Propuesta Compilador
“*Tabby*”

Marzo / 2022

Diego García Rodríguez del Campo
A01700203

Diseño de Compiladores

Visión / Propósito del Proyecto

Este proyecto tiene como propósito crear un lenguaje de programación y su compilador respectivo para reforzar el aprendizaje de todos los conceptos revisados en la clase de *Diseño de Compiladores*, y muchas otras materias cursadas a lo largo de la carrera universitaria, como Matemáticas Computacionales o *Lenguajes de Programación*.

La visión de este proyecto es generar un lenguaje de programación de calidad, que sea fácil de usar e intuitivo, con todos los elementos básicos que engloban a los lenguajes de programación, desde condiciones o ciclos, hasta arreglos simples y funciones, que pueda ser desarrollado a través de 3 meses, y que deje la opción abierta para continuar con su desarrollo y mejoría de este.

Objetivo Principal del Lenguaje

El objetivo principal del lenguaje, o en otras palabras, su enfoque principal es la creación de un código con sintaxis simple, pero poder computacional, y principalmente la implementación en este de estructuras que se puedan utilizar para realizar análisis estadísticos a través de modelos básicos y sus gráficas, de manera similar a lenguajes como R. La idea de *Tabby* es generar un lenguaje de programación fácil de entender, incluso para gente que no está completamente familiarizada con programación, y darles herramientas de manera simple para poder resolver problemas estadísticos.

Requerimientos del Lenguaje

Elementos Básicos

Tabby tiene varios elementos básicos, o tokens, que nos dan las bases para la sintaxis del lenguaje. Los elementos básicos son:

Palabras Clave

Como en cualquier otro lenguaje, existen palabras clave, o reservadas, que el lenguaje utiliza específicamente para identificar ciertas estructuras. Es fácil identificarlas, pues

comienzan con una letra Mayúscula, a diferencia de los identificadores que comienzan con letras minúsculas. Las palabras clave en *Tabby* son:

- Int, Float, Char, Nool
    - Estas se usan para declarar el tipo de variable
- If, Else, While, For
    - Estas se utilizan para ciclos y condicionales
- Or, And
    - Utilizadas en expresiones como lógica booleana
- Program
    - Utilizada al inicio para inicializar el programa
- Arr
    - Para declarar un arreglo
- Fn
    - Usado para declarar un módulo o función
- Return
    - Regresa valores en la función
- Void
    - Para declarar funciones sin valor de retorno
- Tabby
    - Esta palabra es reservada para la función principal del código. Siempre deberá existir una función con este nombre, y esta función es la que se llamará por el compilador para correr el código

Identificadores

Los identificadores se utilizan para reconocer y almacenar variables en el sistema.
Todos los identificadores en *Tabby* consisten únicamente de letras (minúsculas y mayúsculas), siempre comenzando con una letra minúscula, lo cual las diferencia fácilmente de las palabras reservadas. La siguiente expresión regular representa cualquier identificador válido, siempre y cuando no sea una palabra reservada

[a-z][a-zA-Z]*

Símbolos especiales

Existen varios símbolos especiales, que tienen ciertos usos específicos en el lenguaje. Los símbolos son los siguientes:

- [ ]
    - Se utilizan para referenciar posiciones en arreglos
- ( )
    - Se utilizan para indicar llamadas a funciones y parámetros de estas
- { }
    - Utilizados para delimitar bloques de código, ya sea en funciones, condicionales, ciclos, etc
- ,
    - Utilizadas para separar estatutos
- ;
    - Utilizado para definir el fin de un estatuto

Operadores

Los operadores son símbolos que requieren una acción al aplicarlos a variables en el lenguaje.

Operadores Binarios

Operadores que requieren dos operandos y hacen una acción con estos. Se clasifican en

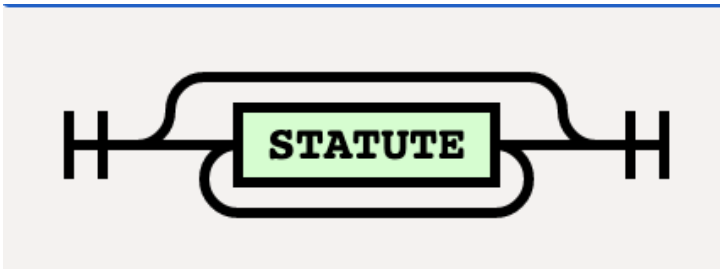
- Aritméticos
- Relacionales
- Lógicos
- Asignación

Diagramas Sintácticos (Gramática)

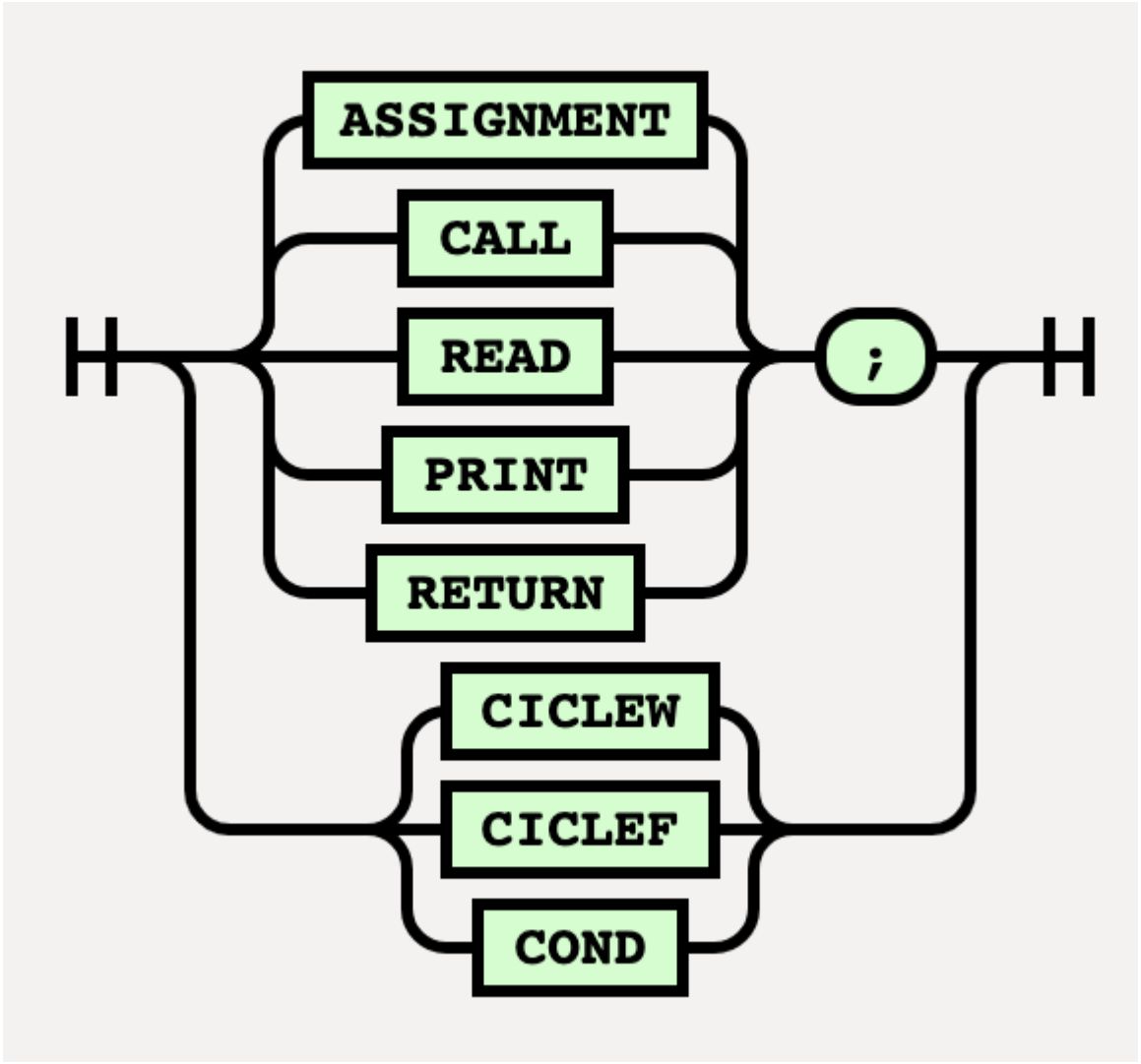
PROGRAM

BLOCK

STATUTES

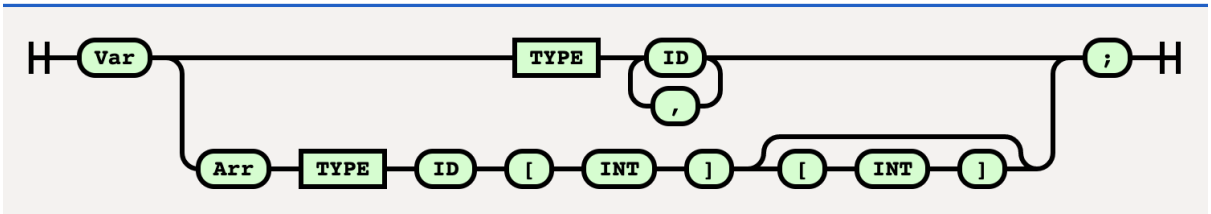


STATUTE

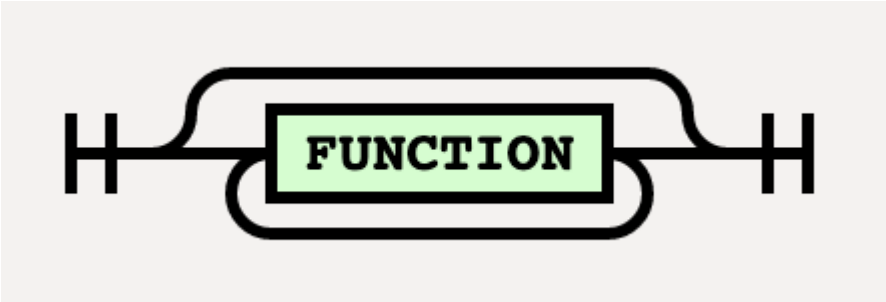


VARDECS

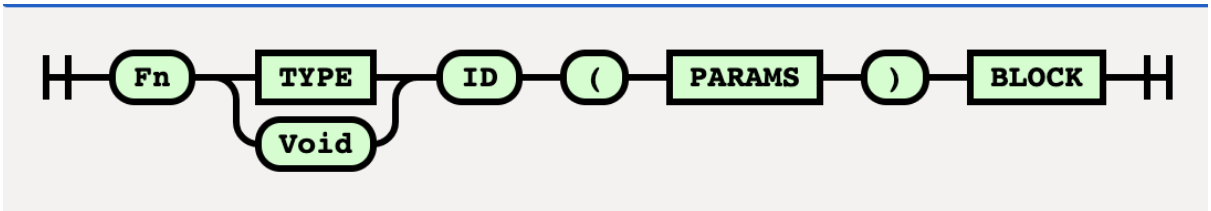
VARDEC



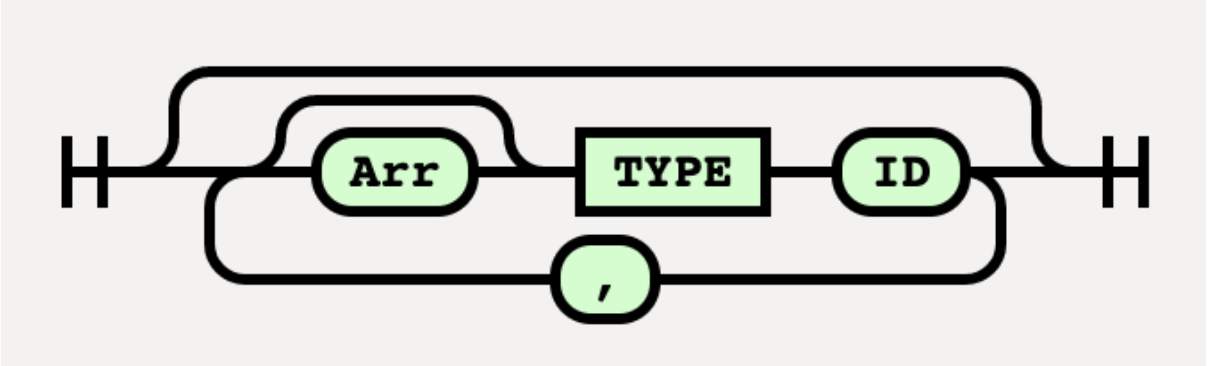
FUNCTIONS



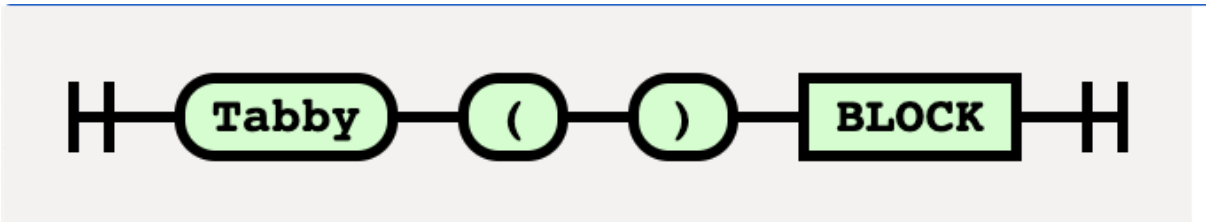
FUCTION



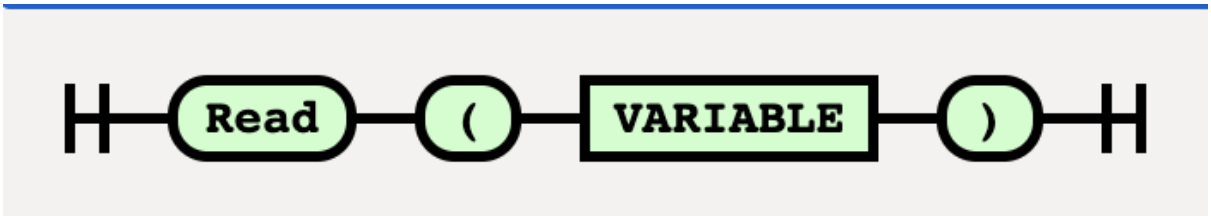
PARAMS



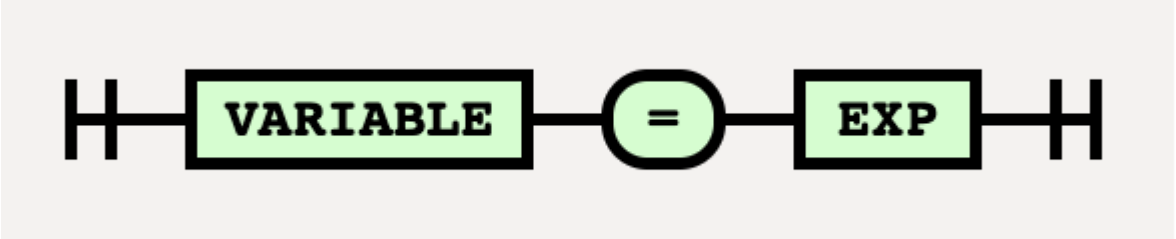
TABBY



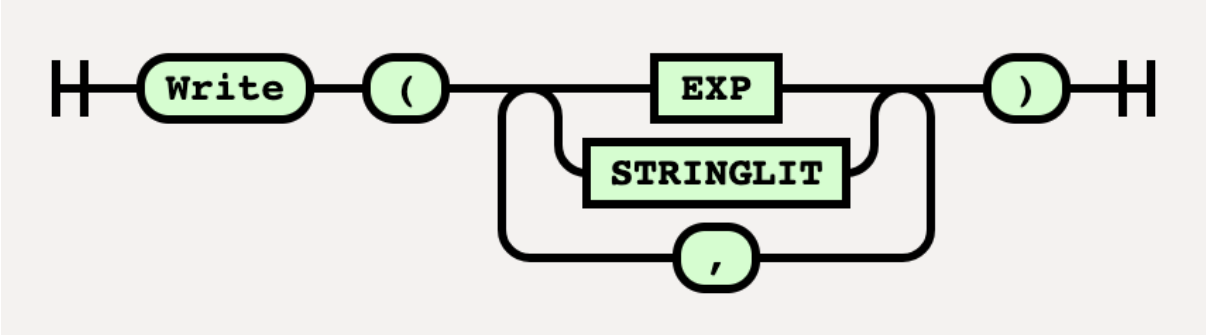
READ



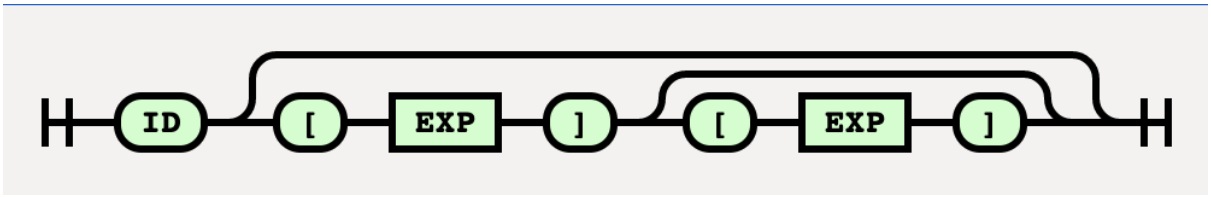
ASSIGNMENT



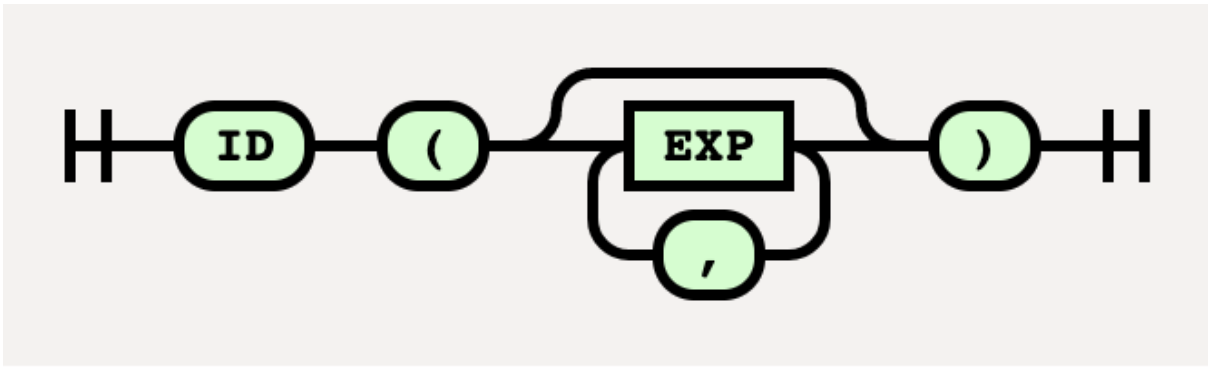
PRINT



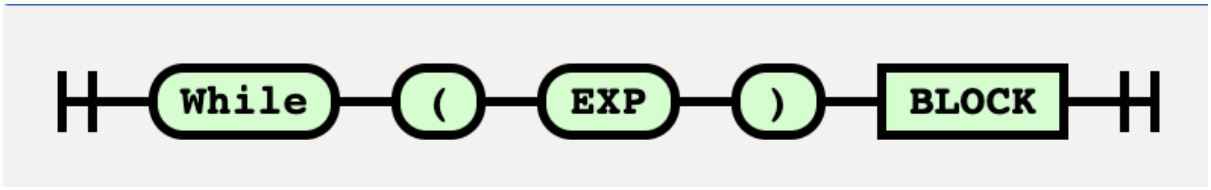
VARIABLE



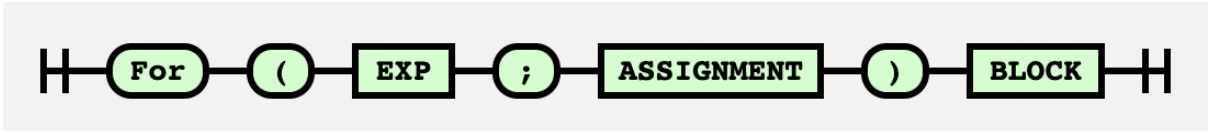
CALL



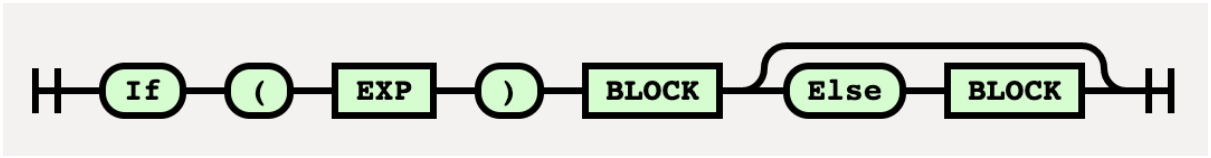
CICLEW



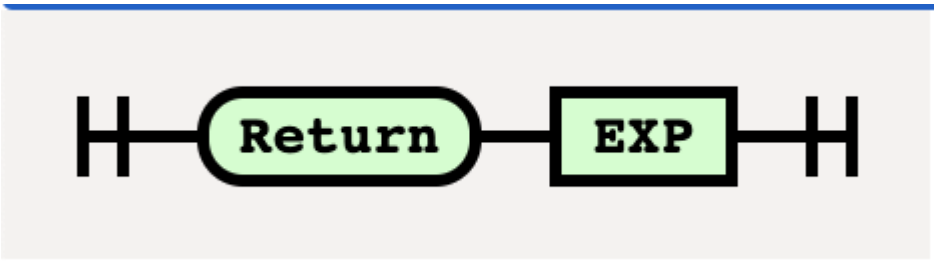
CICLEF



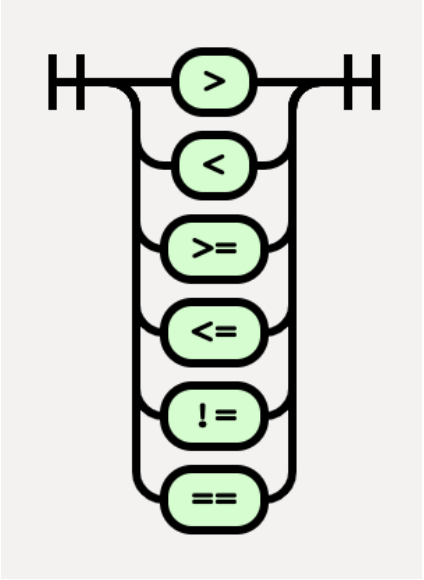
COND



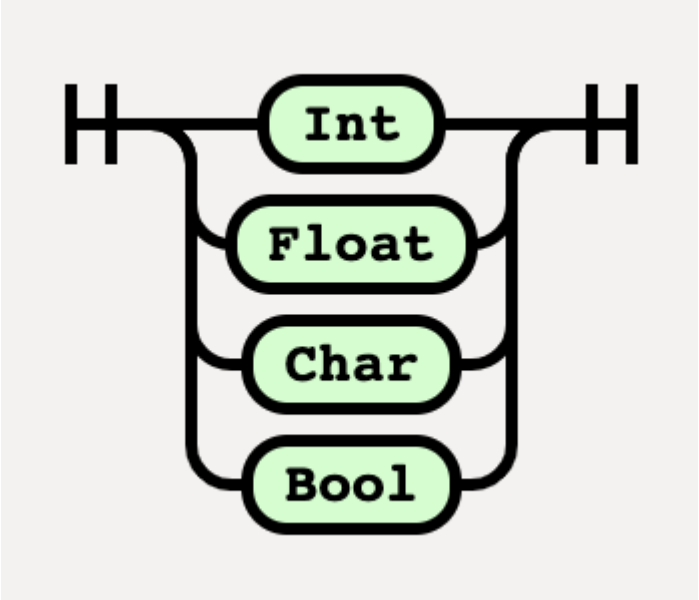
RETURN



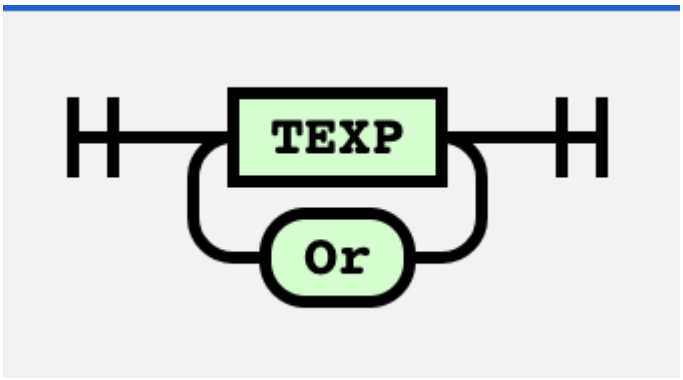
COMP



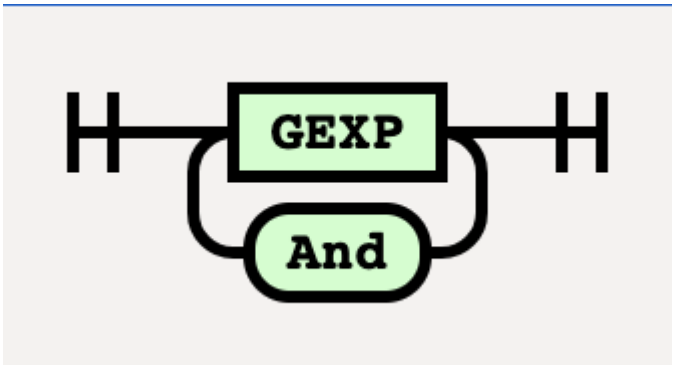
TYPE



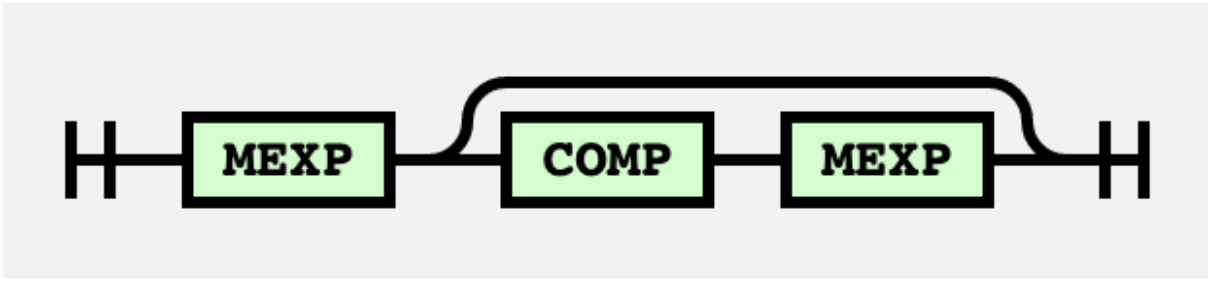
EXP



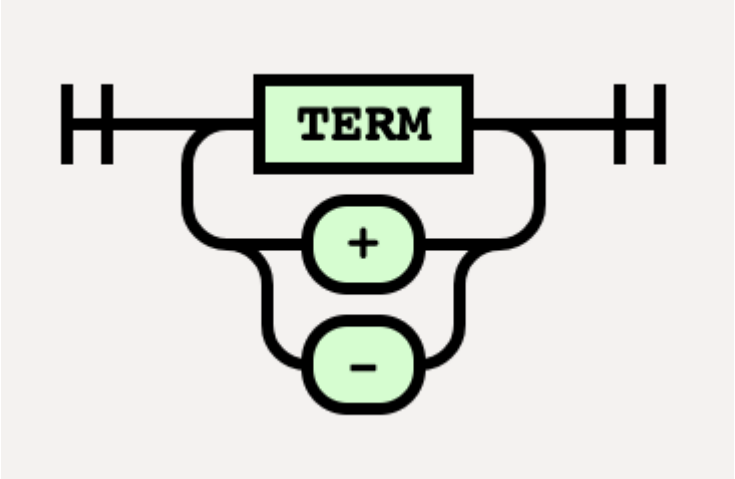
TEXP



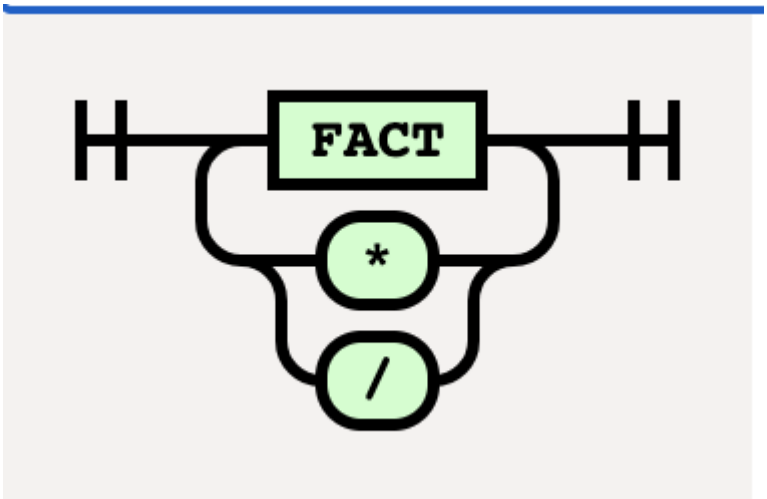
GEXP



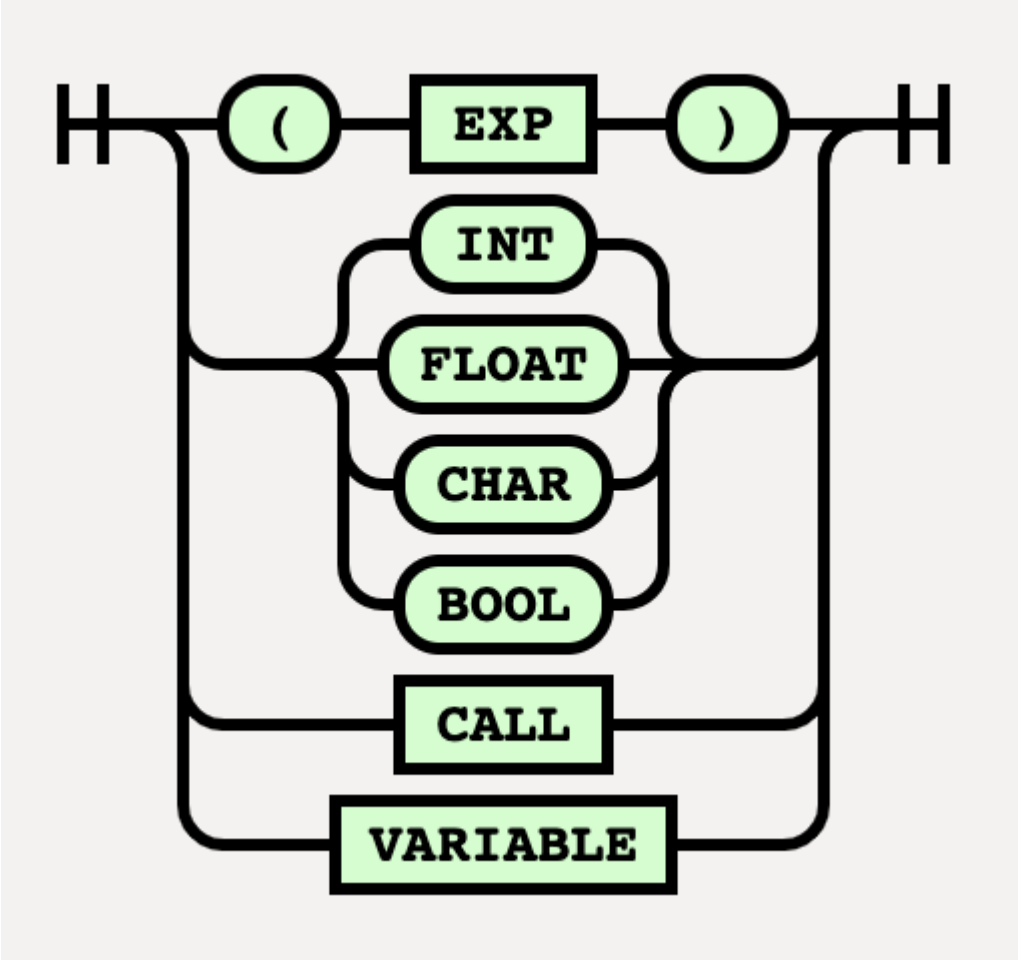
MEXP



TERM



FACT



Terminals

ID

[a-z][a-zA-Z]*

INT

[+-]?[0-9]+

FLOAT

[+-]?[0-9]+\.[0-9]+

CHAR

['].[']

BOOL

True|False

STRINGLIT

\"[^\"]*\"

Tipos de Dato

Como se mencionó anteriormente, solo existen cuatro tipos de datos en Tabby.

- Int
    - Utilizado para representar números enteros
- Float
    - Utilizado para representar números flotantes
- Char
    - Utilizado para representar caracteres
- Bool
    - Utilizado para representar valores Booleanos (True/False)

Además, con el uso de arreglos, se pueden generar grupos de números, o incluso Strings, como un arreglo de Chars

Características semánticas

Los tipos que se podrán mezclar son los siguientes

- float + int
    - En este caso, los ints se tomarán como un float con un 0 después del punto
- int + float
    - En este caso, los floats perderán su valor después del punto
- int/float + bool
    - En este caso, el bool se considerará un 1 si es True, 0 si es False
- bool + int/float
    - En este caso, se suman como números, y el resultado determina el estado. 0 es False, cualquier otra cosa es True
- Char + int/float/bool | int/float/bool + char | char + char
    - Error. No es posible combinar chars con otro tipo de dato, ni hacer comparaciones

Funciones especiales

stats()

Existen varias funciones con el prefijo Stats, usualmente recibiendo dos valores, un arreglo de números (int o float)y el tamaño n de los valores a considerar para las estadísticas. Las funciones son:

- statsMin()
    - Nos regresa el número más pequeño de los datos

- statsMax()
    - Nos regresa el número más grande de los datos
- statsMean()
    - Nos regresa el promedio de los datos proporcionados
- statsMode()
    - Nos regresa la moda de los datos proporcionados
- statsMedian()
    - Nos regresa la mediana de los datos proporcionados
- statsQuantile()
    - Regresa un parámetro extra *p*, un número entre 0 y 1, y regresa el número que delimita el porcentaje de valores menores o iguales que equivale a *p*
- statsStdDev()
    - Nos regresa la desviación estándar de los datos proporcionados
- statsVariance()
    - Nos regresa la varianza de los datos proporcionados

Las siguientes funciones especiales se proponen como opcionales, pues no estoy seguro de la facilidad con la que se puede incorporar un plot o histogram a un compilador, pero la idea suena interesante y factible.

plot()

La función plot da la opción de graficar puntos y líneas en un espacio 2D, mostrando al usuario las gráficas generadas por los puntos proporcionados
Recibe tres parámetros:

- plot(x, y, n)
    - x es un arreglo representando las coordenadas x que se usarán en la gráfica
    - y es un arreglo representando las coordenadas y que se usarán en la gráfica
    - n es el número de puntos a graficar

La función tomará los primeros n pares de puntos, y los graficara como (x, y).
Existe un parámetro opcional “type”, en el que definimos si los puntos están independientes o unidos con una línea.
Además podemos dar títulos a la gráfica y a los ejes con “mainlb”, “xlb” y “ylb” respectivamente.

Se puede agregar más personalización a las gráficas cómo una stretch goal para el compilador

También se podría agregar soporte a varias líneas en la misma gráfica opcionalmente

histogram()

Esta función mostrará un histograma de datos proporcionados por el usuario. La función recibe tres parámetros:

- histogram(x, y, n)

- x es un arreglo representando los valores de cada barra utilizada en la gráfica. Puede ser numérico o carácter
- y es un arreglo representando los valores de cada barra en la gráfica
- n es el número de barras a graficar
-

Además, se puede agregar nombre al histograma con “mainlb”

Lenguaje y OS usado para desarrollo

Para el desarrollo de *Tabby*, se utilizará el lenguaje de programación *Rust*.
Una de las ventajas de Rust es su seguridad de memoria, que puede ser utilizada y transferida a *Tabby*, evitando así problemas de *memory leaks* o *segmentation errors*.

El OS para el desarrollo es MacOS, específicamente MacOS Monterey, el cuál es la 18va versión principal de MacOS

Bibliografía

http://tabatkins.github.io/railroad-diagrams/README-js.html