

NOMBRE: Lesli Yaneth Suazo Pérez

GRADO: 5TO Bachiller en computación

MATERIA: Producción

TRABAJO: Investigación

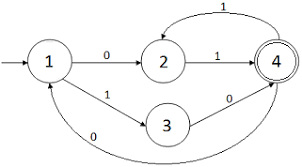
**Temas de Investigación**

**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.

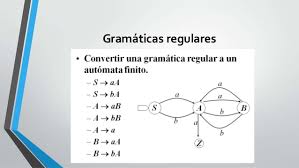
* **Ejemplos y aplicaciones** en informática y ciencias de la computación.



**2. Gramáticas formales**

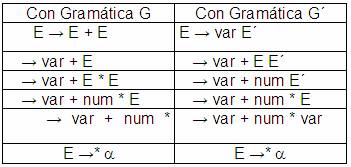
* **Gramáticas regulares**: concepto y ejemplos de su uso en lenguajes de programación.

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 es recursiva **cuando posee al menos una producción recursiva**.



**3. Análisis léxico**

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

En esencia, la tokenización **es como diseccionar una frase para comprender su anatomía**. Al igual que los médicos estudian las células individuales para comprender un órgano, los profesionales de la PNL utilizan la tokenización para diseccionar y comprender la estructura y el significado del texto.

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

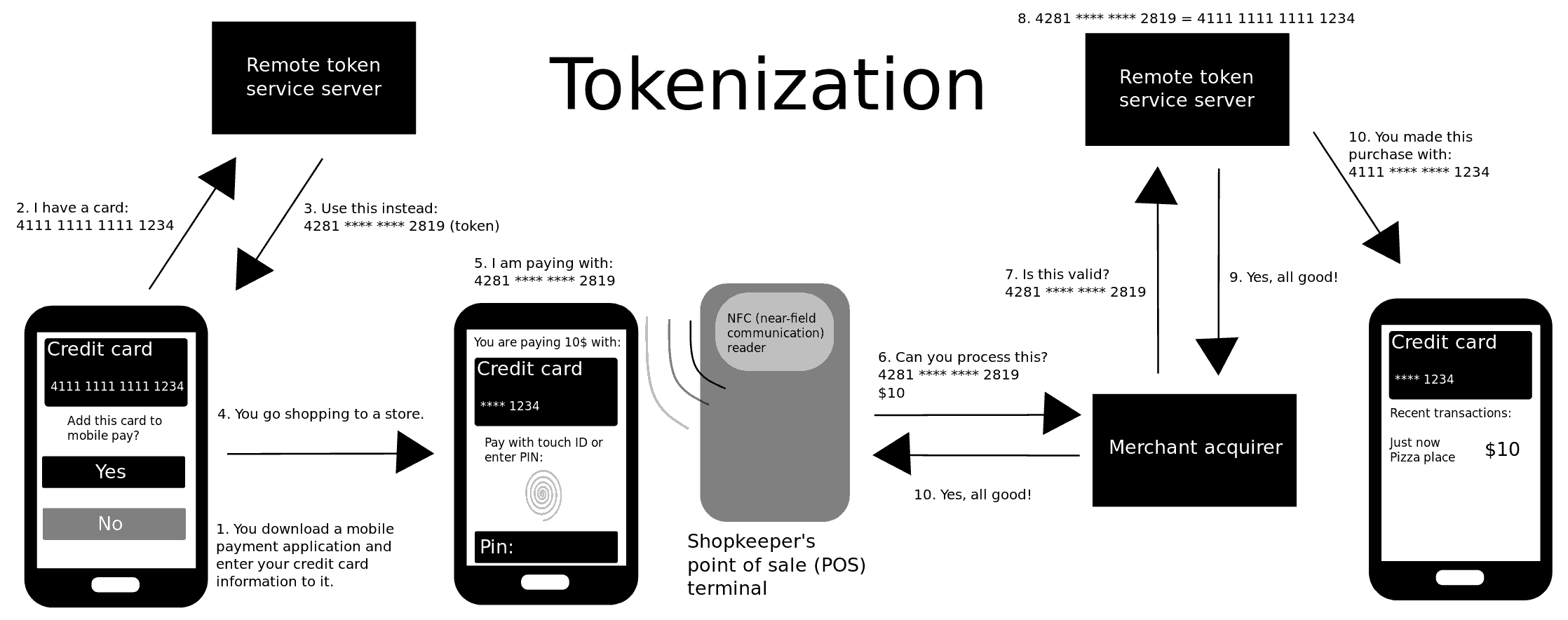
Las expresiones regulares **son una forma de especificar patrones, entendiendo por patrón la forma de describir cadenas de caracteres**. Es la forma de definir los tokens o componentes léxicos y, como veremos, cada patrón concuerda con una serie de cadenas.

* **Reconocimiento de palabras clave**: 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.

Para la autenticación y la autorización, un token es un **objeto digital que contiene información sobre el principal que realiza la solicitud y el tipo de acceso para el que está autorizado**. La autenticación basada en tokens comienza con el inicio de sesión de un usuario en un sistema, dispositivo o aplicación, normalmente utilizando una contraseña o una pregunta de seguridad. Un servidor de autorización valida esa autenticación inicial y luego emite un token de acceso, que es un pequeño dato que le permite a una aplicación de cliente realizar una llamada o señal segura a un servidor API.



**4. Análisis sintáctico**

* **Tipos de análisis sintáctico**:

El **análisis sintáctico** es el **análisis** de las funciones **sintácticas** o relaciones de concordancia y jerarquía que guardan las palabras cuando se agrupan entre sí en forma de sintagmas, oraciones simples y oraciones compuestas de proposiciones.

* + 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.

**Es un tipo de análisis que construye el árbol sintáctico desde las hojas (la parte inferior) hasta la raíz (la parte superior)**.

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

Un árbol de análisis, también conocido como árbol de derivación o árbol de sintaxis, es una estructura de datos utilizada para representar la estructura de una cadena generada por una gramática libre de contexto. Proporciona una representación visual de cómo se puede derivar la cadena a partir de las reglas gramaticales. En el campo de la teoría de la complejidad computacional, los árboles de análisis sintáctico son esenciales para analizar la complejidad de los lenguajes sensibles al contexto y aplicar el lema de bombeo.

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**.

**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 de Boyer-More es considerado el **algoritmo de búsqueda de cadenas más eficiente**, por lo que es implementado en las aplicaciones más habituales, como por ejemplo los editores de texto, los navegadores de internet, los lectores de libros electrónicos.

Funcionamiento

Se precalculan dos tablas que indican cuántas posiciones se pueden saltar hacia delante en cada búsqueda fallida.

Se compara el último carácter del patrón con el carácter del texto.

Si hay una discrepancia, se puede deslizar el patrón m posiciones sin realizar ninguna comparación extra.

Si el carácter discrepante del texto se encuentra dentro del patrón, el patrón podría desplazarse en un número menor de espacios.

Se repite el proceso hasta encontrar el patrón.

El algoritmo de Boyer-Moore es un método para buscar cadenas de caracteres dentro de textos o programas. Se utiliza para encontrar coincidencias entre un patrón y un texto, comparando cada carácter de la cadena o patrón.

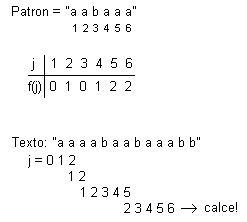
Características del algoritmo de Boyer-Moore

Se preprocesa la cadena objetivo que se busca.

Utiliza la información recopilada en el preprocesamiento para omitir secciones del texto.

Se ejecuta más rápido a medida que aumenta la longitud del patrón.

No necesita comprobar cada carácter de la cadena que se busca, puesto que salta algunos de ellos.



**6. Compiladores e intérpretes**

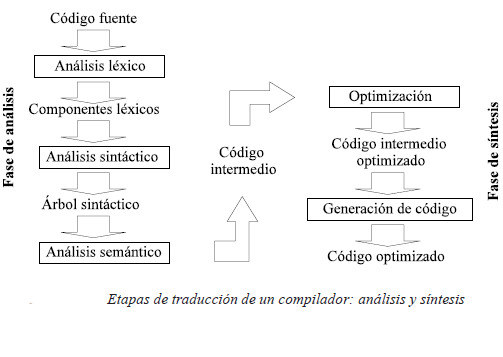
* **Estructura de un compilador**: definición y componentes.

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).

Un compilador está compuesto por dos partes principales: el front-end y el back-end.

Front-end Analizador léxico, Analizador sintáctico, Analizador semántico, Generador de código intermedio.

Back-end Optimización, Generación de código, Ensamblaje



* **Fases del compilador**:
* Las fases del compilador son las etapas que realiza un compilador para traducir un código fuente a código ejecutable. Cada fase tiene una tarea específica y transforma la entrada progresivamente hasta obtener el resultado final.
  + Análisis léxico.
* El análisis léxico es una técnica de análisis del lenguaje que estudia el vocabulario de un discurso. También es la primera etapa de un compilador.
  + Análisis sintáctico.
* El análisis sintáctico es un proceso que permite analizar la estructura de una oración para identificar y clasificar las funciones de sus elementos.
  + Generación de código.
* La generación de código es una fase de la programación que consiste en convertir un programa en instrucciones que una máquina pueda interpretar.
  + Optimización de código.
* La optimización de código es un proceso que mejora la eficiencia y el rendimiento de un programa. El objetivo es que el código se ejecute más rápido y use menos recursos del sistema.
* **Lenguajes de programación utilizados en la creación de compiladores**: como **Lex y Yacc**.
* Los compiladores se pueden escribir en varios lenguajes de programación, como C, C++, C#, Java, BASIC, COBOL, PASCAL, entre otros.
* Lenguajes de programación utilizados para crear compiladores
* C, C++, C#, Java, BASIC, COBOL, PASCAL.

**7. Aplicaciones prácticas de autómatas**

* **Reconocimiento de patrones** en diferentes áreas.

La detección de patrones es un proceso clave en diversas áreas, como **la inteligencia artificial, la minería de datos, la estadística y la investigación científica**, ya que permite descubrir relaciones y tendencias ocultas que de otra manera podrían pasar desapercibidas.

* **Procesamiento de lenguaje natural** y su uso en inteligencia artificial.

El procesamiento del lenguaje natural (PLN) es una rama de la inteligencia artificial que permite a las computadoras comprender, interpretar y generar el lenguaje humano.

¿Cómo funciona?

* Utiliza aprendizaje automático para procesar y analizar grandes cantidades de datos de texto y voz.

 Permite a las máquinas comprender el significado de las palabras y las oraciones.

 Permite a las máquinas tomar decisiones basadas en el significado de los textos.

* **Análisis de texto y minería de datos**: su importancia en el análisis de información.

El análisis de datos y la minería de datos suelen utilizarse indistintamente, pero existe una gran diferencia entre ambos. **El análisis de datos es el proceso de interpretación de los datos para encontrar tendencias y patrones.** **Por otro lado, la minería de datos es el proceso de extracción de información valiosa de un gran conjunto de datos** .

La minería de datos es el proceso de hallar anomalías, patrones y correlaciones en grandes conjuntos de datos para predecir resultados. Empleando una amplia variedad de técnicas, puede utilizar esta información para **incrementar sus ingresos, recortar costos, mejorar sus relaciones con clientes, reducir riesgos y más**.

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

* **Librerías fundamentales**: **STDIO y CONIO,** un resumen de las funciones de cada una.

Las librerías fundamentales son un conjunto de funciones, clases y métodos que se pueden incluir en un código para usar funcionalidades ya predefinidas.

BOOST: proporciona estructuras de datos.

WX WIDGET: crea interfaces graficas.

QT: crea aplicaciones graficas.

OPEN CV: manipula imágenes y realiza operaciones de visión.

FSTREAM: maneja archivos.

IOSTREAM: entrada y salida.

LIST: contiene listas.

VECTOR: contiene vectores.

MATH: matemáticas.

MEMORY: maneja memoria.

STDIO: Stdio.h es un archivo de cabecera que contiene las definiciones de funciones, constantes y macros para realizar operaciones de entrada y salida estándar en el lenguaje de programación C.

Stdio.h significa "standard input-output header" (cabecera estándar de entrada y salida).

CONIO: Conio.h es una librería en C que se utiliza para controlar la entrada y salida de datos por consola. Se usa principalmente en los compiladores de MS-DOS.

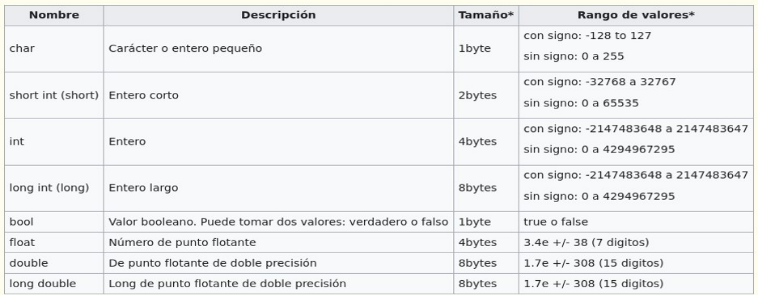
* **Lista de 10 librerías más comunes en C/C++**, además de las ya mencionadas.

Las bibliotecas o librerías en C++ son **archivos que contienen código precompilado para realizar tareas comunes como entrada/salida, funciones matemáticas, y más**. Algunas bibliotecas estándares incluyen iostream, string, y math.

* **Manejadores de formatos para diferentes tipos de datos**, con ejemplos de uso en código.

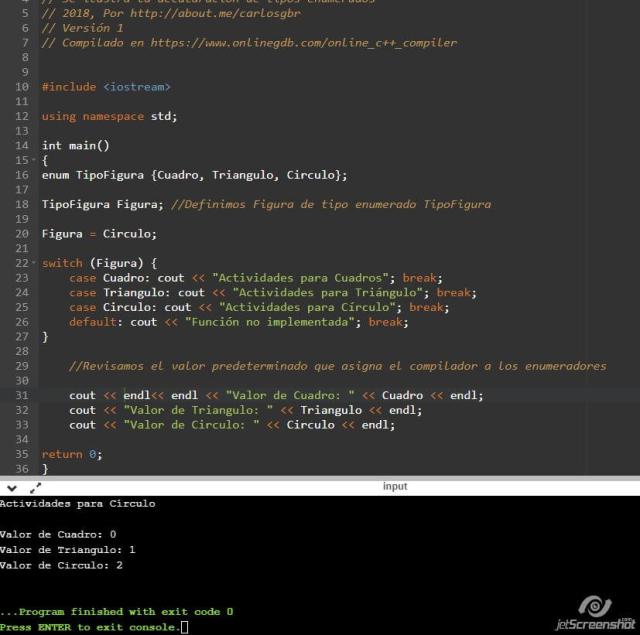
TIPOS DE DATOS SIMPLES EN C++

C++ es un lenguaje de programación que hereda muchos conceptos del lenguaje C, es un lenguaje compilado y fuertemente tipado, lo que significa que en las variables con las que trabajamos hay que indicar el tipo del dato que van a guardar cuando se declaran, lo que puede hacer que tengamos problemas y se generen errores.



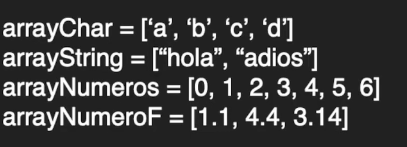
TIPOS DE DATOS ENUMERADOS EN C++

Los tipos enumerados son un mecanismo usado en C++ que nos permite agrupar constantes simbólicas.



TIPOS DE DATOS COMPLEJOS EN C++

Los array son un tipo dato complejo, en el sentido de que nos permiten guardar conjuntos de datos, pero siempre datos del mismo tipo.



PROGRAMACION ORIENTADA A OBJETOS EN C++

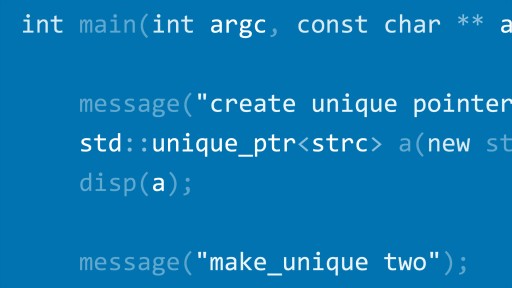
La Programación Orientada a Objetos es la característica más importante de C++.

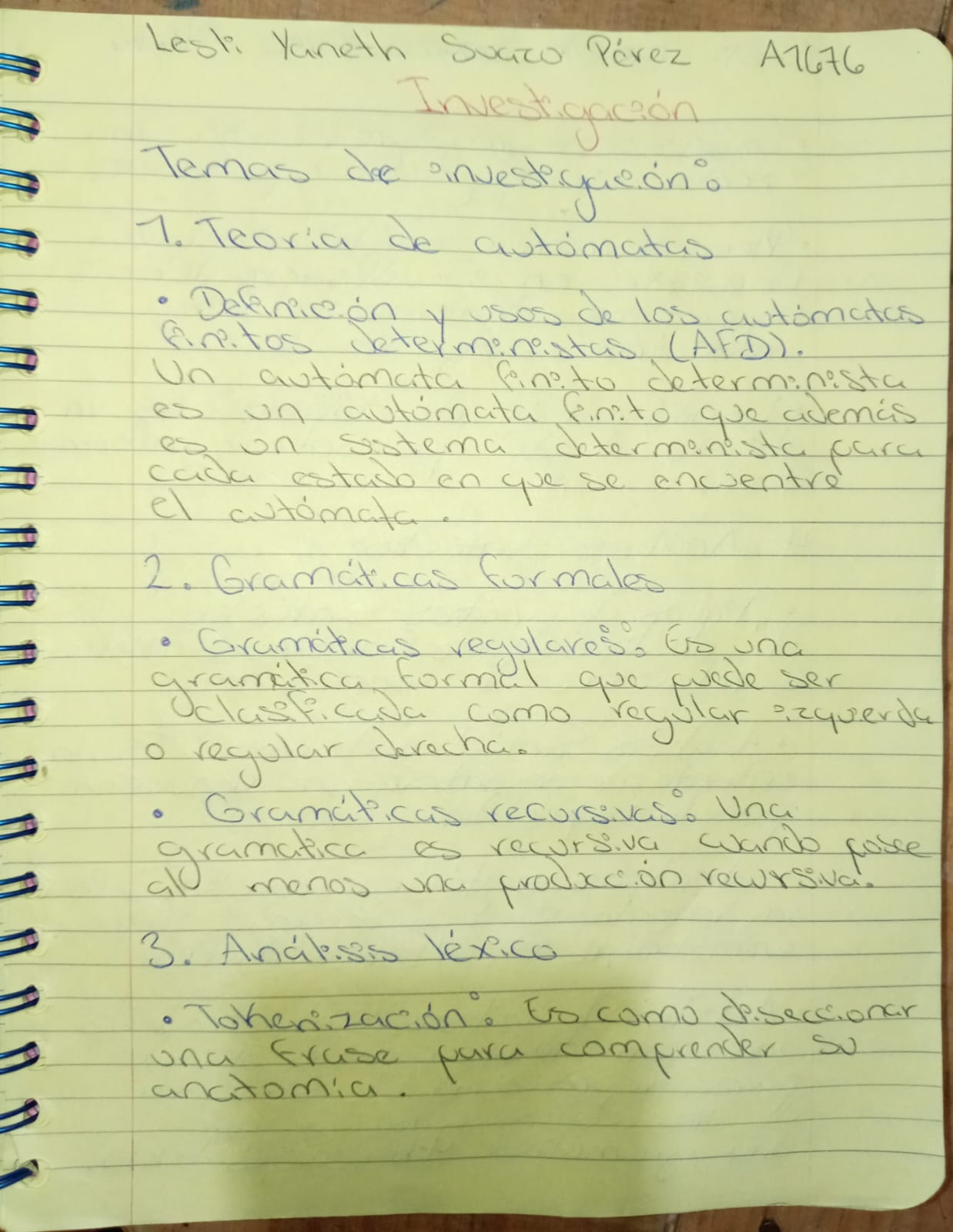
Por ejemplo, la cadena de caracteres, que es otro tipo de datos que utilizamos normalmente en C++, realmente no es un tipo de dato, sino que el string, que representa una cadena de caracteres es una clase.

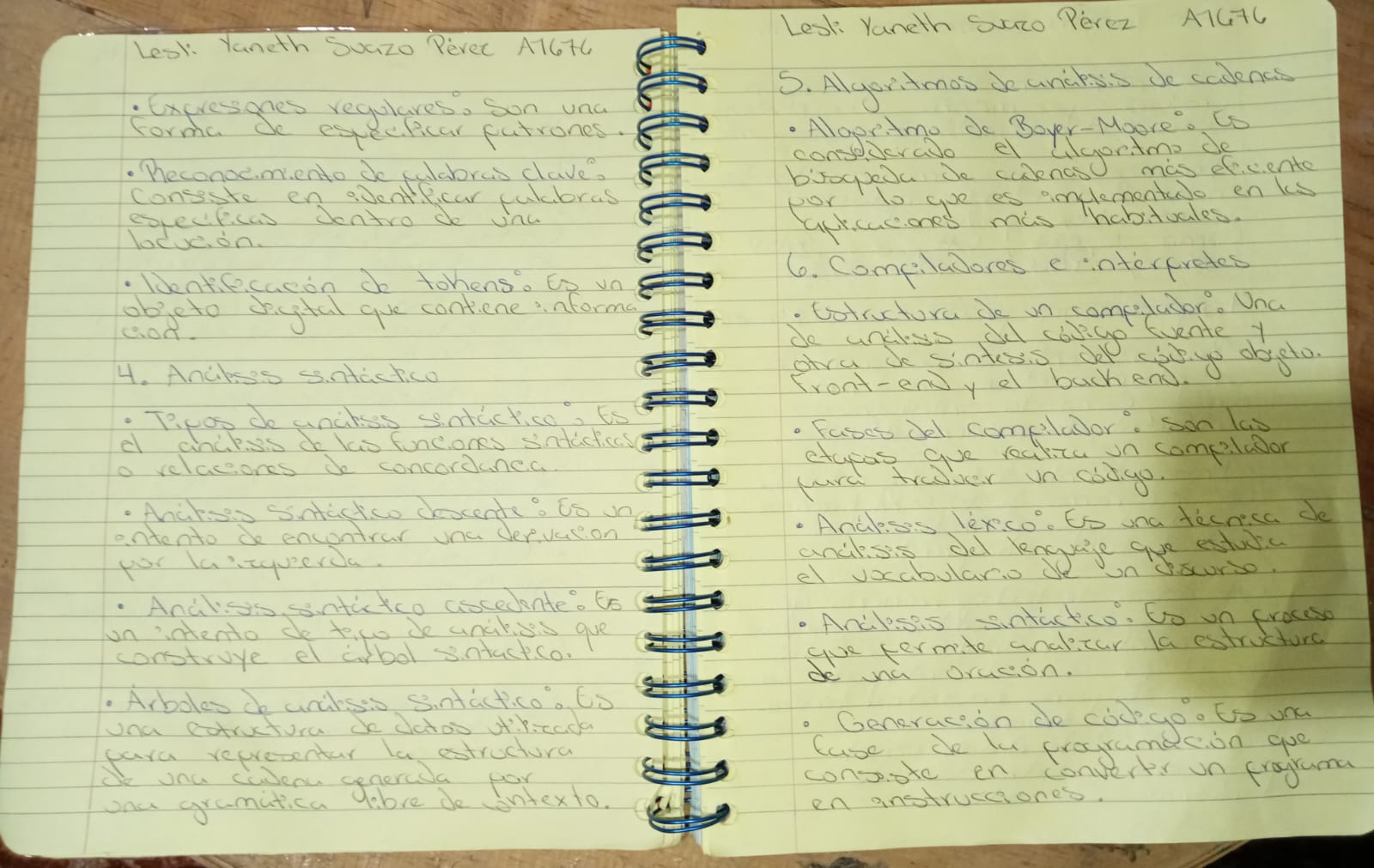


FORMACION EN C+++

Si quieres convertirte en programador en lenguaje C++ esta carrera te resultará de gran interés, ya que está pensada principalmente para no iniciados, aunque si ya tienes conocimientos previos, te permitirá profundizar en ellos. A lo largo de la misma vas a aprender desde los fundamentos de programación hasta los diferentes paradigmas, como la orientación de objetos, la programación genérica y la programación concurrente.







Texto, Carta

Descripción generada automáticamente