Lindsay Mishelle Quirós Enríquez
19 de octubre de 2025

Caso I:

Análisis de Expresiones Aritméticas

SOFT - 10

SCV2

Profesor: Romario Salas Cerdas

Índice

Índice..... 2
1. Introducción..... 2
2. Marco Teórico..... 2
2.1 Estructura de datos pila..... 2
2.2 Validación de expresiones aritméticas y pilas..... 3
3. Desarrollo del programa..... 3
3.1 Clases utilizadas..... 3
3.2 Lógica de tokenización y validación..... 4
3.3 Menú de interacción..... 4
4. Resultados..... 4
5. Representación gráfica del funcionamiento de la pila..... 4
6. Link al repositorio de gitHub..... 5
7. Conclusiones..... 5
Referencias Bibliográficas..... 5

1. Introducción

El siguiente estudio de caso aborda la utilización de la estructura de datos pila para el análisis y validación de expresiones aritméticas. El objetivo es determinar si una expresión compuesta por constantes literales numéricas, variables numéricas, operadores aritméticos y paréntesis es sintácticamente correcta sin necesidad de conocer los valores de las variables. Para ello se implementa una solución en Java que utiliza una pila para gestionar la estructura de la expresión y determinar errores de sintaxis, como operadores consecutivos, paréntesis desbalanceados, o variables sin operador. Este enfoque responde a la necesidad de garantizar que las expresiones cumplen con las reglas de organización y jerarquía de operaciones antes de su evaluación o uso en un programa, además de entender a profundidad cómo funcionan las pilas y su importancia.

2. Marco Teórico

2.1 Estructura de datos pila

Una pila es una estructura lineal de datos en la que las inserciones y eliminaciones se realizan únicamente en un extremo llamado 'tope' o 'cima'. Por ello se describe también como LIFO (Last In, First Out, es decir, el último en entrar es el primero en salir). Las operaciones básicas son: push (apilar un elemento), pop (desapilar el

elemento en el tope o la cima) y peek/top (consultar el elemento en la cima sin eliminarlo). Las pilas se utilizan para diferentes aplicaciones como: control de llamadas de función, reversión de operaciones, recorridos de estructuras y análisis de expresiones.

2.2 Validación de expresiones aritméticas y pilas

La pila es ideal para la validación de expresiones aritméticas porque permite gestionar elementos que deben abrirse y cerrarse en orden inverso, como los paréntesis. Cada vez que se encuentra un paréntesis de apertura se apila, y cada vez que se encuentra un paréntesis de cierre se desapila. Al final, la pila debe quedar vacía, asegurando que todos los paréntesis estén correctamente abiertos y cerrados. Esto es difícil de garantizar con listas o arreglos simples, ya que no respetarán la relación LIFO (primero en entrar primero en salir) necesaria para validar correctamente los paréntesis.

Las ventajas de usar este método son:

1. Seguridad: Evita que paréntesis de cierre aparezcan sin apertura.
2. Orden: Garantiza que los paréntesis se cierren en el orden inverso al que se abrieron.
3. Eficiencia: Las operaciones de apilar y desapilar son un mismo paso que se repite constantemente, por lo que incluso si son expresiones grandes se procesan rápidamente.

3. Desarrollo del programa

3.1 Clases utilizadas

Para la evidenciación de este caso de estudio en el programa se hizo el uso de las siguientes clases.

- Nodo.java: representa cada elemento de la pila.
- Pila.java: implementa las operaciones básicas de la pila.
- Validaciones.java: contiene la lógica de validación de expresiones.
- Main.java: en este se lleva a cabo el uso de menú por parte del usuario.

3.2 Lógica de tokenización y validación

La expresión que sea ingresada por el usuario se separa en tokens: constantes literales numéricas, variables, operadores aritméticos y paréntesis. Durante el recorrido de la pila, se valida la secuencia para evitar operadores consecutivos o finales incorrectos, y se usa la pila para verificar el balance de paréntesis, es decir, si se cuenta con uno de apertura y otro de cierre.

3.3 Menú de interacción

El programa incluye un menú que permite validar expresiones o salir, cumpliendo con los requerimientos del estudio de caso.

4. Resultados

Se realizaron pruebas de caso con varias expresiones:

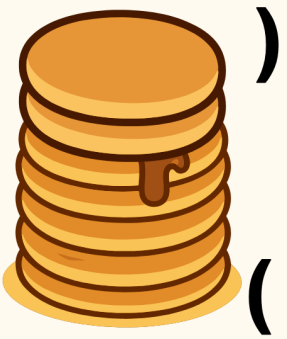
- Expresión válida: 3+num/2 → La expresión es válida.
- Expresión inválida: 3++2 → Error: operador doble '+' en posición 3.
- Expresión inválida: (3+2 → Error: falta cerrar un paréntesis.

5. Representación gráfica del funcionamiento de la pila

A continuación un diagrama para ilustrar el paso a paso de cómo la pila gestiona los paréntesis al analizar la expresión (3+2*x).

(3+2*x)

Pancake 1	(	Pila: (	Apilamos
Pancake 2	3	Pila: (	No cambia nada en la pila
Pancake 3	+	Pila: (	No cambia nada en la pila
Pancake 4	2	Pila: (	No cambia nada en la pila
Pancake 5	*	Pila: (	No cambia nada en la pila
Pancake 6	x	Pila: (	No cambia nada en la pila
Pancake 7	)	Pila → [ ]	Se desapila porque ya se cierra



6. Link al repositorio de gitHub

https://github.com/Lquirose/Casos-de-Estudio-.git

7. Conclusiones

La pila es una herramienta muy eficiente para validar la estructura sintáctica de expresiones aritméticas. Su aplicación garantiza la correcta organización de operadores y paréntesis y en este caso sin depender de los valores de las variables.

Referencias Bibliográficas

Facultad de Estadística e Informática, Universidad Veracruzana. (2021). *Unidad VI. Pilas*. Recuperado de https://www.uv.mx/personal/ermeneses/files/2021/08/Clase7-Pilas.pdf uv.mx

García, M. (s.f.). *ED1 (Pilas)*. Universidad Tecnológica de Morelia. Recuperado de https://www.utm.mx/~mgarcia/ED1(Pilas).pdf

GeeksforGeeks. (2024). *Arithmetic Expression Evaluation*. Recuperado de https://www.geeksforgeeks.org/dsa/arithmetic-expression-evalution/ GeeksforGeeks

GeeksforGeeks. (2025). *Expression Evaluation*. Recuperado de https://www.geeksforgeeks.org/dsa/expression-evaluation/ GeeksforGeeks

Scribd. (s.f.). *Evaluación de Expresiones Aritméticas con Pilas*. Recuperado de https://www.scribd.com/presentation/806890437/8-Evaluacion-de-Expresiones-Aritmeticas-Con-Pilas

Tiliksew, B., Thelwall, J., Khim, J., & Josh, S. (s/f). *Shunting yard algorithm*. Brilliant.org. https://brilliant.org/wiki/shunting-yard-algorithm/