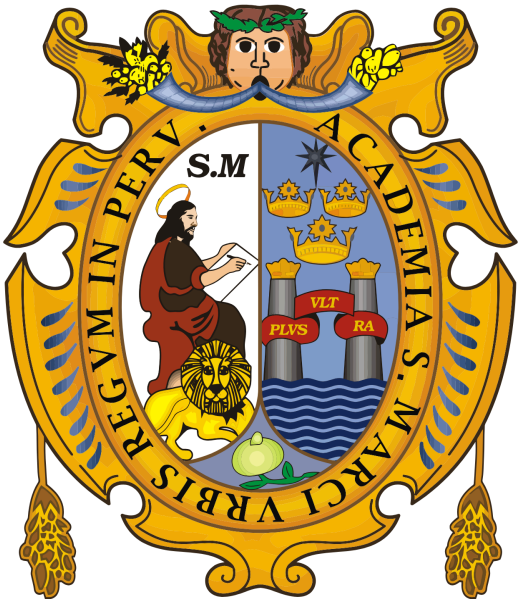
UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, Decana de América)
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA



“Traductor Infija a Postfija con Autómata de Pila”

Asignatura:
TEORÍA DE LA COMPUTACIÓN

Docente:
VICTOR HUGO BUSTAMANTE OLIVERA

Integrantes Grupo 4:

- Verde Jara, Leonel Edwuar - 24200209
- Chamorro Figueroa, Andy Aaron - 24200072
- Sánchez Mendoza, Diego Joaquín - 24200206
- Zapata Llaxa, Jorge Andres - 24200211

13/11/2025

Primer Avance del Proyecto 4
Traductor de Expresiones Infijas a Postfijas (Python)

a. ¿De qué trata el proyecto?

El proyecto que realizaremos en Python se encargará de convertir las expresiones infijas (Notación natural, la que usamos comúnmente) a postfijas (Notación donde los operadores van después de los operandos)

Para esto usaremos el método “Shunting-Yard” creado por el científico en computación Edsger Dijkstra. Su nombre significa “Estación de clasificación” esto es así por su forma en la que hace que los operadores y paréntesis se muevan de manera que respeten la precedencia correcta.

Para resumir, lo que buscaremos es que el programa analice la expresión en notación infija que el usuario le dé paso por paso cumpliendo la prioridad de los operadores y el uso de paréntesis pasando de infijas a postfijas.

b. Plan de trabajo

Seguiremos los siguientes pasos para desarrollar el proyecto:

1. Definir los tokens:
   Veremos qué elementos aceptara el programa en este caso serían números, operadores (+ - * /) y paréntesis.
2. Crear las estructuras base:
   Lo haremos con dos listas en Python:
   - ➔ Una para el resultado en notaciòn postfija
   - ➔ Otra que sería la pila donde se guardarán operadores y paréntesis.
3. Establecer la prioridad de los operadores:
   Esto sería una pequeña tabla para indicar que * y / tienen más prioridad que + y -.
4. Implementar las reglas principales del método:
   Hacer que el programa siga el método Shunting-Yard. Esto incluye decidir qué hacer cuando aparece un número, un operador o un paréntesis.
5. Hacer pruebas:
   Ver si funciona correctamente con expresiones pequeñas, primero sin paréntesis y luego con ellos.
6. Intentar con expresiones erróneas:
   Detectar expresiones incorrectas, como paréntesis sin cerrar.

7. Preparar el informe final:
   Explicar cómo funciona el programa, cómo se usa y el código.

c. Primer avance en código

Mostramos un poco del código donde definimos los tokens iniciales, las estructuras necesarias y la tabla de prioridades.

```
# tokens de ejemplo
tokens = ["2", "+", "3", "*", "4"]

# Estructuras del algoritmo Shunting-Yard
output = []     # Aquì ira la salida en postfijo
stack = []      # Esta es la pila

# Prioridad de operadores
precedencia = {
    "+": 1,
    "-": 1,
    "*": 2,
    "/": 2
}

print("Tokens cargados:", tokens)
print("Estructuras listas para iniciar el algoritmo.")
```