Manual De Usuario

Proyecto 2

Kevin Steve Martinez Lemus
[OLC1] Universidad De San Carlos De Guatemala

Índice

I. Introducción

Objetivos

Objetivo General

- Aplicar los conocimientos sobre la fase de análisis léxico y sintáctico de un compilador para la realización de un intérprete sencillo, con las funcionalidades principales para que sea funcional.

Objetivos Específicos

- Reforzar los conocimientos de análisis léxico y sintáctico para la creación de un lenguaje de programación.
- Aplicar los conceptos de compiladores para implementar el proceso de interpretación de código de alto nivel.
- Aplicar los conceptos de compiladores para analizar un lenguaje de programación y producir las salidas esperadas.
- Aplicar la teoría de compiladores para la creación de soluciones de software.
- Aplicar conceptos de contenedores para generar aplicaciones livianas.
- Generar aplicaciones utilizando arquitecturas Cliente-Servidor.

Requerimientos

- Tener un navegador.
- Se recomienda tener un editor de código (Visual Studio Code u otro) para ver y editar el código de este programa.

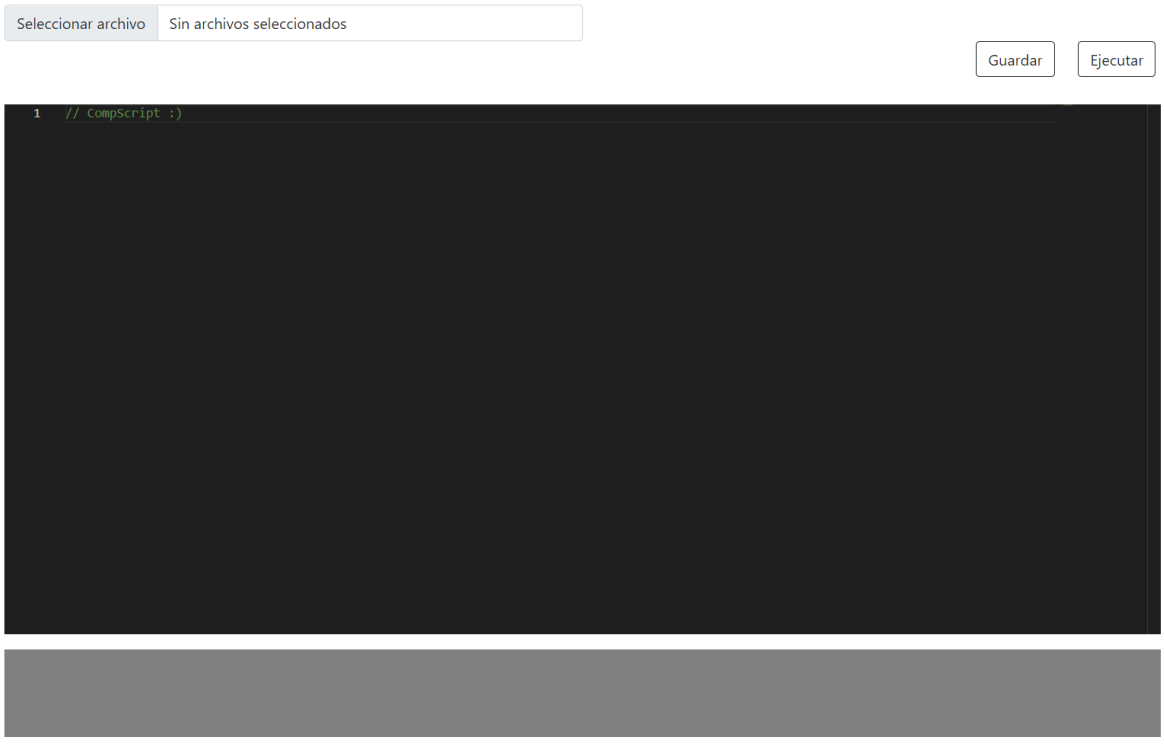
II. Opciones Del Sistema

El presente Manual está organizado de acuerdo con la secuencia de ingreso a las pantallas del sistema de la siguiente manera:

1. Ingreso al Sistema.
2. Archivo.
3. Lenguaje.
4. Analizar Entradas.
5. Ver Tabla de Símbolos

1. Ingreso al Sistema

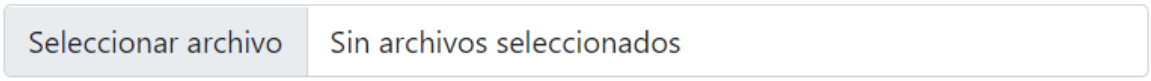
Para poder ejecutar el programa necesitamos abrir una consola nueva, hay que ingresar al directorio donde se encuentra el proyecto y escribir el siguiente comando, para el backend, “node src\index.js” y para el frontend “npm start”; al ejecutar esos comandos se levantará los servidores automáticamente, para empezar a utilizarlo solo es cuestión de entrar al localhost:3000 desde nuestro navegador, al ingresar se mostrará lo siguiente:



En donde se encuentran varios aspectos como: el editor de texto para escribir el código que se desea ejecutar, varias opciones de archivo como abrir y guardar archivos, y demás opciones que se explicaran más adelante.

2. Archivo

Se podrá seleccionar un archivo desde nuestra computadora para empezar a editarlo con la siguiente opción:



Solo se deberá subir el archivo que se desea abrir y listo. También se podrá guardar los archivos que se estén realizando con el botón de guardar:

Esta opción descargará un archivo con todo el código que esté escrito en el editor de texto.

3. Lenguaje

El lenguaje no distinguirá entre mayúsculas o minúsculas. Los comentarios pueden ser:

- De una línea: Estos comentarios deberán comenzar con // y terminar con un salto de línea.
- Multilínea: Estos comentarios deberán comenzar con /* y terminar con */.

```
// Este es un comentario de una línea

/*
        Este es un comentario
        Multilínea
        Para este lenguaje
*/
```

Existen 5 tipos de datos:

TIPO	DEFINICION	DESCRIPCION	EJEMPLO	OBSERVACIONES	DEFAULT
Entero	Int	Este tipo de datos aceptará solamente números enteros.	1, 50, 100, 25552, etc.	Del -2147483648 al 2147483647	0
Doble	Double	Admite valores numéricos con decimales.	1.2, 50.23, 00.34, etc.	Se manejará cualquier cantidad de decimales	0.0
Booleano	Boolean	Admite valores que indican verdadero o falso.	True, false	Si se asigna un valor booleano a un entero se tomará como 1 o 0 respectivamente.	True
Caracter	Char	Tipo de dato que únicamente aceptará un único carácter, y estará delimitado por comillas simples. ' '	'a', 'b', 'c', 'E', 'Z', '1', '2', '^', '%', ')', '=', '!', '&', '/', '\\', '\n', etc.	En el caso de querer escribir comilla simple escribir se escribirá \ y después comilla simple \', si se quiere escribir \ se escribirá dos veces \\, existirá también \n, \t, \r, \".	'\u0000' (carácter 0)
Cadena	String	Es un grupo o conjunto de caracteres que pueden tener cualquier carácter, y este se encontrará delimitado por comillas dobles. " "	"cadena1", "-- ** cadena 1"	Se permitirá cualquier carácter entre las comillas dobles, incluyendo las secuencias de escape: \" comilla doble \\ barra invertida \n salto de linea \r retorno de carro \t tabulación	"" (string vacío)

Los operadores aritméticos son la suma (+), resta (-), multiplicación (*), división (/), potenciación(^), modulo (%) y negación unaria(-). A continuación se muestran las tablas de resultados de cada una de ellas.

+	Entero	Doble	Boolean	Caracter	Cadena
Entero	Entero	Doble	Entero	Entero	Cadena
Doble	Doble	Doble	Doble	Double	Cadena
Boolean	Entero	Doble			Cadena
Caracter	Entero	Doble		Cadena	Cadena
Cadena	Cadena	Cadena	Cadena	Cadena	Cadena

-	Entero	Doble	Boolean	Caracter	Cadena
Entero	Entero	Doble	Entero	Entero	
Doble	Doble	Doble	Doble	Doble	
Boolean	Entero	Doble			
Caracter	Entero	Doble			
Cadena					

*	Entero	Doble	Boolean	Caracter	Cadena
Entero	Entero	Doble		Entero	
Doble	Doble	Doble		Doble	
Boolean					
Caracter	Entero	Doble			
Cadena					

/	Entero	Doble	Boolean	Caracter	Cadena
Entero	Doble	Doble		Doble	
Doble	Doble	Doble		Doble	
Boolean					
Caracter	Doble	Doble			
Cadena					

^	Entero	Doble	Boolean	Caracter	Cadena
Entero	Entero	Doble			
Doble	Doble	Doble			
Boolean					
Caracter					
Cadena					

%	Entero	Doble	Boolean	Caracter	Cadena
Entero	Doble	Doble			
Doble	Doble	Doble			
Boolean					
Caracter					
Cadena					

-num	Resultado
Entero	Entero
Doble	Doble
Boolean	
Caracter	
Cadena	

También existen los operadores relacionales los cuales devuelven un booleano de la siguiente manera:

OPERADOR	DESCRIPCIÓN	EJEMPLO
==	Igualación: Compara ambos valores y verifica si son iguales: - Iguales= True - No iguales= False	1 == 1 "hola" == "hola" 25.5933 == 90.8883 25.5 == 20
!=	Diferenciación: Compara ambos lados y verifica si son distintos. - Iguales= False - No iguales= True	1 != 2, var1 != var2 25.5 != 20 50 != 'F' "hola" != "hola"
<	Menor que: Compara ambos lados y verifica si el derecho es mayor que el izquierdo. - Derecho mayor= True - Izquierdo mayor= False	(5/(5+5))<(8*8) 25.5 < 20 25.5 < 20 50 < 'F'
<=	Menor o igual que: Compara ambos lados y verifica si el derecho es mayor o igual que el izquierdo. - Derecho mayor o igual= True - Izquierdo mayor= False	55+66<=44 25.5 <= 20 25.5 <= 20 50 <= 'F'
>	Mayor que: Compara ambos lados y verifica si el izquierdo es mayor que el derecho. - Derecho mayor= False - Izquierdo mayor= True	(5+5.5)>8.98 25.5 > 20 25.5 > 20 50 > 'F'
>=	Mayor o igual que: Compara ambos lados y verifica si el izquierdo es mayor o igual que el derecho. - Derecho menor o igual= True - Izquierdo menor= False	5-6>=4+6 25.5 >= 20 25.5 >= 20 50 >= 'F'

Se puede utilizar la operación de operador ternario de la siguiente manera:

```
<CONDICION> '?' <EXPRESION> ':' <EXPRESION>

//Ejemplo del uso del operador ternario

int edad = 18;
boolean banderaedad = false;
banderaedad = edad > 17 ? true : false;
```

Los operadores lógicos son los siguientes:

OPERADOR	DESCRIPCIÓN	EJEMPLO	OBSERVACIONES
||	OR: Compara expresiones lógicas y si al menos una es verdadera entonces devuelve verdadero en otro caso retorna falso	(55.5)|| bandera==true Devuelve true	bandera es true
&&	**AND**: Compara expresiones lógicas y si son ambas verdaderas entonces devuelve verdadero en otro caso retorna falso	(flag1) && ("hola" == "hola") Devuelve true	flag1 es true
!	NOT: Devuelve el valor inverso de una expresión lógica si esta es verdadera entonces devolverá falso, de lo contrario retorna verdadero.	!var1 Devuelve falso	var1 es true

Toda instrucción se deberá finalizar con punto y coma (;) y todas las sentencias deben de ir encapsuladas con llaves ({,}). Las variables se declaran de la siguiente manera:

```
<TIPO> identificador;
<TIPO> id1, id2, id3, id4;
<TIPO> identificador = <EXPRESION>;
<TIPO> id1, id2, id3, id4 = <EXPRESION>;
//Ejemplos
int numero;
int var1, var2, var3;
string cadena = “hola”;
char var_1 = ‘a’;
boolean verdadero;
boolean flag1, flag2, flag3 = true;
char ch1, ch2, ch3 = ‘R’;
```

Se pueden realizar casteos:

```
‘(‘<TIPO>’)’ <EXPRESION>
 //Ejemplos
int edad = (int) 18.6; //toma el valor entero de 18
char letra = (char) 70; //tomar el valor ‘F’ ya que el 70 en ascii es F
double numero = (double) 16; //toma el valor 16.0
```

Se puede incrementar y decrementar

```
<EXPRESION>'+''+';'
<EXPRESION> '-''-';'
 //Ejemplos
int edad = 18;
edad++; //tiene el valor de 19
edad--; //tiene el valor 18

int anio=2020;
anio = 1 + anio++; //obtiene el valor de 2022
anio = anio--;  //obtiene el valor de 2021
```

Las sentencias de control pueden ser: La sentencia IF:

```
'if' '(' [<EXPRESION>] ')' '{'
        [<INSTRUCCIONES>]
 '}'
| 'if' '(' [<EXPRESION>] ')' '{'
        [<INSTRUCCIONES>]
  '}' 'else' '{'
        [<INSTRUCCIONES>]
  '}'
| 'if' '(' [<EXPRESION>] ')' '{'
        [<INSTRUCCIONES>]
  '}' 'else' [<IF>]


//Ejemplo de cómo se implementar un ciclo if

if (x <50)
 {
    Println("Menor que 50");
    //Más sentencias
}
```

If else:

```
//Ejemplo de cómo se implementar un ciclo if-else

if (x <50)
 {
    Println("Menor que 50");
    //Más sentencias
}
else
{
    Println("Mayor que 100");
    //Más sentencias
}
```

Las sentencias cíclicas se dividen en 3: while, for y do while.

```
'while' '('[<EXPRESION>] ')' '{'
        [<INSTRUCCIONES>]
 '}'

//Ejemplo de cómo se implementar un ciclo while

while (x<100){
        if (x > 50)
        {
                Print("Mayor que 50");
                //Más sentencias
        }
        else
        {
                Print("Menor que 100");
                //Más sentencias
        }
        X++;
        //Más sentencias

 }
```

```
'for '(' ([<DECLARACION>|<ASIGNACION>])';' [<CONDICION>]';' [<[ACTUALIZACION>] ')' '{'
      [<INSTRUCCIONES>]
  '}'

                //Ejemplo 1: declaración dentro del for con incremento
                for ( int i=0; i<3;i++ ){
                        Println("i="+i)
                        //más sentencias
                }
               /*RESULTADO
                  i=0
                  i=1
                  i=2
                */

//Ejemplo 2: asignación de variable previamente declarada y decremento por asignación
                for ( i=5; i>2;i=i-1 ){
                        Print("i="+i+"\n")
                        //más sentencias
                }
               /*RESULTADO
                  i=5
                  i=4
                  i=3
                */
```

```
  'do' '{'
        [<INSTRUCCIONES>]
   '}' 'while' '('[<EXPRESION>] ')' ';'

  //Ejemplo de cómo se implementar un ciclo do-while
 Int a=5;
 Do{
      If (a>=1 && a <3){
          Println(true)
      }
      Else{
          Println(false)
      }
```

Se puede parar los ciclos con break y continue:

```
break;
//Ejemplo en un ciclo for
for(int i = 0; i < 9; i++){
    if(i==5){
        Println("Me salgo del ciclo en el numero " + i);
        break;
    }
    Println(i);
}
```

```
continue;
//Ejemplo en un ciclo for
for(int i = 0; i < 9; i++){
    if(i==5){
        Println("Me salte el numero " + i);
        continue;
    }
    Println(i);
}
```

Se pueden declarar funciones de la siguiente manera:

```
<ID> '(' [<PARAMETROS>] ')' ':' <TIPO> '{'
        [<INSTRUCCIONES>]
            '}'
    PARAMETROS -> [<PARAMETROS] ',' [<TIPO>] [<ID>]
                    | [<TIPO>] [<ID>]


        //Ejemplo de declaración de una función de enteros
        conversion(double pies, string tipo): double  {
                if (tipo == "metro")
                {
                        return pies/3.281;
                }
                else
                {
                        return -1;
                }
        }
```

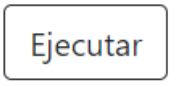
Para imprimir en consola existe el print y println:

```
'Print' '(' <EXPRESION> ')';
//Ejemplo
Print("Hola mundo!!");
Print("Sale compi \n" + valor + "!!");
Print(suma(2,2));
/*
Salida Esperada:
Hola Mundo!!Sale compi
25!!4
*/
// 25 es el valor almacenado en la variable "valor"
```

```
'Println' '(' <EXPRESION> ')';
//Ejemplo
Println("Hola mundo!!");
Println("Sale compi \n" + valor + "!!");
Println(suma(2,2));
/*
Salida Esperada:
Hola Mundo!!
Sale compi
25!!
4
*/
// 25 es el valor almacenado en la variable "valor"
```

4. Analizar Entradas

La entrada se analiza al presionar el botón de ejecutar:



Este botón realiza la petición al backend, el cual devuelve el resultado que se desea imprimiéndolo en la consola:

5. Ver tabla de símbolos

Para ver esta tabla se deberá pulsar al botón de Ver Tabla de Símbolos ubicado en la parte inferior de la página:

Con este botón se mostrará la tabla de símbolos utilizada para ejecutar el código: