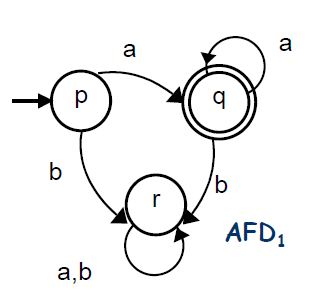
**Temas de Investigación**

**1. Teoría de autómatas**

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

**Modelo matemático que consta de un conjunto de estados, un estado inicial, un alfabeto, un conjunto de estados de aceptación y una función de transición**

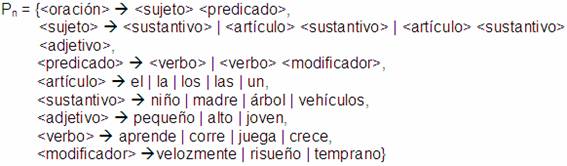
* **Ejemplos y aplicaciones** en informática y ciencias de la computación.

Los autómatas finitos deterministas (AFD) tienen aplicaciones en el análisis léxico, el reconocimiento de patrones, el diseño de circuitos y el reconocimiento de lenguajes.

**2. Gramáticas formales**

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

Una gramática de un lenguaje de programación es un **conjunto de instrucciones sobre cómo escribir declaraciones que sean válidas para ese lenguaje de programación**.



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

Una producción es recursiva por derecha cuando el símbolo no terminal del lado izquierdo de la regla de producción, aparece en el último lugar en el lado derecho de la misma. **Una gramática es recursiva cuando posee al menos una producción recursiva**

Las gramáticas recursivas pueden contener reglas de producción recursivas, mientras que las gramáticas regulares siguen un patrón

**3. Análisis léxico**

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

La tokenización es un proceso que consiste en sustituir datos sensibles por tokens, que son cadenas de símbolos aleatorios y no confidenciales. Se utiliza en la banca, la seguridad de datos, el procesamiento de pagos, y en el ámbito de la economía digital.

Tokenización : durante la tokenización, **el compilador escanea el código fuente de izquierda a derecha y lo divide en tokens**.

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

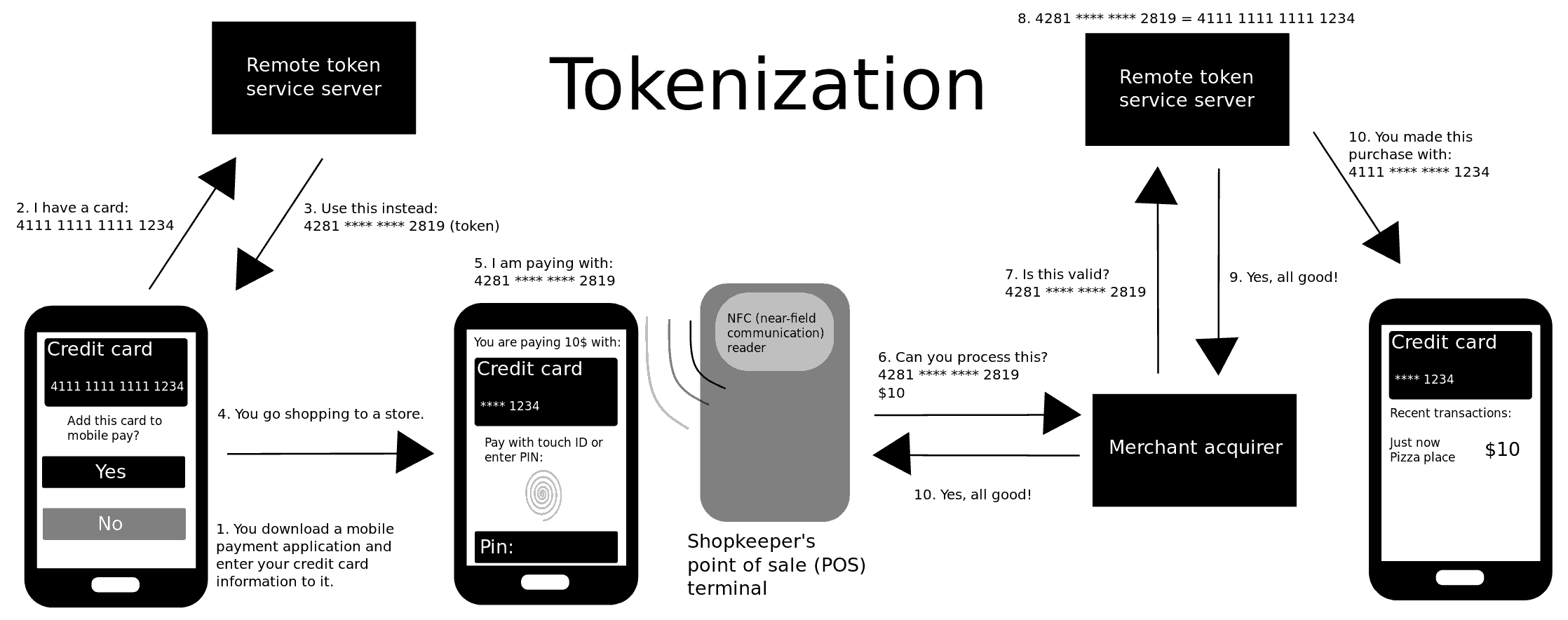
Las expresiones regulares **son una forma de especificar patrones, entendiendo por patrón la forma de describir cadenas de caracteres**.

* **Reconocimiento de palabras clave**: cómo se identifican en los lenguajes de programación.

Las expresiones regulares son patrones que se utilizan para encontrar, validar o manipular cadenas de texto. En el análisis léxico, se usan para identificar y extraer tokens del código fuente.

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

La clasificación de tokens se refiere al **proceso de asignar una etiqueta o categoría a cada token o elemento en una secuencia de texto**



**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**

* + 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)**

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

Un árbol de análisis sintáctico es una representación gráfica de la estructura sintáctica de una cadena de caracteres. También se le conoce como árbol de derivación o árbol de sintaxis concreta

**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 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.

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

* **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