

Proyecto 3

Generación Código Destino

Johan Vargas Quesada

Randall Corella Castillo

Tecnológico de Costa Rica

IC-5701: Compiladores e Intérpretes

Ing. Allan Rodríguez Dávila

Contenido

[Manual de usuario 3](#_Toc167138015)

[Pruebas de funcionalidad 11](#_Toc167138016)

[Descripción del problema 14](#_Toc167138017)

[Librerías usadas 14](#_Toc167138018)

[Análisis de resultados 15](#_Toc167138019)

[Bitácora 16](#_Toc167138020)

# Manual de usuario

Este proyecto está realizado en java utilizando 2 librerías, la manera correcta de compilar este proyecto es con 2 librerías muy importantes las cuales son: java cup y jflex, estas 2 es muy sencillo de instalar, primero se descargan desde su página oficial: [CUP (tum.edu)](https://www2.cs.tum.edu/projects/cup/) este es el link para descargar java cup

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 1: Página donde se descarga java cup

En la figura 1 se muestra la página oficial para descargar java cup.

[JFlex - JFlex The Fast Scanner Generator for Java](https://jflex.de/) link oficial para descargar JFlex.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 2: Página donde se descarga JFlex

En la figura 2 se muestra la página oficial para descargar JFlex.  
luego se debe descomprimir los archivos descargados.

Texto

Descripción generada automáticamente con confianza media

Figura 3: Archivos descargados

En la figura 3 se muestran los archivos descargados en las paginas mostradas en las figuras 1 y 2.

y por último se agregan en el IDLE utilizado, para este caso es NETBEANS para agregarlos simplemente se van abajo en el proyecto donde dice Libraries como se muestra en la figura 4.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 4: Paquete donde se deben implementar las librerías descargadas

Con click derecho sobre el paquete señalado en la figura 4 aparecerá una ventada, en esa ventana se presiona la opción que dice add jar/folder como se muestra en la figura 5.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figura 5: Opciones para añadir las librerías

Y se busca donde se descomprieron los archivos descargados y se agregan.

Además, se debe asegurar de contar con el JDK de java en el equipo.

Para poder utilizar este proyecto se debe tener en cuenta que cuenta con los siguientes archivos mostrados en la figura 6.  
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 6: Archivos del programa

El archivo que se debe correr es el PY01Compiladores.java que se muestra en la quinta posición, pero para poder utilizarse correctamente debe de utilizarse con un archivo de texto como por ejemplo test.txt que se encuentra al final de la figura 6, el cual debe contener un código similar al de la figura 7.

Imagen que contiene Tabla

Descripción generada automáticamente

Figura 7: archivo donde se escribe el código de prueba

Luego se ejecuta el proyecto, este se puede correr dándole click derecho sobre el archivo PY01Compiladores.java como en la figura 8 o tocar el botón de correr como en la figura 9, se debe considerar que dependiendo del equipo cuando se producen cambios en código y en pruebas, se debe ejecutar 2 veces para que se actualicen los cambios.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 8: Correr el archivo con el archivo dándole click derecho

Interfaz de usuario gráfica, Texto, Aplicación, Word, Correo electrónico

Descripción generada automáticamente

Figura 9: Correr archivo dándole el botón Run.

Una vez se ejecuta el programa se generarán varios archivos, el codigoTresDirecciones.txt y el codigoMips.s que se muestran en la figura 6 en la posición 3 y 2 respectivamente.

Figura 10: generación del código intermedio.

Texto

Descripción generada automáticamente  
En la figura 10 se muestra como se ve el código intermedio dentro del archivo.

Texto

Descripción generada automáticamente con confianza media

Figura 11: generación del código mips.

El codigo mips de la figura 11 se genera a partir del código intermedio de 3 direcciones.

Por último, una vez ejecutado el programa se mostrará mensajes de código ejecutado y en caso de algún error se mostrará el error con fila y columna además del tipo de error, en este caso los errores serán de tipo semántico y sintactico.

Figura 12: Errores semánticos.

Texto

Descripción generada automáticamente

En la figura 12 se muestran los errores semánticos que proporciona el código en la figura 7.

# Pruebas de funcionalidad

Figura 13: código de prueba.

Texto

Descripción generada automáticamente con confianza baja

En la figura 13 se muestra el código de prueba, el cual esta en un archivo de texto, en caso de que su compilación sea exitosa generará 1 archivo el del código intermedio. En esta ocasión fue exitosa por lo tanto se deberían de ver como en la figura 10.

En la figura 12 se pueden ver errores de semántica que existen en el código de prueba, en caso de que el analizador semántico se encuentre con un error, este mostrará el error, la fila y la columna, en caso de que sea exitoso, se mostrara un mensaje de éxito y se generará el código intermedio como el de la figura 10 y finalmente se generará el código Mips que se puede ver en la figura 11, este último se genera cuando ya no existen errores de ningún tipo en el código.

# Descripción del problema

# El objetivo principal es desarrollar un generador de código MIPS a partir de código de tres direcciones generado por el proyecto anterior (Proyecto II). Este código de tres direcciones es el resultado del análisis sintáctico y semántico realizado por el compilador.

# Para comprobar el desarrollo de esta fase el programa a presentar deberá tomar un archivo fuente y realizar lo siguiente:

# Preservar y corregir los alcances de los Proyectos I y II

# Indicar si el archivo fuente puede o no ser generado por la gramática. Tomando en cuenta la gramática, sintaxis y semántica.

# Reportar los errores léxicos, sintácticos y semánticos encontrados. Debe utilizar la técnica de Recuperación en Modo Pánico.

# Escribir en un archivo el código MIPS para el archivo fuente. El código generado corresponde semánticamente al código fuente.

# Librerías usadas

java\_cup.runtime: Librería necesaria para el uso de java cup.

java.nio.file.Files, java.nio.file.Path, java.nio.file.Paths: Librerias necesarias para el uso de ubicaciones de los archivos.

java.io.IOException: Librería para manejar algunos errores.

java.io.FileReader, java.io.FileWriter: Librerias para leer y escribir en archivos.

java.util.ArrayList: Librería necesaria para la creación de arraylist.

java.util.HashMap: Se utiliza para almacenar pares de datos en forma de clave-valor.

java.io.PrintWriter; sirve para imprimir líneas completas de texto y diferentes tipos de datos.

java.util.regex.Matcher; se utiliza con la librería a continuación permite aplicar patrones de expresiones regulares a cadenas de texto.

java.util.regex.Pattern; se utiliza para definir patrones de búsqueda que luego pueden ser aplicados sobre cadenas de texto utilizando la librería anterior “Matcher”.

# Análisis de resultados

|  |  |  |
| --- | --- | --- |
|  | Objetivos alcanzados | Objetivos no alcanzados |
| Preservar y corregir los alcances de los Proyectos I y II | X |  |
| Indicar si el archivo fuente puede o no ser generado por la gramática. Tomando en cuenta la gramática, sintaxis y semántica. | X |  |
| Reportar los errores léxicos, sintácticos y semánticos encontrados. Debe utilizar la técnica de Recuperación en Modo Pánico. | X |  |
| Escribir en un archivo el código MIPS para el archivo fuente. El código generado corresponde semánticamente al código fuente. |  | X se hicieron algunas partes, pero no es funcional en su totalidad. |

# Bitácora

Enlace a la bitácora: [RandallCorellaCastillo/PY03Compi (github.com)](https://github.com/RandallCorellaCastillo/PY03Compi)