PRACTICA 1

PRÁCTICA 1. ANÁLISIS DE EXPRESIONES REGULARES

IDENTIFICACIÓN:

NOMBRE DE LA PRÁCTICA:	ANÁLISIS DE EXPRESIONES REGULARES
NO. DE PRÁCTICA:	1
NO. DE SESIONES:	2
NO. DE INTEGRANTES MÁXIMO POR EQUIPO:	1

NOMBRE: SANCHEZ CHAVEZ OJANNA A.

INTRODUCCIÓN

Una Gramática Formal definida sobre un alfabeto ∑ es una tupla de la forma:

G = {∑**T**, ∑**N**, **S**, **P**} donde:

- o ∑**T** es un alfabeto de símbolos terminales
- o ∑**N** es un alfabeto de símbolos no terminales
- o ***S*** es el símbolo inicial de la gramática
- o **P** es un conjunto de producciones gramaticales

Además, se cumple:

S ∈ ∑**N** ∑**T** ∩ ∑**N** = ∅ ∑ = ∑**T** ∪ ∑**N**

La gramática formal **G** permite generan un lenguaje **L** = {**x** ∈ ∑**T** ∗ |**S** ∗ → **x**}

Por lo tanto, las palabras del lenguaje estarán formadas por cadenas de símbolos terminales generadas a partir del símbolo inicial de la gramática utilizando las producciones que la definen. Una expresión regular es una notación normalizada para representar lenguajes regulares, es decir, lenguajes generados por gramáticas regulares (Tipo 3). Las expresiones regulares permiten describir con exactitud y sencillez cualquier lenguaje regular.

OBJETIVO GENERAL

Construir programas en un lenguaje de programación de alto nivel que realicen el análisis de cadenas de símbolos que forman palabras basado en la implementación de estructuras de datos y algoritmos de análisis, además del uso de expresiones regulares como segunda estrategia de implementación.

OBJETIVOS ESPECÍFICOS

★Identificar el proceso básico de análisis de cadenas de símbolos de un alfabeto.

★ Implementar estructuras de datos y algoritmos de análisis de cadenas de símbolos en un lenguaje de programación de alto nivel.

★ Implementar expresiones regulares para el análisis de cadenas de símbolos en un lenguaje de programación de alto nivel.

EQUIPO Y SOFTWARE A UTILIZAR

Equipo de cómputo con software para programación a elección de los alumnos.

DESARROLLO

1.Utilizar un lenguaje de programación para construir programas basados en estructuras de datos para analizar cadenas de símbolos con el propósito de identificar:

- Número entero
- Palabras en minúsculas
- Palabras en mayúsculas
- Identificadores (Nombres de variables)

2.Definir el alfabeto de símbolos y las expresiones regulares que permita identificar las siguientes tipos de palabras de un lenguaje regular:

- Números enteros
- Palabras en minúsculas
- Palabras en mayúsculas
- Identificador (Nombres de variables)
- Símbolos 3.

3.Utilizar un lenguaje de programación para construir programas basados en expresiones regulares para analizar cadenas de símbolos con el propósito de clasificar y contabilizar las palabras en las categorías:

- Número entero
- Palabras en minúsculas
- Palabras en mayúsculas
- Identificador (Nombre de variable)
- Símbolo

CUESTIONARIO

- ¿Cúal es utilidad de las expresiones regulares en la identificación de palabras de un lenguajes?

  nos ayuda a definir reglas gramaticales

- ¿Cuáles elementos del lenguaje de programación utilizaste para la implementación de las estructuras de datos?

  arreglos, ciclos y condicionales

- ¿Cómo las utilizas?

  los arreglos para almacenar los token que dividen en palabras mi texto, el ciclo for para recorrer el arreglo llamado token que almacena mis tokens

- ¿Cuál es el proceso que seguiste para analizar las cadenas de símbolos?

  utilizando el alfabeto de guia

- ¿Qué elementos del lenguaje de programación te permitieron implementar expresiones regulares?

  Secuencias de escape(\s+) para dividirlo en palabras y expresiones regulares

- ¿Qué condiciones o recomendaciones se deben seguir al escribir las expresiones regulares?

  deben de ser claras, definir si la entrada va a reconocer uno o más, y en qué formato, ya sean números, letras o símbolos

RESULTADOS

CÓDIGO 1

```
import java.util.Scanner;

public class MinMayus {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Ingresa un texto");

        String entrada = scanner.nextLine();

        String tokens[] = entrada.split("\s+");
```

```
        for (String token : tokens) {
            if (token.matches("[0-9]+")) {
                System.out.println(token + " -es un ENTERO");

            } else if (token.matches("[A-Z]+")) {
                  System.out.println(token + " -esta escrito en
MAYUSCULAS");

            } else if (token.matches("[a-z]+")) {
                  System.out.println(token + " -esta escrito en
MINUSCULAS");

            } else if (token.matches("[a-zA-Z0-9]+")) {
                        System.out.println(token + " -es un
IDENTIFICADOR");

            } else if (token.trim().isEmpty())

                continue;
        }
    }

}
```

CÓDIGO 2

```
import java.util.Scanner;

public class MinMayusSim {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Ingresa un texto");

        String entrada = scanner.nextLine();

        String tokens[] = entrada.split("\s+");
```

```
        for (String token : tokens) {
            if (token.matches("[0-9]+")) {
                System.out.println(token + " -es un ENTERO");

            } else if (token.matches("[A-Z]+")) {
                    System.out.println(token + " -esta escrito en
MAYUSCULAS");

            } else if (token.matches("[a-z]+")) {
                    System.out.println(token + " -esta escrito en
MINUSCULAS");

            } else if (token.matches("[a-zA-Z0-9]+")) {
                            System.out.println(token + " -es un
IDENTIFICADOR");

            } else if (token.matches("[^a-zA-Z0-9]+")) {
                System.out.println(token + " -es un SIMBOLO");

            } else if (token.trim().isEmpty())

                continue;
        }
    }

}
```

CONCLUSIONES

Se utilizaron expresiones regulares para definir ciertas reglas gramaticales y así poder conseguir que se identificaran las palabras en mayúsculas, minúsculas, identificadores y hasta los símbolos.

BIBLIOGRAFÍA

Giró, J., Vázquez, J., Meloni, B., Constable, L. (2015). Lenguajes formales y teoría de autómatas. Editorial Alfaomega. Argentina.

Ruiz Catalán, J. (2010). Compiladores: teoría e implementación. Editorial Alfaomega. México.

Brookshear, J. G. (1995). Teoría de la Computación. Lenguajes formales, autómatas y complejidad. Editorial Addison-Wesley Iberoamericana. Primera edición. U.S.A.