

Proyecto Final: Fase 1



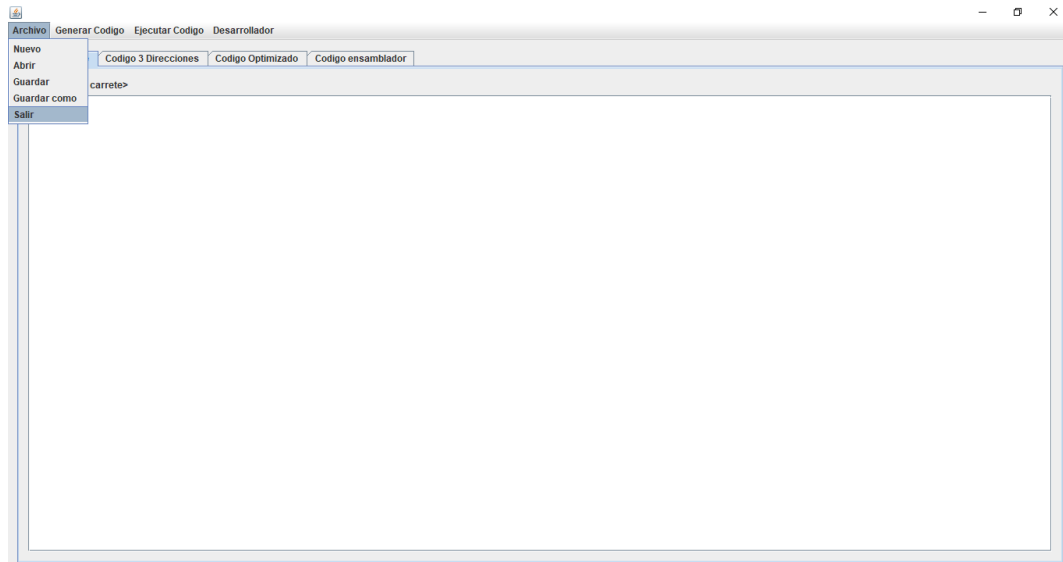
MANUAL DE USUARIO

- ¿Qué es?

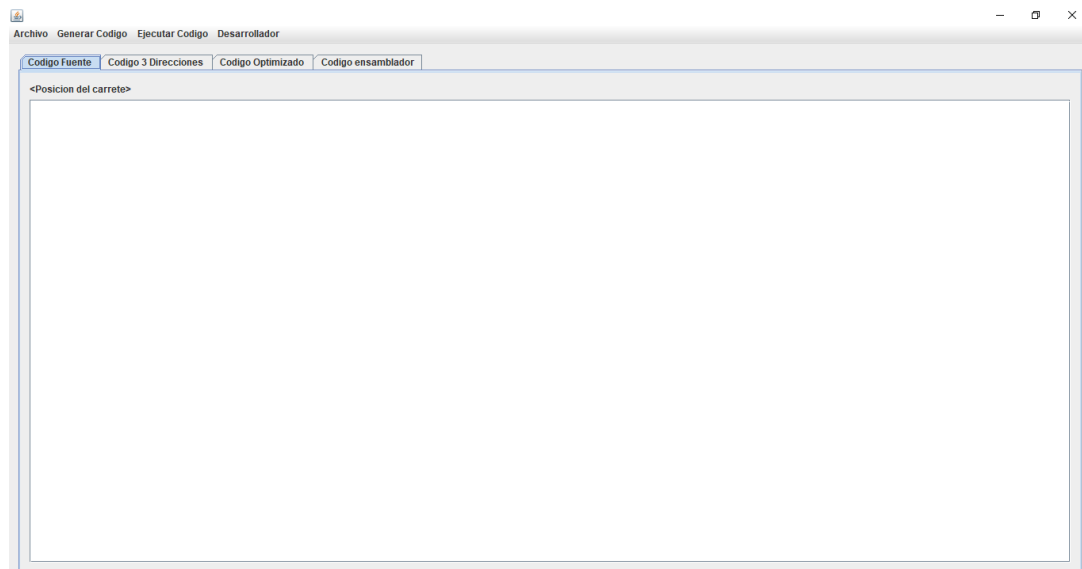
El proyecto es un generador de código intermedio en base a una gramática específica, esta se compone por 4 secciones claramente diferenciadas por las sintaxis de diferentes lenguajes, entre los cuales se encuentran: Visual Basic, Python, Java y C.

- ¿Cómo funciona?

Consta de un editor de texto para poder crear y modificar un archivo con extensión .mlg, el cual será analizado léxica, sintáctica y semánticamente para poder verificar que el mismo se encuentre libre de errores.

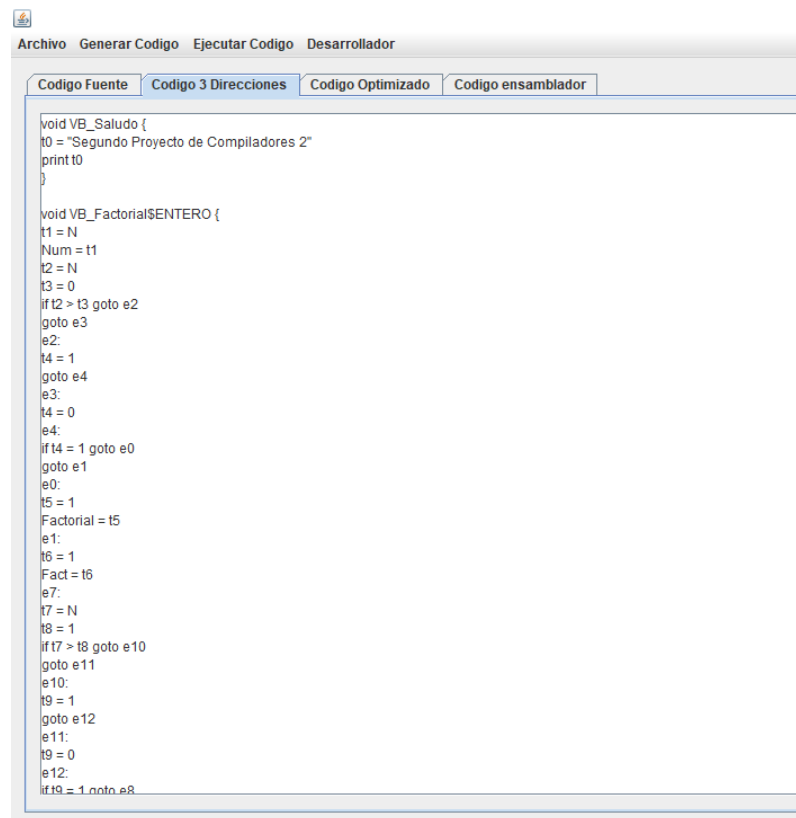


Una vez terminado esto se generará el código 3 direcciones equivalente a todo lo que se haya escrito en cualquiera de los otros 4 lenguajes.



- Funcionalidades
 - Nuevo: permite al usuario crear un archivo nuevo desde 0.
 - Abrir: permite al usuario acceder a un archivo que este posea con anterioridad y abrirlo en el editor de texto brindado.
 - Guardar: permite al usuario guardar los cambios realizados en el archivo que se encuentre actualmente abierto.
 - Guardar como: permite al usuario guardar los cambios realizados en un archivo distinto, para no modificar el original.
 - Salir: cierra la aplicación.
 - El editor de texto cuenta con todas las funcionalidades de hacer y rehacer, cortar, pegar, copiar.

- Sección de código 3 direcciones



The screenshot shows a compiler window with a menu bar (Archivo, Generar Código, Ejecutar Código, Desarrollador) and a tabbed interface. The 'Codigo 3 Direcciones' tab is active, displaying assembly code for two functions: VB_Saludo and VB_Factorial\$ENTERO. The code uses 3-address instructions with labels (e0-e12) for control flow.

```

void VB_Saludo {
t0 = "Segundo Proyecto de Compiladores 2"
print t0
}

void VB_Factorial$ENTERO {
t1 = N
Num = t1
t2 = N
t3 = 0
if t2 > t3 goto e2
goto e3
e2:
t4 = 1
goto e4
e3:
t4 = 0
e4:
if t4 = 1 goto e0
goto e1
e0:
t5 = 1
Factorial = t5
e1:
t6 = 1
Fact = t6
e7:
t7 = N
t8 = 1
if t7 > t8 goto e10
goto e11
e10:
t9 = 1
goto e12
e11:
t9 = 0
e12:
if t9 = 1 goto e8

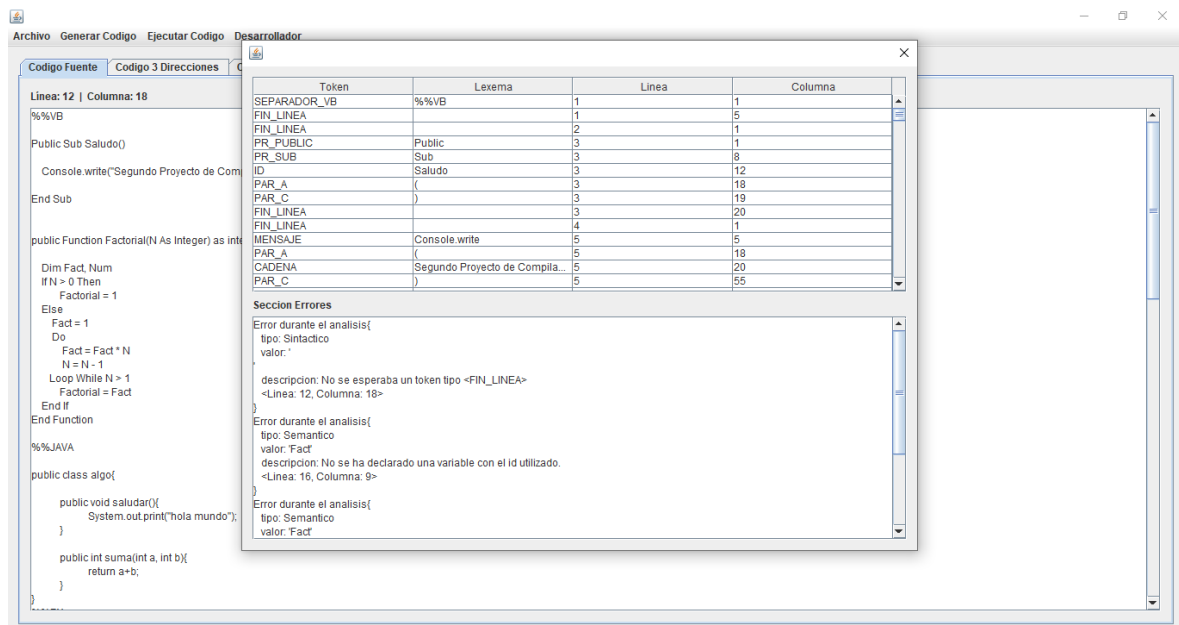
```

- Sección de errores

Esta consta de dos divisiones una se encarga de listar todos los caracteres encontrados, y que tokens representan, teniendo una lista de las entradas léxicas que ha reconocido el analizador léxico.

En la segunda división encontramos un área de texto, en la cual se anotarán todos los errores que se vayan encontrando en el archivo de entrada, errores léxicos, sintácticos y semánticos.

A tener en cuenta que si se encuentran errores no se generará ningún tipo de código 3 direcciones.



- Estructura del archivo .mlg

Sección de código para Vb, inicia con el separador %% VB y finaliza con %%JAVA

Sección de código para Java inicia con el separador %%JAVA y finaliza con %%PY

Sección de código para Python inicia con el separador %%PY y finaliza con %%PROGRAMA

Sección de código para el programa principal inicia con el separador %%PROGRAMA

Cada sección acepta la sintaxis del lenguaje que “representa”, donde el programa principal utiliza la sintaxis del lenguaje de programación C.

*Nota: para poder usar el código declarado en las demás secciones dentro del programa principal es necesario importar las librerías #include “<libreriaQueDeseaImportar>”

- Limitantes de los lenguajes de programación
 - Expresiones Aritmeticas +,-,*,/,%
 - Mensajes a pantalla
 - Solicitud de datos
 - Ciclos
 - Manejo de condiciones
 - Declaracion de variables
 - Asignación de variables
 - Declaración de Funciones
 - Declaracion de Procedimientos

- Extras en el lenguaje de C
 - Llamadas a métodos
 - Creación de objetos
 - Creación de constantes
 - Arreglos multidimensionales

NOTA: Los únicos tipos de datos aceptados son int (enteros), float (flotantes) y char (caracteres). Se recomienda leer las palabras reservadas de cada lenguaje para no tener problemas al codificar en cada uno de los lenguajes que se pueden utilizar.