A close up of a logo

Description automatically generated

Instituto Tecnológico de Costa Rica

Escuela de Computación

Compiladores e intérpretes

Etapa 3

Profesor:

[Kirstein](https://tecdigital.tec.ac.cr/dotlrn/community-member?user_id=409264) Gätjens Soto

Estudiantes:

David Acuña López – 2020426228

Deylan Sandoval – 2020234274

Cartago, Costa Rica

30 de mayo del 2025, I semestre

Índice

[Concurso de nombres 3](#_Toc199537188)

[Definición del lenguaje 4](#_Toc199537189)

[Índice de pruebas 33](#_Toc199537190)

[Listado de palabras reservadas 35](#_Toc199537191)

[Algoritmos de conversión 36](#_Toc199537192)

[Gramática (actualización etapa 3) 37](#_Toc199537193)

[Símbolos semánticos 45](#_Toc199537194)

[Tabla de símbolos 49](#_Toc199537195)

[Tabla de precedencia 50](#_Toc199537196)

[Listado de errores y recuperación 52](#_Toc199537197)

[Autómatas 53](#_Toc199537198)

[Autómata en código 59](#_Toc199537199)

[Instrucciones del analizador sintáctico 67](#_Toc199537200)

[Manejo del análisis semántico 68](#_Toc199537201)

[Símbolos semánticos utilizados 68](#_Toc199537202)

[Pruebas del análisis sintáctico 69](#_Toc199537203)

# Concurso de nombres

**Propuesta del nombre:** Notch Engine

**Logo:**



**Extensión:** .ne

**Descripción:** El nombre fue escogido en homenaje a Markus Persson, también conocido como Notch, creador y padre fundador de Minecraft como un título independiente, autoría de Mojang, compañía que le perteneció desde 2009 hasta 2014. El logo representa un bloque similar a los que se encuentran en el videojuego, con líneas transversales que lo atraviesan en todas sus caras, simulando los procesos y el transporte de información que se da a lo interno una vez se programa en Notch Engine.

# Definición del lenguaje

1. Estructura del título del programa: **WorldName <id>:**

Se escoge porque es el inicio del programa y da el nombre del programa, al igual que el worldname



1. Sección de constantes: **Bedrock**

Se escoge porque es un bloque que no se puede cambiar ni mover



1. Sección de tipos: **ResourcePack**

Se escoge porque porque los tipos también son los recursos para el porgrama

Texto

El contenido generado por IA puede ser incorrecto.

1. Sección de variables: **Inventory**

Se escoge porque los recursos que se guardan pueden estarse cambiando

Texto

El contenido generado por IA puede ser incorrecto.

1. Sección de prototipos: **Recipe**

Se escoge porque es como la formula o base para crear después un objeto

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

1. Sección de rutinas: **CraftingTable**

Se escoge porque es donde se puede poner los recibe que ahora si hagan el objeto

Texto

El contenido generado por IA puede ser incorrecto.

1. Punto de entrada del programa: **SpawnPoint**

Se escoge porque es el lugar donde todo empieza



1. Sistema de asignación de constantes: **Obsidian <tipo> <id> <value>**

Se escoge porque cuando se coloca una obsidiana va ser muy difícil que cambie



1. Sistema de asignación de tipos: **Anvil <id> -> <tipo>**

Se escoge porque es como colocar los objetos de herremientas

Texto

El contenido generado por IA puede ser incorrecto.

1. Sistema de declaración de variables: **<tipo> id = <lit> , id = <lit>**

Se escoge porque es una asignación común en los nuevos lenguajes

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo de dato entero: **Stack**

Se escoge porque el stack una cantidad entera de objetos

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo de dato caracter: **Rune**

Se escoge porque la runa siempre es un solo simbolo

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo de dato string: **Spider**

Se escoge porque una arana pueden unirse cosas, en este caso caracteres

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo de dato booleano: **Torch**

Se escoge porque es un objeto que esta en dos estados on y off

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo de dato conjunto: **Chest**

Se escoge porque es un lugar donde se pueden guardar datos

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo de dato archivo de texto: **Book**

Se escoge porque un libro se puede decir que es igual a un archivo de texto

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo de datos números flotantes: **Ghast**

Se escoge porque puede andar flotando por el mundo

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo de dato arreglos: **Shelf**

Se escoge porque es un lugar donde se puede ordenar/colocar cosas

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo de dato registros: **Entity**

Se escoge porque un registro se puede decir que es una entidad

Texto

El contenido generado por IA puede ser incorrecto.

1. Literales booleanas: **On/Off**

Se escoge porque son los dos estados de la lampara

Texto

El contenido generado por IA puede ser incorrecto.

1. Literales de conjuntos: **{: :}**

Se escoge porque un delimitador común es { } y para diferenciar de los otros tipos se agrega :

Texto

El contenido generado por IA puede ser incorrecto.

1. Literales de archivos: **{/ /}**

Se escoge porque un delimitador común es { } y para diferenciar de los otros tipos se agrega /

Texto

El contenido generado por IA puede ser incorrecto.

1. Literales de números flotantes: **-3.45**
   1. Se escoge porque es una forma convencional de redactar números, con – para negativos y . para diferenciar cuando se varios parámetros

Texto

El contenido generado por IA puede ser incorrecto.

1. Literales de enteros: **5,-5**

Se escoge porque es la forma normal de hacerlo

Texto

El contenido generado por IA puede ser incorrecto.

1. Literales de caracteres: **'K'**

Se escoge porque en los nuevos lenguajes los caracteres se represetan con una comilla sencilla

Texto

El contenido generado por IA puede ser incorrecto.

1. Literales de strings: **"Hola Mundo"**

Se escoge porque la forma común en todos los lenguajes se hace con comillas dobles

Texto

El contenido generado por IA puede ser incorrecto.

1. Literales de arreglos: **[ ]**

Se escoge porque es igual a la nomenclatura en java

Texto

El contenido generado por IA puede ser incorrecto.

1. Literales de registros: **{<id>: <value>, <id>: <value>, …}**

Se escoge porque un delimitador común es { } y con una estructura comun a los lenguajes modernos

Texto

El contenido generado por IA puede ser incorrecto.

1. Sistema de acceso arreglos: **[#]**

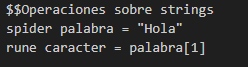
Se escoge porque es la nomenclatura común en los lenguajes alto nivel

Texto

El contenido generado por IA puede ser incorrecto.

1. Sistema de acceso strings: **[#]**

Se escoge porque es una forma común en los lenguajes modernos



1. Sistema de acceso registros: **'@ (e.g.: registro@campo)**

Se escoge porque @ es un carácter común y que no se usan en otro lado

Texto

El contenido generado por IA puede ser incorrecto.

1. Asignación y Familia: **'= (i.e: =, +=, -=, \*=, /=, %=)**

Se escogen porque es la forma similar a Python y de fácil entendimiento

Texto

El contenido generado por IA puede ser incorrecto.

1. Operaciones aritméticas básicas de enteros: **+ - \* % //**

Se escoge porque son la nomenclatura básica de símbolos en Python

Texto

El contenido generado por IA puede ser incorrecto.

1. Incremento y Decremento: **soulsand, magma**

Se escoge porque soulsand te baja la velocidad y elmagma en el agua te hace subir

Texto

El contenido generado por IA puede ser incorrecto.

1. Operaciones básicas sobre caracteres: **isEngraved, isInscribed, etchUp, etchDown**

Se escoge porque es una forma de fácil entendimiento y esta relacionado con minecraft

Texto

El contenido generado por IA puede ser incorrecto.

1. Operaciones lógicas solicitadas: **and, or, not, xor**

Se escoge porque son los nombres con los operadores con letras

Texto

El contenido generado por IA puede ser incorrecto.

1. Operaciones de Strings solicitadas: **bind, #, from ##,##, seek**

Se escoge porque es una forma fácil de entender que hacen las operaciones:

concatenar, largo, cortar, recortar, buscar. Cortar y Recortar son ternarias y usan doble # para evitar ambigüedad con el length

Texto

El contenido generado por IA puede ser incorrecto.

1. Operaciones de conjuntos solicitadas: **add, drop, items, feed, map, kill**

Se escoge porque es una forma fácil de entender que hacen las operaciones:

agregar, eliminar, unión, intersección, pertenece, vacío

Texto

El contenido generado por IA puede ser incorrecto.

1. Operaciones de archivos solicitadas: **unlock, lock, craft, gather, forge**

Se escoge porque es una forma fácil de entender que hacen las operaciones:

abrir, cerrar, crear, leer, escribir

Texto

El contenido generado por IA puede ser incorrecto.

1. Operaciones de números flotantes: **:+, :-, :\*, :%, ://**

Se escoge porque porque es la nomenclatura básica en Python, y se usa : para diferenciarlo de los enterops

Texto

El contenido generado por IA puede ser incorrecto.

1. Operaciones de comparación solicitadas:  **<, >, <=, >=, is, isNot**

Se escoge porque es la nomenclatura común en los lenguajes alto nivel

Texto

El contenido generado por IA puede ser incorrecto.

1. Manejo de Bloques de más de una instrucción: **PolloCrudo PolloAsado**

Se escoge porque al inicio está crudo y, al final del proceso, cocido o asado

Texto

El contenido generado por IA puede ser incorrecto.

1. Instrucción while: **repeater <cond> craft <instrucción>**

Se escoge porque un repetidor manda señales de redstone constantemente Ojo con el craft que es parte de la instrucción

Texto

El contenido generado por IA puede ser incorrecto.

1. Instrucción if-then-else: **target <cond> craft hit <inst> miss <inst>**

Se escoge porque similar a cómo el bloque objetivo evalúa si se dio en el centro o no. Recuerden que el hit y el miss pueden ir en orden inverso y ambos son opcionales, pero al menos debe haber uno.

Texto

El contenido generado por IA puede ser incorrecto.

1. Instrucción switch: **jukebox <condition> craft , disc <case> : , silence**

Se escoge porque similar a la caja tocadiscos que tiene varias opciones para seleccionar canciones. Le agregué el craft y cambié el default por silence. Recuerden que el silence puede ir en cualquier lugar y que es obligatorio usar el polloCrudo y polloAsado

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

1. Instrucción Repeat-until: **spawner <instrucciones> exhausted <cond> ;**

Se escoge porque similar al generador de monstruos que spawnea criaturas hasta que se cumpla una condición

Texto

El contenido generado por IA puede ser incorrecto.

1. Instrucción For: **walk VAR set <exp> to <exp> step <exp> craft <instrucción>**

Se escoge porque caminar es repetir un a cantidad de pasos

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

1. Instrucción With: **wither <Referencia a Record> craft <instrucción>**

Se escoge porque enemigo que de casualidad tiene un nombre similar a with

1. Instrucción break: **creeper**

Se escoge porque puede explotar y salir de donde esta

Texto

El contenido generado por IA puede ser incorrecto.

1. Instrucción continue: **enderPearl**

Se escoge porque saltar el resto de una iteración es similar a teletransportarse a Ender Pearl

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

1. Instrucción Halt: **ragequit**

Se escoge porque ragequit transmite la idea de terminar repentinamente la ejecución, como abandonar el juego por enojo

Texto

El contenido generado por IA puede ser incorrecto.

Este es el mismo ejemplo que el anterior pero con la instrucción halt en medio que detiene todo el programa.

1. Encabezado de funciones: **Spell <id>(<parameters>) -> <tipo>**

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

1. Encabezado de procedimientos: **Ritual <id>(<parameters>)**

Se escoge porque es un procedimiento que solo invoca

Texto

El contenido generado por IA puede ser incorrecto.

Manejo de parámetros formales: **( <type> :: <name>, <name>; <type> ref <name>; … )**

Se escoge porque es la nomenclatura común en los lenguajes de alto nivel

Texto

El contenido generado por IA puede ser incorrecto.

1. Manejo de parámetros reales: **(5,A,4,B)**

Se escoge porque es la nomenclatura común en los lenguajes de alto nivel

Texto

El contenido generado por IA puede ser incorrecto.

1. Instrucción return: **respawn**

Se escoge porque como devolver cuando se termina

Texto

El contenido generado por IA puede ser incorrecto.

1. Operación de size of: **chunk <exp> o <tipo>**

Se escoge porque un chunk es como la medida para el mundo

Una captura de pantalla de un celular con texto e imagen

El contenido generado por IA puede ser incorrecto.

1. Sistema de coherción de tipos: **<exp> >> <tipo>**

Se escoge porque es una forma intuitiva de hacerlo

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

1. Manejo de la entrada estándar: **x = hopper<TipoBásico>()**

Se escoge porque una Hopper recibe objetos

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

1. Manejo de la salida estándar: **dropper<tipoBásico>(dato)**

Se escoge porque un objeto que suelta cosas

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

1. Terminador o separador de instrucciones - Instrucción nula: **;**

Se escoge porque es la nomenclatura común en los lenguajes

Texto

El contenido generado por IA puede ser incorrecto.

1. Todo programa se debe cerrar con un: **worldSave**

Se escoge porque cuando sales del mundo en minecraft lo guardas

Texto

El contenido generado por IA puede ser incorrecto.

1. Comentario de Bloque: **$\* comentario \*$**

Se escoge porque un es un símbolo fácil de escribir y diferenciar

Texto

El contenido generado por IA puede ser incorrecto.

1. Comentario de Línea: **$$ comentario**

Se escoge porque un es un símbolo fácil de escribir y diferenciar

Texto

El contenido generado por IA puede ser incorrecto.

1. Tipo creativo: temperatura

* Este tipo de dato sirve para manipular información de temperaturas en grados celcius, Fahrenheit y kelvin.

A black background with white text

AI-generated content may be incorrect.

1. Literal para el tipo creativo:

* Las literales para el tipo creativos son números enteros normales, simplemente se le añade una C, F o K según el tipo de temperatura que sea.

A screenshot of a computer

AI-generated content may be incorrect.

**Operaciones**

**Transformar:** transf<entrada><salida>(valor)

* Se utiliza el transf seguido de las letras en mayúscula de la temperatura de entrada y la temperatura de salida
* Transforma un tipo de temperatura a otra. Es importante mencionar que no es lo mismo que simplemente escribir el otro tipo de temperatura en la misma variable ya que no se haría la conversión
* Este es un ejemplo de Celsius a Kelvin

A black background with white text

AI-generated content may be incorrect.

**Ajustar:** ajus<entrada><salida>(<valor1>, <valor2>)

* Se utiliza el ajus seguido de las letras en mayúscula de la temperatura de entrada y la temperatura de salida
* Este permite sumar dos tipos de temperaturas distintas

A black screen with white text

AI-generated content may be incorrect.

**Reducir:** red<temp entrada><temp salida>(<exp1>, <exp2>)

* Se utiliza el red seguido de las letras en mayúscula de la temperatura de entrada y la temperatura de salida
* Este permite saber la diferencia entre dos tipos de temperaturas

A black screen with white text

AI-generated content may be incorrect.

**Comparación:** comp<entrada><salida>(<valor1>, <valor2>)

* Se utiliza el comp seguido de las letras en mayúscula de la temperatura de entrada y la temperatura de salida
* Este permite comparar dos tipos de temperaturas para saber si son iguales.

**A screen shot of a computer

AI-generated content may be incorrect.**

1. Dos Operaciones adicionales sobre el tipo entero:
2. **Promedio de temperaturas:** promtemp<entrada>(<valor1>, <valor1>, …)

* Este nos permite sacar un promedio de un tipo de temperatura
* Las temperaturas deben de ser de un mismo tipo

A screen shot of a computer

AI-generated content may be incorrect.

1. **Diferencia absoluta:** difabs<entrada><salida>(<valor1>, <valor2>)
   * Este nos permite sacar un promedio de un tipo de temperatura
   * Las temperaturas deben de ser de un mismo tipo

A black background with white text

AI-generated content may be incorrect.

1. Dos Instrucciones que no sean fácilmente implementables con lo que ya tenga el lenguaje:
2. **Sensación térmica:** termtemp<entrada>(<dato1>, <dato2>, <factor>)
3. **Interpolación de temperaturas:** interptemp<entrada>(<dato1>, <dato2>, <factor>)

* Permite interpolar una temperatura entre dos puntos y un factor de interpolación entre 0 y 1

A black background with white text

AI-generated content may be incorrect.

# Índice de pruebas

* test-mcr-01-Programa inicial
* test-mcr-02-Seccion de constantes
* test-mcr-03-Seccion de tipos
* test-mcr-04-Seccion de variables
* test-mcr-05-Prototipo
* test-mcr-06-Funcion
* test-mcr-07-Arreglo
* test-mcr-08-Operaciones de string
* test-mcr-09-Registro
* test-mcr-10-Asignacion y familia
* test-mcr-11-Operaciones enteros
* test-mcr-12-Incremento y decrement
* test-mcr-13-Operaciones caracteres
* test-mcr-14-Operaciones conjuntos
* test-mcr-15-Operaciones archivos
* test-mcr-16-Operaciones flotantes
* test-mcr-17-Operaciones comparacion
* test-mcr-18-While con if-else
* test-mcr-19-For con switch
* test-mcr-20-Repeat until
* test-mcr-21-Procedimiento
* test-mcr-22-Creativo
* test-mcr-23-Logica
* test-mcr-24-Halt-Sizeof

# Listado de palabras reservadas

**Elementos:**

WorldName           Bedrock           ResourcePack           Inventory           Recipe           CraftingTable           SpawnPoint           Obsidian           Anvil           Stack           Rune           Spider           Torch           Chest           Book           Ghast           Shelf           Entity           soulsand           magma           isEngraved           isInscribed           etchUp           etchDown           bind           from           except           seek           add           drop           items           feed           map           biom           kill           unlock           lock           craft           gather           forge           expand           PolloCrudo           PolloAsado           repeater           craft           target           hit           miss           jukebox           disc           silence           spawner           walk           set           wither           creeper           enderPearl           ragequit           Spell           Ritual           respawn           chunk           hopper           dropper           worldSave  magma soulsand

**Operadores binarios:**

=           +=           \*=           -=         %=           /=

+           -           \*           %           //

%+           %-           %\*           %/           %%

>           <           >=           <=           is       isNot

**Operadores ternarios**

**Comentarios**

$$               $\* \*$

# Algoritmos de conversión

Tipo Stack (int)

Spider -> Stack: El string tiene que contener solo caracteres que sean un digito los crea en un numero entero, y no es válido devuelve un error

Ej:



Rune -> Stack: devuelve el valor ascii del carácter ingresado

Ej:



Torck -> Stack: Si el dato ingresado es 0, devuelve un cero, cualquier otro dato será un 1

Ej:



# Gramática (actualización etapa 3)

<inicio> ::= WORLDNAME #CrearTLG <identificador> : <seccion>

<seccion> ::= <seccion1\_bedrock> <seccion1\_resourcepack> <seccion1\_inventory> <seccion1\_recipe> <seccion1\_craftingtable> <punto\_entrada> <final>

<seccion> ::=

<seccion1\_bedrock> ::= BEDROCK <sistema\_asignacion\_constante>

<seccion1\_bedrock> ::=

<seccion1\_resourcepack> ::= RESOURCEPACK <sistema\_asignacion\_tipos>

<seccion1\_resourcepack> ::=

<seccion1\_inventory> ::= INVENTORY <sistema\_asignacion\_variables>

<seccion1\_inventory> ::=

<seccion1\_recipe> ::= RECIPE <encabezado>

<seccion1\_recipe> ::=

<seccion1\_craftingtable> ::= CRAFTINGTABLE <seccion1\_craftingtable2>

<seccion1\_craftingtable> ::=

<seccion1\_craftingtable2> ::= <identificador> <bloque> <seccion1\_craftingtable2>

<seccion1\_craftingtable2> ::=

<sistema\_asignacion\_constante> ::= OBSIDIAN #AlmacenarTipoConstante <tipo> #ValidarExistenciaIdentificadorConstante <identificador> = #ValidarTipoValorConstante <valor> <terminador> <sistema\_asignacion\_constante>

<sistema\_asignacion\_constante> ::=

<sistema\_asignacion\_tipos> ::= ANVIL #ValidarExistenciaIdentificadorTipo <identificador> -> #ValidarValorTipo <tipo> <terminador> <sistema\_asignacion\_tipos>

<sistema\_asignacion\_tipos> ::=

<sistema\_asignacion\_variables> ::= #AlmacenarTipoVariable <tipo> <sistema\_asignacion\_variables2>

<sistema\_asignacion\_variables> ::= #AlmacenarTipoVariable <identificador> <sistema\_asignacion\_variables2>

<sistema\_asignacion\_variables> ::=

<sistema\_asignacion\_variables2> ::= <lista\_ids\_valores> #BorrarTipoVariable <terminador> <sistema\_asignacion\_variables>

<encabezado> ::= SPELL #AlmacenarFuncion <identificador> <parentesis> -> #AlmacenarTipoFuncion <tipo> <terminador> <encabezado>

<encabezado> ::= RITUAL #AlmacenarProcedimiento <identificador> <parentesis> #BorrarIdentificador <terminador> <encabezado>

<encabezado> ::=

<parentesis> ::= ( <parametros> )

<parametros> ::= <parametros\_formales>

<parametros> ::= <parametros\_reales>

<parametros> ::=

<parametros\_formales> ::= #AlmacenarTipoParametroFormal <tipo> :: #ValidarExistenciaIdentificadorParametroFormal <identificador> <mas\_parametros\_formales>

<mas\_parametros\_formales> ::= , #AlmacenarTipoParametroFormal <tipo> :: #ValidarExistenciaIdentificadorParametroFormal <identificador> <mas\_parametros\_formales>

<mas\_parametros\_formales> ::=

<parametros\_reales> ::= <identificador> <mas\_parametros\_reales>

<mas\_parametros\_reales> ::= , <identificador> <mas\_parametros\_reales>

<mas\_parametros\_reales> ::=

<lista\_ids\_valores> ::= #ValidarExistenciaIdentificadorVariable <identificador> = #ValidarTipoValorVariable <valor> <mas\_lista\_ids\_valores>

<mas\_lista\_ids\_valores> ::= , #ValidarExistenciaIdentificadorVariable <identificador> = #ValidarTipoValorVariable <valor> <mas\_lista\_ids\_valores>

<mas\_lista\_ids\_valores> ::=

<tipo> ::= STACK

<tipo> ::= RUNE

<tipo> ::= SPIDER

<tipo> ::= TORCH

<tipo> ::= CHEST

<tipo> ::= BOOK

<tipo> ::= GHAST

<tipo> ::= SHELF

<tipo> ::= ENTITY

<literal\_booleano> ::= ON

<literal\_booleano> ::= OFF

<literal\_flotante> ::= LITERAL\_FLOTANTE

<literal\_entero> ::= LITERAL\_ENTERO

<literal\_caracter> ::= LITERAL\_CARACTER

<literal\_string> ::= LITERAL\_STRING

<literal\_arreglo> ::= LITERAL\_ARREGLO

<literal\_registro> ::= LITERAL\_REGISTRO

<punto\_entrada> ::= SPAWNPOINT <bloque>

<valor> ::= <literal\_booleano>

<valor> ::= <literal\_flotante>

<valor> ::= <literal\_entero>

<valor> ::= <literal\_caracter>

<valor> ::= <literal\_string>

<valor> ::= <literal\_arreglo>

<valor> ::= <literal\_registro>

<terminador> ::= ;

<instruccion> ::= <bloque>

<instruccion> ::= <variable> <instruccion>

<instruccion> ::= ANVIL <identificador> -> <tipo> <terminador> <sistema\_asignacion\_tipos>

<instruccion> ::=

<bloque> ::= POLLOCRUDO <instruccion> POLLOASADO <terminador>

<operador> ::= +

<operador> ::= -

<operador> ::= \*

<operador> ::= /

<operador> ::= %

<operacion\_incre\_decre> ::= SOULSAND

<operacion\_incre\_decre> ::= MAGMA

<incre\_decre> ::= <operacion\_incre\_decre> <identificador> <terminador>

<operacion\_caracter> ::= ISENGRAVED

<operacion\_caracter> ::= ISINSCRIBED

<operacion\_caracter> ::= ETCHUP

<operacion\_caracter> ::= ETCHDOWN

<operacion\_logica> ::= AND

<operacion\_logica> ::= OR

<operacion\_logica\_not> ::= NOT

<operacion\_logica> ::= XOR

<operacion\_string> ::= BIND

<operacion\_string> ::= FROM

<operacion\_string> ::= EXCEPT

<operacion\_string> ::= SEEK

<operacion\_flotante> ::= %

<operacion\_comparacion> ::= <

<operacion\_comparacion> ::= >

<operacion\_comparacion> ::= <=

<operacion\_comparacion> ::= >=

<operacion\_comparacion> ::= IS

<operacion\_comparacion> ::= ISNOT

<operacion\_sizeof> ::= CHUNK <tipo\_sizeof>

<tipo\_sizeof> ::= <identificador>

<tipo\_sizeof> ::= <tipo>

<variable> ::= #AntesVerificarAsignacion <identificador> <variable2> #VerificarAsignacion

<variable2> ::= = <variable\_asignacion>

<variable2> ::= <asignacion\_operando> <variable\_asignacion\_operando>

<variable2> ::= <operacion\_incre\_decre> <terminador>

<variable2> ::= <operacion\_sizeof> <terminador>

<variable\_asignacion> ::= <expresion\_booleana> <terminador>

<variable\_asignacion> ::= <expresion\_caracter> <terminador>

<variable\_asignacion> ::= <expresion\_string> <terminador>

<variable\_asignacion> ::= <expresion\_arreglo> <terminador>

<variable\_asignacion> ::= <expresion\_registro> <terminador>

<variable\_asignacion> ::= <expresion\_flotante> <terminador>

<variable\_asignacion> ::= <expresion\_entera> <terminador>

<variable\_asignacion\_operando> ::= <expresion\_flotante> <terminador>

<variable\_asignacion\_operando> ::= <expresion\_entera> <terminador>

<expresion\_booleana> ::= <literal\_booleano>

<expresion\_flotante> ::= <literal\_flotante> <mas\_expresion\_flotante>

<mas\_expresion\_flotante> ::= <operacion\_flotante> <literal\_flotante> <mas\_expresion\_flotante>

<mas\_expresion\_flotante> ::=

<expresion\_entera> ::= <literal\_entero> <mas\_expresion\_entera>

<mas\_expresion\_entera> ::= <operador> <literal\_entero> <mas\_expresion\_entera>

<mas\_expresion\_entera> ::= <operacion\_logica> <literal\_entero> <mas\_expresion\_entera>

<mas\_expresion\_entera> ::= <operacion\_comparacion> <literal\_entero> <mas\_expresion\_entera>

<mas\_expresion\_entera> ::=

<expresion\_caracter> ::= <literal\_caracter> <mas\_expresion\_caracter>

<mas\_expresion\_caracter> ::= <operacion\_caracter> <literal\_caracter> <mas\_expresion\_caracter>

<mas\_expresion\_caracter> ::=

<expresion\_string> ::= <literal\_string> <mas\_expresion\_string>

<mas\_expresion\_string> ::= <operacion\_string> <literal\_string> <mas\_expresion\_string>

<mas\_expresion\_string> ::=

<expresion\_arreglo> ::= <literal\_arreglo>

<expresion\_registro> ::= <literal\_registro>

<asignacion\_operando> ::= <operador> =

<identificador> ::= IDENTIFICADOR

<final> ::= WORLDSAVE

<instruccion> ::= <incre\_decre> <instruccion>

<instruccion> ::= <ciclo\_while> <instruccion>

<instruccion> ::= <ciclo\_for> <instruccion>

<instruccion> ::= <ciclo\_repeat\_until> <instruccion>

<instruccion> ::= <instruccion\_halt> <terminador> <instruccion>

<instruccion> ::= <instruccion\_break> <terminador> <instruccion>

<instruccion> ::= <instruccion\_continue> <terminador> <instruccion>

<instruccion> ::= <instruccion\_switch> <terminador> <instruccion>

<instruccion> ::= <instruccion\_with> <instruccion>

<instruccion> ::= <instruccion\_dropper> <terminador> <instruccion>

<instruccion> ::= <instruccion\_return> <instruccion>

<instruccion> ::= <if\_then\_else> <instruccion>

<instruccion\_halt> ::= RAGEQUIT <terminador>

<instruccion\_break> ::= CREEPER <terminador>

<instruccion\_continue> ::= ENDERPEARL <terminador>

<instruccion\_dropper> ::= DROPPER <tipo> ( <identificador> )

<instruccion\_return> ::= RESPAWN <instruccion\_return2>

<instruccion\_return2> ::= <terminador>

<instruccion\_return2> ::= <valor> <terminador>

<instruccion\_return2> ::= <identificador> <terminador>

<ciclo\_while> ::= REPEATER <condicion> CRAFT <bloque>

<ciclo\_for> ::= WALK <identificador> SET <literal\_entero> TO <literal\_entero> <ciclo\_for2>

<ciclo\_for2> ::= STEP <literal\_entero> CRAFT <bloque>

<ciclo\_for2> ::= CRAFT <bloque>

<ciclo\_repeat\_until> ::= SPAWNER <bloque> EXHAUSTED <condicion> <terminador>

<instruccion\_with> ::= WITHER <identificador> CRAFT <bloque>

<instruccion\_switch> ::= JUKEBOX <condicion> CRAFT <bloque\_switch>

<bloque\_switch> ::= POLLOCRUDO <casos\_switch> <caso\_switch\_final> POLLOASADO

<casos\_switch> ::= DISC <literal\_entero> : <instruccion> <casos\_switch>

<casos\_switch> ::=

<caso\_switch\_final> ::= SILENCE <instruccion>

<condicion> ::= <valor\_condicion> <operacion\_condicion> <valor\_condicion> <mas\_condicion>

<condicion> ::= <operacion\_logica\_not> <valor\_condicion> <operacion\_condicion> <valor\_condicion> <mas\_condicion>

<mas\_condicion> ::= <operacion\_condicion> <mas\_condicion\_con\_sin\_not>

<mas\_condicion> ::=

<mas\_condicion\_con\_sin\_not> ::= <operacion\_logica\_not> <valor\_condicion> <mas\_condicion>

<mas\_condicion\_con\_sin\_not> ::= <valor\_condicion> <mas\_condicion>

<operacion\_condicion> ::= <operacion\_comparacion>

<operacion\_condicion> ::= <operacion\_logica>

<valor\_condicion> ::= <identificador>

<valor\_condicion> ::= <valor>

<valor\_condicion> ::= ( <tipo\_expresion> )

<tipo\_expresion> ::= <expresion\_entera>

<tipo\_expresion> ::= <expresion\_flotante>

<tipo\_expresion> ::= <expresion\_string>

<if\_then\_else> ::= TARGET <condicion> MISS <bloque> <else>

<else> ::= HIT <bloque>

<else> ::=

# Símbolos semánticos

**CrearTLG**

Se encarga de al inicio del programa crear la tabla de símbolos para el alcance global

**AlmacenarTipoConstante**

Se encarga se identificar y almacenar el tipo de datos que se le desea asignar al identificador del siguiente símbolo numérico semántico, que se espera que sea con una variable

**ValidarExistenciaIdentificadorConstante**

Se encarga de recibir y analizar el identificador de la constante, verificando que no exista ya en la tabla de símbolos, de no ser así lo crea con el tipo anteriormente almacenado y la categoría constante

**ValidarTipoValorConstante**

Se encarga de recibir y analizar el valor que se le quiere dar a la constante validando que sea una literal del mismo tipo y asignándola al identificador correspondiente de la tabla de símbolos

**ValidarExistenciaIdentificadorTipo**

Se encarga de recibir y analizar el identificador del tipo de la sección de tipos, verificando que no exista ya en la tabla de símbolos, de no ser así lo crea la categoría tipo

**ValidarValorTipo**

Se encarga de recibir y analizar el valor que se le quiere dar al tipo validando que sea una literal del mismo tipo y asignándola al identificador correspondiente de la tabla de símbolos

**AlmacenarTipoVariable**

Se encarga se identificar y almacenar el tipo de datos que se le desea asignar al identificador del siguiente símbolo semántico, que se espera que sea con una o varias variables

**ValidarExistenciaIdentificadorVariable**

Se encarga de recibir y analizar el identificador de la variable, verificando que no exista ya en la tabla de símbolos, de no ser así lo crea con el tipo anteriormente almacenado y la categoría variable

**ValidarTipoValorVariable**

Se encarga de recibir y analizar el valor que se le quiere dar a la variable validando que sea una literal del mismo tipo y asignándola al identificador correspondiente de la tabla de símbolos

**BorrarTipoVariable**

Se encarga de borrar el tipo de variable que se almaceno anteriormente para la creación de las variables

**AlmacenarFuncion**

Se encarga de recibir y analizar el identificador de una función, verificando que no exista ya en la tabla de símbolos, de no ser así lo crea la categoría función, guardando el identificador para el posterior uso en la creación de los parámetros formales y el retorno de la función

**AlmacenarTipoFuncion**

Se encarga de recibir y analizar el valor de retorno que se le quiere dar a la función, asignándola al identificador correspondiente de la tabla de símbolos

**AlmacenarProcedimiento**

Se encarga de recibir y analizar el identificador del procedimiento, verificando que no exista ya en la tabla de símbolos, de no ser así lo crea la categoría procedimiento, guardando el identificador para el posterior uso en la creación de los parámetros formales.

**BorrarIdentificador**

Se encarga de borrar identificador de función que se almaceno anteriormente para la asignación del valor de retorno

**AlmacenarTipoParametroFormal**

Se encarga se identificar y almacenar el tipo de dato que se le desea asignar al identificador del siguiente símbolo numérico semántico, que se espera que sea con un parámetro formal

**ValidarExistenciaIdentificadorParametroFormal**

Se encarga de recibir y analizar el identificador del tipo del parámetro formal, verificando que no exista ya en los parámetros formales de la función/procedimiento y , de no ser así lo crea la categoría parámetro formal, como un atributo de la función/procedimiento

**VerificarAsignacion**

Se encarga de hacer las validaciones de la familia se asignación, pudiendo ser con o sin operadores, valida que el primer argumento sea un identificador que exista en la tabla de símbolos, si existe que sea de la categoría variable, también revisa si es una asignación con un operador para que de ser también revise que el identificador sea un stack, luego por ultimo revisa que el segundo parámetro sea una literal del mismo tipo que el primer identificador

**ValidarExistenciaIdentificadorParametroReal**

Se encarga de recibir y validar que exista el identificador del parámetro real, así también de comprobar que el dato enviado sea del mismo tipo que el parámetro formal ya establecido

**ValidarIdentificadorIncrementoDecremento**

Se encarga de recibir y validar que exista el identificador a decrementar o incrementar, así también de comprobar que el dato enviado sea un stack

**ValidarOperacionCaracter**

Se encarga de comprobar que la literal antes de la operación de carácter se una literal, variable, constante valida de carácter

**ValidarOperacionEntero**

Se encarga de comprobar que la literal antes y después de la operación de entero se una literal, variable, constante valida de entero

**ValidarOperacionString**

Se encarga de comprobar que la literal antes y después de la operación de string se una literal, variable, constante valida de string

**ValidarOperacionComparacion**

Se encarga de comprobar que la literal antes y después de la operación de comparación se una literal, variable, constante válida para compara y que sean del mismo tipo

**ValidarOperacionLogica**

Se encarga de comprobar que la literal antes y después de la operación de logica se una literal, variable, constante valida de lógica

**EmpezarAnalizarReturn**

Se encarga de empezar la variable para ver si ha y un return en false y almacenar la función actual

**SiHayReturn**

Se encarga de guardar en la variable return un true para decir que en la función actual hay un return

**ComprobarReturn**

Se encarga de comprobar si en la función anterior hay un return

**ValidarIdentificadorForRecorrido**

Se encarga se comprobar que el identificador para correr el for sea aceptable, y exista para usar como variable

**ValidarLiteralEnteraFor**

Se encarga de validar que la literal que viene sea de tipo stack, se usa para set, to y step del for

**ValidarTopeFor**

Se encarga de comprobar si el for ya llego al final, comparando el recorrido con el tope

**ValidarIdentificadorWith**

Se encarga se comprobar que el identificador para el with sea aceptable, y exista para usar como variable

**EmpezarAnalizarDefaultSwitch**

Se encarga de empezar la variable para ver si hay un default en false y el switch actual

**HayDefaultSwitch**

Se encarga de guardar en la variable default un true para decir que en el siwtch actual si hay default

**SiHayDefaultSwitch**

Se encarga de comprobar si en el switch actual si hay default

**ValidarCondicionRepeat**

Se encarga de validar si la condición del repeat until sea válida y si todavía se cumple para repetir

**ValidarCondicionWhile**

Se encarga de validar si la condición del while sea válida y si todavía se cumple para repetir

# Tabla de símbolos

Esta implementación está basada en una **tabla hash con listas enlazadas ordenadas** para manejar colisiones eficientemente y permite realizar operaciones básicas como inserción, búsqueda, modificación y eliminación de símbolos.

**Estructura:**

* **Tabla Hash**: Un arreglo (lista) de tamaño fijo, donde cada posición representa un "bucket" o contenedor.
* **Listas Enlazadas Ordenadas**: Cada bucket puede contener una lista enlazada de nodos distribuyendo los identificadores en la tabla tomando en cuenta el peso de cada letra según su posición en la cadena, reduciendo así las colisiones.
* **Nodo (símbolo)**:
  + nombre: Identificador (clave principal)
  + categoria: Tipo de identificador (ej. variable, función, etc.)
  + atributos: Diccionario con información adicional (ej. tipo de dato, valor, categoria, parámetros.)
  + siguiente: Puntero al siguiente nodo en la lista enlazada

**Control de Colisiones**

Las colisiones son manejadas mediante **listas enlazadas ordenadas** por nombre y peso dentro de cada posición del hash. Esto garantiza:

* Inserción ordenada
* Búsqueda eficiente dentro del bucket
* Evita duplicación de identificadores

# Tabla de precedencia

Esta es la tabla de precedencia, aquí se explica como orden que deben de seguir las palabras o símbolos que hay en el lenguaje sobre operaciones y demás, ayuda a entender como el compilador elije que orden ejecutarlas en caso de la presencia de más de una en el mismo lugar

|  |  |  |  |
| --- | --- | --- | --- |
| **Nivel** | **Operadores** | **Descripción** | **Asociatividad** |
| 1 | () . | paréntesis | Izquierda |
| 2 | + - not | Signo (unario), negación lógica | Derecha |
| [ ] @ | Acceso a un elemento (unarios): puntero y dirección | Derecha |
| magma souldan | Incremento y decremento | Derecha |
| (cast) sizeof | Conversión de tipo (casting) y tamaño de un elemento | Derecha |
| 3 | \* // % | Producto, división, módulo | Izquierda |
| 4 | + - | Suma y resta | Izquierda |
| 5 | < <= >= > | Comparaciones de superioridad e inferioridad | Izquierda |
| 6 | is isNot | Comparaciones,de igualdad | Izquierda |
| 7 | and | And logico | Izquierda |
| 8 | or | Or logico | Izquierda |
| 9 | = += -= \*= //= %= | Asignaciones | Derecha |
| 10 | , | Coma | Izquierda |

# Listado de errores y recuperación

**-1. Carácter monstruo:** Carácter no reconocido por el autómata. **Solución:** Se reporta el error y se avanza al siguiente carácter.

**-2. Comentario que nunca cierre:** No se encuentra el símbolo de cierre (\*$). **Solución:** Se sigue leyendo hasta el final del documento y se reporta un error si no se cierra.

**-3. Número mal formado:** Secuencia numérica que contiene caracteres inválidos (ejemplo 12a3). **Solución:** Se consume hasta un separador válido (espacio, salto de línea) y se reporta el error.

**-4. Token desconocido:** Lexema que no corresponde a ningún estado de aceptación. **Solución:** Se consume hasta el próximo separador y se reporta el error.

**-5. Operador inválido:** Secuencia de símbolos que no forma un operador permitido (ejemplo \*\*, %%). **Solución:** Se analiza solo el primer símbolo como operador válido y se reporta el error.

**-6. Identificador inválido:** Identificador que comienza con un número (ejemplo 4variable). **Solución:** Se lee hasta un separador y se reporta el error.

**-7. Asignación incorrecta:** Uso incorrecto del signo igual, como =; o =-. **Solución:** Se reporta un mensaje de error si no se sigue la secuencia del automata

**-8. Terminador mal formado:** Se esperaba un ; o : y aparece otro carácter.  
**Solución:** Se reporta el error y se continúa leyendo los siguientes lexemas.

# Autómatas

El autómata del proyecto es demasiado grande por lo que hemos tomado la decisión de segmentarlo por parte. Nuestro estado inicial va a ser s0, de ahí se va expandiendo a diferentes estados. **Una nota importante** es que más adelante se explicara los errores o identificadores ya que si se explica en el mismo autómata podría ser confuso. Por cada imagen de autómatas se explicará que se hizo, secciones utilizadas y las palabras reservadas representadas. **Otra segunda nota** es que no se está tomando en esta segmentación el orden de la numeración de los estados para poder representar cada sección o grupo de tokens, sin embargo en este orden el autómata irá creciendo

**Estructura del título del programa y secciones**

Palabras:

* worldname
* bedrock
* resourcepack
* récipe
* craftingtable

A line of circles and dots

AI-generated content may be incorrect.

**Puntos de entrada del sistema y sistemas de asignación**

Palabras:

* spawnpoint
* obsidian
* anvil

A diagram of a network

AI-generated content may be incorrect.

**Tipos**

* palabras:
* stack
* rune
* spider
* torch
* chest
* shelf
* entity

A diagram of a computer

AI-generated content may be incorrect.

**Literales**

**Literal booleana**

**A diagram of a network

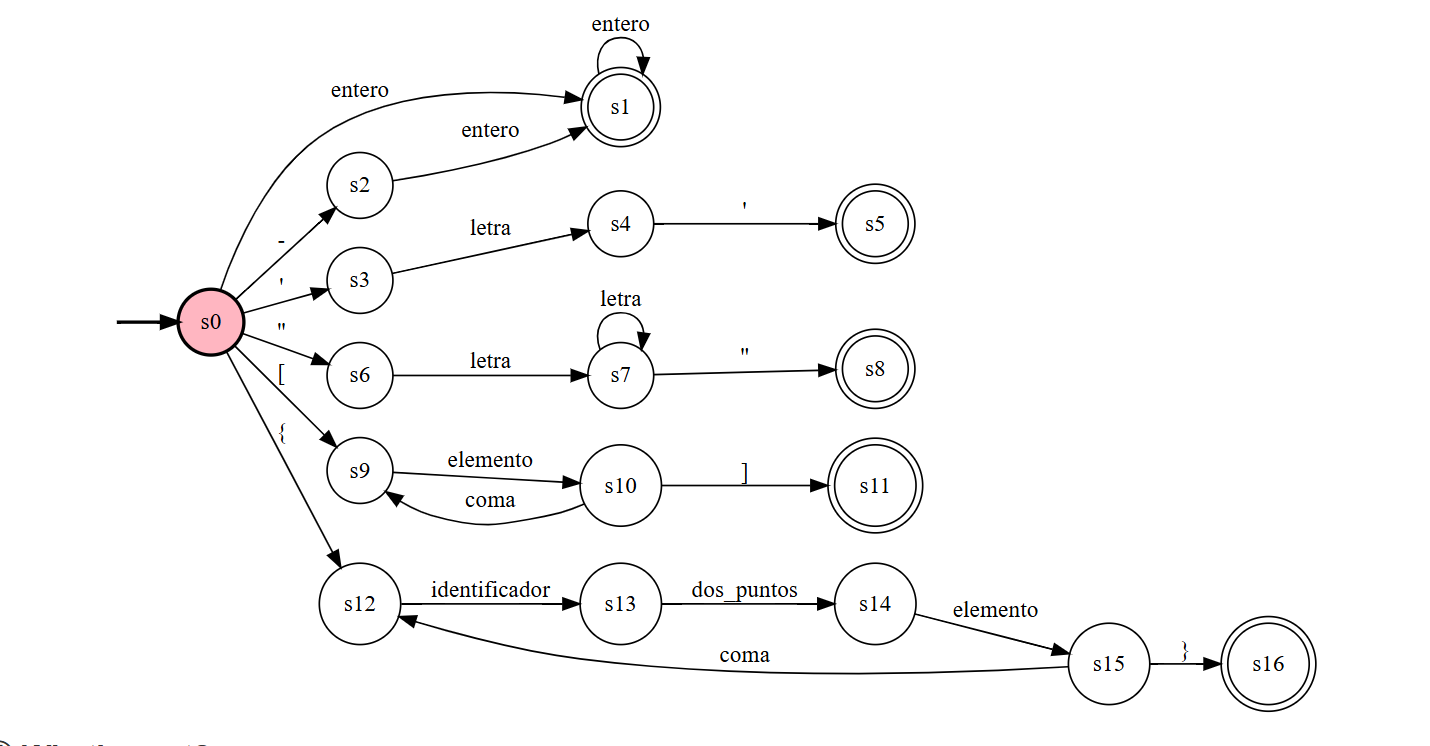
AI-generated content may be incorrect.**

**Literal de conjunto**

**A diagram of a diagram

AI-generated content may be incorrect.**

**Literal de enteros, char, string, arreglos y registros**

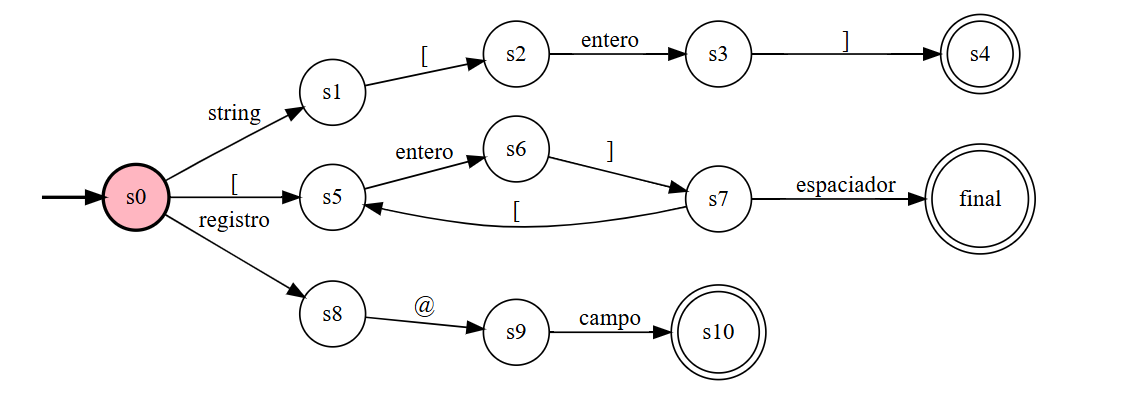
****

**Sistema de acceso**

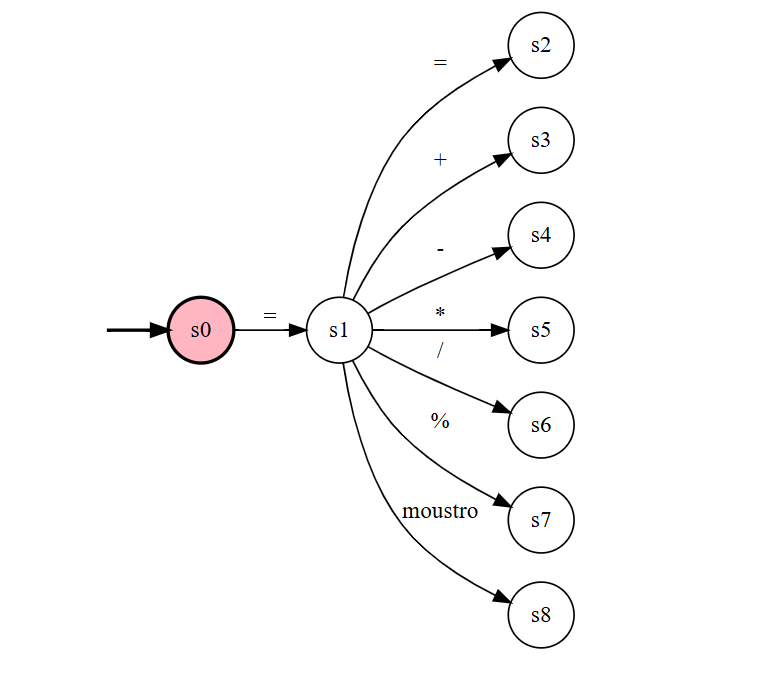
Sistema de acceso de arreglos

Sistema de acceso de string

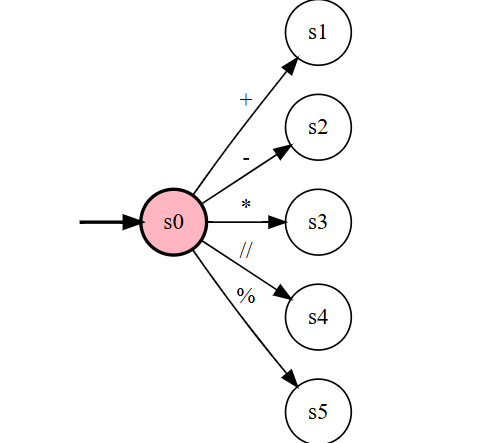
Sistema de acceso de registros



**Asignación y familia**

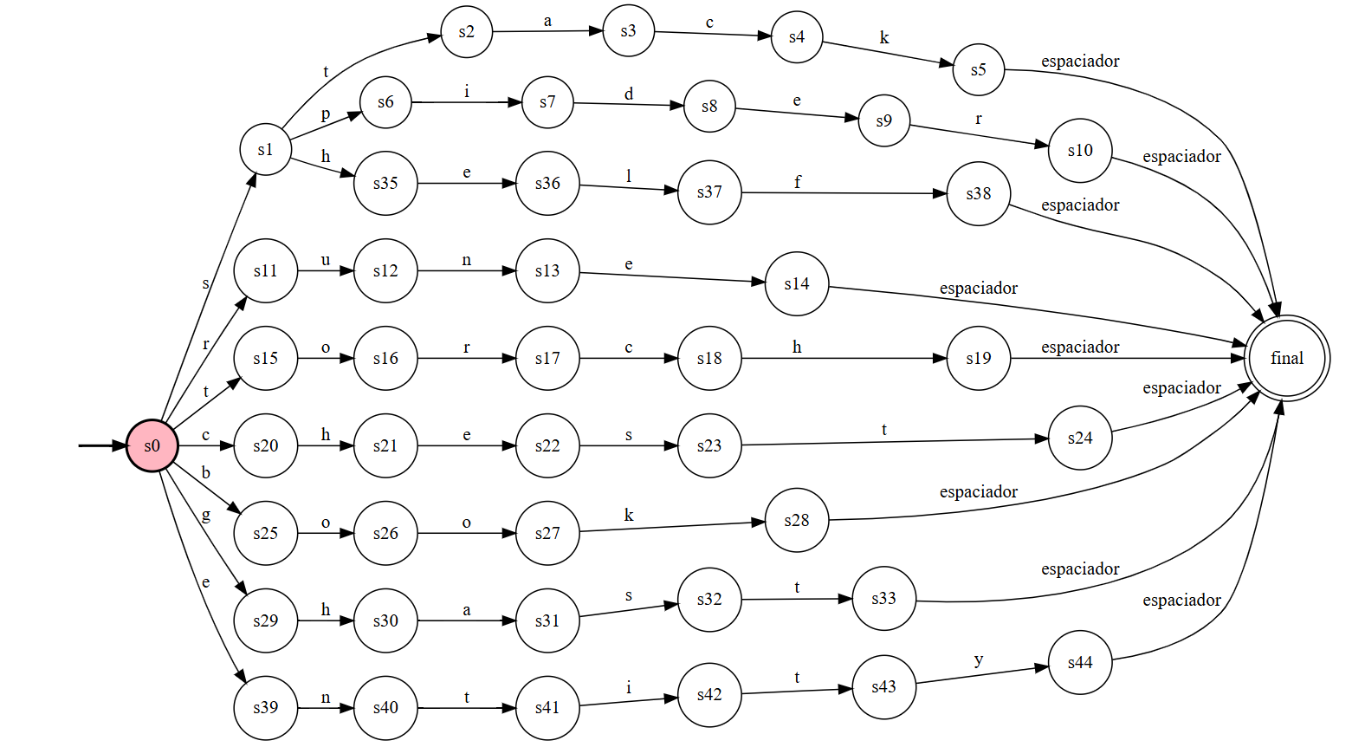
****

**Operaciones aritméticas básicas de enteros**

****

**Tipos**

* Stack
* Rune
* Spider
* Torch
* Chest
* Book
* Ghast
* Shelf
* Entity



# Autómata en código

100: {' ': 100, '\n': 100, 'w': 101, 'b': 110, 'r': 118, 'i': 130, 'c': 143, 's': 156, 'o': 166, 'a': 174, 't': 190, 'g': 202, 'e': 211, 'm': 227, 'n': 263, 'x': 266, 'f': 272, 'd': 283, 'k': 297, 'p': 304, 'h': 330, 'j': 336,

*# Espacios y saltos de línea*

*#'1': 50, '2': 50, '3': 50, '4': 50, '5': 50, '6': 50, '7': 50, '8': 50, '9': 50, '0': 50,*

*# Asignacion y aritmetica*

'=': 26,

*#Operaciones aritmeticas*

*#'+': 27, '-': 28, '\*': 29, '/': 30, '%': 31,*

*# Operacores de comparacion*

'<': 46, '>': 47,

*#terminador*

';': 76, ':': 94,

*# Abre parentesis*

'(': 70,

*# Cierra parentesis*

')': 71,

*# Abre llave*

'{': 59,

*# Cierra llave*

'}': 60,

*# Comilla*

',': 89,

},

*# worldname*

101: {'o': 102, 'a': 255, 'i': 261},

102: {'r': 103},

103: {'l': 104},

104: {'d': 105},

105: {'n': 106, 's': 409},

106: {'a': 107},

107: {'m': 108},

108: {'e': 0},

*# bedrock*

110: {'e': 111, 'o': 199, 'i': 269},

111: {'d': 112},

112: {'r': 113},

113: {'o': 114},

114: {'c': 115},

115: {'k': 1},

*# resourcepack*

118: {'e': 119, 'u': 183, 'a': 279, 'i': 389},

119: {'s': 120, 'c': 139, 'p': 219},

120: {'o': 121, 'p': 294},

121: {'u': 122},

122: {'r': 123},

123: {'c': 124},

124: {'e': 125},

125: {'p': 126},

126: {'a': 127},

127: {'c': 128},

128: {'k': 2},

*# inventory*

130: {'n': 131, 's': 232, 't': 288},

131: {'v': 132},

132: {'e': 133},

133: {'n': 134},

134: {'t': 135},

135: {'o': 136},

136: {'r': 137},

137: {'y': 3},

*# recipe*

139: {'i': 140},

140: {'p': 141},

141: {'e': 4},

*# craftingtable*

143: {'r': 144, 'h': 195},

144: {'a': 145, 'e': 266},

145: {'f': 146},

146: {'t': 147},

147: {'i': 148, ' ': 54},

148: {'n': 149},

149: {'g': 150},

150: {'t': 151},

151: {'a': 152},

152: {'b': 153},

153: {'l': 154},

154: {'e': 5},

*# spawnpoint*

156: {'p': 157, 't': 179, 'h': 207, 'o': 220, 'e': 280},

157: {'a': 158, 'i': 186, 'e': 386},

158: {'w': 159},

159: {'n': 160},

160: {'p': 161, 'e': 246},

161: {'o': 162},

162: {'i': 163},

163: {'n': 164},

164: {'t': 6},

*# obsidian*

166: {'b': 167, 'n': 18, 'f': 216, 'r': 22},

167: {'s': 168},

168: {'i': 169},

169: {'d': 170},

170: {'i': 171},

171: {'a': 172},

172: {'n': 7},

*# anvil*

174: {'n': 175, 'd': 283},

175: {'v': 176, 'd': 22},

176: {'i': 177},

177: {'l': 8},

*# stack*

179: {'a': 180, 'e': 259},

180: {'c': 181},

181: {'k': 9},

*# rune*

183: {'n': 184},

184: {'e': 10},

*# spider*

186: {'d': 187},

187: {'e': 188},

188: {'r': 11},

*# torch*

190: {'o': 191, 'a': 325},

191: {'r': 192, ' ': 93},

192: {'c': 193},

193: {'h': 12},

*# chest*

195: {'e': 196, 'u': 298},

196: {'s': 197},

197: {'t': 13},

*# book*

199: {'o': 200},

200: {'k': 14},

*# ghast*

202: {'h': 203},

203: {'a': 204},

204: {'s': 205},

205: {'t': 15},

*# shelf*

207: {'e': 208},

208: {'l': 209},

209: {'f': 16},

*# entity*

211: {'n': 212, 't': 250, 'x': 276},

212: {'t': 213, 'd': 271},

213: {'i': 214},

214: {'t': 215},

215: {'y': 17},

*# on*

*#6*

*# off*

216: {'f': 19},

*# soulsand*

220: {'u': 221},

221: {'l': 222},

222: {'s': 223},

223: {'a': 224},

224: {'n': 225},

225: {'d': 32},

*# magma*

227: {'a': 228, 'i': 333},

228: {'g': 229, 'p': 24},

229: {'m': 230},

230: {'a': 33},

*# 12 es de comparacion solicitada*

*# isengraved*

232: {'e': 233, 'i': 241, ' ': 50, 'n': 301},

233: {'n': 234},

234: {'g': 235},

235: {'r': 236},

236: {'a': 237},

237: {'v': 238},

238: {'e': 239}, *#21*

239: {'d': 34},

*# isinscribed*

241: {'n': 242},

242: {'s': 243},

243: {'c': 244},

244: {'r': 245},

245: {'i': 246},

246: {'b': 247},

247: {'e': 248},

248: {'d': 35},

*# etchup*

250: {'c': 251},

251: {'h': 252},

252: {'u': 253, 'd': 255},

253: {'p': 36},

*# etchdown*

255: {'o': 256},

256: {'w': 257},

257: {'n': 37},

*# and*

259: {' ': 38},

*# or*

262: {' ': 39},

*# not*

263: {'o': 264},

264: {'t': 40},

*# xor*

266: {'o': 267},

267: {'r': 41},

*# bind*

269: {'n': 270, 'o': 294},

270: {'d': 42},

*# from*

272: {'r': 273, 'e': 291, 'e': 291},

273: {'o': 274},

274: {'m': 43},

*# except*

276: {'c': 277, 'h': 248},

277: {'e': 278},

278: {'p': 279},

279: {'t': 44},

*# seek*

280: {'e': 281, 't': 92},

281: {'k': 45},

*# add*

283: {'d': 11, 'i': 243, 'r': 285},

284: {' ': 100},

*# drop*

285: {'o': 286},

286: {'p': 11},

*#este no se si tengo que borrarlo*

287: {'p': 306},

*# item*

288: {'e': 289},

289: {'m': 11},

*# feed*

291: {'e': 292},

292: {'d': 11},

*# map*

293: {'p': 11},

*# biom*

294: {'m': 11},

*# kill*

297: {'i': 298},

298: {'l': 299},

299: {'l': 11},

*# is -> 232*

*# isnot*

301: {'o': 302},

302: {'t': 51},

*# pollocrudo*

304: {'o': 305},

305: {'l': 306},

306: {'l': 307},

307: {'o': 308},

308: {'c': 309, 'a': 314},

309: {'r': 310},

310: {'u': 311},

311: {'d': 312},

312: {'o': 52},

*# polloasado*

314: {'s': 315},

315: {'a': 316},

316: {'d': 317},

317: {'o': 53},

*# repeater*

319: {'e': 320},

320: {'a': 321},

321: {'t': 322},

322: {'e': 323},

323: {'r': 57},

*# craft -> 143*

*# target*

325: {'r': 326},

326: {'g': 327},

327: {'e': 328},

328: {'t': 58},

*# hit*

330: {'i': 331, 'o': 301},

331: {'t': 96},

*# miss*

333: {'s': 334},

334: {'s': 55},

*# jukebox*

336: {'u': 337},

337: {'k': 338},

338: {'e': 339},

339: {'b': 340},

340: {'o': 341},

341: {'x': 59},

*# disk*

343: {'s': 344},

344: {'k': 60},

*# spawner*

346: {'r': 61},

*# exhausted*

348: {'a': 349},

349: {'u': 350},

350: {'s': 351},

351: {'t': 352},

352: {'e': 353},

353: {'d': 90},

*# walk*

355: {'l': 356},

356: {'k': 62},

*# set -> 280*

*# to -> 190*

*# step*

359: {'p': 63},

*# wither*

361: {'t': 362},

362: {'h': 363},

363: {'e': 364},

364: {'r': 64},

*# creeper*

366: {'e': 367},

367: {'p': 368},

368: {'e': 369},

369: {'r': 91},

*# enderpearl*

371: {'e': 372},

372: {'r': 373},

373: {'p': 374},

374: {'e': 375},

375: {'a': 376},

376: {'r': 377},

377: {'l': 87},

*# ragequit*

379: {'g': 380},

380: {'e': 381},

381: {'q': 382},

382: {'u': 383},

383: {'i': 384},

384: {'t': 0},

*# spell*

386: {'l': 387},

387: {'l': 68},

*# ritual*

389: {'t': 390},

390: {'u': 391},

391: {'a': 392},

392: {'l': 69},

*# respawn*

394: {'a': 395},

395: {'w': 396},

396: {'n': 72},

*# chunk*

398: {'n': 399},

399: {'k': 73},

*# hopper*

401: {'p': 402},

402: {'p': 403},

403: {'e': 404},

404: {'r': 74},

*# dropper*

406: {'e': 407},

407: {'r': 75},

*# worldsave*

409: {'a': 410},

410: {'v': 411},

411: {'e': 77},

*# disk*

413: {' ': 60},

}

Si observamos, el autómata va recorriendo cada estado hasta llegar al estado final, donde reconocerá el código que tendrá mi token. Mi estado inicial va a ser el número 100, ya que los primeros números van a estar reservados para precisamente asignar el código a estas terminales

# Instrucciones para correr el analizador el programa

Para poder correr el archivo llamado **parcer.py** tienes dos opciones, la primera es cambiar la ruta de la variable **path** en el \_\_main\_\_ por la ruta del archivo código fuente que se quiera probar. En este caso ya hay un archivo de prueba llamado **prueba.ne** por lo que nada más debe de copiar la ruta de ese archivo y correr el archivo **parcer.py**. Este código se encuentra al final del archivo

A black screen with white text

AI-generated content may be incorrect.

La otra opción es ejecutar el archivo **MAIN.py** que te va desplegar un menú en consola donde podrás probar diferentes archivos de prueba.

A screenshot of a computer

AI-generated content may be incorrect.

# Manejo del análisis semántico

En la gramatica se manejarán símbolos semánticos que nos ayudaran a realizar el análisis semántico. Por ejemplo, podemos observar en la siguiente imagen que se utilizan símbolos como **#CrearTLG** para establecer puntos importantes dentro de la gramatica. Estos se pueden ver dentro del archivo de Excel llamado **x2.xls** presente en este proyecto.

A screenshot of a computer

AI-generated content may be incorrect.

## Símbolos semánticos utilizados

CrearTLG = 185

EmpezarAnalizarReturn = 186

ComprobarReturn = 187

AlmacenarTipoConstante = 188

ValidarExistenciaIdentificadorConstante = 189

ValidarTipoValorConstante = 190

ValidarExistenciaIdentificadorTipo = 191

ValidarValorTipo = 192

AlmacenarTipoVariable = 193

BorrarTipoVariable = 194

AlmacenarFuncion = 195

AlmacenarTipoFuncion = 196

AlmacenarProcedimiento = 197

BorrarIdentificador = 198

AlmacenarTipoParametroFormal = 199

ValidarExistenciaIdentificadorParametroFormal = 200

ValidarExistenciaIdentificadorParametroReal = 201

ValidarExistenciaIdentificadorVariable = 202

ValidarTipoValorVariable = 203

ValidarIdentificadorIncrementoDecremento = 204

AntesVerificarAsignacion = 205

VerificarAsignacion = 206

ValidarOperacionEntero = 207

ValidarOperacionComparacion = 208

ValidarOperacionCaracter = 209

ValidarOperacionString = 210

SiHayReturn = 211

ValidarCondicionWhile = 212

ValidarIdentificadorForRecorrido = 213

ValidarLiteralEnteraFor = 214

ValidarTopeFor = 215

ValidarCondicionRepeat = 216

ValidarIdentificadorWith = 217

EmpezarAnalizarDefaultSwitch = 218

HayDefaultSwitch = 219

SiHayDefaultSwitch = 220

ValidarOperacionLogica = 221

ValidarValorIdentificador = 222

ValidarValorValor = 223

ValidarTipoEntera = 224

ValidarTipoFlotante = 225

ValidarTipoString = 226

Como se puede observar en el código anterior, esos fueron los símbolos semánticos utilizados con su respectivo código. Estos código son diferentes de los terminales y no terminales. Cada uno fue explicado anteriormente.

Pruebas del análisis sintáctico

Como puede observar, al correr el programa, se puede observar el correcto funcionamiento del parcer y del análisis semántico el cual es implementado dentro de este.  
A computer screen shot of a black screen

AI-generated content may be incorrect.