# Universidad Centroamericana “José Simeón Cañas”

# Facultad de Ingeniería y Arquitectura


- **Estudiantes:**

| | |
|---|---|
| • Emilio José Leiva Estrada | 00064114 |
| • Rafael Gerardo Leiva Estrada | 00065010 |
| • Luis Roberto Romualdo Menjívar | 00005513 |
| • Xavier Gerardo Figueroa Soriano | 00013812 |
| • César Samael Portillo Cabrera | 00013814 |

## Catedrático:

Néstor Santiago Aldana Rodríguez


## Presentación Informal del Lenguaje de Programación

## Trabajo presentado para la materia de

Teoría de Lenguajes de Programación

Ciclo 02/2019

- GENERALIDADES:
  - **Nombre del lenguaje:** AURIGA
  - **Nombre de IDE:** CHARIOT

- MANEJO DE VARIABLES:

Por convención de cada uno de los miembros del grupo de trabajo, se decidió que el lenguaje a ser implementado será un lenguaje de semántica fuertemente tipeada para el manejo de variables.

El inicio del programa estará identificado por el uso del lexema INICIO el cual debe ser único e independiente de uso de cualquier otro lexema o carácter reservado en su línea. El programa será finalizado por medio del uso del lexema FINCODIGO independiente de uso de cualquier otro lexema o carácter reservado en su línea.

El lenguaje AURIGA es un lenguaje de programación simple, que posee una semántica fuertemente tipeada y una sensibilidad alta de caracteres CAPITALIZABLES para el manejo de variables

Para el manejo más efectivo de las diversas variables, así como la equidad en las definiciones de las mismas, se llegó a las siguientes convenciones:

La definición de una variable comenzara con un lexema el cual estará asociado a uno de los **tipos de datos** soportados por el lenguaje, separado por un espacio por el **lexema** que se adjudicara de nombre a la variable a asignar, seguido por un identificador de asignación que será **":="** , posteriormente se encontrara el **valor** a relacionar con la variable, finalizando con el lexema de finalización de comando **";"**.

Las variables serán estrictamente **alfabéticas** y no soporta el ingreso de caracteres especiales en la conformación del nombre de la variable, cualquier otro símbolo fuera de los caracteres alfabeticos serán inválidos para ser asignados como nombre a una variable.

El lenguaje solo permitirá **una asignación por comando**, si el usuario desea asignar múltiples variables deberá aplicarlas en múltiples líneas de código separadas por el lexema de finalización de comando **";"**.

Las variables en ningún caso, las variables podrán cambiar de tipo de dato, pero si pueden cambiar de valor, las constantes se asignarán incluyendo el lexema "CONST" al principio de la asignación.

**Ejemplo de identificación:**

```
INICIO
ENT Variable := 10 ;
FLOT Variable := 12.9 ;
CAD cadena := "cadenaDeCaracteres" ;
CAR Respuesta := 'a' ;
BOOL Varo := VERDADERO ;
```

Donde variable= A+…+Z+a+…+z+_

- TIPOS DE DATOS:

AURIGA maneja una serie de identificadores a la hora de tipificar las variables que contendrán los datos que se manipularán a la hora de ejecutar el programa. Las definiciones de dichas tipificaciones están basadas en el término en español y mostradas de forma abreviada para la conveniencia del usuario.

Los tipos de datos usados por AURIGA son:

| DEFININICION | LEXEMA | Variables aceptadas | EJEMPLO |
|---|---|---|---|
| ENTEROS | ENT | Datos de tipo entero.<br>Datos con punto o coma decimal serán rechazados.<br>Datos con caracteres alfabéticos<br>Los enteros tendrán como máximo 4 dígitos | `ENT variable := 6 ;`<br>`ENT zero := 0 ;`<br>`ENT variable := 5678 ;` |
| FLOTANTES | FLOT | Datos de tipo flotante.<br>Admitidos datos con punto decimal.<br>Datos con caracteres alfanuméricos serán rechazados.<br>Los enteros tendrán como máximo 9 dígitos incluido el punto decimal | `FLOT variableA:= 7.0 ;`<br>`FLOT variableB := 566.846 ;` |

| | | | |
|---|---|---|---|
| CARACTERES | CAR | Datos de tipo carácter.<br>Admitidos caracteres individuales de tipo alfabético.<br>El dato debe estar entre comillas simples<br>El intento de poner dos caracteres será rechazado | `CAR caracter := ‘a’ ;`<br>`CAR caracterB := ‘8’ ;` |
| CADENAS | CAD | Datos de tipo cadena.<br>Admitidos caracteres individuales y múltiples de tipo alfabetico<br>El dato debe estar entre comillas<br>El intento de poner dos cadenas separadas por espacio o numeros será rechazado. | `CAD hi := “Hello” ;`<br>`CAD PIN := “JUFRT” ;` |
| BOOLEANOS | BOOL | Datos de tipo Booleano<br>Admitidos valores VERDADERO para true y FALSO para valor false.<br>Cualquier intento de asociar el dato a valores booleanos (0-1, v-f, true-false) será rechazado) | `BOOL V := VERDADERO ;`<br>`BOOL FALSE := FALSO ;` |

Las variables no admiten valores nulos

- INSTRUCCIONES DE SELECCIÓN:

AURIGA maneja una serie de instrucciones de selección a la hora de crear las diversas condiciones y resultados generados a partir del cumplimiento de dicha condición, en el flujo normal de los datos e instrucciones del programa. Las definiciones de dichas instrucciones de selección están basadas en el término en español y mostradas de forma abreviada para la conveniencia del usuario.

Por convención de cada uno de los miembros del grupo de trabajo, se decidió que El lenguaje poseerá las siguientes instrucciones de selección:

| DEFININICION | LEXEMA | Variables aceptadas | EJEMPLO |
|---|---|---|---|
| IF | SI | Evaluador inicial de la condición en la instrucción de selección. Debe cumplir una condición de tipo BOOL para decidir el resultado del flujo de los datos | SI ( variable IGUAL 5 ) {<br>SI ( Variable < Variable ) {<br>Respuesta := "esmayor";<br>RET Respuesta;<br>} |
| ELSE | SINO | capturador del flujo en los casos de que la condición en “SI” no se haya cumplido. | SINO<br>VARIABLE := 7; |
| == | IGUAL | Evaluará entre dos variables cualesquiera y verificará si estas son del mismo tipo y posteriormente evaluando si poseen el mismo valor, retornando una variable “BOOL” dependiendo el caso. | (Variable IGUAL 6)<br>VarA IGUAL VarB; |
| ENDIF | FINSI | Determinará el final de la condicional iniciada con la instrucción SI | }<br>FINSI; |

CARACTERES RESERVADOS

| LEXEMA | FUNCIO QUE APLICA |
|---|---|
| IGUAL | == |
| NOIGUAL | != |
| MAYOR | > |
| MENOR | < |

- INSTRUCCIONES ITERATIVAS:

AURIGA maneja una serie de instrucciones iterativas a la hora de determinar procesos repetitivos que cumplan una condición inicial o que se ejecuten mientras una condición exista dentro del contexto de ejecución del programa. Las definiciones de instrucciones están basadas en el término en español y mostradas de forma abreviada para la conveniencia del usuario.

| DEFININICION | LEXEMA | Variables aceptadas | EJEMPLO |
|---|---|---|---|
| FOR | PARA | ocupado para hacer iteraciones programadas para una cantidad determinada previamente de repeticiones, y dichas repeticiones serán efectuadas en una porción establecida de código | PARA ( 0 , 10 , 2 ) {<br>Respuesta := "esMayor";<br>RET Respuesta;<br>} |
| WHILE | MIENTRAS | Usado para hacer iteraciones durante el cumplimiento de una condición la cual retornará un valor del tipo “BOOL”. Dichas repeticiones serán efectuadas en una porción establecida de código. | MIENTRAS (VAR IGUAL 1) {<br>Respuesta := "es mayor";<br>RET Respuesta;<br>} |

| | | | |
|---|---|---|---|
| DO | HACER | Indicará el inicio del bloque de código que será ejecutado mientras la condición declarada en la línea de MIENTRAS sea verdad | HACER {<br>…<br>} |

- INSTRUCCIONES DE ENTRADA Y SALIDA DE DATOS:

AURIGA maneja una serie de instrucciones de entrada y salida de datos generales del sistema. La salida de datos se relaciona con las variables manejadas en el sistema o puede ser una cadena introducida de forma manual. Las definiciones de instrucciones están basadas en el término en español y mostradas de forma abreviada para la conveniencia del usuario.

| DEFININICION | LEXEMA | Variables aceptadas | EJEMPLO |
|---|---|---|---|
| SCANF | ESCR | Lexema que contendrá haciendo uso de llaves cuáles serán los valores que procederán a ser enviados a consola de código | ESCR ( “HOLAMUNDO” ) ;<br>ESCR ( variable ) ; |
| PRINTF | LECT | Lexema que contendrá haciendo uso de llaves cuáles serán los valores que procederán a ser introducidos desde consola de código. | ESCR ( “Escribateclado” ) ;<br>LECT ( tecladoIN) ; |

- CREACIÓN DE SUB-ALGORITMOS:

AURIGA maneja una serie de instrucciones para declarar funciones y subprocesos generales del sistema. Las definiciones de instrucciones están basadas en el término en español y mostradas de forma abreviada para la conveniencia del usuario.

| DEFININICION | LEXEMA | Variables aceptadas | EJEMPLO |
|---|---|---|---|
| PROCESO | PROC | Contendrá una porción de código el cual será delimitado por llaves que servirían para crear sub algoritmos para modular de forma más efectiva el código establecida de código | `PROC ( ENTRADA ){`<br>`BLOQUE`<br>`}` |
| FUNCION | FUNC | contendrá una porción de código el cual será delimitado por llaves que servirá para crear sub algoritmos para modular el código teniendo la capacidad de retornar un VALOR | `FUNC ( ENTRADA ){`<br>`BLOQUE`<br>`DEV variable ;`<br>`}` |
| RETORNAR | DEV | Servirá para el retorno de valores necesario para la porción de código delimitado por el lexema “FUNC” | `DEV VARIABLE ;` |

- MANEJO DE ARREGLOS:

AURIGA maneja una serie de instrucciones para manejo de arreglos generales del sistema. Las definiciones de instrucciones están basadas en el término en español y mostradas de forma abreviada para la conveniencia del usuario.

| DEFININICION | LEXEMA | Variables aceptadas | EJEMPLO |
|---|---|---|---|
| DEFINICION | *DEF | Contendrá una porción de código el cual será delimitado por llaves que servirían para crear sub algoritmos para modular de forma más efectiva el código establecida de código | `*DEF ENT arreglo ;` |
| DIMENSION | *DIF | contendrá una porción de código el cual será delimitado por llaves que servirá para crear sub algoritmos para modular el código teniendo la capacidad de retornar un VALOR | `*DIM arreglo 8 ;` |
| INGRESAR | *ING | Servirá para el retorno de valores necesario para la porción de código delimitado por el lexema “FUNC” | `*ING arreglo variable 6 ;` |
| LEER | *LEER | Lexema que permite sustraer un valor en una posición determinada del arreglo | `*LEER arreglo 5 ;` |

Los arreglos en el lenguaje inician por una posición 0 y estos van aumentando unitariamente. La posición de los arreglos debe ser estrictamente entero.

- MANEJO BÁSICO DE ARCHIVOS (APERTURA Y VOLCADO):

AURIGA contiene una serie de instrucciones que le otorgan la capacidad de manejar archivos. Los lexemas están basados en nomenclatura al castellano para una relación más sencilla con el usuario.

| DEFININICION | LEXEMA | Variables aceptadas | EJEMPLO |
|---|---|---|---|
| **ARIR ARCHIVO** | ABRIR | Lexema que abre un archivo, si no existe lo crea. El archivo debe existir en la misma carpeta que el ejecutable | `ABRIR ( c:/user/teo /cosa.txt ) ;` |
| **CERRAR ARCHIVO** | GUARDAR | Lexema que permite cerrar el flujo de datos del archivo. La instrucción también permite la guardado de los cambios del archivo | `GUARDAR ( c:/user/teo /cosa.txt ) ;` |
| **LEER ARCHIVO** | LEER | Lexema que permite la lectura total o parcial del archivo abierto. | `LEER ( c:/user/teo /cosa.txt ) ;` |

- FUNCIONES PREDEFINIDAS:

| | | |
|---|---|---|
| COSENO | COS | `COS (VARIABLE);` |
| SENO | SEN | `SEN (VARIABLE);` |
| TANGENTE | TAN | `TAN (VARIABLE);` |
| Pi | PI | `PI;` |
| Logaritmo neperiano | LOGE | `LOGE (VARIABLE);` |
| Potenciación | POT | `POT (VARIABLE 5);` |
| Y booleano | LOGY | `LOGY (EXPRESION EXPRESION);` |
| O booleano | LOGO | `LOGY (EXPRESION EXPRESION);` |
| Negación booleana | NEG | `NEG (EXPRESION);` |

- COMENTARIOS

AURIGA contiene una serie de caracteres especiales que permiten declarar comentarios en el código. Los comentarios de AURIGA se define de la siguiente forma:

Los comentarios de bloque los cuales apertura con el lexema #/ y cierran con el lexema /#

La línea en la que se encuentren los lexemas de apertura y cierre de comentario de bloque deben estar libres de cualquier otro lexema.

Las líneas que estén por debajo del lexema de apertura de comentario y por encima del lexema de cierre de comentario serán ignoradas por el compilador:

```
#/
ENT variableA 8;
BOOL VERDADERO;
/#
```

- # GENERALIDADES BASICAS DEL LENGUAJE

**INTRUCCION ELEMENTAL BASICA:** La asignación

**MECANISMOS DE ESTRUCTURACIÓN:** Lenguaje de tipo estructurado

**RECURSION**: Recursión directa

**ESTRUCTURA DE DATOS**: Simple y definida por tipos de valores según tipo de variable

**OPERADORES**: Operadores Aritméticos con jerarquía estándar

**INSTRUCCIONES DE I/O**: LECT ESCR

**LONGITUD DE VARIABLES**: 15 caracteres

**COMENTARIOS**: #/ /# para comentarios de bloque

**AMBITO:** Basado en C y C++

**IDENTIFICADORES:** TIPO nombreVariable valor ;

LENGUAJE CON IDENTIFICADORES **SENSIBLE A MAYUSCULAS y ALFABETICOS**

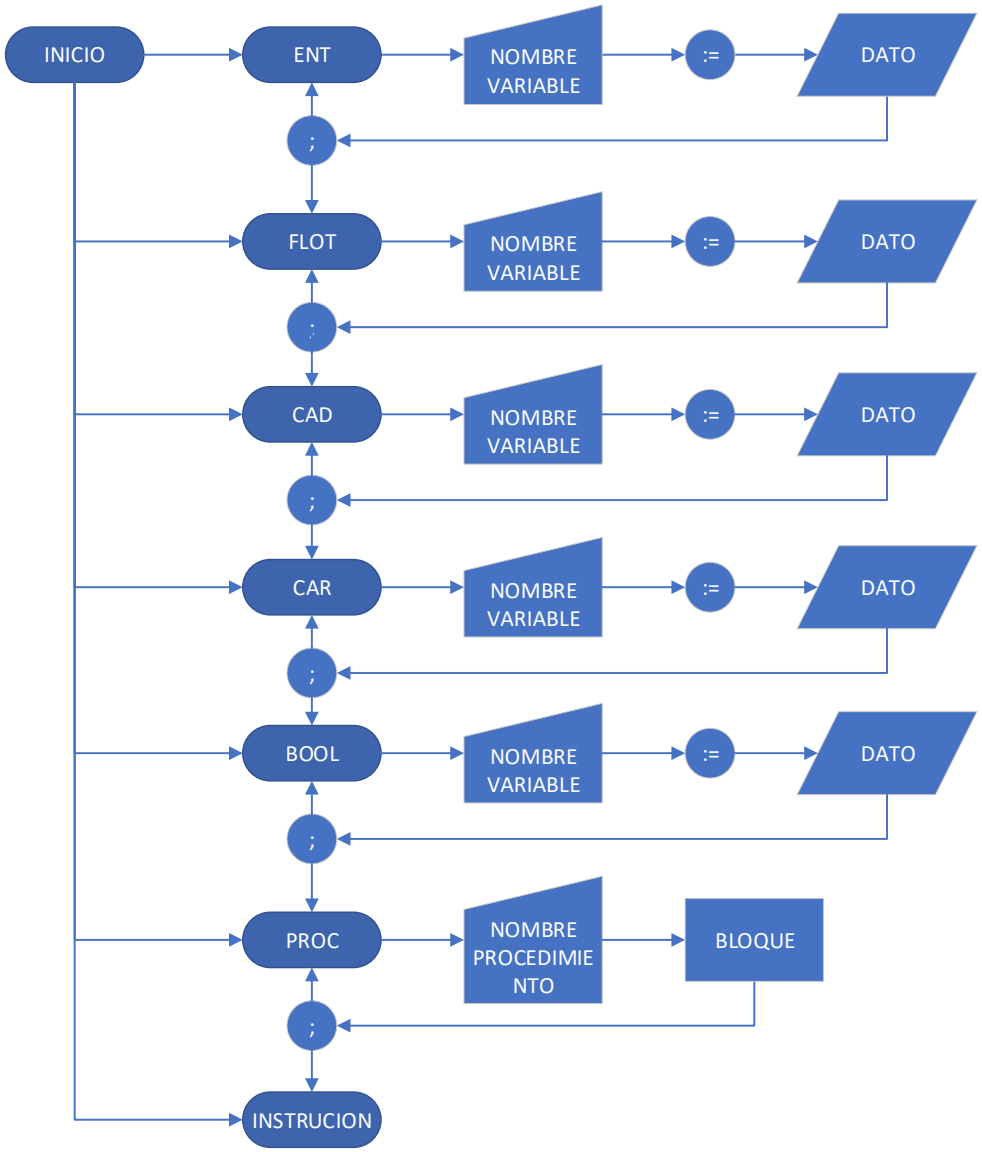
**JERARQUIA DE OPERACIONES ARITMETICAS**

| Prioridad | Operador | Significado | Ejemplo |
|---|---|---|---|
| **1** | ( ) | Paréntesis | `(2+3)*5 = 25` |
| **2** | POT | Exponenciación | `POT(5 2);` |
| | MOD | MODULO | `MOD (5 2);` |
| **3** | * | Multiplicación | `2*4 = 8` |
| | / | División | `5/2 = 2.5` |
| **4** | + | Suma | `3 + 4 = 7` |
| | - | Resta | `8 - 5 = 3` |

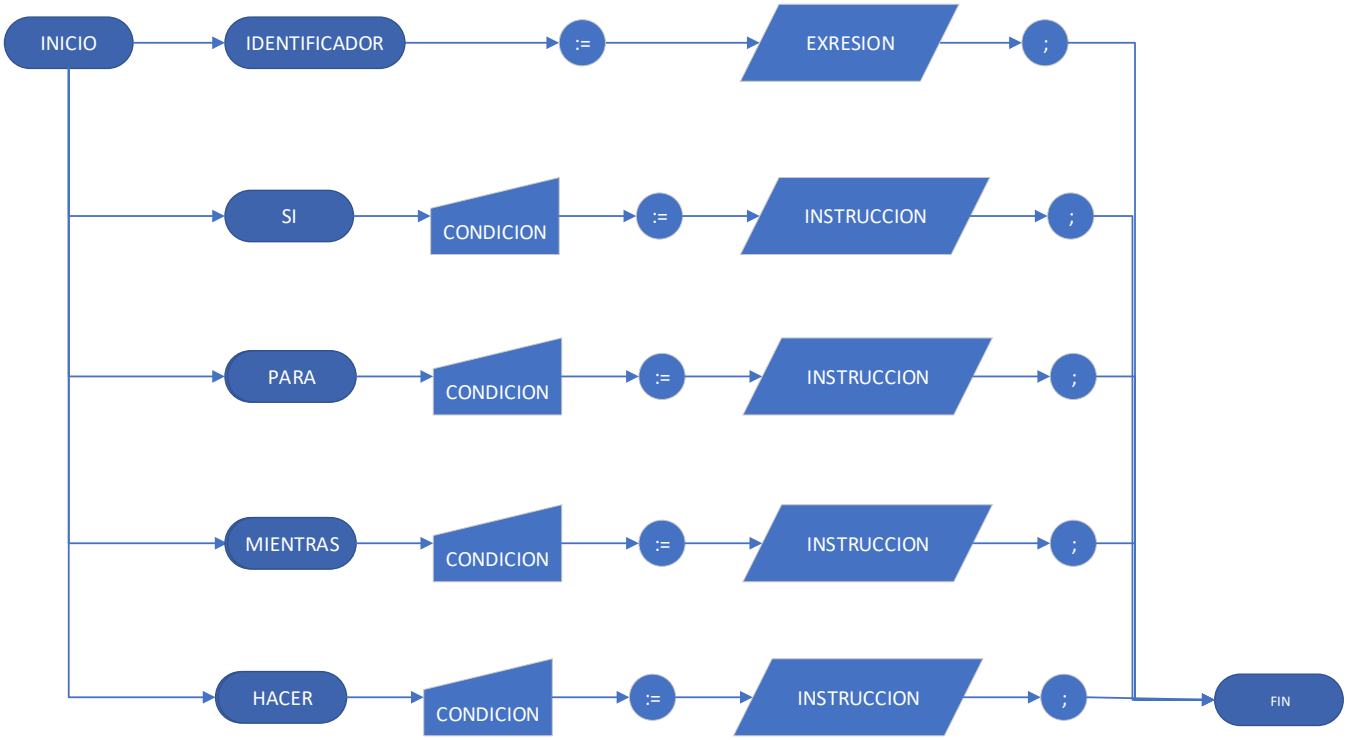
DIAGRAMAS DE SINTAXIS

# PROGRAMA

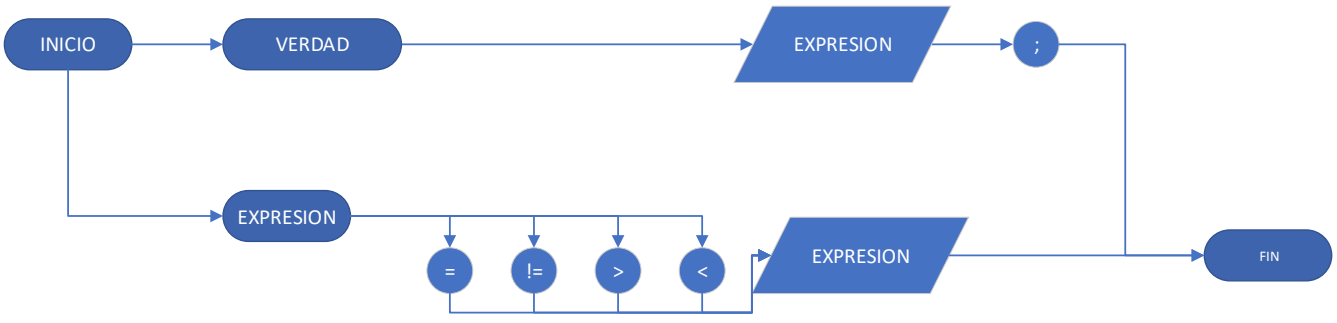

# BLOQUE



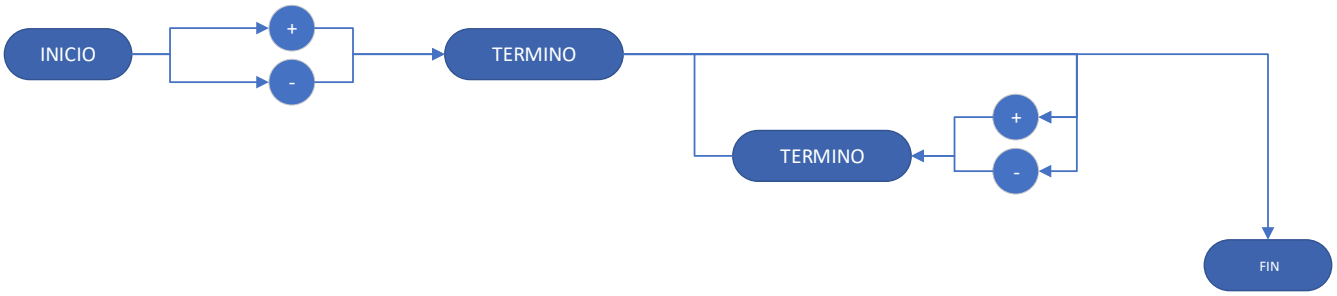
# INSTRUCCIÓN



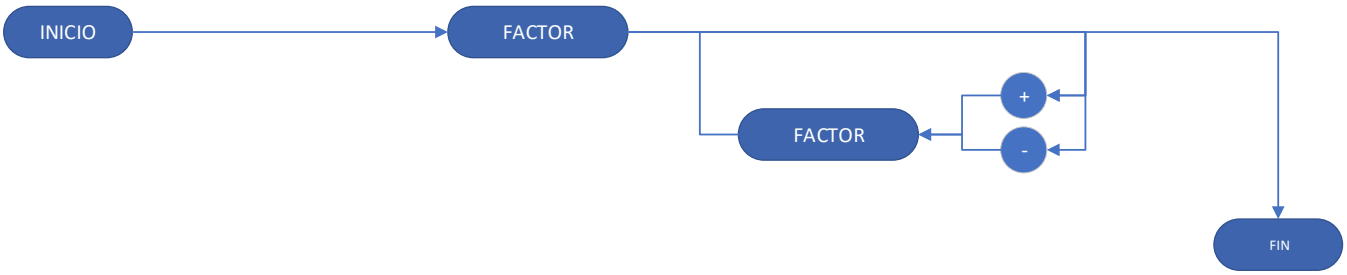

# CONDICION



## EXPRESION




## TERMINO




## FACTOR