Escola Politécnica da PUC-Campinas

Faculdade de Análise de Sistemas
Curso de Sistemas de Informação

Trabalho de ALP e de ERD1
Trabalho 'Expressões Aritméticas' – 2º Semestre de 2025

Objetivo

O propósito deste trabalho é, usando **pilhas** e **filas**, construir em C o programa de uma **Calculadora de Expressões Aritméticas**, devidamente estruturado em TADs.
O programa deverá ser capaz de receber uma expressão algébrica digitada pelo usuário, avaliá-la e exibir o resultado final.

Exemplo de execução:

Entrada:

10 + (2 * 3 - 4) ^ 2 / 4 + 6 * 2

Saída:

23

Introdução

Para solucionar o problema da **ordem de precedência dos operadores** e o uso de **parênteses** em expressões matemáticas, o matemático polonês **Jan Łukasiewicz** desenvolveu uma forma alternativa de representação que dispensa parênteses e torna a avaliação das expressões mais simples.
Essa representação é conhecida como **notação polonesa** (prefixa) e **notação polonesa reversa** (pós-fixa).

Existem três formas principais de representar uma expressão:

1. **Notação infixa:** o operador aparece entre os operandos.
   Exemplo: (1 + 2)
2. **Notação prefixa:** o operador aparece antes dos operandos.
   Exemplo: (+ 1 2)
3. **Notação pós-fixa:** o operador aparece depois dos operandos.
   Exemplo: (1 2 +)

A notação pós-fixa é a mais eficiente para construção de algoritmos que avaliam expressões matemáticas, pois elimina ambiguidades relacionadas à precedência e aos parênteses.

Modus operandi

A calculadora deve:

1. Ler uma **expressão aritmética na notação infixa**, digitada pelo usuário.
2. Converter essa expressão para a **notação pós-fixa (RPN)**.
3. Calcular o resultado da expressão pós-fixa.

Para isso, deverão ser utilizadas **pilhas** e **filas** como estruturas de dados básicas.
Expressões malformadas devem ser detectadas e sinalizadas.

Operadores aceitos

A calculadora deve reconhecer os seguintes operadores:

+ (adição)
- (subtração)
* (multiplicação)
/ (divisão)
^ (exponenciação)
( ) (parênteses)

Precedência dos operadores

A calculadora deve respeitar a seguinte ordem de precedência (da maior para a menor):

1. Parênteses
2. Exponenciação
3. Multiplicação e divisão
4. Adição e subtração

Quando dois operadores têm a mesma precedência, a expressão deve ser resolvida **da esquerda para a direita**, exceto ^, que é associativo à direita.

Todas as regras ficam resumidas no uso da seguinte tabela:

		Primeiro símbolo da fila de entrada						
		(	**^**	*****	**/**	**+**	**-**	**)**
Símbolo que está no topo da pilha	**(**	F	F	F	F	F	F	T
	^	F	F	T	T	T	T	T
	*****	F	F	T	T	T	T	T
	/	F	F	T	T	T	T	T
	+	F	F	F	F	T	T	T
	-	F	F	F	F	T	T	T
	)	F	F	F	F	F	F	F

Etapa 1 – Leitura e quebra da expressão

O programa deve:

1. Solicitar ao usuário a digitação de uma expressão aritmética.
2. Remover todos os espaços em branco que não estejam entre dígitos (se houver espaços em branco entre dígitos, deve ser dado um erro e a expressão não deve ser calculada).
3. Quebrar a expressão em pedaços (tokens), identificando **números**, **operadores** e **parênteses**, dispondo-os na fila de entrada.

Exemplo:

Entrada:

10 + (2 * 3 - 4) ^ 2 / 4 + 6 * 2

Após remover os espaços:

10+(2*3-4)^2/4+6*2

Após quebrar, armazene os pedaços na uma fila de entrada:

10, +, (, 2, *, 3, -, 4,), ^, 2, /, 4, +, 6, *, 2

Etapa 2 – Conversão da notação infixa para pós-fixa

Alem da **fila de entrada**, deve-se usar **uma pilha de operadores** e **uma fila de saída**.

Quando pegamos da fila de entrada os tipos de itens abaixo, o algoritmo segue as seguintes regras gerais:

- Números → vão para a **fila de saída**.
- Parênteses abertos → são empilhados.
- Parênteses fechados → transferem tudo da pilha de operadores para a fila de saída até desempilhar o correspondente (.
- Operadores → acabam sendo empilhados na pilha de operadores, mas antes, causam o desempilhamento de outros operadores, tantos quanto possível, até a pilha ficar vazia ou a tabela acima parar de indicar T (desempilhamento).

Exemplo passo a passo:

No início:
Fila de entrada: 10, +, (, 2, *, 3, -, 4,), ^, 2, /, 4, +, 6, *, 2
Fila de saída: (vazia)
Pilha de operadores: (vazia)

Passo 1:
Fila de entrada: 10, +, (, 2, *, 3, -, 4,), ^, 2, /, 4, +, 6, *, 2
Ao pegar o 10 da fila de entrada, por ser número, ele vai direto para a fila de saída.
Fila de saída: 10
Pilha de operadores: (vazia)
Fila de entrada: +, (, 2, *, 3, -, 4,), ^, 2, /, 4, +, 6, *, 2

Passo 2:
Fila de entrada: +, (, 2, *, 3, -, 4,), ^, 2, /, 4, +, 6, *, 2
Para pegar o + da fila de entrada, por ser operador, ele vai acabar sendo empilhado na pilha de operadores, mas, antes, 0 ou mais operadores serão de lá desempilhados e postos na fila de saída, dependendo unicamente da pilha não estar vazia e da tabela acima indicar T para o desempilhamento.
Fila de saída: 10
Pilha: (vazia)
Fila de entrada: +, (, 2, *, 3, -, 4,), ^, 2, /, 4, +, 6, *, 2

2. Pega o +. Como a pilha está vazia, nem tem sentido pensar em desempilhar alguma coisa antes de empilhar o + → simplesmente empilha +.
Fila: 10
Pilha: +

Fila de entrada: (, 2, *, 3, -, 4,), ^, 2, /, 4, +, 6, *, 2

3. Pega o (. → considerando a tabela, nada é desempilhado; empilha (.
Fila: 10
Pilha: +, (

Fila de entrada: 2, *, 3, -, 4,), ^, 2, /, 4, +, 6, *, 2

4. Pega o 2. É número → vai para a a fila de saída.
Fila: 10, 2
Pilha: +, (

Fila de entrada: *, 3, -, 4,), ^, 2, /, 4, +, 6, *, 2

5. Pega o *. → considerando a tabela, nada é desempilhado; empilha *.
Fila: 10, 2
Pilha: +, (, *

Fila de entrada: 3, -, 4,), ^, 2, /, 4, +, 6, *, 2

6. Pego o 3. É número → vai para a fila de saida.
Fila: 10, 2, 3
Pilha: +, (, *

Fila de entrada: -, 4,), ^, 2, /, 4, +, 6, *, 2

7a. Pego o -. considerando a tabela, desempilha *, que vai para a fila.
Fila: 10, 2, 3, *
Pilha: +, (

7b. Tendo lido -. considerando a tabela, nada mais é desempilhado; Empilha -.
Fila: 10, 2, 3, *
Pilha: +, (, -

Fila de entrada: 4,), ^, 2, /, 4, +, 6, *, 2

8. Pego o 4. É número → vai para a fila.
Fila: 10, 2, 3, *, 4
Pilha: +, (, -

Fila de entrada:), ^, 2, /, 4, +, 6, *, 2

9. Pega o). → desempilha e põe na fila até encontrar (.
Remove (.
Fila: 10, 2, 3, *, 4, -
Pilha: +

Fila de entrada: ^, 2, /, 4, +, 6, *, 2

10. Pega o ^. → considerando a tabela, nada é desempilhado; empilha ^.
Fila: 10, 2, 3, *, 4, -
Pilha: +, ^

Fila de entrada: 2, /, 4, +, 6, *, 2

11. Pego o 2. É número → vai para a fila.
Fila: 10, 2, 3, *, 4, -, 2
Pilha: +, ^

Fila de entrada: /, 4, +, 6, *, 2

12a. Pego o /. → considerando a tabela, desempilha ^, que vai para a fila.
Fila: 10, 2, 3, *, 4, –, 2, ^
Pilha: +

12b. Tendo lido /. → considerando a tabela, nada mais é desempilhado; Empilha /.
Fila: 10, 2, 3, *, 4, -, 2, ^
Pilha: +, /

Fila de entrada: 4, +, 6, *, 2

13. Pego o 4. É número → vai para a fila.
Fila: 10, 2, 3, *, 4, -, 2, ^, 4
Pilha: +, /

Fila de entrada: +, 6, *, 2

14a. Pego o +. → considerando a tabela, desempilha /, que vai para a fila.
Fila: 10, 2, 3, * ,4 –, 2, ^, 4, /
Pilha: +

14a. Tendo pego o +. → considerando a tabela, desempilha +, que vai para a fila.
Fila: 10, 2, 3, * ,4 –, 2, ^, 4, /, +
Pilha: (vazia)

14b. Tendo pego o +. → considerando que a pilha está vazia, nada mais é desempilhado; Empilha +.
Fila: 10, 2, 3, *, 4, -, 2, ^, 4, /, +
Pilha: +

Fila de entrada: 6, *, 2

15. Pego o 6. É número → vai para a fila.
Fila: 10, 2, 3, *, 4, -, 2, ^, 4, /, +, 6
Pilha: +

Fila de entrada: *, 2

16. Pega o *. → considerando a tabela, nada é desempilhado; empilha *.
Fila: 10, 2, 3, *, 4, -, 2, ^, 4, /, +, 6
Pilha: +, *

Fila de entrada: 2

17. Pega o 2. É número → vai para a fila.
Fila: 10, 2, 3, *, 4, -, 2, ^, 4, /, +, 6, 2
Pilha: +, *

Fila de entrada: (vazia)
Fim da expressão.
Desempilha tudo e vai passando para a fila.
Fila final (pós-fixa): 10, 2, 3, *, 4, -, 2, ^, 4, /, +, 6, 2, *, +
Pilha: (vazia)

Etapa 3 – Cálculo da expressão

Agora avaliamos a expressão pós-fixa com **uma pilha de resultados**.

Fila final (pos-fixa): 10, 2, 3, *, 4, -, 2, ^, 4, /, +, 6, 2, *, +

1. Pegou o 10. É numero? Empilha 10 → Pilha: 10

Fila final (pos-fixa): 2, 3, *, 4, -, 2, ^, 4, /, +, 6, 2, *, +

2. Pegou o 2. É numero? Empilha 2 → Pilha: 10, 2

Fila final (pos-fixa): 3, *, 4, -, 2, ^, 4, /, +, 6, 2, *, +

3. Pegou o 3. É numero? Empilha 3 → Pilha: 10, 2, 3

Fila final (pos-fixa): *, 4, -, 2, ^, 4, /, +, 6, 2, *, +

4. Pegou o * → desempilha 3 e 2 → 2 * 3 = 6 → empilha 6 → Pilha: 10, 6

Fila final (pos-fixa): 4, -, 2, ^, 4, /, +, 6, 2, *, +

5. Pegou o 4. É numero? Empilha 4 → Pilha: 10, 6, 4

Fila final (pos-fixa): -, 2, ^, 4, /, +, 6, 2, *, +

6. Pegou o - → desempilha 4 e 6 → 6 - 4 = 2 → empilha 2 → Pilha: 10, 2

Fila final (pos-fixa): 2, ^, 4, /, +, 6, 2, *, +

7. Pegou o 2. É numero? Empilha 2 → Pilha: 10, 2, 2

Fila final (pos-fixa): ^, 4, /, +, 6, 2, *, +

8. Pegou o ^ → desempilha 2 e 2 → 2 ^ 2 = 4 → empilha 4 → Pilha: 10, 4

Fila final (pos-fixa): 4, /, +, 6, 2, *, +

9. Pegou o 4. É numero? Empilha 4 → Pilha: 10, 4, 4

Fila final (pos-fixa): /, +, 6, 2, *, +

10. Pegou a / → desempilha 4 e 4 → 4 / 4 = 1 → empilha 1 → Pilha: 10, 1

Fila final (pos-fixa): +, 6, 2, *, +

11. Pegou o + → desempilha 1 e 10 → 10 + 1 = 11 → empilha 11 → Pilha: 11

Fila final (pos-fixa): 6, 2, *, +

12. Pegou o 6. É numero? Empilha 6 → Pilha: 11, 6

Fila final (pos-fixa): 2, *, +

13. Pegou o 2. É numero? Empilha 2 → Pilha: 11, 6, 2

Fila final (pos-fixa): *, +

14. Pegou o * → desempilha 2 e 6 → 6 * 2 = 12 → empilha 12 → Pilha: 11, 12

Fila final (pos-fixa): +

15. Pegou o + → desempilha 12 e 11 → 11 + 12 = 23 → empilha 23 → Pilha: 23

Observe que os operadores são desempilhados numa órdem mas são usados pelos operadores em ordem contrária!

Resultado final: 23

Considerações finais

- Este trabalho deverá ser desenvolvido EM DUPLA e deverá ser entregue e DEMONSTRADO impreterivelmente na aula do dia 28/outubro/2025
- A pilha deve terminar com **apenas um valor**, que é o resultado final da expressão.
- Expressões **malformadas** e **divisões por zero** devem ser tratadas adequadamente, com mensagens de erro claras.

Bom Trabalho!
Prof. André Luís
Campinas, 16/outubro/2025