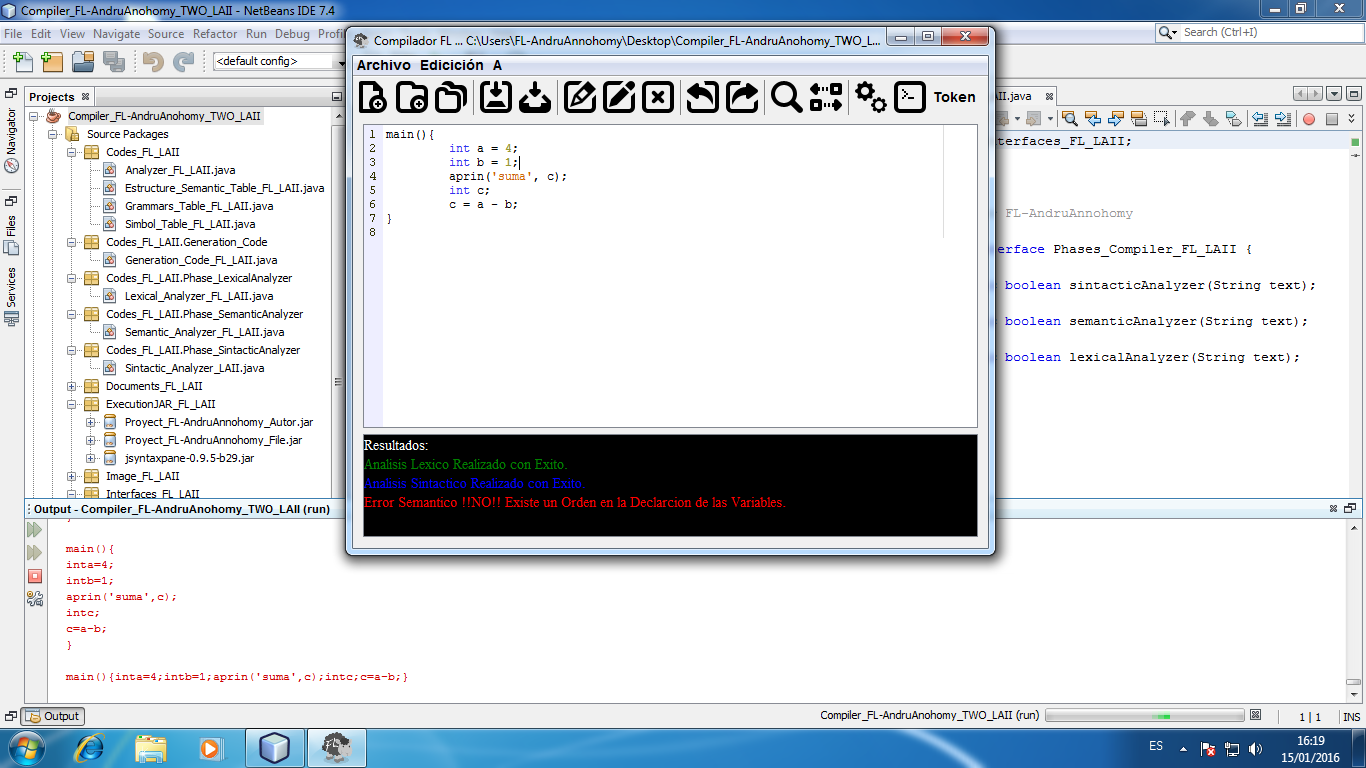


Figura 21. Error semántico orden de sentencias.

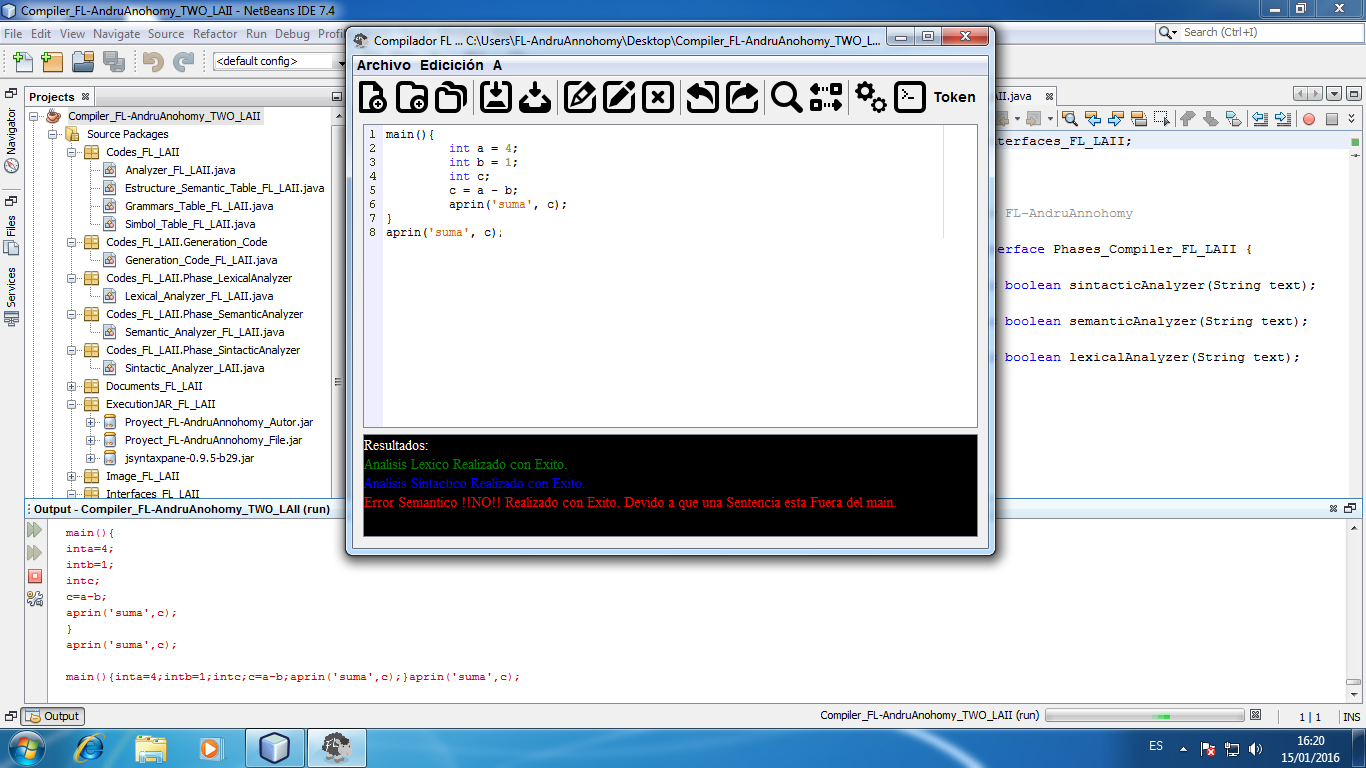


Figura 22. Error semántico sentencia fuera del main.

* Error variable no existe o duplicada, esto consisten en la declaración de una variable que no exista o no este declara, así como si se realiza la declaración de una segunda variable con el mismo nombre, un ejemplo en la figura 23 y 24.

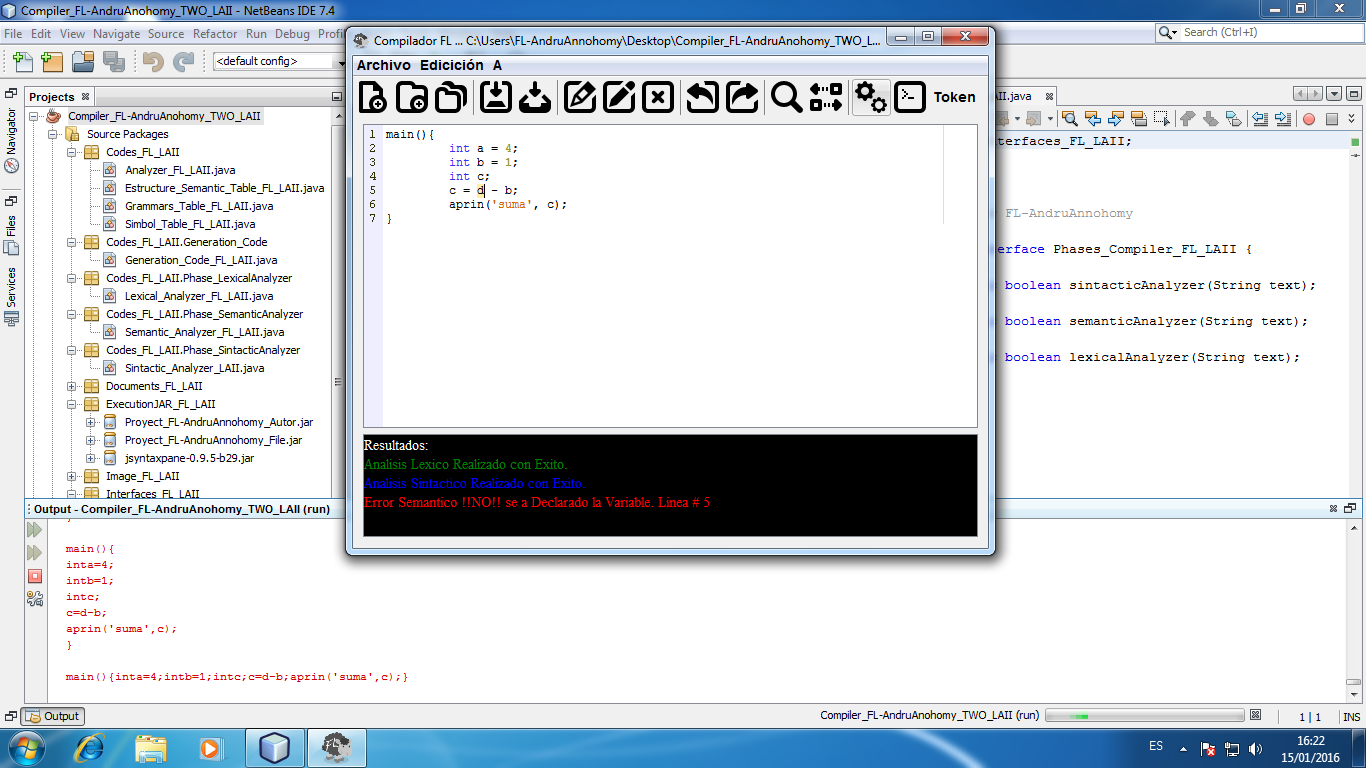


Figura 23. Error semántico variable no declarada.

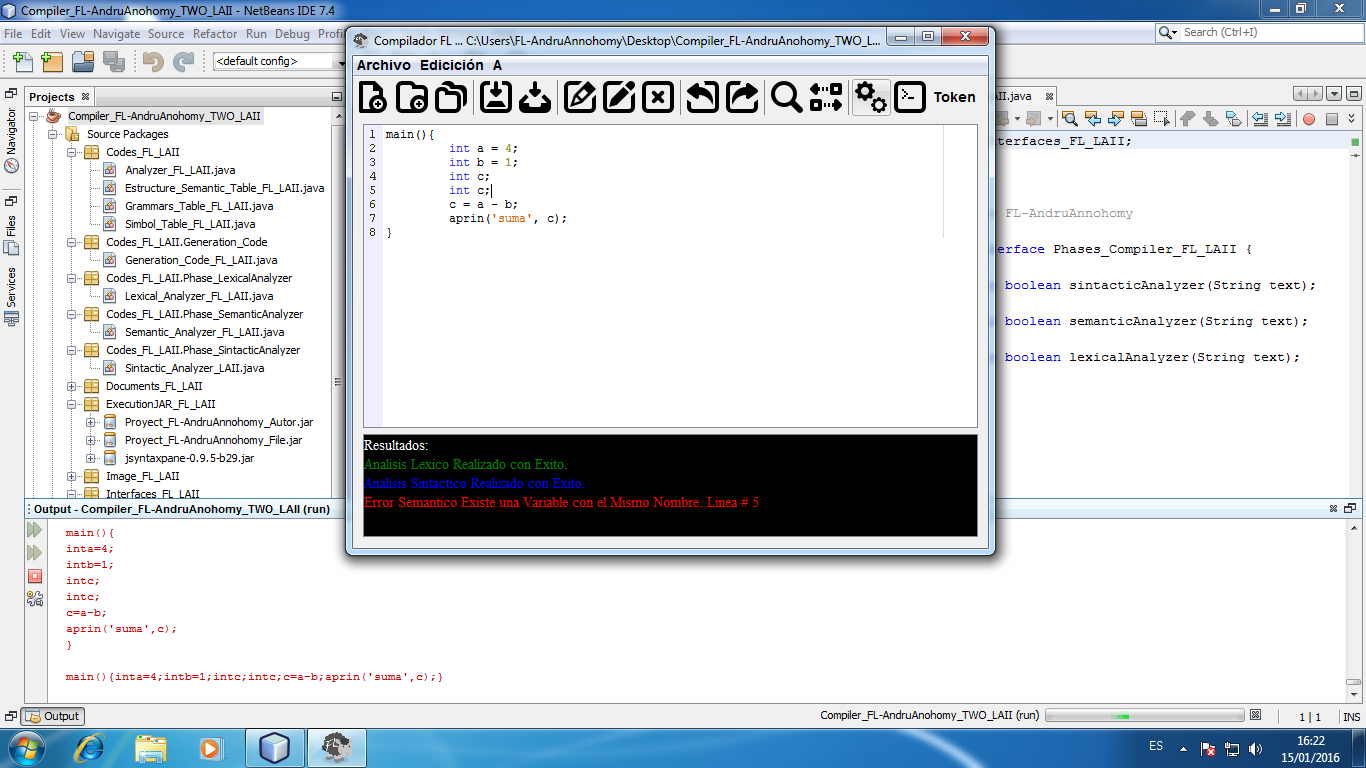


Figura 24. Error semántico variable duplicada.

* Valores no utilizados, este consiste en la declaración de sentencias no utilizada en el programa como por ejemplo, la declaración de una variable sin utilizar, un ejemplo en la figura 25.

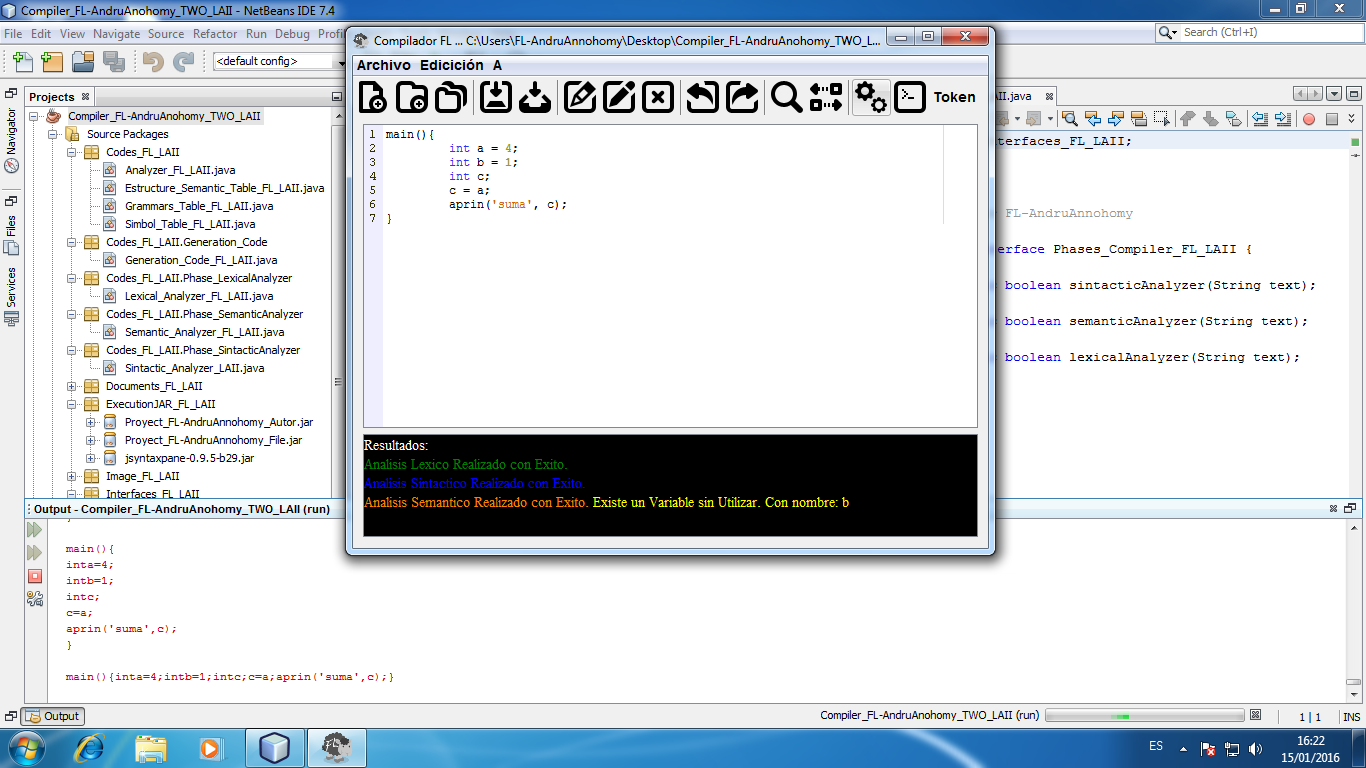


Figura 25. Valores no utilizados.

Tabla de tokens del lenguaje.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | TOKEN | LEXEMA | PATRON |
| 1 | LETRAS | A,B,C,…,Z,a,b,c,…,z,Ñ,ñ | A|B|C|,…,|Z|a|b|c|,…,|z|Ñ|ñ |
| 2 | DIGITOS | 0,1,2,3,4,5,6,7,8,9 | 0|1|2|3|4|5|6|7|8|9 |
| 3 | CARACTERES | !,\,#,/,(,),",|,[,],\_,.,:,, | !|\|#|/|(|)|"|||[|]|\_|.|:|, |
| 4 | IGNORAR | “ ”,\n | “ ”|\n |
| 5 | OPERADORES | +,-,\*,/,^,=,<,> | +|-|\*|/|^|=|<|> |
| 6 | IDENTIFICADOR | A24, contador, a1,… | LETRAS(LETRAS|DIGITOS)\* |
| 7 | CADENA | ‘HOLA’, ‘HOY 24’,… | ‘(DIGITOS|LETRAS|CARACTERES)\*’ |
| 8 | NUMERO | 1234, 35112, 1, 34,63,… | (DIGIT)+ |
| 9 | MAIN | main | Main |
| 10 | APRIN | aprin | aprin |
| 11 | APRINL | aprinl | aprinl |
| 12 | STRING | string | string |
| 13 | INT | int | int |
| 14 | FL | fl | fl |
| 15 | GET | get | Get |
| 16 | MAY | may | May |
| 17 | OPERACIONES | 3 + 5, 5 – A, a \* T, … | (IDENTIFICADOR|NUMERO) OPERADORES (IDENTIFICADOR|NUMERO) |
| 18 | DECLARACION | int a, string b, int a2, … | (INT|STRING) IDENTIFICADOR |
| 19 | ASIGANACION | a = 45, b = ‘HOLA’, … | IDENTIFICADOR = (NUMERO|CADENA) |
| 20 | DECLARACION  ASIGANCION | int a = 45, string = ‘HOLA’, … | (INT|STRING) IDENTIFICADOR  = (CADENA|NUMERO) |
| 21 | CONACTENACION | ,a3 , , h4 , … | ,IDENTIFICADOR |
| 22 | IMPRIMIR | aprin(‘uno’,d2), aprin(‘DOS’), aprinl(‘uno’,d2), aprinl(F6), … | APRIN ( (CADENA|IDENTIFICADOR)  (CONCATENACION)\*) |
| 23 | MOSTRAR | fl(‘HOLA’) | FL(CADENA) |
| 24 | COMANDOS | int s = 0;  aprin(‘hola’);  int c; , … | ((DECLARACION | ASIGNACION | DECLARACIONASIGANCION | IMPRIMIR | MOSTRAR);)+ |
| 25 | FOR | D2>F4: fl(‘HOLA’); may; | OPERACIONES : COMANDOS MAY; |