NOMBRE: Osman Neemias Gómez Sagastume

GRADO: Quinto bachillerato en ciencias y letras con orientación en computación.

MATERIA: Producción.

1. Teoría de Autómatas

Definición de Autómatas Finitos Deterministas (AFD)

Un **Autómata Finito Determinista (AFD)** es un modelo matemático utilizado en informática para representar sistemas que tienen un **número finito de estados** y que cambian de estado al recibir una entrada.

Se llama **determinista** porque **para cada estado y símbolo de entrada existe una única transición posible**.

Características

- Tiene un conjunto de **estados**.
- Tiene un **estado inicial**.
- Puede tener **uno o varios estados finales**.
- Usa un **alfabeto de entrada**.
- Tiene una **función de transición** que indica cómo cambiar de estado.

Usos de los AFD

- Validación de cadenas de texto.
- Diseño de compiladores.
- Procesamiento de lenguaje.
- Reconocimiento de patrones.

Ejemplo

Un AFD que reconoce números binarios que terminan en **1**.

Entradas posibles:

0

1

Cadenas aceptadas:

1 101 111 1001

2. Gramáticas Formales

Las **gramáticas formales** son reglas utilizadas para definir la estructura de los lenguajes de programación o lenguajes formales.

Gramáticas Regulares

Son el tipo de gramática más simple y se usan para describir **lenguajes regulares**.

Características

- Sus reglas siguen una estructura simple.
- Son usadas en el **análisis léxico de compiladores**.

Ejemplo:

S → aS

S → b

Esto genera cadenas como:

ab

aab

Aaab

Uso en lenguajes de programación

- Reconocimiento de **identificadores**
- Reconocimiento de **números**
- Reconocimiento de **palabras clave**

Gramáticas Recursivas

Una **gramática recursiva** es aquella donde una regla **se llama a sí misma**.

Ejemplo

E → E + T

E → T

Esta gramática permite crear expresiones como:

a + b

a + b + c

Diferencia con las gramáticas regulares:

Gramática Regular	Gramática Recursiva
Más simple	Más compleja
Describe lenguajes regulares	Describe lenguajes más complejos
Usada en análisis léxico	Usada en análisis sintáctico

3. Análisis Léxico

El **análisis léxico** es la primera fase del compilador.

Se encarga de **leer el código fuente y dividirlo en partes llamadas tokens**.

Tokenización

La **tokenización** es el proceso de dividir el código en unidades llamadas **tokens**.

Ejemplo de código:

int x = 10;

Tokens:

int

x

=

10

;

Expresiones Regulares

Las **expresiones regulares** son patrones que se usan para identificar textos.

Ejemplo:

[0-9]+

Reconoce números como:

10

25

300

Se usan para reconocer: Números, Identificadores y Palabras Clave.

Reconocimiento de Palabras Clave

Las **palabras clave** son palabras reservadas del lenguaje.

Ejemplos en C:

int

float

if

while

return

El compilador las reconoce mediante **tablas de símbolos o comparaciones directas**.

Identificación de Tokens

Los tokens se clasifican en diferentes tipos.

Ejemplos:

Token	Tipo
int	palabra clave
x	identificador
25	número
+	operador
;	símbolo

4. Análisis Sintáctico

El **análisis sintáctico** verifica si la estructura del programa es correcta según la gramática del lenguaje.

Tipos de Análisis Sintáctico

Análisis Sintáctico Descendente

Analiza el programa **desde la raíz hacia las hojas**.

También se conoce como **Top-Down Parsing**.

Ejemplo:

Parser LL

Análisis Sintáctico Ascendente

Analiza el programa **desde las hojas hacia la raíz**.

También se conoce como **Bottom-Up Parsing**.

Ejemplo:

Parser LR

Árboles de Análisis Sintáctico

Un **árbol sintáctico** muestra la estructura de una expresión o programa.

Ejemplo:

Expresión:

a + b

Árbol:

```
 +
/ \
a   b
```

- Sirve para: Entender la estructura del programa, generar código y optimizar código.

5. Algoritmo de Boyer-Moore

El **algoritmo Boyer-Moore** es un método eficiente para **buscar una cadena dentro de un texto**.

Características

- Compara la cadena **de derecha a izquierda**.
- Salta partes del texto para acelerar la búsqueda.

Ejemplo

Texto: Programación

Patrón: RAM

El algoritmo encuentra rápidamente la posición de la palabra.

Aplicaciones

- Buscadores de texto
- editores de código
- análisis de programas

6. Compiladores e Intérpretes

Compilador

Un **compilador** es un programa que traduce un lenguaje de programación a **lenguaje máquina**.

Ejemplo:

C → código máquina

Estructura de un Compilador

Un compilador se compone de varias fases.

Fases del Compilador:

1. Análisis Léxico

Divide el código en tokens.

2. Análisis Sintáctico

Verifica la estructura del programa.

3. Generación de Código

Convierte el programa a código máquina.

4. Optimización de Código

Mejora el código para que sea **más rápido y eficiente**.

Herramientas para crear compiladores

Lex

Herramienta para generar **analizadores léxicos**.

Yacc

Herramienta para generar **analizadores sintácticos**.

7. Aplicaciones Prácticas de Autómatas

Reconocimiento de Patrones

Se utilizan para reconocer patrones en:

- búsqueda de texto
- seguridad informática
- análisis de datos

Procesamiento de Lenguaje Natural

Se usan en sistemas de **inteligencia artificial** para entender lenguaje humano.

Ejemplos:

- traductores automáticos
- chatbots
- asistentes virtuales

Análisis de Texto y Minería de Datos

Se utiliza para analizar grandes cantidades de información.

Aplicaciones:

- análisis de redes sociales
- análisis de opiniones
- detección de spam

8. Lenguaje de Programación C

Librería STDIO

La librería **stdio.h** permite realizar operaciones de **entrada y salida**.

Funciones comunes:

printf()

scanf()

getchar()

putchar()

Ejemplo:

```
#include <stdio.h>

int main() {
    printf("Hola mundo");
    return 0;
}
```

Librería CONIO

La librería **conio.h** permite manejar la consola.

Funciones comunes:

clrscr()

getch()

getche()

Ejemplo:

```
#include <conio.h>

getch();
```

10 librerías comunes en C

1. stdio.h
2. stdlib.h
3. string.h
4. math.h
5. time.h
6. ctype.h
7. stdarg.h
8. signal.h
9. locale.h
10. Assert.h

Manejadores de Formato en C

Los **formatos** se usan en printf y scanf.

Formato	Tipo de dato
%d	entero
%f	flotante
%c	carácter
%s	cadena

Ejemplo:

```
#include <stdio.h>

int main() {
    int edad = 20;
    float altura = 1.75;
     printf("Edad: %d\n", edad);
    printf("Altura: %f", altura);
    return 0;
}
```

ID: 2567 Osman Gomez

Resumen Investigación

7. Teoria de Autómatas:

La teoria de autómatas estudia modelos matematicos que permiten representar modelos de computación. Los Autómatas Finitos Deterministas AFD son sistemas con un número limitado de estados que cambian según las entradas que reciben. Se utilizan en informatica para reconocer patrones, validar datos y en el diseño de compiladores.

2. Gramaticas Formales:

Las gramaticas formales son reglas que definen la estructura de los lenguajes. Las gramaticas regulares describen estructuras simples y las recursivas permiten crear estructuras más complejas.

3. Análisis léxico:

Es la primera fase del compilador. Divide el codigo en tokens como palabras clave, identificadores, números y operadores usando expresiones regulares.

4. Analisis Sintático.

Verifica que el programa esté escrito correctamente según las reglas del lenguaje. Puede ser descendente o ascendente y usa arboles sintacticos.

5. Algoritmo Boyer - Moore.

Es un algoritmo eficiente para buscar palabras o cadenas dentro de un texto, comparando caracteres para encontrar coincidencias más rapido.

6. Compiladores e Interpretes

Un compilador traduce el código de un lenguaje de programación a lenguaje máquina. Sus fases principales son análisis lexico, sintactico, generación y optimización de codigo.

7. Aplicaciones de Autómatas

Se usan en reconocimiento de patrones, inteligencia Artificial, procesamiento de lenguaje natural y analisis de texto.