

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Organización de Lenguajes y Compiladores
Segundo Semestre 2022

MFM Script

Nombre: albertt wosveli itzep raymundo

carnet: 201908658



INTRODUCCIÓN

Para el mejor entendimiento con respecto a los alumnos ingresantes a la universidad que están llevando los primeros cursos de programación, desarrollamos una herramienta que les pueda apoyar para el aprendizaje ágil como efectivo, y está en si es un lenguaje natural compilado en javascript nombrado MFMScript .

MFM Script

Por parte de la universidad de San Carlos de Guatemala se 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.

Arquitectura General del proyecto

Para una mayor facilidad de empleo de esta herramienta, esta es desplegada por medio de un sistema web que le permite escribir código de forma accesible y rápida sin tener que instalar o configurar un entorno para su próxima ejecución.

Descripción de Interface

Funcionalidades dentro de la interfaz:

- Crear archivos: El editor deberá ser capaz de crear archivos en blanco.
- Abrir archivos: El editor deberá abrir archivos [.olc]
- Guardar el archivo: El editor deberá guardar el estado del archivo en el que se estará trabajando.
- Podrá generar los reportes del árbol AST, tabla de símbolos y Errores que se generan dentro del sistema.

Área de consola

En esta área se mostrarán los resultados, mensajes y todo lo que sea indicado dentro del lenguaje.

Descripción de Lenguaje

Case Insensitive

El lenguaje no distinguirá entre mayúsculas o minúsculas.

Comentarios

Los comentarios son una forma elegante de indicar que función tiene cierta sección del código que se ha escrito simplemente para dejar algún mensaje en específico. El lenguaje deberá soportar dos tipos de comentarios que son los siguientes:

- Comentarios de una línea
 - Estos comentarios deberán comenzar con `//` y terminar con un salto de línea.
- Comentarios de una línea
 - Estos comentarios deberán comenzar con `/*` y terminar con `*/`.

Tipos de Datos

Los tipos de dato que soportará el lenguaje en concepto de un tipo de variable se definen a continuación:

- Int
- Double
- Boolean
- Char
- String

Operadores Aritméticos

Se podrá realizar las siguientes operaciones Aritméticas:

- Suma $:= a + b$

- Resta := $a - b$
- Multiplicación := $a * b$
- División := a / b
- Potencia := $a ^ b$
- Módulo := $a \% b$
- Negación Unaria := $-a$

Operadores Relacionales

Se podrá realizar las siguientes operaciones relacionales:

- Mayor que := $a > b$
- Menor que := $a < b$
- Mayor o igual que := $a \geq b$
- Menor o igual que := $a \leq b$
- Igual que := $a == b$
- Diferente := $a != b$
- Operador Ternario := Condicional ? Instrucción A: Instrucción B

Operadores Lógicos

Se podrá realizar las siguientes operaciones lógicas:

- Or := $a || b$
- And := $a \&\& b$
- Not := $! \text{var}$

Declaración y asignación de variables

El lenguaje será fuertemente tipado (el tipo de dato de una variable no puede cambiar, siempre será el mismo tipo). Cada una de las variables deberá de ser declarada antes de poder ser utilizada, de lo contrario será un error semántico. Las variables cuentan con un identificador y no podrán repetirse en el mismo entorno, para los identificadores es válido usar caracteres del abecedario [A-Z], números y guiones bajos. Las variables

podrán ser declaradas global y localmente. Durante la declaración de variables también se tendrá la opción de poder crear múltiples variables al mismo tiempo, al crear múltiples variables al mismo tiempo se tendrá la opción de crear todas las variables con un mismo valor, para ello se realizará una asignación al final del listado de las variables, en caso de no indicar esta asignación se dejará el valor por defecto para cada variable.

Ejemplos :

```
string curso_ = "organización de compiladores 1 2022";
```

```
char var_111 = 'a';
```

```
celular = true;
```

```
int var1, var2, var3;
```

```
boolean flag_1, flag2_, flag33 = false;
```

```
char ch_1, ch_2, ch3, qwert123 = 'M';
```

```
universidad= "usac";
```

```
boolean flag_personalizada;
```

Casteos

Los casteos son una forma de indicar al lenguaje que convierta un tipo de dato en otro, por lo que, si queremos cambiar un valor a otro tipo, es la forma adecuada de hacerlo. Para hacer esto, se colocará la palabra reservada del tipo de dato destino entre paréntesis seguido de una expresión.

Ejemplos :

```
Int edad = (Int) 18.6;
```

```
Int intBool = (Int) false;
```

```
Int intDecimal = (Int) '0';
```

```

Int intCadena = (Int) "100";

Double doubleInt = (Double) 16;

Double doubleString = (Double) "16.0";

Boolean boolInt = (Boolean) 0;

Boolean boolDouble = (Boolean) 1.0;

```

Incremento y Decremento

Los incrementos y decrementos nos ayudan a realizar la suma o resta continua de un valor de uno en uno, es decir si incrementamos una variable, se incrementará de uno en uno, mientras que, si realizamos un decremento, hará la operación contraria.

Ejemplos :

```

int edad = 18;

edad++;

int anio=2020;

anio = 1 + anio++;

anio = anio--;

```

Vectores

Declaración de Vectores:

Ejemplos:

```

Int [ ] vector1 = new Int[4];

Char [ ][ ] vectorDosd = new Char [(Int) "4"] [4] ;

String [ ] vector2 = {"hola", "Mundo"};

```

```
Char [ ][ ] vectordosd2 = {{ 0 ,0},{0 , 0}};
```

Acceso a Vectores

Ejemplos:

```
String [ ] vector2 = {"hola", "Mundo"};
```

```
String valorPosicion = vector2[1];
```

```
Char [ ][ ] vectorDosd = new char [4] [4] ;
```

```
Char valor = vectorDosd[1][1];
```

Modificación de Vectores

Ejemplos:

```
String [ ] vector2 = {"hola", "Mundo"};
```

```
Int [ ] vectorNumero = {2020,2021,2022};
```

```
vector2[1] = "OLC1 ";
```

```
vector2[2] = "2do Semestre "+vectorNumero[2];
```

Sentencias de Control

if

Ejemplos:

```
if (x <50){
```

```
    Println("Menor que 50");
```

```
}
```

```
if (x <50){
```

```
    Println("Menor que 50");
```

```
}else{
```



```

    println("Mayor que 100");

}

if (x > 50){
    Print("Mayor que 50");
}elif (x <= 50 && x > 0){
    Print ("Menor que 50");
    if (x > 25){
        Print("Número mayor que 25");
    }else{
        Print("Número menor que 25");
    } } else {
        Print("Número negativo");
    }
}

```

Switch case

Ejemplos:

```

int edad = 18;

switch( edad ) {

Case 10:

    println("Tengo 10 años.");

    Break;

Case 18:

    Print("Tengo 18 años.\n");
}

```

Case 25:

```
Println("Tengo 25 anios.");
```

```
Break;
```

Default:

```
Print("No se que edad tengo. :(\n");
```

```
Break;
```

```
}
```

Ciclos

While

Ejemplo:

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

For

Ejemplo:

```
for ( i=5; i>2;i=i-1 ){  
    Print("i="+i+"\n");  
}
```

DO WHILE

Ejemplo:

```
do{  
    if (a>=1 && a <3){  
        Println(true)  
    } else {  
        Println(false)  
    }  
    a--;  
} while (a>0);
```

DO UNTIL

Ejemplo:

```
do{  
    if (a>=1 && a <3){  
        Println(true)  
    } else {
```

```

        println(false)
    }
    a--;
} while (a>0);

```

Break

Ejemplo:

```

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

```

Return

Ejemplo:

```

for(int i = 0; i < 9; i++){
    if(i==5){
        println("Me salgo del ciclo en el numero " + i);
        return;
    }
    println(i);    } return n3;

```

Funciones

Ejemplo:

```
conversion (double pies, string tipo): double {  
    if (tipo == "metro"){  
        return pies/3.281;  
    }else{  
        return -1;  
    }  
}
```

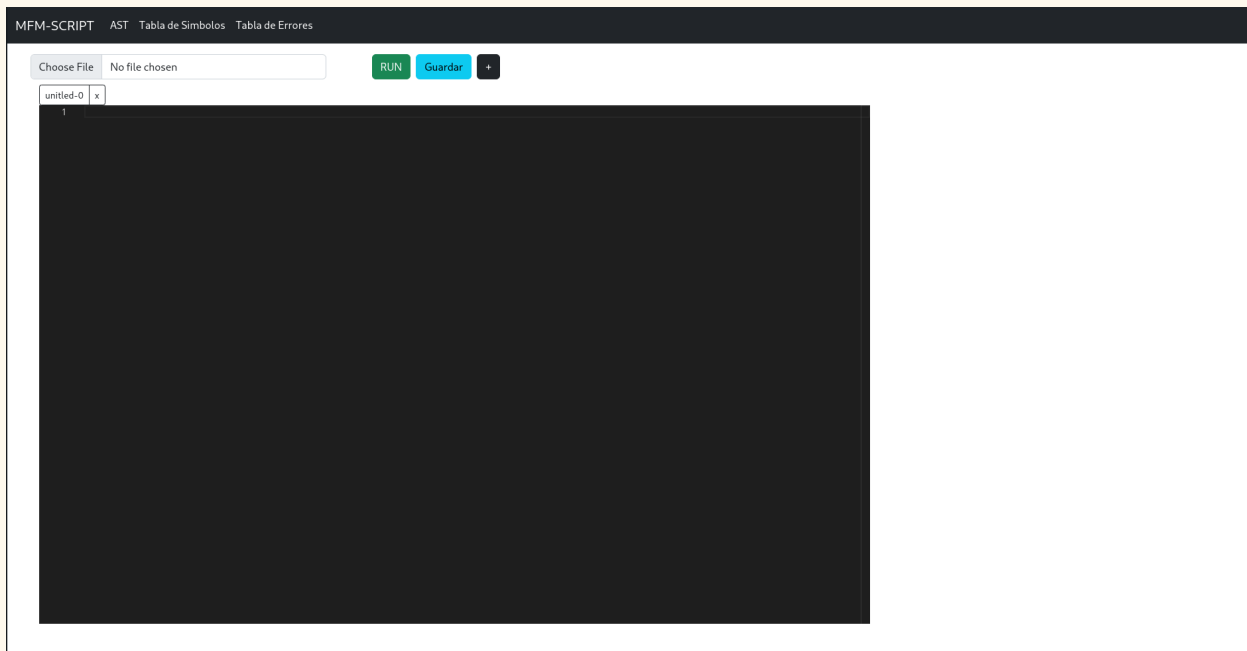
Métodos

Ejemplo:

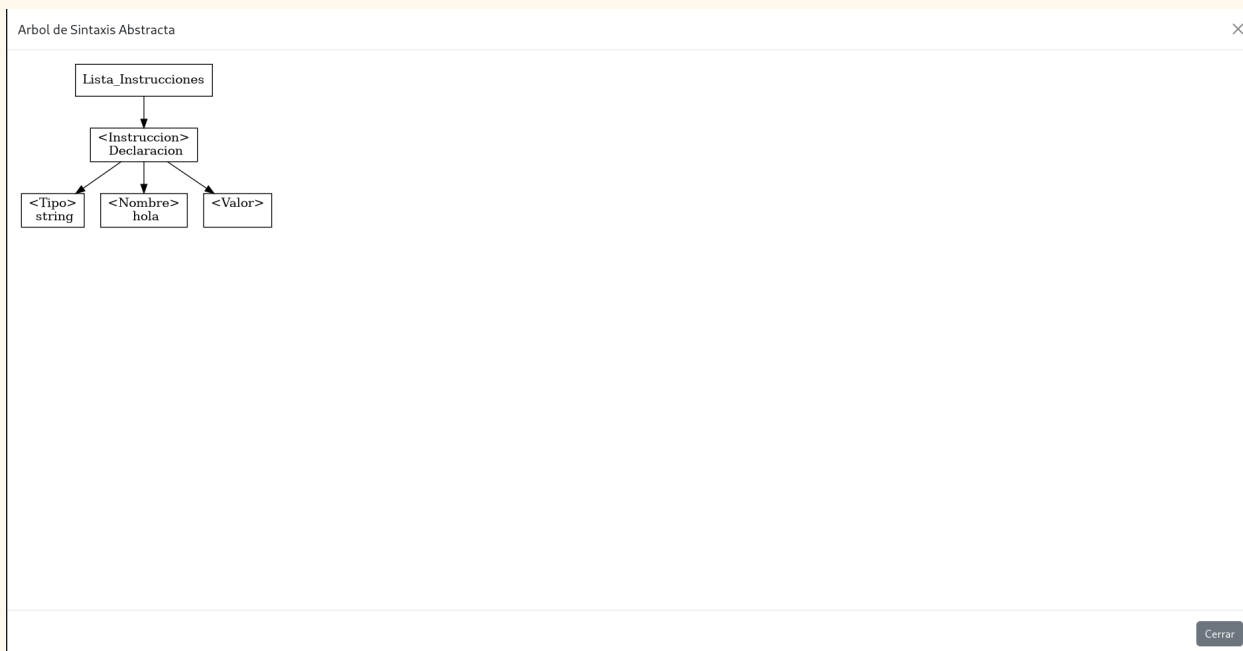
```
holamundo(): void {  
    Print("Hola mundo");  
}
```

Anexos

- **Interface de usuario:**



- **AST**



- Símbolos

Tabla de Símbolos			
<input type="checkbox"/>	ID	TIPO	COLUMNA
Sorry, no matching records found			
Rows per page: 10 0-0 of 0 < >			

Cerrar

- Errores

Tabla de errores			
<input type="checkbox"/>	TIPO	DESCRIPCION	COLUMNA
Sorry, no matching records found			
Rows per page: 0-0 of 0 < >			

Cerrar