



INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Computo

PROYECTO

Generador de figuras

Asignatura:
Compiladores

Presenta:
Bernal Ramírez Brian Ricardo
Escalona Zuñiga Juan Carlos

Profesor:
Tecla Parra Roberto

25 / Junio /2025



Contenido:

• Introducción	3
• Requisitos	4
• ¿Cómo obtener Acceso de Administrador (Root)?	5
• ¿Cómo obtener byacc, lex o flex?	6
• ¿Cómo verificar versión de Java?	7
• Descargar el compilador	8
• Ejecución del programa	G
• Interfaz	11
• Cuadro Gris	11
• Cuadro Blanco	12
• Botón Compilar	13
• Botón Borrar	14
• Botón “Cargar archivo”	15
• Resumen del Funcionamiento del Programa	16
○ Interfaz de usuario	
○ Visualización	
• Conclusión	17

Introducción:

En la materia de “Compiladores”, se nos enseñaron las bases y fundamentos que tienen por detrás los compiladores que usamos al día con día en nuestra vida de programadores. A lo largo del curso, hicimos diversas actividades que nos ayudaron a reforzar y obtener conocimientos esenciales para el buen uso y desarrollo de un compilador.

Se nos propusieron varios proyectos a elaborar, de los cuales, optamos por el de crear un compilador desde cero, haciendo uso de los conocimientos adquiridos en la materia. De la misma forma, con la implementación de interfaces y conocimientos de diseño, logramos crear un compilador muy intuitivo para el usuario. Esto con la finalidad de poder llegar a usuarios de todas las edades y fomentar el uso de este, haciéndolo atractivo visualmente como en funcionalidades.

La finalidad de nuestro proyecto va más allá de obtener una nota aprobatoria en la materia, sino también buscamos plasmar nuestros conocimientos de una forma divertida y atractiva como lo es la “creación de logos” o “creación de figuras”.

Lo antes mencionado, se logra con ayuda de nuestro compilador el cuál espera que el usuario ingrese ciertos datos y funciones que permitan al compilador interpretar lo que ingresa el usuario a lenguaje máquina, y este le pueda traducir a la computadora lo que el usuario ingreso para su ejecución y le muestre al usuario mediante una interfaz gráfica lo que se generó con las entradas que dio.

A lo largo del documento, veremos como nuestros usuarios pueden hacer uso de nuestro compilador guiándolos con texto e imágenes para un mejor entendimiento y satisfacción a nuestros usuarios.

Requisitos:

Nuestro compilador está pensado y realizado para ser ejecutable de manera fluida y nativa en cualquier computadora que cuente con Java 6, 2 gb de ram, un procesador de 4 núcleos y una distro de Linux la que nos permita la compilación y ejecución de java.

Sin embargo, es preferible que la maquina de nuestro usuario cuente con las siguientes características para una eficacia y fluidez agradable para nuestros usuarios. Los requisitos son los siguientes:

- Ubuntu o derivados del mismo.
- 4 gb de Ram.
- 2 gb de almacenamiento.
- Procesador con 4 o más núcleos.
- Editor de texto.
- Acceso de Administrador (root)
- Tener la última versión de su distro de Linux, así como la última versión de byacc, lex o flex.

En caso de no contar con alguno de los dos últimos, seguir los siguientes pasos para instalar correctamente estos requisitos indispensables para la compilación y ejecución del programa.

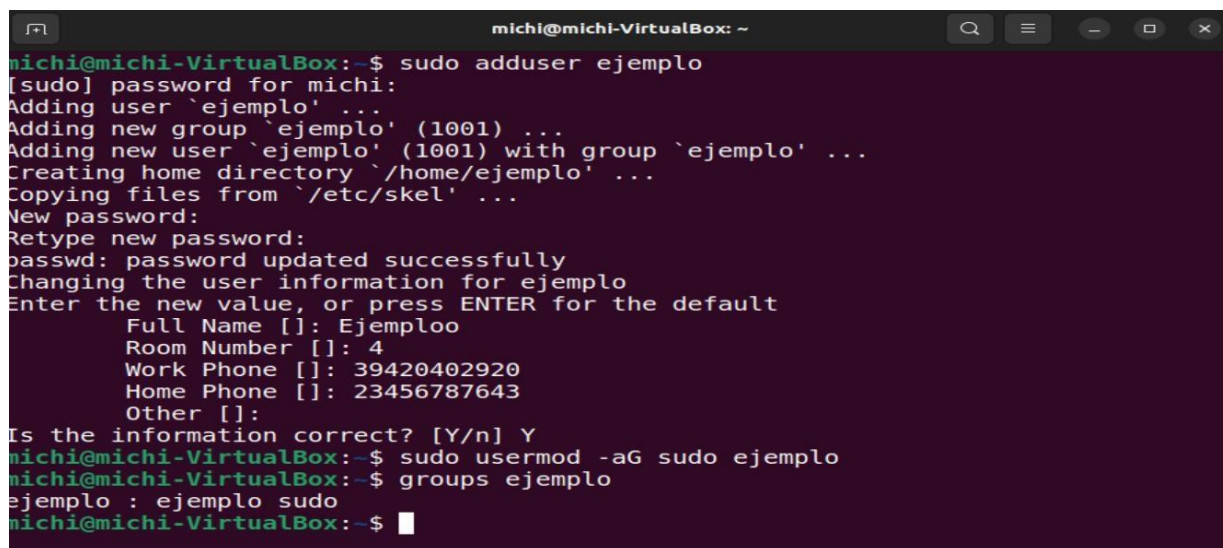
¿Cómo obtener Acceso de Administrador (Root)?

En caso de no tener un super usuario, deberá seguir los siguientes pasos:

Crear un nuevo usuario

1. **Abrir una terminal:** Puedes abrir una terminal en Ubuntu presionando `Ctrl + Alt + T` o buscando "Terminal" en el menú de aplicaciones.
2. **Crear el usuario:** Utiliza el siguiente comando para crear un nuevo usuario. Reemplaza `ejemplo` con el nombre que desees para el nuevo usuario. Este comando creará un nuevo usuario con nombre `ejemplo`. Te pedirá que ingreses una contraseña para este usuario y otros detalles opcionales como el nombre completo, número de teléfono, etc.
3. **Asignar privilegios de sudo (opcional):** Por defecto, el nuevo usuario creado no tendrá privilegios de sudo. Para darle acceso a los comandos de administración, como mencioné anteriormente, puedes agregarlo al grupo `sudo`. Ejecuta el siguiente comando para agregar el usuario al grupo `sudo`:

Al seguir estos pasos, habrás creado un nuevo usuario en Ubuntu con privilegios de superusuario, lo que te permitirá administrar el sistema y realizar tareas que requieren permisos elevados de manera segura y organizada. En este caso, es necesario tener acceso a estos privilegios para ejecutar el compilador.



```
michi@michi-VirtualBox: ~  
michi@michi-VirtualBox:~$ sudo adduser ejemplo  
[sudo] password for michi:  
Adding user `ejemplo' ...  
Adding new group `ejemplo' (1001) ...  
Adding new user `ejemplo' (1001) with group `ejemplo' ...  
Creating home directory `/home/ejemplo' ...  
Copying files from `/etc/skel' ...  
New password:  
Retype new password:  
passwd: password updated successfully  
Changing the user information for ejemplo  
Enter the new value, or press ENTER for the default  
Full Name []: Ejemplo  
Room Number []: 4  
Work Phone []: 39420402920  
Home Phone []: 23456787643  
Other []:  
Is the information correct? [Y/n] Y  
michi@michi-VirtualBox:~$ sudo usermod -aG sudo ejemplo  
michi@michi-VirtualBox:~$ groups ejemplo  
ejemplo : ejemplo sudo  
michi@michi-VirtualBox:~$
```

¿Cómo obtener byacc, lex o flex?

Para obtener `byacc`, `lex` o `flex` en sistemas basados en Ubuntu (como Ubuntu, Linux Mint, etc.), puedes instalar estos programas a través del gestor de paquetes `apt`. Aquí te explico cómo hacerlo:

Instalar `byacc`

`byacc` es un generador de analizadores sintácticos basado en Yacc (Yet Another Compiler Compiler).

Una vez instalado, puedes usar `byacc` para generar analizadores sintácticos.

```
michi@michi-VirtualBox:~$ sudo apt install byacc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
byacc is already the newest version (1:2.0.20220114-1).
0 upgraded, 0 newly installed, 0 to remove and 59 not upgraded.
```

Instalar `lex`

`lex` es un generador de analizadores léxicos.

En Ubuntu, `flex` es la implementación de `lex`. Una vez instalado, puedes usar `flex` para generar analizadores léxicos.

```
michi@michi-VirtualBox:~$ sudo apt install flex
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfl-dev libfl2
```

Verificar la instalación

Después de instalar `byacc` o `flex`, puedes verificar si se han instalado correctamente ejecutando los comandos correspondientes en la terminal:

```
michi@michi-VirtualBox:~$ byacc -V
byacc - 2.0 20220114
michi@michi-VirtualBox:~$ flex -V
flex 2.6.4
michi@michi-VirtualBox:~$
```

Estos comandos deberían mostrar la versión del software instalado, confirmando que la instalación fue exitosa.

Notas adicionales

- Si estás en una distribución diferente de Linux que no usa `apt` como gestor de paquetes (por ejemplo, CentOS/RHEL usa `yum` o `dnf`), los comandos pueden ser diferentes. Consulta la documentación de tu distribución para obtener instrucciones específicas.

- Si necesitas `lex` específicamente en lugar de `flex`, puedes crear un enlace simbólico `lex` que apunte a `flex`, ya que en Ubuntu `flex` proporciona la funcionalidad de `lex`:

```
michi@michi-VirtualBox:~$ sudo ln -s /usr/bin/flex/usr/bin/lex
michi@michi-VirtualBox:~$
```

Esto te permite usar `lex` como si fuera `flex`.

Siguiendo estos pasos, deberías poder instalar y utilizar `byacc` y `flex` (o `lex`) en tu sistema Ubuntu sin problemas.

¿Cómo verificar versión de Java?

Como se mencionó un inicio, se necesita Java 6 en adelante, por lo que verificaremos la versión que tenemos. En caso de ser una versión previa a Java 6, se necesitará instalar una versión más reciente.

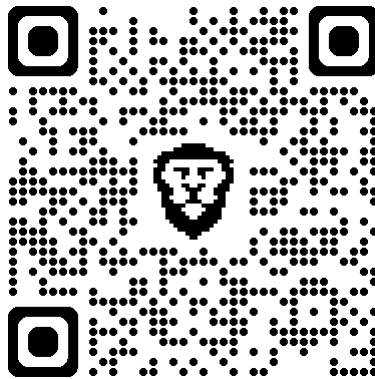
Con los siguientes comandos, podemos verificar e instalar Java sin problema alguno.

```
michi@michi-VirtualBox:~$ java -version
openjdk version "11.0.23" 2024-04-16
OpenJDK Runtime Environment (build 11.0.23+9-post-Ubuntu-1ubuntu122.04.1)
OpenJDK 64-Bit Server VM (build 11.0.23+9-post-Ubuntu-1ubuntu122.04.1, mixed mode, sharing)
```

```
michi@michi-VirtualBox:~$ sudo apt install openjdk-17-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  openjdk-17-jdk-headless openjdk-17-jre openjdk-17-jre-headless
```

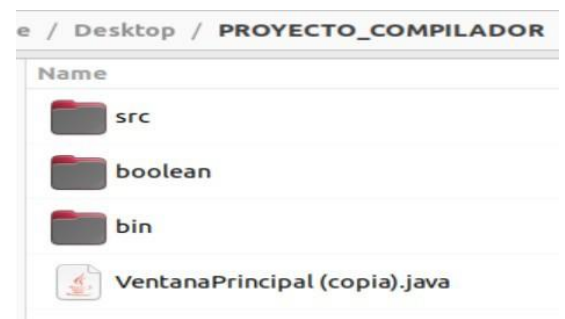
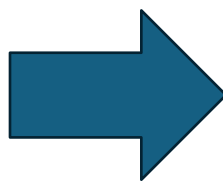
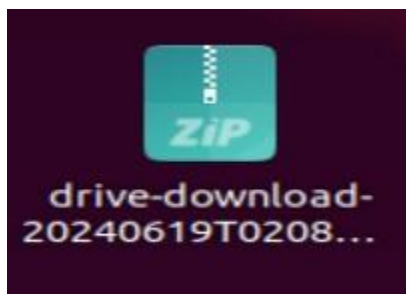
Descargar el compilador

Una vez verificado que hayamos cumplido con los requisitos, es momento de descargar nuestro compilador. Escanea el siguiente QR para poder dirigirte en donde está alojado el “.zip” donde está contenido el compilador.



Una vez descargado, nos aparecerá todo comprimido en una carpeta, la cuál con ayuda de algún programa propio de su sistema operativo o con uno externo (como rar), nos aparecerán todos los archivos que conforman el compilador; Desde las imágenes que se ocuparon para la interfaz, como el código que conforman al compilador.

NOTA: Es de suma importancia que no mueva nada en los archivos que se encontraban y generaron antes y después de la compilación, ya que la más mínima modificación podría generar errores que sólo se corregirían con algún respaldo de la carpeta o la reinstalación de esta.

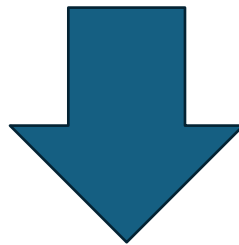
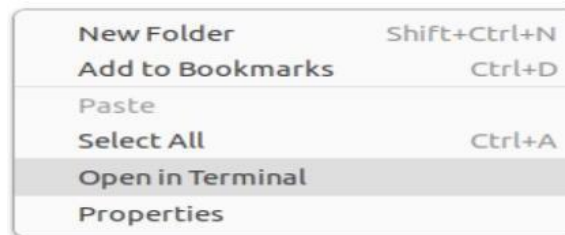


Ejecución del programa

Una vez que estamos dentro de la carpeta “PROYECTO_COMPILADOR”, iremos a la carpeta “src” que se encuentra en la parte superior de la carpeta. Comúnmente siempre es la carpeta que encabeza la lista.

Name	Size	Modified
src	6 items	mar ☆
boolean	8 items	13 may ☆
bin	5 items	mar ☆
VentanaPrincipal (copia).java	7.7 kB	9 jun ☆

Una vez dentro de esta, vamos a abrir una terminal. Esto se hace dando clic derecho en un espacio en blanco. Se abrirá un menú y en la opción “open in Terminal” damos clic para seleccionar la opción. Tardará unos segundos y nos abrirá nuestra terminal de forma fácil y sencilla en el directorio que es necesario para los siguientes pasos.



```
michi@michi-VirtualBox: ~/Desktop/PROYECTO_COMPILADOR/src
michi@michi-VirtualBox:~/Desktop/PROYECTO_COMPILADOR/src$
```

Estando en la terminal, daremos paso a ingresar los siguientes comandos para lograr la correcta ejecución de nuestro programa. Es muy importante que los siguientes pasos se hagan de la manera que se muestran a continuación:

Crear el directorio de salida para los archivos compilados:

```
michi@michi-VirtualBox:~/Desktop/PROYECTO_COMPILADOR/src$ mkdir -p ../bin
```

Compilar el proyecto:

```
michi@michi-VirtualBox:~/Desktop/PROYECTO_COMPILADOR/src$ javac -d ../bin $(find . -name "*.java")
```

Navegar al directorio de archivos compilados (bin):

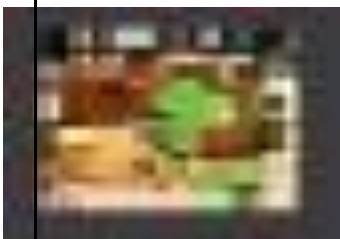
```
michi@michi-VirtualBox:~/Desktop/PROYECTO_COMPILADOR/src$ cd ../bin
```

Ejecutar el proyecto:

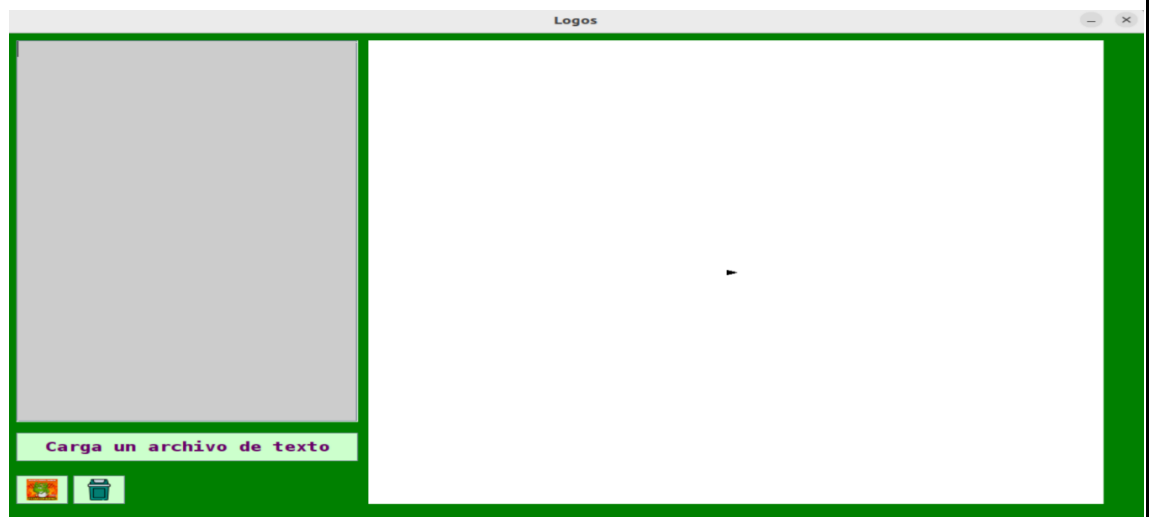
```
michi@michi-VirtualBox:~/Desktop/PROYECTO_COMPILADOR/bin$ java Logos.Logos
```

Si todo se realizó de manera correcta y ordenada, nos deberá tardar en un periodo máximo de 5-8 segundos en aparecernos una ventana verde con un cuadrado blanco. Esta ventana no es más que la interfaz de nuestro compilador, el cual se explicará en unos momentos a fondo sus partes y funciones que lo integran.

Icono de la app

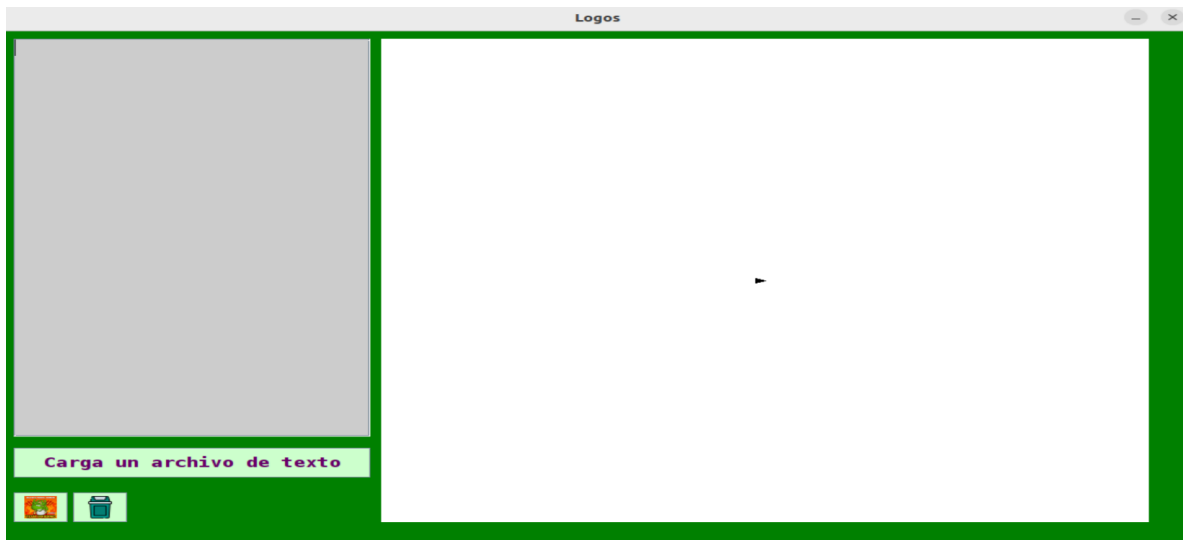


Interfaz maximizado del compilador



Interfaz

Una vez dentro, tendremos la interfaz que se mostró en la hoja anterior. Aquí podremos ver que está compuesta por 3 botones principales y dos cuadros, los cuáles cuentan con diferentes funciones que a continuación explicaremos a lujo de detalle.



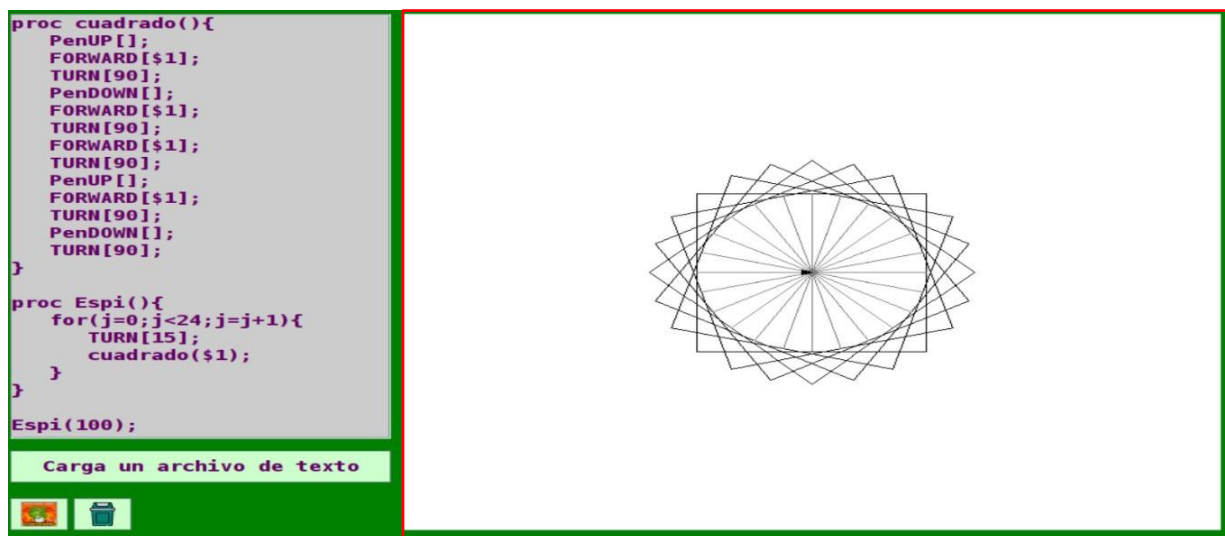
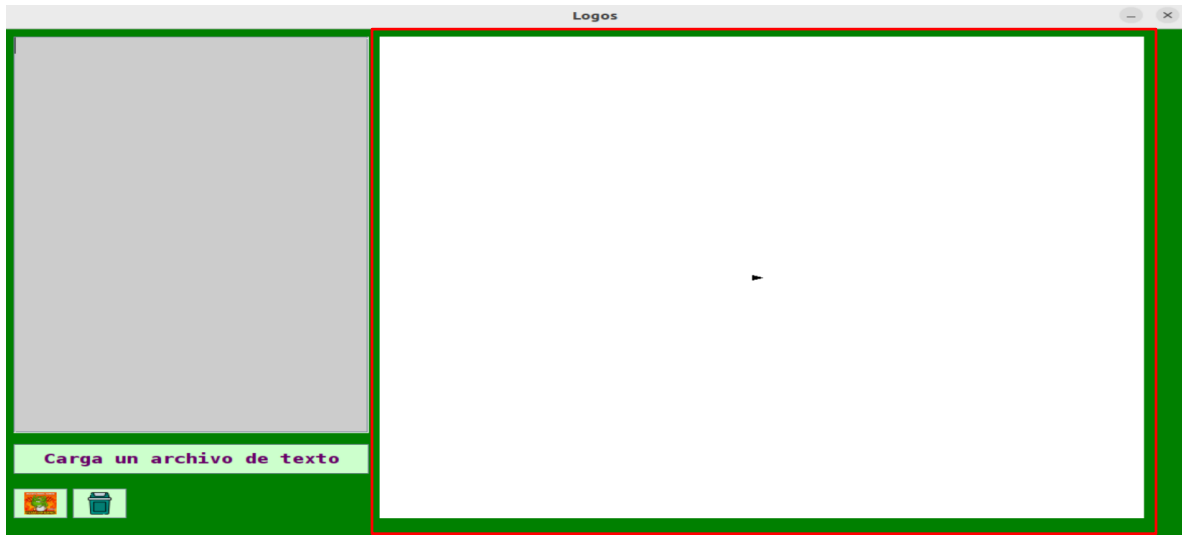
Cuadro Gris

Este cuadro cuenta con la funcionalidad de permitirnos escribir y borrar líneas de código, con las cuales haremos las figuras que queramos con las dimensiones, colores y coordenadas que queramos, siempre y cuando la sintaxis sea correcta. En caso de tener un problema de sintaxis, el compilador se cerrará y nos mandará "error" desde la terminal. Para volver a usarlo, sólo basta volver abrirlo desde la terminal con el comando: `java Logos.Logos`



Cuadro Blanco

Este cuadro tiene la funcionalidad de ser una especie de “pizarrón” o “proyector”, en dónde podremos visualizar el resultado final de nuestro trabajo que hayamos realizado gracias al código que hayamos ingresado en el “Cuadro Gris”. En dado caso de que no se muestre la figura y no haya arrojado la terminal el mensaje “error” o se haya cerrado el compilador, favor de volver a compilarlo con el botón “Compilar” para que se muestre de manera correcta.



Botón Compilar

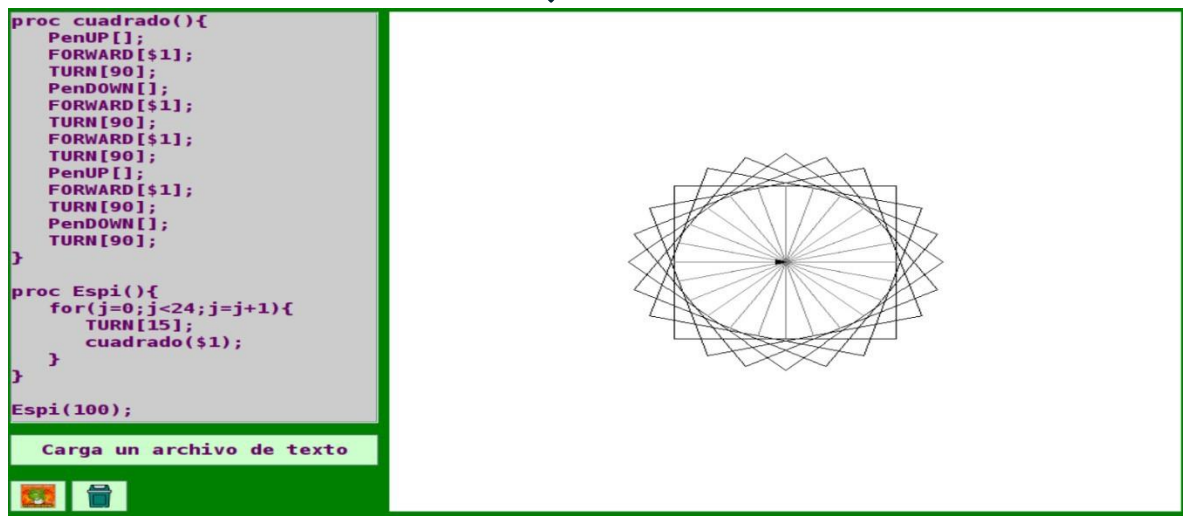

Este botón de una rana con fondo naranja que se encuentra en la esquina inferior izquierda hace la función de “Compilar” el código que hayamos escrito en el “Cuadro Gris” que se mencionó anteriormente. Una vez que presionemos el botón, empezará a leer y comparar línea por línea, carácter por carácter que esté bien la sintaxis del código que se haya ingresado para evitar errores al momento de generar el logo. En caso de que todo esté bien y orden, el compilador podrá generar el logo y lo mostrará en el “Cuadro blanco” como se muestra a continuación:

```
proc cuadrado(){
  PenUP [];
  FORWARD [$1];
  TURN [90];
  PenDOWN [];
  FORWARD [$1];
  TURN [90];
  FORWARD [$1];
  TURN [90];
  FORWARD [$1];
  TURN [90];
  PenUP [];
  FORWARD [$1];
  TURN [90];
  PenDOWN [];
  TURN [90];
}

proc Espi(){
  for(j=0;j<24;j=j+1){
    TURN [15];
    cuadrado($1);
  }
}

Espi(100);
```

Carga un archivo de texto



Botón Borrar

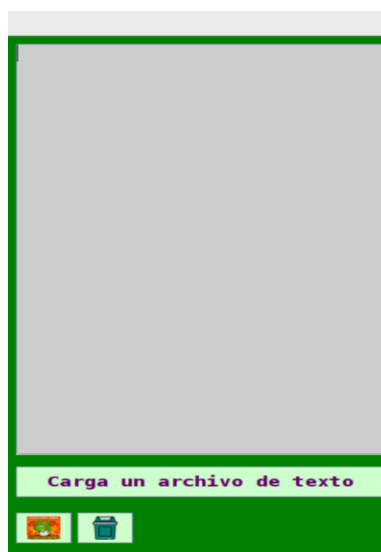

Este botón con icono de “bote de basura” que se encuentra de lado derecho del “Botón Compilar” tiene la funcionalidad de poder borrar todo el código que hayas ingresado en el “Cuadro Gris”. Esto nos sirve para ahorrar tiempo a la hora de que queramos ingresar otro código, ya que con un solo clic te evitarás de tener que borrar línea por línea de código.

```
proc cuadrado(){
  PenUP[];
  FORWARD[$1];
  TURN[90];
  PenDOWN[];
  FORWARD[$1];
  TURN[90];
  FORWARD[$1];
  TURN[90];
  PenUP[];
  FORWARD[$1];
  TURN[90];
  PenDOWN[];
  TURN[90];
}

proc Espi(){
  for(j=0;j<24;j=j+1){
    TURN[15];
    cuadrado($1);
  }
}

Espi(100);
```

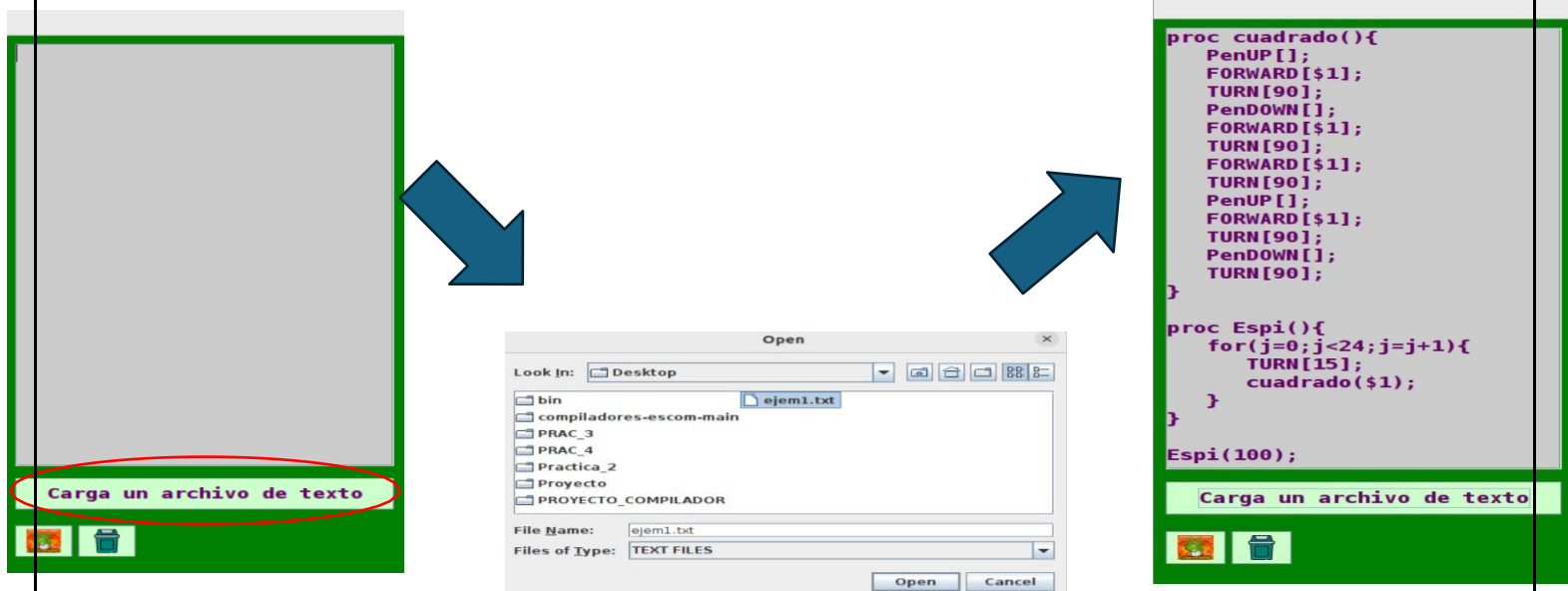
Carga un archivo de texto



Botón “Cargar archivo”

Este botón de forma rectangular que se encuentra en la parte inferior del programa. Su función es poder abrir los archivos de nuestra computadora para poder cargar en el programa algún documento de texto que con tenga algún código que forme alguna figura que nosotros hayamos hecho previamente. Es importante que el documento sea “.txt” para que funcione y sea leído e interpretado de forma correcta por el compialador.

Una vez cargado el archivo, veremos que se mostrará lo que hay en el archivo (en este caso el código de una figura). Ya estando seguro de que se cargó correctamente, usaremos el botón con la imagen de rana o “Botón compilador” que hará que el compilador verifique y compile nuestro código, así mostrándonos en la parte derecha de la ventana nuestra figura con los colores y coordenadas que le brindamos.



Resumen del Funcionamiento del Programa

Interfaz de Usuario

El usuario ingresa código Logos en un área de texto dentro de la VentanaPrincipal.

El usuario puede cargar archivos de texto que contienen código Logos y ejecutar el código directamente desde la interfaz gráfica.

Visualización

Las instrucciones que afectan la posición, ángulo, y color de la "Rana" generan líneas que se dibujan en el PanelDeDibujo.

La Configuración mantiene el estado actual del dibujo, incluyendo todas las líneas trazadas y la posición de la "Rana".

Conclusión

Este proyecto de compilador e intérprete para el lenguaje Logos es una excelente demostración de los conceptos fundamentales de los compiladores. Desde el análisis léxico y sintáctico hasta la ejecución de instrucciones en una máquina de pila, el código muestra cómo se pueden combinar varias técnicas y estructuras de datos para crear una aplicación completa y funcional.

El uso de la interfaz gráfica facilita la interacción del usuario y hace que la programación sea accesible y visualmente intuitiva, especialmente para los principiantes en programación. Este enfoque gráfico no solo enseña conceptos de programación, sino que también introduce a los usuarios en el mundo de los compiladores y la ejecución de código.

“ GRACIAS POR LEER Y UTILIZAR NUESTRO PROGRAMA ;) ”