



USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

PROYECTO 2

MANUAL USUARIO

**Organización de
Lenguajes y
Compiladores 2**

Miguel Adrian Tubac Agustin
202101927

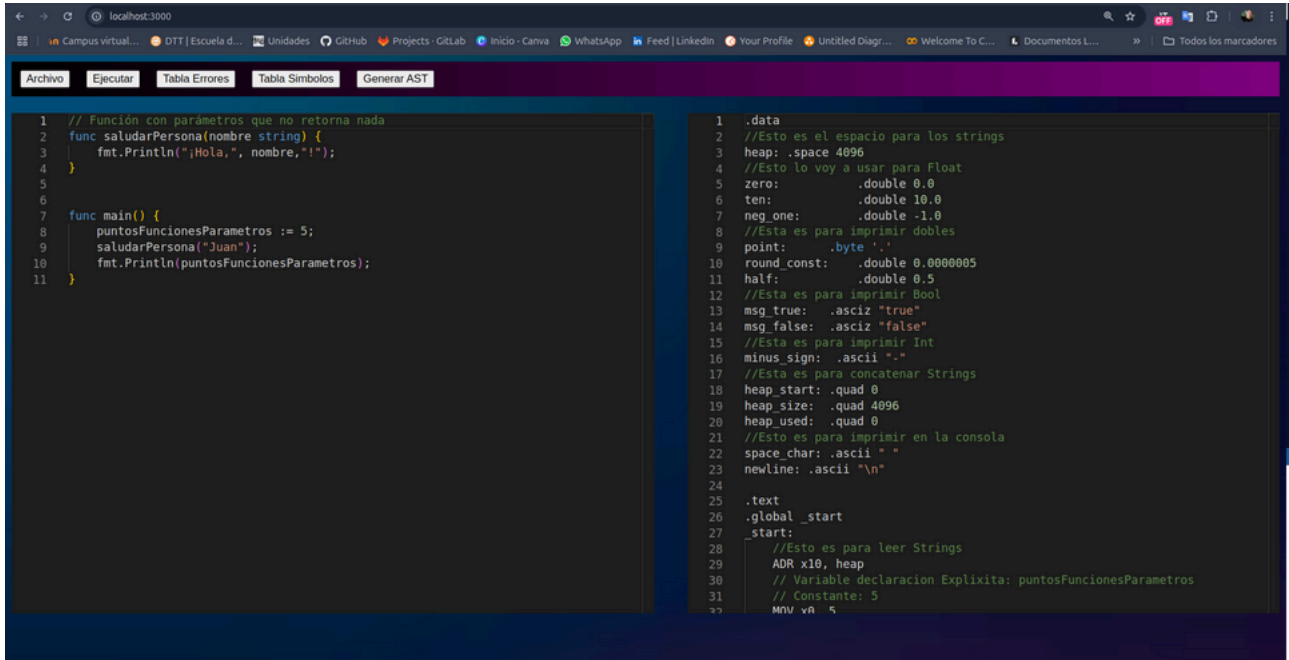
Descripción general del programa



El programa resuelve el problema de un compilador que realiza la generación de código ensamblador ARM64 desde un alto nivel, ya que convierte Go a ensamblador. Lo que permitirá comprender el proceso de traducción de un lenguaje de alto nivel a instrucciones de bajo nivel en una arquitectura de conjunto de instrucciones reducido (RISC).

Pasos para el uso del sistema

Entorno principal:

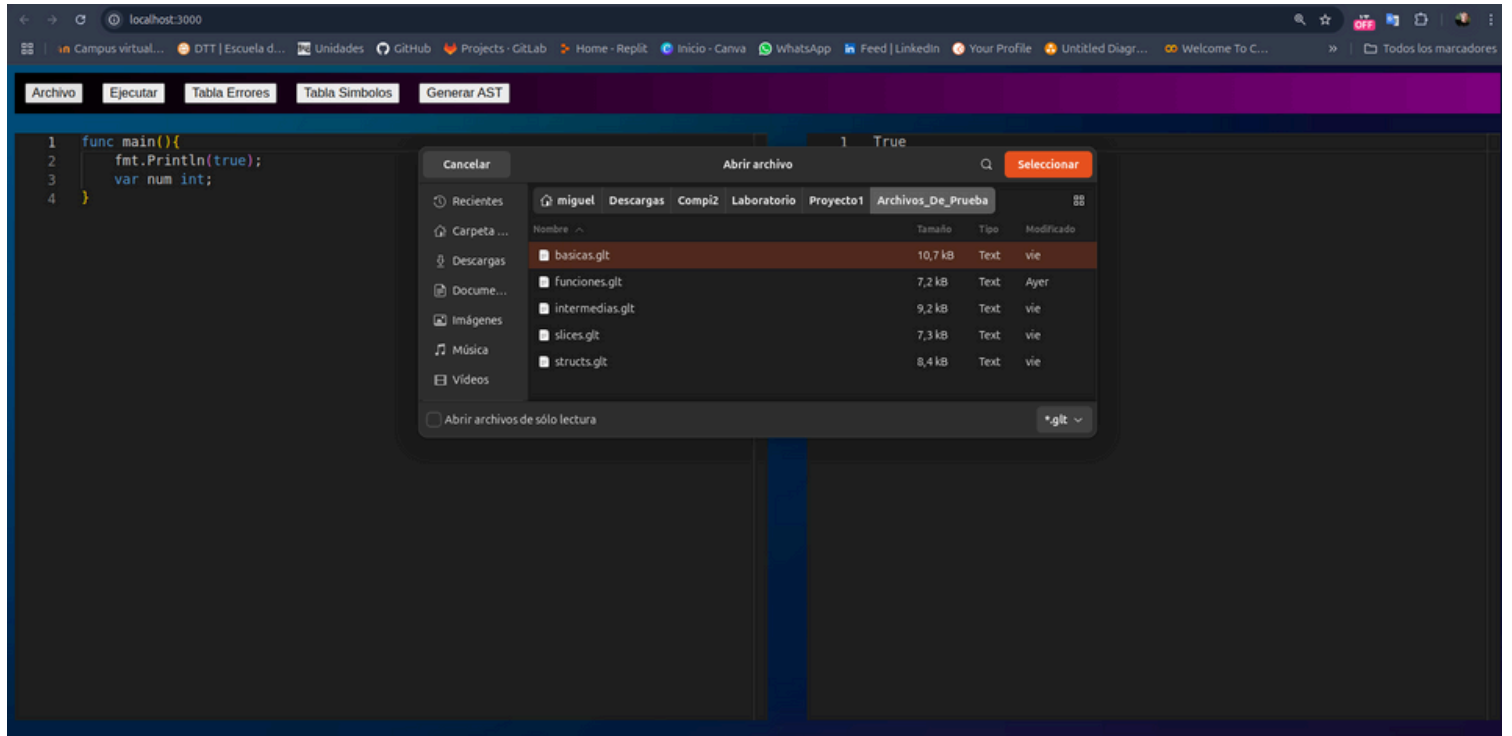


Al iniciar el programa se mostrará la ventana principal con los menús correspondientes a cada entidad del sistema, cada una de las entidades cuenta con las siguientes opciones:

- Archivo
- Ejecutar
- Tabla Errores
- Tabla Simbolos
- Generar AST

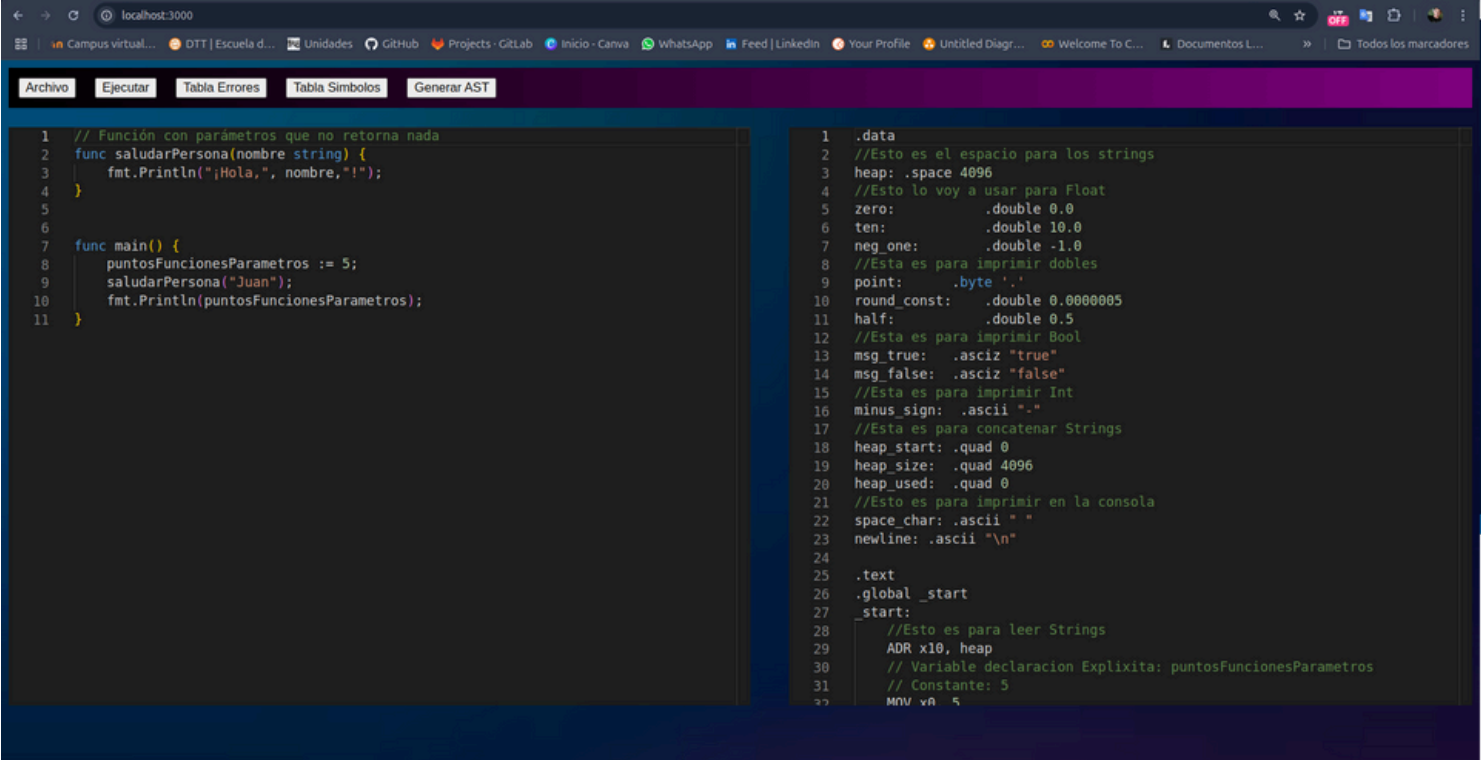
Los botones están disponibles desde el inicio del programa.

Archivo:



Al momento de seleccionar la opción de Archivo este mostrará un explorador de archivos en donde se encuentren los archivos con extensión .glt, posteriormente al seleccionar un archivo este se mostrara en la primera área de texto del sistema.

Ejecutar:



The screenshot shows a web-based IDE interface. The top bar contains navigation links like 'Campus virtual...', 'OTT | Escuela d...', 'Unidades', 'GitHub', 'Projects · GitLab', 'Inicio · Canva', 'WhatsApp', 'Feed | LinkedIn', 'Your Profile', 'Untitled Diagram...', 'Welcome To C...', 'Documentos L...', and 'Todos los marcadores'. Below the top bar is a menu with 'Archivo', 'Ejecutar', 'Tabla Errores', 'Tabla Simbolos', and 'Generar AST'. The main area is split into two panels. The left panel shows Go code:

```
1 // Función con parámetros que no retorna nada
2 func saludarPersona(nombre string) {
3     fmt.Println("¡Hola,", nombre, "!");
4 }
5
6
7 func main() {
8     puntosFuncionesParametros := 5;
9     saludarPersona("Juan");
10    fmt.Println(puntosFuncionesParametros);
11 }
```

 The right panel shows assembly code:

```
1 .data
2 //Esto es el espacio para los strings
3 heap: .space 4096
4 //Esto lo voy a usar para Float
5 zero: .double 0.0
6 ten: .double 10.0
7 neg_one: .double -1.0
8 //Esta es para imprimir dobles
9 point: .byte '.'
10 round_const: .double 0.000005
11 half: .double 0.5
12 //Esta es para imprimir Bool
13 msg_true: .asciz "true"
14 msg_false: .asciz "false"
15 //Esta es para imprimir Int
16 minus_sign: .ascii "-"
17 //Esta es para concatenar Strings
18 heap_start: .quad 0
19 heap_size: .quad 4096
20 heap_used: .quad 0
21 //Esta es para imprimir en la consola
22 space_char: .ascii " "
23 newline: .ascii "\n"
24
25 .text
26 .global _start
27 _start:
28     //Esto es para leer Strings
29     ADR x10, heap
30     // Variable declaracion Explixita: puntosFuncionesParametros
31     // Constante: 5
32     MOV vA, 5
```

Al momento de seleccionar la opción de Ejecutar, el programa enviará la información al backend en la cual posteriormente este devolverá el resultado. El resultado se mostrará en la segunda área de texto, cabe resaltar que en esta segunda área de texto no será posible editar.

Tabla Errores:

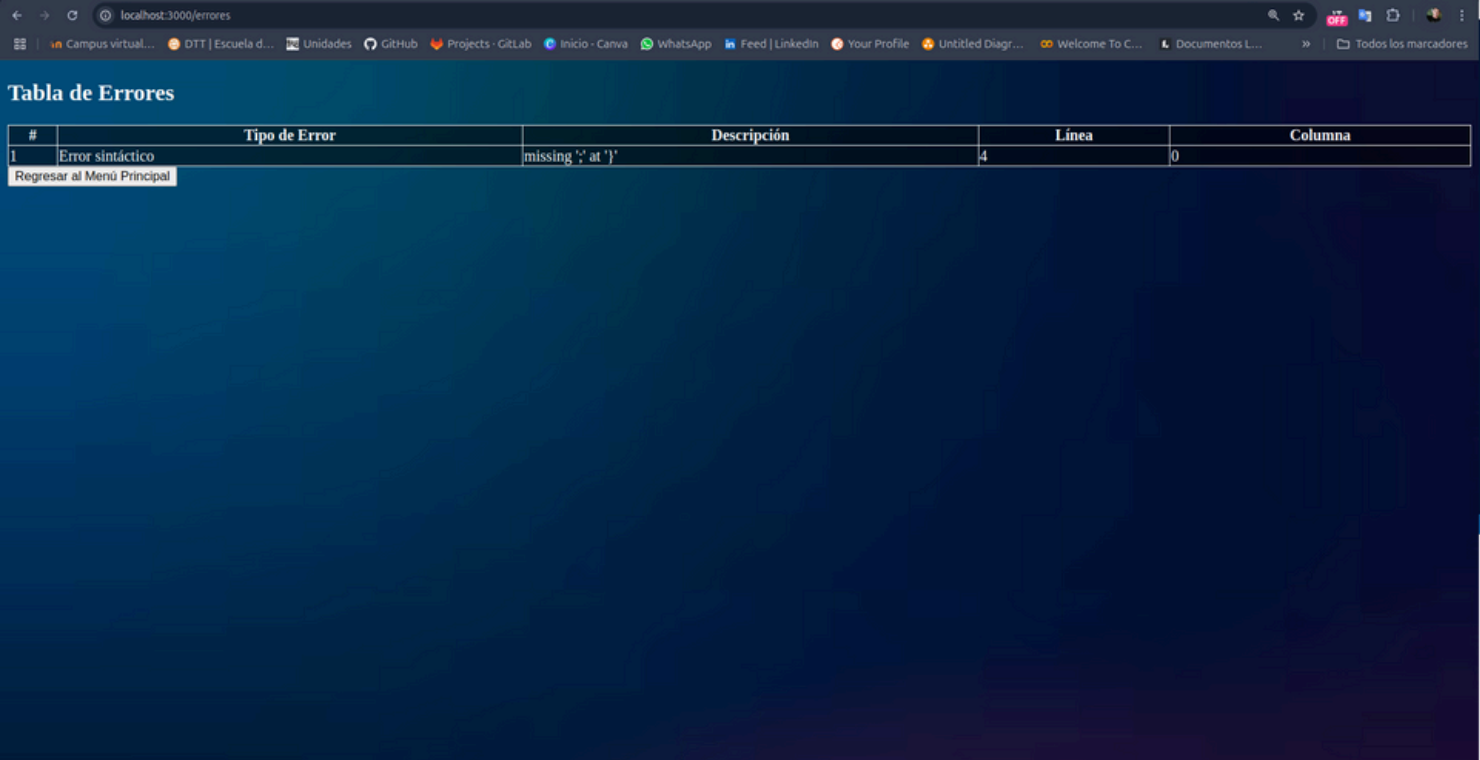


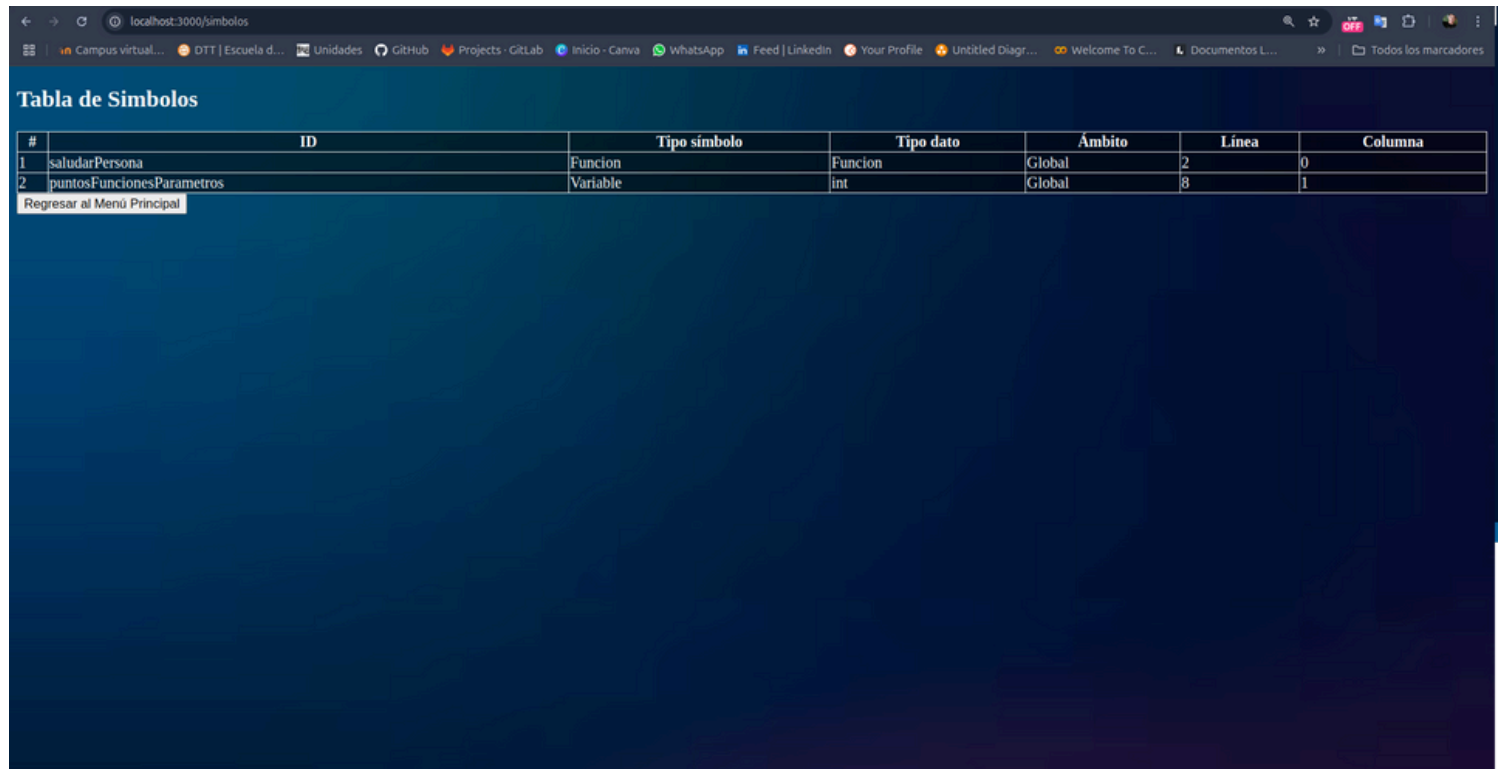
Tabla de Errores

#	Tipo de Error	Descripción	Línea	Columna
1	Error sintáctico	missing ':' at ''	4	0

[Regresar al Menú Principal](#)

Este es el ejemplo, se muestra un error que el sistema detecto, en el cual se muestra a detalle lo que capturo. También muestra la línea y columna en donde se encuentra el error. Esto es de bastante ayuda al momento de desarrollar un sistema de gran envergadura.

Tabla Simbolos:

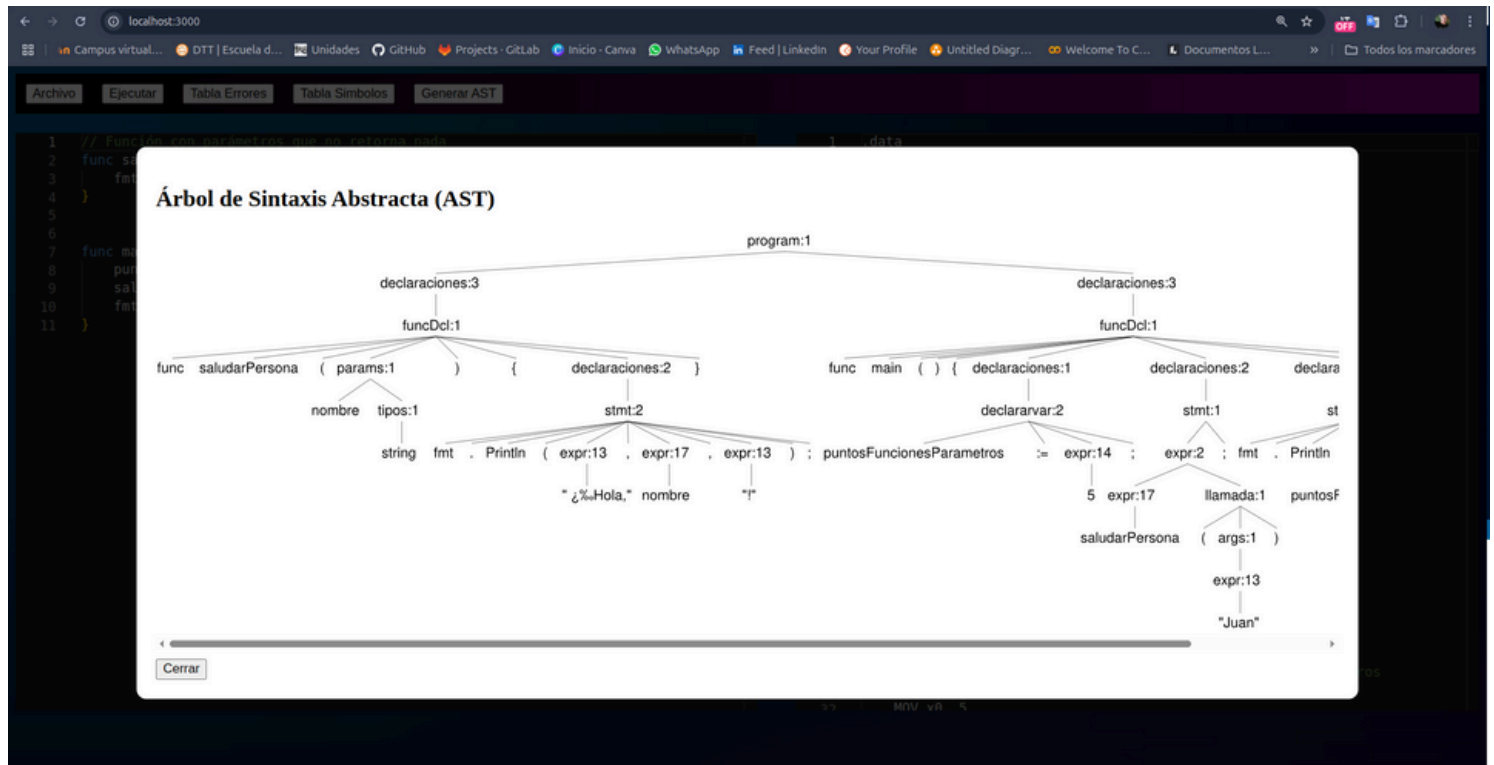


#	ID	Tipo símbolo	Tipo dato	Ámbito	Línea	Columna
1	saludarPersona	Funcion	Funcion	Global	2	0
2	puntosFuncionesParametros	Variable	int	Global	8	1

[Regresar al Menú Principal](#)

Este es el ejemplo de una lista de símbolos que se detectaron en la ejecución de un programa, en la misma lista se describe con claridad el tipo de variable y en que ámbito fue declarado, también se muestra en que línea y en que columna este fue declarado o reasignado.

Generar AST:



Este es el ejemplo de un AST de un programa de prueba en el cual se muestra con detalle los niveles que se aplicaron en la gramática. Cabe mencionar que el árbol es de gran tamaño, por lo que es necesario desplazarnos dentro de la ventana para visualizar el árbol completo.