**Instituto Nacional De Educación Diversificada INED Santa Cruz Naranjo**

**Centro GNet**

**Tema principal: Teoría de Autómatas Finitos Deterministas**

**alumna: Estefany Fabiola Donis Herrera**

**ID: 1694**

**Materia: Producción**

**Grado: 5to Bachillerato**

**Catedrático: Gustavo Blanco**

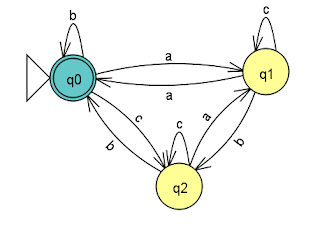
**Fecha: 5/04/2025**

**1. Teoría de autómatas**

* **Definición y usos de los autómatas finitos deterministas (AFD)**

. Un autómata finito determinista es un autómata finito que además es un sistema determinista; es decir, para cada estado en que se encuentre el autómata, y con cualquier símbolo del alfabeto leído, existe siempre no más de una transición posible desde ese estado y con ese símbolo.

* **Ejemplos y aplicaciones** en informática y ciencias de la computación.



Son conjunto de señales recibidas por el procesador de un ordenador, el que está compuesto por un número **finito** de compuertas(estados), cada una de las cuales pueden utilizar dos condiciones posibles, por lo general denotadas por 0 y 1(transiciones).

**2. Gramáticas formales**

* **Gramáticas regulares**: concepto y ejemplos de su uso en lenguajes de
* programación.
* *Una* **gramática regular** *es una cuádruple G* = ( *N* , Σ, *P* , *S* ), *donde*
* 1.
* *N es un alfabeto llamado el conjunto de* **no terminales.**
* 2.
* El alfabeto Σ *wan se denomina conjunto de* **terminales** , *con* Σ ∩ *N* = Ø.
* 3.
* *P es un conjunto finito de* **producciones** *o* **reglas** *de la forma A* → *w* , *donde A* ∈ *N y w* ∈ Σ\* *N* ∪ Σ\*.
* 4.
* *S es el* **símbolo de** inicio , *S* ∈ *N.*
* Nótese que las producciones en una gramática regular tienen como máximo un no terminal en el lado derecho y que este no terminal siempre ocurre al final de una producción.
* **Gramáticas recursivas**: explicación y diferencias con las gramáticas regulares.
* Un **autómata finito** (**AF**) o **máquina de estado finito** es un [modelo computacional](https://es.wikipedia.org/wiki/Modelo_computacional) que realiza [cómputos](https://es.wikipedia.org/wiki/Proceso_(inform%C3%A1tica)) en forma automática sobre una [entrada](https://es.wikipedia.org/wiki/Entrada) para producir una [salida](https://es.wikipedia.org/wiki/Salida_(inform%C3%A1tica)).
* Este modelo está conformado por un [alfabeto](https://es.wikipedia.org/wiki/Alfabeto), un conjunto de [estados](https://es.wikipedia.org/wiki/Estado_(inform%C3%A1tica)) finito, una [función de transición](https://es.wikipedia.org/wiki/Tabla_de_transici%C3%B3n_de_estados), un estado inicial y un conjunto de estados finales. Su funcionamiento se basa en una función de transición, que recibe a partir de un *estado inicial* una [cadena de caracteres](https://es.wikipedia.org/wiki/Cadena_de_caracteres) pertenecientes al alfabeto (la entrada), y que va leyendo dicha cadena a medida que el autómata se desplaza de un estado a otro, para finalmente detenerse en un *estado final* o *de aceptación*, que representa la salida.
* La finalidad de los autómatas finitos es la de reconocer [lenguajes regulares](https://es.wikipedia.org/wiki/Lenguaje_regular), que corresponden a los [lenguajes formales](https://es.wikipedia.org/wiki/Lenguaje_formal) más simples según la [Jerarquía de Chomsky](https://es.wikipedia.org/wiki/Jerarqu%C3%ADa_de_Chomsky).
* Una gramática recursiva es aquella que contiene reglas de producción recursivas, mientras que una gramática regular es una gramática formal que genera lenguajes regulares.

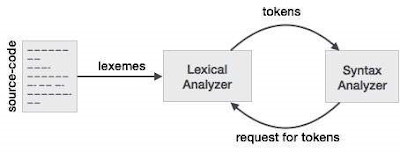
**3. Análisis léxico**

* **Tokenización**: qué es y cómo se usa en el proceso de compilación.

La **tokenización es el proceso de sustituir los datos sensibles o los activos físicos por símbolos de identificación únicos** que conservan toda la información esencial de los datos o los activos sin comprometer su seguridad.Todo este proceso está basado en la tecnología de blockchain, que es una cadena de bloques que almacena y registra las transacciones de forma descentralizada, distribuida, inmutable y transparente.Aquí, **cada token representa un activo o un dato que está vinculado a un contrato inteligente**, que es un código que define las reglas y las condiciones de uso del token. De esta forma, se garantiza la autenticidad, la trazabilidad, la propiedad y la seguridad de los tokens.

* **Expresiones regulares**: su función en el análisis léxico.
* En [cómputo teórico](https://es.wikipedia.org/wiki/Ciencia_computacional_te%C3%B3rica) y teoría de [lenguajes formales](https://es.wikipedia.org/wiki/Lenguaje_formal), una **expresión regular** o **expresión racional** [[1]](https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular#cite_note-Mitkov2003-1)​[[2]](https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular#cite_note-Lawson2003-2)​ (también son conocidas como **regex** o **regexp**,[[3]](https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular#cite_note-Regex_info,_2017-3)​ por su contracción de las palabras inglesas ***reg****ular* ***ex****pression)* es una secuencia de [caracteres](https://es.wikipedia.org/wiki/Car%C3%A1cter_(tipo_de_dato)) que conforma un [patrón](https://es.wikipedia.org/wiki/Patr%C3%B3n_(estructura)) de búsqueda. Se utilizan principalmente para la [búsqueda de patrones](https://es.wikipedia.org/wiki/B%C3%BAsqueda_de_patrones) de cadenas de caracteres u operaciones de sustituciones.
* Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una [cadena de texto](https://es.wikipedia.org/wiki/Cadena_de_texto). Las expresiones regulares proporcionan una manera muy flexible de buscar o reconocer cadenas de texto. Por ejemplo, el grupo formado por las cadenas *Handel*, [*Händel*](https://es.wikipedia.org/wiki/H%C3%A4ndel) y *Haendel* se describe con el patrón H(a|ä|ae)ndel.
* La mayoría de las formalizaciones proporcionan los siguientes constructores: una expresión regular es una forma de representar los [lenguajes regulares](https://es.wikipedia.org/wiki/Lenguaje_regular) (finitos o infinitos) y se construye utilizando [caracteres](https://es.wikipedia.org/wiki/Tipograf%C3%ADa) del [alfabeto](https://es.wikipedia.org/wiki/Alfabeto) sobre el cual se define el [lenguaje](https://es.wikipedia.org/wiki/Lenguaje).
* **Reconocimiento de palabras clave**: cómo se identifican en los lenguajes de programación.

1. El analizador léxico rompe sintaxis en una serie de **fichas**, eliminando cualquier espacio en blanco o comentarios en el código fuente.
2. Si el analizador léxico encuentra un simbólico inválido, genera un error. Lee los flujos de caracteres del código fuente, comprueba simbólicos legales y pasa los datos al analizador de sintaxis cuando lo exige.



* **Identificación de tokens**: clasificación y ejemplos prácticos
* El identificador de token digital (DTI) es un estándar que identifica tokens digitales, como criptoactivos. El DTI se estableció en la Organización Internacional de Normalización (ISO) bajo la norma ISO 24165.

.

**4. Análisis sintáctico**

* **Tipos de análisis sintáctico**:
  + Análisis sintáctico descendente.
* El análisis sintáctico descendente es un método que construye un árbol de análisis sintáctico de arriba hacia abajo. Se utiliza para analizar gramáticas y determinar si una cadena de entrada es válida.
  + Análisis sintáctico ascendente.
* es un método de análisis que construye un árbol sintáctico de abajo hacia arriba. Se parte de la cadena de entrada y se intenta llegar al símbolo inicial de la gramática.
* **Árboles de análisis sintáctico**: qué son y cómo ayudan en la estructura de los programas.

es una representación gráfica de la estructura de un programa o texto. Ayuda a analizar y manipular el código fuente, lo que es útil para el diseño de compiladores.

**5. Algoritmos de análisis de cadenas**

* **Algoritmo de Boyer-Moore**: concepto, funcionamiento y su aplicación en la búsqueda de cadenas dentro de textos o programas.

**6. Compiladores e intérpretes**

* **Estructura de un compilador**: definición y componentes.
* **Fases del compilador**:
  + Análisis léxico.
  + Análisis sintáctico.
  + Generación de código.
  + Optimización de código.
* **Lenguajes de programación utilizados en la creación de compiladores**: como **Lex y Yacc**.

**7. Aplicaciones prácticas de autómatas**

* **Reconocimiento de patrones** en diferentes áreas.
* **Procesamiento de lenguaje natural** y su uso en inteligencia artificial.
* **Análisis de texto y minería de datos**: su importancia en el análisis de información.

**8. Lenguaje de programación C y C++**

* **Librerías fundamentales**: **STDIO y CONIO,** un resumen de las funciones de cada una.
* **Lista de 10 librerías más comunes en C/C++**, además de las ya mencionadas.
* **Manejadores de formatos para diferentes tipos de datos**, con ejemplos de uso en código.