**1. Teoría de autómatas**

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

 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.

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](https://es.wikipedia.org/wiki/Teor%C3%ADa_de_aut%C3%B3matas) es un conjunto finito de reglas que describen toda la secuencia de símbolos pertenecientes a un [lenguaje](https://es.wikipedia.org/wiki/Lenguaje) específico *L*. Dos gramáticas que describan el mismo lenguaje se llaman gramáticas equivalentes.



**3. Análisis léxico**

* **Tokenización**: qué es y cómo se usa en el proceso de compilación.
* **Expresiones regulares**: su función en el análisis léxico.
* **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.

**Requisitos adicionales**

✅ **Formato del documento**: Arial 12, con buena estructura y organización.
✅ **Carátula**: debe incluir su nombre, grado, materia y un borde decorativo y logotipo GNet en la esquina derecha superior.
✅ **Numeración de página** en todo el documento.
✅ **Resumen escrito en el cuaderno** de cada uno de los **7 temas principales**. (del 1 al 7 nada mas)