Lenguajes Formales

TP de Compiladores

Trabajo práctico Nº2 de Lenguajes Formales y Teoría de la Computación.

Profesor: Ing. Mario S. Moreno.

ININF. Universidad del CEMA.

Alumnos: Matias Casullo, Isabella Marafioti y Luca Sucri.

Introducción:

En este informe se va a detallar la ejecución del segundo projecto práctico de la asignatura de Lenguajes Formales y Teoría de la Computación.

El propósito de este trabajo es poner en práctica los conceptos enseñados en clases hasta ahora mediante el desarollo de un compilador utilizando un lenguaje creado por los miembros del grupo.

El programa demostrará la funcionalidad del compilador a través de la parte semántica, sintáctica o ambas.

Especificaciones:

- ➔ Cada instrucción debe terminar con un punto y coma “;”.
- ➔ Para que la compilación sea la correcta, todas las variables tienen que tener primero el tag, seguido de un nombre, “=” y un número o un valor booleano o valores alfanuméricos (texto). Además, tiene que haber un espacio entre cada uno.

- ➔ Al tipo “number” (int) se le debe asignar un valor numérico.
- ➔ Al tipo “string” se le debe asignar un valor alfanumérico o una cadena de texto encerrados entre comillas dobles.
- ➔ Al tipo “boolean” se le debe asignar un valor booleano (true o false).

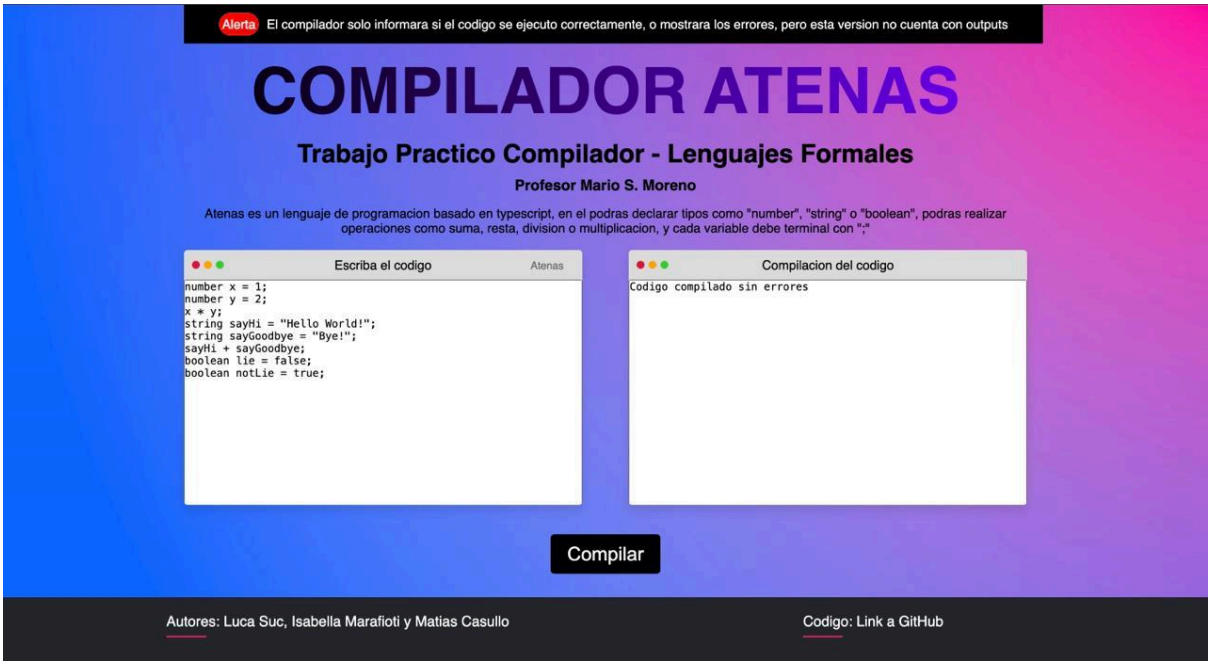
Instrucciones válidas:

- ☑ Asignarle un valor a una variable
- ☑ Sumas, restas, multiplicación y división
- ☑ Concatenar alfanuméricos o números con números.

Modo de uso:

.

- ➔ Podemos compilar un código en un lenguaje de programa ficticio (“Atenas”) basado en typescript.
- ➔ Link para el compilado => Link



Detalles:

El compilador funciona a partir de un lenguaje ficticio creado por los integrantes del grupo, llamado “Atenas”, definido dentro del archivo “script.js”. Si bien se encuentra en una fase inicial con gran potencial de mejora a lo largo del curso, la implementación actual permite compilar solo ciertas funciones básicas:

- Declaración de variables:
  → Numeros enteros => *number x = 150304;*
  → Cadena de texto => *string Nombre = “Juan Manuel”*
  → Valores booleanos => *boolean Estado = true;*

- Operaciones matemáticas básicas:
  → Suma, resta, multiplicación y divisiones => *int ejemploOperacion = 3 * 2 + 14 / 7*

```
const symbolsMatrix = [
    { tag: "number", type: "declaration", rule: (i, p, iSimbolo, iInstruccion, program) => { numberTagVerification(i, p, iSimbolo, iInstruccion, program) } },
    { tag: "string", type: "declaration", rule: (i, p, iSimbolo, iInstruccion, program) => { stringTagVerification(i, p, iSimbolo, iInstruccion, program) } },
    { tag: "boolean", type: "declaration", rule: (i, p, iSimbolo, iInstruccion, program) => { booleanTagVerification(i, p, iSimbolo, iInstruccion, program) } },
    { tag: "=", type: "asignation", rule: (i, p, iSimbolo, iInstruccion, program) => { asignarValidacion(i, p, iSimbolo, iInstruccion, program) } },
    { tag: "+", type: "operation", rule: (i, p, iSimbolo, iInstruccion, program) => { validarOperacionSuma(i, p, iSimbolo, iInstruccion, program) } },
    { tag: "-", type: "operation", rule: (i, p, iSimbolo, iInstruccion, program) => { validarOperacionResta(i, p, iSimbolo, iInstruccion, program) } },
    { tag: "/", type: "operation", rule: (i, p, iSimbolo, iInstruccion, program) => { validarOperacionDivicion(i, p, iSimbolo, iInstruccion, program) } },
    { tag: "*", type: "operation", rule: (i, p, iSimbolo, iInstruccion, program) => { 6validarOperacionMultiplicar(i, p, iSimbolo, iInstruccion, program) } },
  { tag: ";", type: "break", rule: () => { } },
```

];

Ejemplos:

- Ejemplos exitosos:

```
number operacionPrueba = 2 * 4 + 6 / 2 - 1;
numer numero = 2024;
string hola = "Hola";
string mundo = "Mundo";
boolean valorVerdadero = true;
hola + mundo;
```

- Nos devolverá:

```
Codigo compilado correctamente. 0 errores
```

- Ejemplos de instrucciones no validas

```
boolean string = "true";
boolean numero = 1234;
number string = "hola mundo";
number simbolo = ???;
number = 1234;
```

- Nos devolverá:

```
Error, linea 1: El tag string no esta al principio de la instruccion
Error, linea 1: La variable tiene que tener un nombre
Error, linea 1: El valor de la variable no es booleana
Error, linea 2: El valor de la variable no es booleana
Error, linea 3: La variable tiene que tener un nombre
Error, linea 3: El tag string no esta al principio de la instruccion
Error, linea 3: El valor de la variable no es numerico
Error, linea 4: El valor de la variable no es numerico
Error, linea 5: La variable tiene que tener un nombre
```