

Universidad de San Carlos de Guatemala  
Organización de lenguajes y compiladores 1  
Ing: Kevin Lajpop  
Aux: Sandra Jiménez

# **MANUAL DE USUARIO**

## **-SYSCOMPILER-**

Nombre: Osmar Abdel Peña Santizo

Carnet: 201801619

Guatemala, 2 de noviembre de 2021

# CONTENIDO

INTRODUCCION .....	3
DESCRIPCION DEL SOFTWARE .....	4
AQUITECTURA GENERAL DEL SOFTWARE.....	4
ENTORNO PRINCIPAL DE LA WEAPP.....	4
PANTALLA PRINCIPAL .....	4
FUNCIONES NATIVAS DEL LENGUAJE.....	5
METODOS Y FUNCIONES .....	6
SENTENCIAS DE TRANSFERENCIA .....	6
SENTENCIAS CONDICIONALES Y BUCLES.....	6
TIPOS DE DATOS.....	7
OPERACIONES ARITMETICAS.....	7
ANEXOS .....	8

## **INTRODUCCION**

En el curso de organización de Lenguajes y Compiladores 1, se solicitó crear un nuevo Lenguaje de programación, y así aplicar los conocimientos sobre la fase de análisis léxico y sintáctico de un compilador. Con esto lograr hacer un intérprete sencillo.

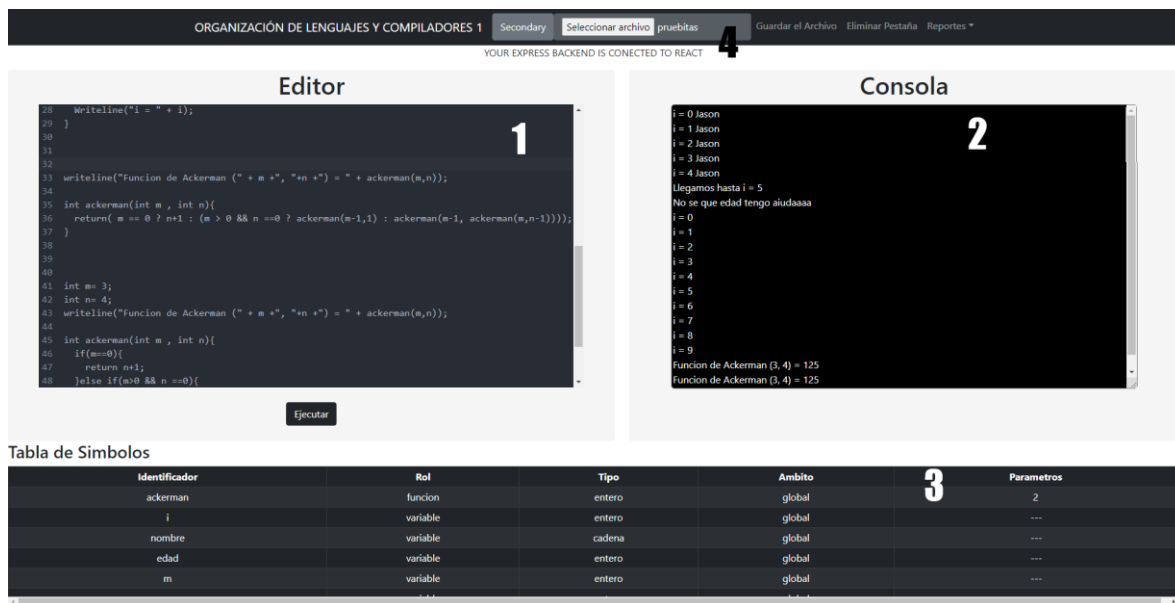
## Descripción del software

El curso de Organización de Lenguajes y Compiladores 1, ha puesto en marcha un nuevo proyecto, requerido por la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería, que consiste en crear un lenguaje de programación para que los estudiantes, del curso de Introducción a la Programación y Computación 1, aprendan a programar y tener conocimiento de todas las generalidades de un lenguaje de programación. Cabe destacar, que este lenguaje será utilizado para generar sus primeras prácticas de laboratorio del curso antes mencionado. Por lo tanto, a usted, que es estudiante del curso de Compiladores 1, se le encomienda realizar el proyecto llamado SysCompiler, dado sus altos conocimientos en temas de análisis léxico, sintáctico y semántico.

## Arquitectura General del software

El software tendrá una arquitectura Cliente-Servidor, el cual tendrá la funcionalidad de compilar un lenguaje de programación el cual su resultado se podrá visualizar en una consola. Se podrán cargar archivos hacia el editor o escribir manualmente para luego ejecutarlo.

## Entorno Principal de la web app



ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1 Secondary Seleccionar archivo pruebas Guardar el Archivo Eliminar Pestalla Reportes \*

YOUR EXPRESS BACKEND IS CONNECTED TO REACT 4

**Editor** 1

```
28 WriteLine("i = " + i);
29 }
30
31
32
33 WriteLine("Funcion de Ackerman (" + m + ", " + n + ") = " + ackerman(m,n));
34
35 int ackerman(int m , int n){
36     return( m == 0 ? n+1 : (m > 0 && n ==0 ? ackerman(m-1,1) : ackerman(m-1, ackerman(m,n-1))));
37 }
38
39
40
41 int m= 3;
42 int n= 4;
43 WriteLine("Funcion de Ackerman (" + m + ", " + n + ") = " + ackerman(m,n));
44
45 int ackerman(int m , int n){
46     if(m==0){
47         return n+1;
48     }else if(m>0 && n ==0){
```

Ejecutar

**Consola** 2

```
i = 0 Jason
i = 1 Jason
i = 2 Jason
i = 3 Jason
i = 4 Jason
Llegamos hasta i = 5
No se que edad tengo aludaaa
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
Funcion de Ackerman (3, 4) = 125
Funcion de Ackerman (3, 4) = 125
```

**Tabla de Simbolos** 3

Identificador	Rol	Tipo	Ambito	Parametros
ackerman	funcion	entero	global	2
i	variable	entero	global	---
nombre	variable	cadena	global	---
edad	variable	entero	global	---
m	variable	entero	global	---

## Pantalla Principal

1. Editor: En este componente se podrá observar el programa fuente del programa que se va a compilar. Así como crear programas desde cero, o subiendo un archivo.
2. Consola: En este componente se podrá visualizar la salida de la ejecución del código.

3. Menú de opciones En este componente se podrán visualizar varias opciones que dispone el programa, las cuales son:
- a. Seleccionar archivo: En este componente se podrá cargar un archivo con extensión del lenguaje de programación dado (.sc), para ser visualizado en el editor
  - b. Reportes
    - i. Reporte AST: Esta opción generará un reporte de las acciones realizadas en una imagen.

## SINTAXIS DEL LENGUAJE DE PROGRAMACION

Como es primera vez que se usará el lenguaje de programación se dará un listado de palabras reservadas que se utilizan en dicho lenguaje de programación

### Funciones nativas del lenguaje

Palabra Reservada	Descripción
Start With	Ejecuta el método seleccionado
Writeline	Imprimirá en consola el contenido dentro de esta función
toLowerCase	Esta función recibe como parámetro una expresión de tipo cadena y retorna una nueva cadena con todas las letras minúsculas
toUpperCase	Esta función recibe como parámetro una expresión de tipo cadena retorna una nueva cadena con todas las letras mayúsculas.
Length	Esta función recibe como parámetro un vector, una lista o una cadena y devuelve el tamaño de este.
Truncate	Esta función recibe como parámetro un valor numérico. Permite eliminar los decimales de un número, retornando un entero
Round	Esta función recibe como parámetro un valor numérico. Permite redondear los números decimales según las siguientes reglas: <ul style="list-style-type: none"><li>- Si el decimal es mayor o igual que 0.5, se aproxima al número superior</li><li>- Si el decimal es menor que 0.5, se aproxima al número inferior</li></ul>
Typeof	Esta función retorna una cadena con el nombre del tipo de dato evaluado.
ToString	Esta función permite convertir un valor de tipo numérico o booleano en texto.

toCharArray	Esta función permite convertir una cadena en un arreglo de caracteres
-------------	---

## Métodos y funciones

Palabra Reservada	Descripción
Void	Declaración e inicio de un Método
Return	La sentencia return finaliza la ejecución de un método o función y puede especificar un valor para ser devuelto a quien llama a la función.

## Sentencias de transferencia

Palabra Reservada	Descripción
Break	La sentencia break hace que se salga del ciclo inmediatamente, es decir que el código que se encuentre después del break en la misma iteración no se ejecutara y este se saldrá del ciclo.
Continue	La sentencia continue puede detener la ejecución de la iteración actual y saltar a la siguiente.

## Sentencias condicionales y bucles

Palabra Reservada	Descripción
Do-While	El ciclo o bucle Do-While, es una sentencia que ejecuta al menos una vez el conjunto de instrucciones que se encuentran dentro de ella y que se sigue ejecutando mientras la condición sea verdadera.
for	El ciclo o bucle for, es una sentencia que nos permite ejecutar N cantidad de veces la secuencia de instrucciones que se encuentra dentro de ella.
while	El ciclo o bucle While, es una sentencia que ejecuta una secuencia de instrucciones mientras la condición de ejecución se mantenga verdadera.
switch	Estructura principal del switch, donde se indica la expresión a evaluar.
case	Estructura que contiene las diversas opciones a evaluar con la expresión establecida en el switch.
Default	Estructura que contiene las sentencias si en dado caso no haya salido del switch por medio de una sentencia break.

if	La sentencia if ejecuta las instrucciones sólo si se cumple una condición. Si la condición es falsa, se omiten las sentencias dentro de la sentencia
else	Sentencia que se ejecutara si la condición es falsa.

## Tipos de datos

Tipo de dato	Palabra Reservada	Descripción
Entero	Int	Este tipo de datos aceptará solamente números enteros
Doble	Double	Admite valores numéricos con decimales.
Booleano	Boolean	Admite valores que indican verdadero o falso.
Carácter	Char	Tipo de dato que únicamente aceptará un único carácter, y estará delimitado por comillas simples. '
Cadena	String	Es un grupo o conjunto de caracteres que pueden tener cualquier carácter, y este se encontrará delimitado por comillas dobles. " "

## Operaciones Aritmeticas

Operación	Signo	Descripción
Suma	+	Es la operación aritmética que consiste en realizar la suma entre dos o más valores.
Resta	-	Es la operación aritmética que consiste en realizar la resta entre dos o más valores
Multiplicación	*	Operación aritmética que consiste en sumar un número (multiplicando) tantas veces como indica otro número (multiplicador).
División	/	Operación aritmética que consiste en partir un todo en varias partes, al todo se le conoce como dividendo, al total de partes se le llama divisor y el resultado recibe el nombre de cociente.
Potencia	^	Es una operación aritmética de la forma $a^b$ donde a es el valor de la base y b es el valor del exponente que nos indicará cuantas veces queremos multiplicar el mismo número.
Modulo	%	Es una operación aritmética que obtiene el resto de la división de un numero entre otro.
Negación Unaria	-	Es una operación que niega el valor de un número, es decir que devuelve el contrario del valor original.

# ANEXOS

Editor:

Editor

```
34
35 int ackerman(int m , int n){
36     return( m == 0 ? n+1 : (m > 0 && n ==0 ? ackerman(m-1,1) : ackerman(m-1, ackerman(m,n-1))));
37 }
38
39
40
41 int m= 3;
42 int n= 4;
43 writeline("Funcion de Ackerman (" + m +", "+n +") = " + ackerman(m,n));
44
45 int ackerman(int m , int n){
46     if(m==0){
47         return n+1;
48     }else if(m>0 && n ==0){
49         return ackerman(m-1,1);
50     }else{
51         return ackerman(m-1,ackerman(m,n-1));
52     }
53 }
54 }
```

Ejecutar

Consola

Consola

```
i = 0 Jason
i = 1 Jason
i = 2 Jason
i = 3 Jason
i = 4 Jason
Llegamos hasta i = 5
No se que edad tengo aiudaaaa
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
Funcion de Ackerman (3, 4) = 125
Funcion de Ackerman (3, 4) = 125
```



## Tabla de símbolos:

Tabla de Símbolos

Identificador	Rol	Tipo	Ambito	Parametros
ackerman	funcion	entero	global	2
i	variable	entero	global	---
nombre	variable	cadena	global	---
edad	variable	entero	global	---
m	variable	entero	global	---
n	variable	entero	global	---
ackerman	funcion	entero	Local	2
i	variable	entero	Local	---
nombre	variable	cadena	Local	---
edad	variable	entero	Local	---
m	variable	entero	Local	---
n	variable	entero	Local	---
i	variable	entero	Local	---
ackerman	funcion	entero	Local	2
i	variable	entero	Local	---
nombre	variable	cadena	Local	---
edad	variable	entero	Local	---
m	variable	entero	Local	---
n	variable	entero	Local	---

## Arbol AST:

### Arbol AST

