**Instituto Nacional De Educacion Diversificada**

**Santa Cruz Naranjo, Santa Rosa**

**INED**

**Catedra:** Produccion

**Catedratico:** Gustavo Blanco

**Nombre:** Danna Belen Garcia

**Tema:**

Investigacion

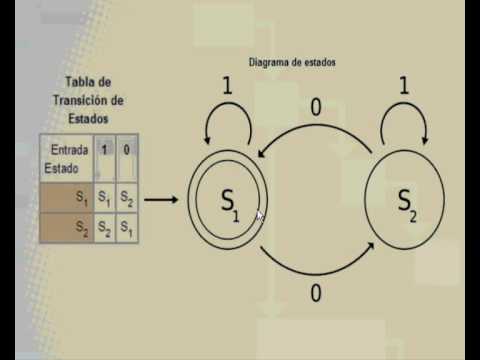
**Grado:** Quinto Bachillerato

**Seccion:** única

**Ciclo:** 2025

1. **Teoría de autómatas**

* **Definición y usos de los autómatas finitos deterministas (AFD)**.La teoría de autómatas es el estudio de máquinas abstractas y su uso para resolver problemas computacionales. Los autómatas finitos deterministas (AFD) son modelos computacionales que realizan cálculos automáticos sobre entradas para producir salidas.



### Ejemplos:

Elevadores que memorizan las llamadas de cada piso para optimizar sus ascensos y descensos

Termostatos

Pilotos automáticos de aeronaves, Ejemplos de autómatas en la informática:

Autómatas finitos que reconocen patrones de cadenas en lenguajes de programación

Máquinas de Turing que representan la universalidad computacional

Autómatas que ayudan a analizar y comprender los lenguajes humanos

Ejemplos de autómatas en la automatización industrial

Autómatas programables que controlan maquinaria y procesos industriales de forma automática

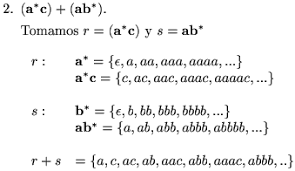
* **Aplicaciones**La teoría de autómatas se utiliza en aplicaciones de procesamiento de texto como motores de búsqueda, filtros de spam y procesamiento del lenguaje natural. Esta teoría proporciona una forma de analizar y manipular texto mediante expresiones regulares y otros lenguajes formales.

### Gramáticas formales

Una gramática formal es un conjunto de reglas que define cómo se pueden combinar símbolos para formar cadenas de caracteres en un lenguaje formal.

**Gramáticas regulares**: Una gramática regular es una estructura formal que se utiliza para generar cadenas de símbolos en un lenguaje. Es fundamental para definir la sintaxis de los lenguajes de programación. Uso en lenguajes de programación

* Las gramáticas son conjuntos de instrucciones que especifican cómo se pueden colocar caracteres y palabras para formar enunciados válidosLas gramáticas son fundamentales para definir la sintaxis de los lenguajes de programación
* **Ejemplos:**
* Un ejemplo de su uso es en el tratamiento de texto y la búsqueda de patrones en cadenas de caracteres.



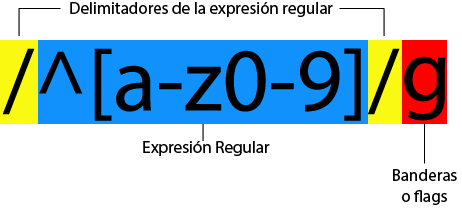
* **Gramáticas recursivas**: Una gramática recursiva es un conjunto de reglas de producción que permiten repetir estructuras dentro de un lenguaje. Esto facilita la creación de oraciones y frases complejas.

**Diferencia de las Gramáticas Regulares y Gramaticas Recursivas:** Las gramáticas recursivas son aquellas que contienen reglas de producción que pueden generar cadenas que incluyan el mismo no terminal. Las gramáticas regulares son un tipo de gramática formal que se define recursivamente.

**3.Análisis léxico**

El análisis léxico es un proceso que analiza una cadena de caracteres para generar una secuencia de símbolos llamados tokens. Es el primer paso en el procesamiento de texto y es clave para que las máquinas entiendan el lenguaje humano.

* **Expresiones regulares**: Las expresiones regulares son una forma de especificar patrones, entendiendo por patrón la forma de describir cadenas de caracteres. Es la forma de definir los tokens o componentes léxicos y, como veremos, cada patrón concuerda con una serie de cadenas.



* **Reconocimiento de palabras clave:**El reconocimiento de palabras clave es una técnica de inteligencia artificial que identifica palabras o frases en un audio. Se utiliza para activar asistentes virtuales por voz, como Cortana.

**¿Cómo se identifican en los lenguajes de programación?** Los lenguajes de programación se identifican por medio de su alfabeto, reglas gramaticales, semánticas, y notaciones específicas.

Cómo se clasifican los lenguajes de programación

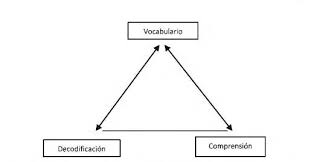
**Nivel de abstracción**: Los lenguajes pueden ser de bajo o alto nivel.

**Paradigma**: Los lenguajes pueden ser imperativos, declarativos, orientados a objetos, funcionales, o lógicos.

**Tipado**: Los lenguajes pueden ser estáticos o dinámicos.

**Dominio de aplicación**: Los lenguajes pueden ser generales o específicos.

**Nivel de complejidad**: Los lenguajes pueden ser de bajo, medio, o alto nivel.



* **Identificación de tokens:** La identificación de tokens es un método de autenticación que permite verificar la identidad de un usuario o dispositivo mediante el uso de tokens de seguridad.

**Clasificación:** Los tokens criptográficos se pueden dividir en cinco categorías: de utilidad, de seguridad, de activos reflejados, tokens no fungibles (NFT) y «monedas meme». Cada uno cumple una función específica en el entorno blockchain y ofrece ciertos beneficios a participantes e inversores.

**ejemplos prácticos: Fichas de casino**: Se utilizan en máquinas recreativas

**Vales**: Se pueden usar para acceder a un producto o servicio

**Puntos de bonificación**: Se pueden obtener en programas de fidelización

**Certificados de acciones**: Representan la propiedad de un activo

**Bonos**: Representan la propiedad de un activo

**Membresía de un club**: Permite acceder a un club

**Reservas**: Permiten reservar un lugar para cenar

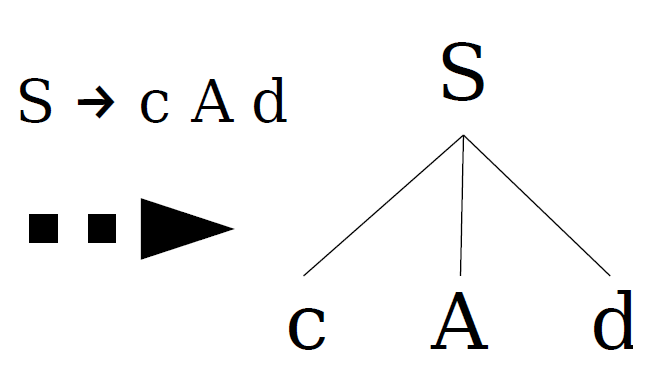
**Binance Coin (BNB)**: Se usa para pagar tarifas en la plataforma de Binance

**Token de Atención Básica (BAT)**: Proporciona acceso a un servicio o producto dentro de un ecosistema basado en blockchain.

**4.Análisis sintáctico**

El análisis sintáctico es un proceso que consiste en analizar la estructura de las oraciones para identificar y clasificar las funciones de las palabras que las forman

* **Tipos de análisis sintáctico**:
  + **Análisis sintáctico descendente:** El análisis sintáctico descendente es una técnica que construye un árbol de análisis de arriba hacia abajo. Es una de las técnicas más sencillas de análisis sintáctico.



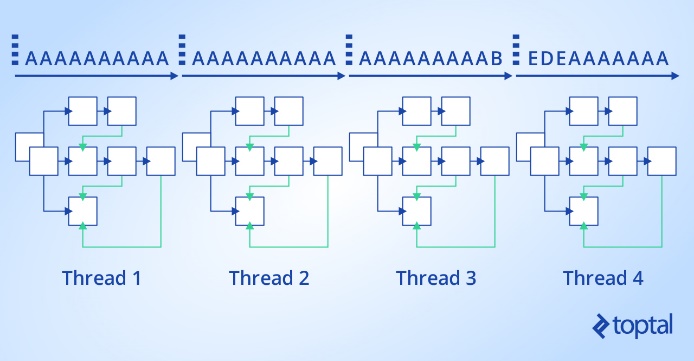
* + **Análisis sintáctico ascendente:** El análisis sintáctico ascendente es un método que construye un árbol sintáctico desde las hojas hasta la raíz. Se utiliza para analizar una cadena de entrada y llegar al símbolo inicial de una gramática.

* **Árboles de análisis sintáctico:** Los árboles de análisis son útiles en varias aplicaciones, incluido el procesamiento del lenguaje, el diseño del compilador y el análisis de sintaxis. Proporcionan una representación estructural de la cadena de entrada, lo que permite el análisis y la manipulación de sus propiedades sintácticas.

**¿cómo ayudan en la estructura de los programas?**  Proporcionan una representación estructural de la cadena de entrada, lo que permite el análisis y la manipulación de sus propiedades sintácticas.

**5.Algoritmos de análisis de cadenas**

Los algoritmos de análisis de cadenas son métodos que se utilizan para encontrar coincidencias entre patrones y textos.



* **Algoritmo de Boyer-Moore**: El algoritmo de Boyer-Moore utiliza la información recopilada durante El preprocesamiento para omitir secciones del texto, lo que resulta en un factor constante menor que el de muchos otros algoritmos de búsqueda de cadenas.

**Funcionamiento:** Funciona buscando de izquierda a derecha y comparando con el patrón de derecha a izquierda.

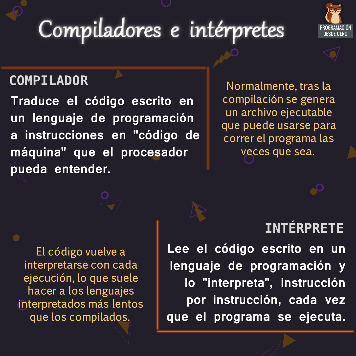
**su aplicación en la búsqueda de cadenas dentro de textos o programas:**  es una técnica de búsqueda de cadenas que encuentra patrones en textos o programas. Es eficiente y se usa como estándar en la literatura de búsqueda de cadenas.

se utiliza para encontrar patrones en textos, secuencias de ADN, y en otros ámbitos.

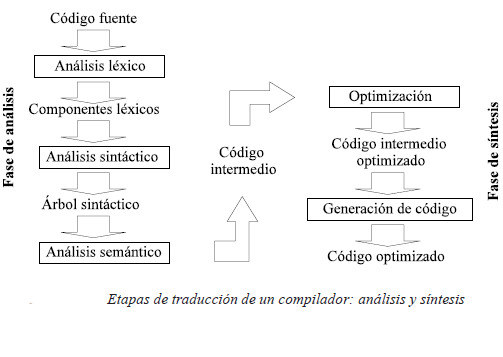
Aplicaciones Editores de texto y motores de búsqueda, Análisis de secuencias de ADN, Algoritmos de compresión de datos, Sistemas de detección de intrusiones.

**6.Compiladores e intérpretes**

Un compilador y un intérprete son programas que traducen el código fuente de un programa a un lenguaje máquina. La principal diferencia entre ambos es que el intérprete ejecuta el código línea por línea, mientras que el compilador traduce todo el código de una vez.



* **Estructura de un compilador**: Un compilador es un programa que traduce el código fuente de un programa a código máquina. Su estructura tiene dos partes principales: análisis del código fuente y síntesis del código objeto.



**Componentes:** **Analizador léxico**: Divide el código fuente en tokens, que son símbolos que representan palabras clave, operadores, identificadores, etc.

**Analizador sintáctico**: Verifica que los tokens generados por el analizador léxico cumplan con lagramática del lenguaje de programación

**Analizador semántico**: Verifica que el código fuente sea coherente y tenga sentido lógico

**Generador de código intermedio**: Transforma el código fuente validado en una representación intermedia

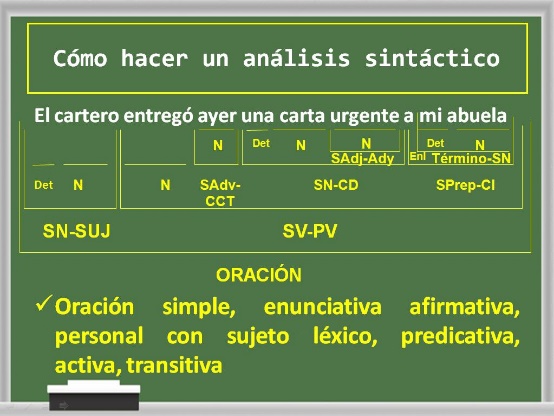
**Optimizador de código**: Mejora la eficiencia del código intermedio o del código objeto

**Generador de código objeto**: Genera el código objeto que se puede ejecutar en la computadora

* **Fases del compilador**:

Durante la compilación se traduce el lenguaje de alto nivel en lenguaje de máquina. Dentro de esta etapa pueden distinguirse las siguientes fases: Análisis léxico, análisis sintáctico, aná- lisis semántico y generación de código.

* + **Análisis léxico:**El análisis léxico es un proceso que consiste en descomponer una secuencia de caracteres en unidades significativas. Se utiliza en el procesamiento del lenguaje natural y en la compilación de programas.
  + **Análisis sintáctico:**El análisis sintáctico es un proceso que permite analizar la estructura de una oración y la función de las palabras que la componen.



* + **Generación de código:**La generación de código es un proceso que crea código de programa a partir de datos de entrada. Se puede realizar de forma automática, utilizando herramientas como los generadores de código o compiladores.
  + **Optimización de código:** La optimización de código es un proceso que mejora la eficiencia de un programa informático. El objetivo es que el código se ejecute más rápido y use menos recursos de cálculo.

**Lenguajes de programación utilizados en la creación de compiladores**: como **Lex y Yacc**: Los compiladores como Yacc y Lex se crean usando el lenguaje de programación C.

Yacc

Yacc (Yet Another Compiler Compiler) es un generador de analizadores sintácticos.

El código fuente generado por Yacc está escrito en C.

Yacc ha sido usado para compilar lenguajes como C++, Fortran 77, AWK, Ratfor, APL, bc, y m4.

La entrada de Yacc es una gramática libre de contexto (CFG).

La salida de Yacc es un analizador sintáctico ascendente LALR(1) en C y una cabecera de símbolos o tokens.

Lex

Lex es una herramienta de los sistemas UNIX/Linux.

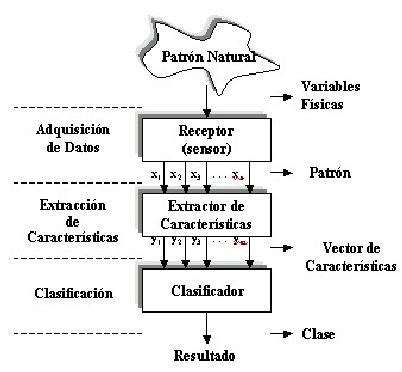
Lex genera código C que se puede compilar y enlazar con un programa.

Lex permite asociar acciones descritas en C a la localización de las Expresiones Regulares que se le definen.

Un compilador es un software que traduce un programa escrito en un lenguaje de alto nivel a un lenguaje de bajo nivel.

**7. Aplicaciones prácticas de autómatas:** Los autómatas tienen aplicaciones en la industria, en el software y en la vida cotidiana

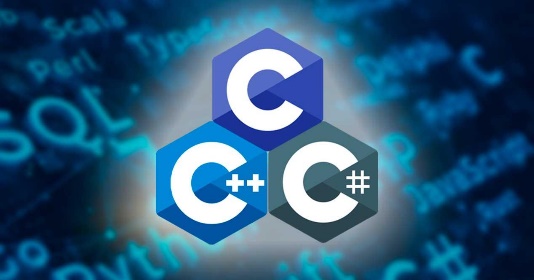
* **Reconocimiento de patrones:** El reconocimiento de patrones tiene aplicación en visión artificial, segmentación de imágenes, detección de objetos, procesamiento de datos de radar, reconocimiento de voz, clasificación de texto y mucho más.



* **Procesamiento de lenguaje natural y su uso en inteligencia artificial:** El procesamiento del lenguaje natural (PLN) es una rama de la inteligencia artificial (IA) que permite a las computadoras comprender y comunicarse con el lenguaje humano.
* **Análisis de texto y minería de datos**: La minería y el análisis de texto ayudan a las organizaciones a encontrar información empresarial potencialmente valiosa en documentos corporativos, correos electrónicos de clientes, registros de centros de llamadas, comentarios textuales de encuestas, publicaciones en redes sociales, historiales médicos y otras.

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

C++ es un lenguaje de programación de propósito general, que se basa en el lenguaje C y se caracteriza por ser un lenguaje orientado a objetos. Combina la eficiencia y la potencia del lenguaje C con la flexibilidad y la abstracción de la programación orientada a objetos.



* **Librerías fundamentales**:

**STDIO:**

**Fopen:** Abre un archivo para lectura, escritura o adición

**Fprintf:** Formatea e imprime caracteres y valores en la secuencia de salida

**Fputc:** Imprime un carácter en la secuencia de salida

**Fclose:** Cierra un archivo a través de su puntero

**Remove:** Elimina un archivo

**Rename:** Cambia el nombre de un archivo

**Rewind:** Coloca el indicador de posición de archivo al comienzo del archivo

**Tmpfile:** Crea y abre un archivo temporal que se borra al cerrar

**Feof:** Comprueba el indicador de final de archivo

**Ferror:** Comprueba el indicador de errores

* **CONIO:**

**getch()**: Lee un carácter del teclado

**clrscr()**: Borra la pantalla de la consola

**textcolor()**: Cambia el color del texto

**cprintf()**: Imprime la salida formateada en la consola

Características de la librería conio.h

* **Lista de 10 librerías más comunes en C/C++:**

**<algorithm>**: Contiene algoritmos de la plantilla estándar

**<chrono>**: Permite medir el tiempo

**<complex>**: Permite usar números complejos

**<exception>**: Contiene excepciones estándar

**<functional>**: Contiene objetos para funciones

**<initializer\_list>**: Permite inicializar listas

**<iterator>**: Permite definir iteradores

**<limits>**: Calcula límites numéricos

**<locale>**: Permite usar localizaciones

**<memory>**: Contiene elementos de memoria

* **Manejadores de formatos para diferentes tipos de datos:** Los manejadores de datos son herramientas que permiten transformar formatos de datos específicos. Se pueden utilizar cuando no hay una transformación estándar disponible.

## Ejemplo

Utilice el estilo C printf para imprimir tres números rodeados entre corchetes:

[**IMPRIMIR**](https://www.nv5geospatialsoftware.com/docs/print.html),[**FINDGEN**](https://www.nv5geospatialsoftware.com/docs/findgen.html)( 3 ), FORMATO = 'Los valores son: [ %f %f %f]'

Impresiones IDL:

Los valores son: [ 0.000000 1.000000 2.000000 ]

