

Manual Técnico

Marvin Eduardo Catalán Véliz

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Organización de Lenguajes y Compiladores 1

Ing. Mario Bautista

Aux. Emely García

Aux. José Morán

MANUAL TECNICO FIUSAC SYSCOMPILER

Marvin Eduardo Catalán Véliz

201905554

Guatemala, 6 de nov. de 21

Índice

REQUERIMIENTO TÉCNICO PARA EJECUCION	2
HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO	3
Angular.....	3
¿Qué es?	3
Versiones utilizadas en el framework	3
Editor de texto	3
¿Qué es?	3
Editor de código utilizado: Visual Studio Code	4
LAPTOP	4
Librerías	5
Jison:.....	5
Version Jison:.....	5
Funcionalidad y explicación.....	6
¿Cómo inicializar el proyecto?.....	6
Front-End.....	6
App.component.html:	6
Style.css:	6
Back-end.....	6
Expresion.....	7
Expresion.ts	7
Retorno.ts	8
AccesoAmbito.ts	8
Aritmetica.ts	8
Literal.ts	8
Logica.ts y Relacional.ts	8
Ambito.....	8
Grammar.....	9
Instrucción.....	9
Metodos abstractos de Instrucción y Expresion.....	10
Public abstract execute(ámbito:Ambito null):Retorno null{}	10
Public abstract getCodigoAST():{codigo,nombre}	10

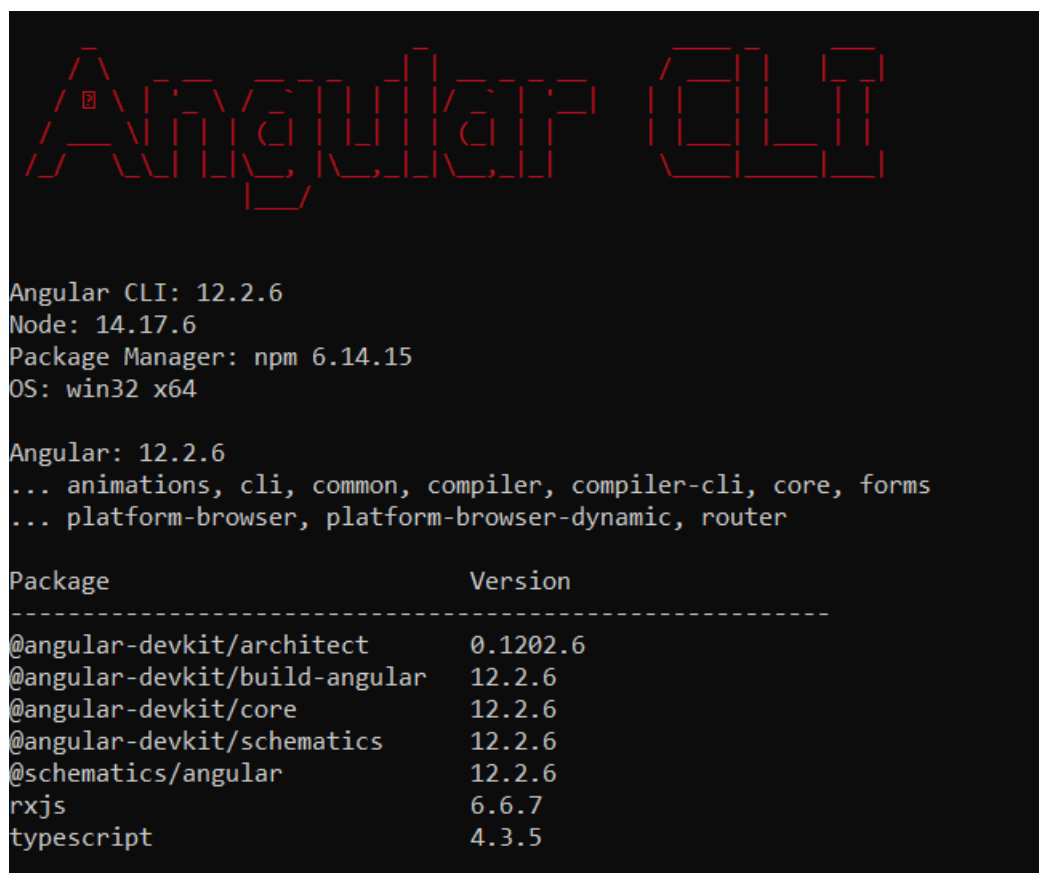
HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO

Angular

¿Qué es?

Es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

Versiónes utilizadas en el framework



```
Angular CLI: 12.2.6
Node: 14.17.6
Package Manager: npm 6.14.15
OS: win32 x64

Angular: 12.2.6
... animations, cli, common, compiler, compiler-cli, core, forms
... platform-browser, platform-browser-dynamic, router

Package                                Version
-----
@angular-devkit/architect              0.1202.6
@angular-devkit/build-angular          12.2.6
@angular-devkit/core                   12.2.6
@angular-devkit/schematics             12.2.6
@schematics/angular                    12.2.6
rxjs                                    6.6.7
typescript                             4.3.5
```

Editor de texto

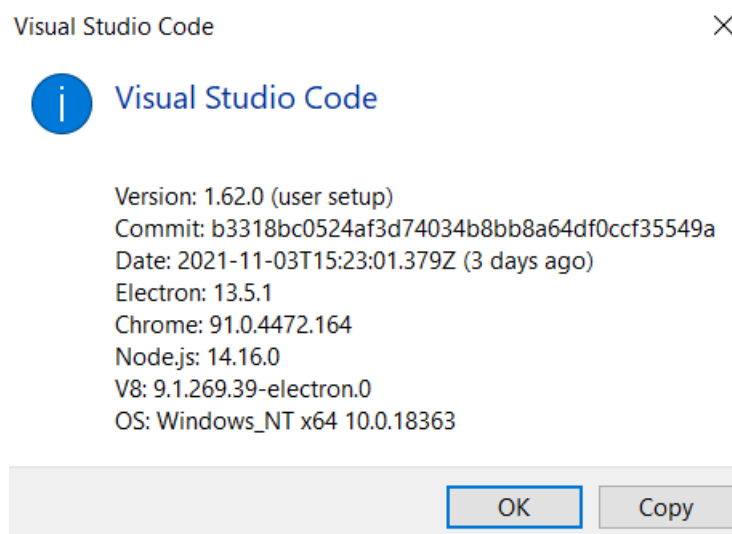
¿Qué es?

Editor de texto es un programa informático que permite crear y modificar archivos digitales compuestos únicamente por textos sin formato, conocidos comúnmente como archivos de texto o "texto plano". El programa lee el archivo e interpreta los bytes leídos según el código de caracteres que usa el editor. Es comúnmente de 7- u 8-bits en ASCII o UTF-8, rara vez EBCDIC.

Editor de código utilizado: Visual Studio Code

Es un editor de código fuente que permite trabajar con diversos lenguajes de programación, admite gestionar tus propios atajos de teclado y refactorizar el código. Es gratuito, de código abierto y nos proporciona una utilidad para descargar y gestionar extensiones con las que podemos personalizar y potenciar esta herramienta.

Las extensiones de Visual Studio Code nos otorgan infinidad de opciones, como colorear tabulaciones, etiquetas o recomendaciones de autocompletado. También hay extensiones que nos ayudan con el lenguaje de programación que vayamos a usar, como por ejemplo para Python, C / C++, JavaScript, etc.



LAPTOP

Especificaciones del dispositivo

Nombre del dispositivo	DESKTOP-KMQA5BN
Procesador	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
RAM instalada	12.0 GB (11.9 GB utilizable)
Id. del dispositivo	369A16F7-DB15-4299-B4CD-780C88B080B8
Id. del producto	00325-96001-12028-AAOEM
Tipo de sistema	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil	Compatibilidad con entrada táctil con 10 puntos táctiles

Librerías

Jison:

Jison toma una gramática libre de contexto como entrada y produce código JavaScript capaz de parsear el lenguaje descrito por dicha gramática. Una vez se tenga el script generado podemos usarlo para parsear la entrada y aceptarla, rechazarla o ejecutar acciones con base en la entrada. Si se está familiarizado con Bison, Yacc o algún otro similar ya se está listo para iniciar. Jison genera tanto el analizador léxico como el analizador sintáctico.

La principal tarea de un analizador léxico es leer los caracteres de entrada del programa fuente, agruparlos en lexemas y producir como salida una secuencia de tokens.

- Un *token* es un par que consiste en un nombre de token y un valor de atributo opcional.
- Un *lexema* es una secuencia de caracteres en el programa fuente, que coinciden con el patrón para un token y que el analizador léxico identifica como una instancia de este tóken.
- Un *patrón* es una descripción de la forma que pueden tomar los lexemas de un token.

El analizador sintáctico obtiene una cadena de tokens del analizador léxico y verifica que dicha cadena pueda generarse con la gramática para el lenguaje fuente. Una gramática proporciona una especificación precisa y fácil de entender de un lenguaje de programación.

En Jison se definen tanto el analizador léxico como el sintáctico. Esto es una gran ventaja pues podemos trabajar en una sola herramienta.

Version Jison:

```
C:\Users\marvi\OneDrive\Escritorio\Segundo semestre 2021\Compi 1\Laboratorio\Proyecto 2\OLC1_SYSCOMPILER_201905554\SYSCompiler>jison --version
0.4.18
```

Funcionalidad y explicación

El programa está basado en su mayoría utilizando programación orientada a objetos, o al menos es lo que se intentó plasmar para poder tener un código, optimo y organizado. El programa está dividido en distintas clases los cuales se describen a continuación:

¿Cómo inicializar el proyecto?

El Proyecto esta realizado en angular por lo cual el comando para ejecutar el código sería

```
ng serve
```

Si en dado caso se necesita modificar la gramática escrita en el `grammar.jison` para implementar alguna nueva funcionalidad o cambiar la gramática como tal, luego de haber guardado cambios se debe ejecutar un comando que compila todo el código y lo traduce a JavaScript el comando es: `jison grammar.jison`.

Front-End

App.component.html:

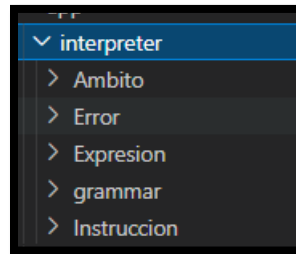
Esta clase es un componente de angular la cual ejecuta la primera plantilla que esta siendo utilizada para ejecutar el interprete y sus salidas por medio de una consola o diferentes tablas.

Style.css:

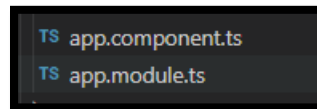
Esta clase es un componente de angular donde se encuentran encapsuladas todas las sentencias css que se ejecutan en todo el ámbito del proyecto.

Back-end

El back-end esta dividido en seis grandes pilares que son los siguientes mostrados en la imagen:



Los cinco paquetes generados que manejan toda la lógica del lenguaje, así como los errores, ámbitos, etc. De los cuales se hablará posteriormente.

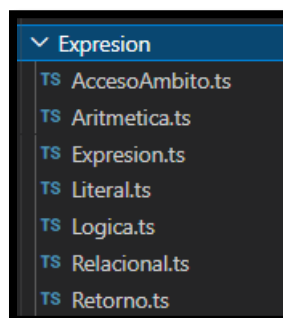


El app.component.ts y app.module.ts estos son los encargados de ejecutar el código y hacer la conexión del back-end con el front-end.

Ahora entrando más a profundidad que contiene cada paquete y sus funcionalidades.

Expresion

Este paquete contiene todas aquellas clases que sean de tipo expresión, como por ejemplo operadores aritméticos, relacionales, lógicos, ósea todo aquello que pueda ir dentro de una expresión como tal.



Expresion.ts

Esta clase es utilizada como una Interface que hereda todos sus atributos y métodos abstractos hacia todas las demás clases que extiendan de esta misma.

Retorno.ts

Esta clase es super importante ya que esta por medio de matrices nos devuelve todos los tipos de retorno que hay entre un tipo de dato con otro. En este lenguaje existen 5 tipos de variables que se muestran a continuación.

```
export enum Type {  
  INT = 0,  
  DOUBLE = 1,  
  BOOLEAN = 2,  
  CHAR = 3,  
  STRING = 4,  
  ERROR=5  
}
```

El tipo Error es utilizado para cuando 2 tipos de datos no se pueden operar en cualquier tipo de acción aritmética, al momento de que retorne un tipo error se lanzará una excepción.

AccesoAmbito.ts

Esta clase es utilizada para saber si una variable esta en el ámbito a consultar ya sea en cualquier parte del código.

Aritmetica.ts

Esta clase es utilizada para realizar operaciones aritméticas entre 2 tipos de datos, retornan el tipo de dato dominante que se obtenga desde la clase Retorno.ts.

Literal.ts

Esta clase retorna el tipo de literal que entra desde la gramática, es la encargada de inicializar el tipo de la gramática dependiendo con que expresión regular declarada en el grammar.jison venga de entrada.

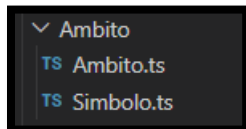
Logica.ts y Relacional.ts

Estas dos clases como bien lo indica su nombre están declaradas para realizar validaciones u operaciones entre dos datos, ya sean relaciones lógicas y relacionales.

Ambito

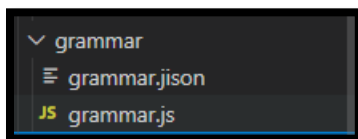
Este paquete es utilizado para delcarar los ámbitos y las variables que existen dentro de cada ámbito, contiene dos clases que son Ambito.ts y Simbolo.ts el símbolo solamente es una clase

de objeto que almacena los datos de las variables y el ámbito es aquel que almacena las variables pero también almacena los métodos, con la diferencia que los métodos solamente pueden estar en el ámbito que su anterior apunte a *null* esto quiere decir que no le precede ningún otro ámbito, ósea es el ámbito *Global*.



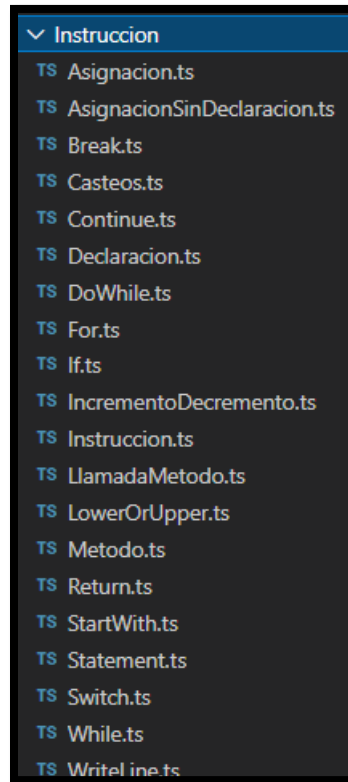
Grammar

Dentro de este paquete se encuentra la gramática que sigue nuestro lenguaje, esta gramática esta escrita en la sintáxis de la herramienta Jison de la cuál ya se habló con anterioridad, se encuentra en el archivo `Grammar.jison`, el archivo `grammar.js` es el que se crea al compilar la gramática si no existe ningún error de precedencia o ambigüedad.



Instrucción

Este paquete contiene la mayoría de las instrucciones fuera de las operaciones o comparaciones que puede haber en el lenguaje. Aquí se encuentra la lógica de todas las acciones que se pueden realizar. El código o la solución para que funcione es bastante intuitivo sabiendo que producción produce el objeto de la clase valga la redundancia, por lo cual no se procede a explicar cada solución para cada instrucción ya que cualquier programador debe de tener esas bases claras.



Metodos abstractos de Instrucción y Expresion

Public abstract

`execute(ambito:Ambito|null):Retorno|null{}`

Este metodo abstracto es utilizado para ejecutar cada una de las sentencias y acciones que extienden de cualquiera de estas dos clases, cabe destacar que todo lo que esta en expresión necesitan de un ámbito como de un retorno ya que las expresiones son recursivas, mientras que las instrucciones también pero crean su propio ámbito dentro de sus Statements.

Public abstract `getCodigoAST():{codigo,nombre}`

Este método abstracto es utilizado para ejecutar y almacenar el código de cada sentencia e instrucción para ir almacenándolo y generar un AST al final de la ejecución del programa.