**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.

Las **ciencias de la computación** o **ciencias de la informática** son las [ciencias formales](https://es.wikipedia.org/wiki/Ciencias_formales) que abarcan las bases teóricas de la [información](https://es.wikipedia.org/wiki/Informaci%C3%B3n) y la [computación](https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_computaci%C3%B3n), así como su aplicación en los [sistemas informáticos](https://es.wikipedia.org/wiki/Sistemas_de_informaci%C3%B3n)



**2. Gramáticas formales**

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

En [informática](https://es.wikipedia.org/wiki/Inform%C3%A1tica) una **gramática regular** es una [gramática formal](https://es.wikipedia.org/wiki/Gram%C3%A1tica_formal) (V, Σ, R, S) que puede ser clasificada como regular izquierda o regular derecha. Las gramáticas regulares sólo pueden generar a los [lenguajes regulares](https://es.wikipedia.org/wiki/Lenguaje_regular) de manera similar a los autómatas finitos y las expresiones regulares.

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

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** L.

**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.

* **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.

El **reconocimiento de palabras clave**, conocido en [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s) como ***wordspotting***, es una parte de la [inteligencia artificial](https://es.wikipedia.org/wiki/Inteligencia_artificial) que consiste en identificar palabras específicas dentro de una locución. Es un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) basado en la búsqueda de palabras clave en archivos de audio, por ejemplo llamadas telefónicas, donde localiza e indexa contenido de audio ganando un ahorro importante de tiempo ante la búsqueda manual. El sistema wordspotting se diferencia entre el reconocimiento de palabras aisladas y el reconocimiento de habla continua, es decir, las palabras son reconocidas en un flujo continuo de [fonemas](https://es.wikipedia.org/wiki/Fonemas).

* **Identificación de tokens**: clasificación y ejemplos prácticos.

**unidad de valor basada en criptografía y emitida por una entidad privada en una 'blockchain', como Bitcoin o Ethereum**.



**4. Análisis sintáctico**

* **Tipos de análisis sintáctico**:
	+ Análisis sintáctico descendente.

Se puede considerar el análisis sintáctico descendente como un **intento de encontrar una derivación por la izquierda para una cadena de entrada** También se puede considerar como un intento de construir un árbol de análisis sintáctico para la entrada comenzando desde la raíz y creando los nodos del árbol en orden previo.

* + Análisis sintáctico ascendente.

Análisis Ascendente (BU): **Es un tipo de análisis que construye el árbol sintáctico desde las hojas (la parte inferior) hasta la raíz (la parte superior)**. Aplica reducciones (pasos de la derivación en reversa) y desplazamientos (lecturas de tokens de entrada).

* **Árboles de análisis sintáctico**: qué son y cómo ayudan en la estructura de los programas.

Es la **fase del analizador que se encarga de chequear el texto de entrada en base a una gramática dada**. Y en caso de que el programa de entrada sea válido, suministra el árbol sintáctico que lo reconoce.

**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.

El algoritmo Boyer-Moore **utiliza información recopilada durante el paso de preprocesamiento para omitir secciones del texto** , lo que da como resultado un factor constante más bajo que muchos otros algoritmos de búsqueda de cadenas. En general, el algoritmo se ejecuta más rápido a medida que aumenta la longitud del patrón.

**6. Compiladores e intérpretes**

* **Estructura de un compilador**: definición y componentes.

El funcionamiento de un compilador está dividido fundamentalmente en dos partes: **Una de análisis del código fuente y otra de síntesis del código objeto**, para lo cual, al igual que el ensamblador, debe construir y analizar varias tablas (ver Figura 1).

* **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.