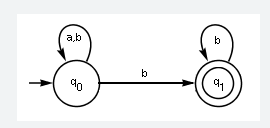
**1. Teoría de autómatas**

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

Un **autómata finito determinista** (abreviado **AFD**) 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.

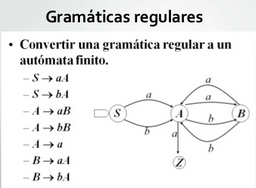


En el campo de la computación, **tienen múltiples aplicaciones en el diseño de circuitos digitales y la programación, por ejemplo, en el diseño de compiladores de lenguajes y reconocimiento automático de éstos**.

**2. Gramáticas formales**

* **Gramáticas regulares**: concepto y ejemplos de su uso en lenguajes de programación.

Una gramática ("G") desde el punto de vista de la teoría de autómatas es un **conjunto finito de reglas que describen toda la secuencia de símbolos pertenecientes a un lenguaje específico**



* **Gramáticas recursivas**: explicación y diferencias con las gramáticas regulares.

La recursividad **consiste en realizar una definición de un concepto en términos del propio concepto que se está definiendo**.

Ejemplos: Los números naturales se pueden definir de la siguiente forma: 0 es un Número natural y el sucesor de un número natural es también un número natural

**3. Análisis léxico**

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

La tokenización es un facilitador clave de las transacciones seguras en línea y con monedero digital. **Permite pagos más rápidos y seguros, y reduce el fraude**. Como resultado, la tokenización ayuda a disminuir las tasas de carritos abandonados, mejorando la experiencia de compra general de los clientes

* **Expresiones regulares**: su función en el análisis léxico.

Las expresiones regulares son un **equivalente algebraico para un autómata**. Utilizado en muchos lugares como un lenguaje para describir patrones en texto que son sencillos pero muy útiles. Pueden definir exactamente los mismos lenguajes que los autómatas pueden describir: Lenguajes regulares.

* **Reconocimiento de palabras clave**: cómo se identifican en los lenguajes de programación.
* **Identificación de tokens**: clasificación y ejemplos prácticos.

**4. Análisis sintáctico**

* **Tipos de análisis sintáctico**:
  + Análisis sintáctico descendente.
  + Análisis sintáctico ascendente.
* **Árboles de análisis sintáctico**: qué son y cómo ayudan en la estructura de los programas.

**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.