Trabalho I de Estrutura de Dados
Forma Infixa e Posfixa

Alunos:
Tiago L. P. de Pádua - 12/1042457
Ronaldo S. Ferreira Jr. - 09/48721
Alex Leite - 05/97694
Professor:
Eduardo A. P. Alchieri

Forma Infixa e Posfixa

- Forma infixa:
  - Prós:
    - Didadicamente mais eficiente.
    - Visualização da expressão matemática de forma instanânea pelo cérebro humano.
  - Contras:
    - Computacionalmente ineficiente.
    - Ordem das operações são mais complexas de serem implementadas pela forma infixa em um *software*.
  - Exemplo:
    - 5*10 + 85 * (48+9 + 6/2)
      - Ordem das operações: { (5*10) + {85*[(48+9) + (6/2)]} }

Forma Infixa e Posfixa

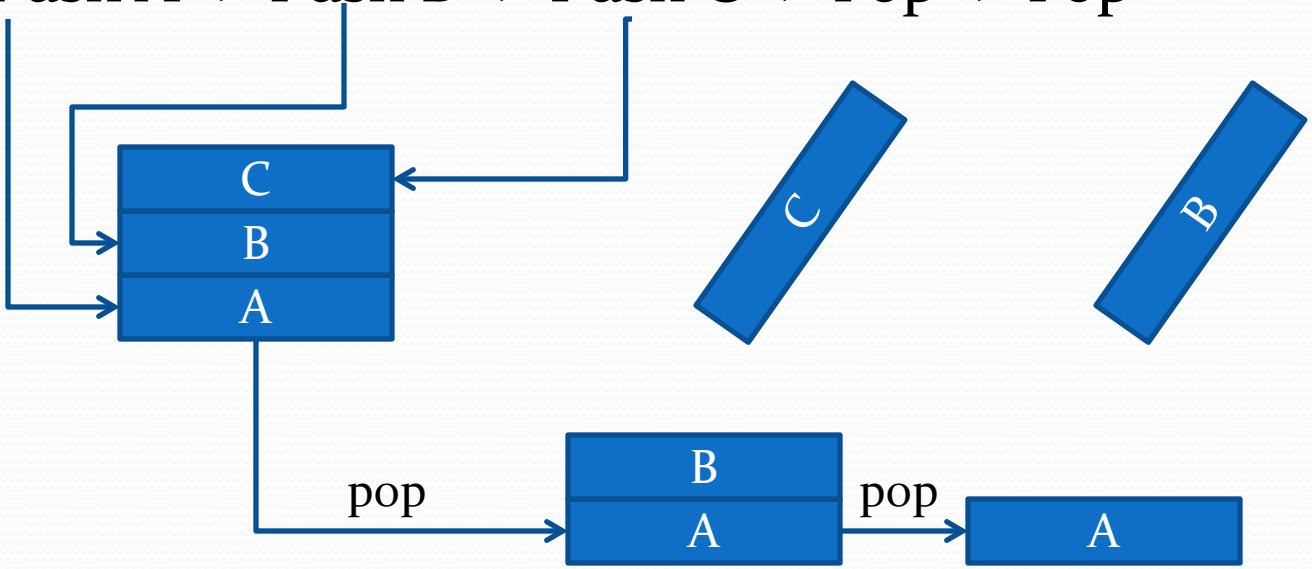
- Forma posfixa:
  - Prós:
    - Computacionalmente mais eficiente.
    - Disposição dos elementos em Pilha.
  - Contras:
    - Cérebro humano requer treinamento mais detalhado para interpretar.
  - Exemplo:
    - 5*10 + 85 * (48+9 + 6/2)
      - 5 10 * 6 2/ 48 9 + 85 * +
      - Ordem das operações: (5*10) , [(6/2) + (48+9)] * 85, soma.

Pilhas

- Pilhas
  - LIFO – *Last In - First Out*, o último elemento que entra é o primeiro que sai.
    - Operações básicas:
      - Empilha, *push;*
      - Desempilha, *pop*;
      - getTopo( );
      - getTamanho( );
      - isVazio( );

Pilhas

- Exemplo:
  - Push A -> Push B -> Push C -> Pop -> Pop

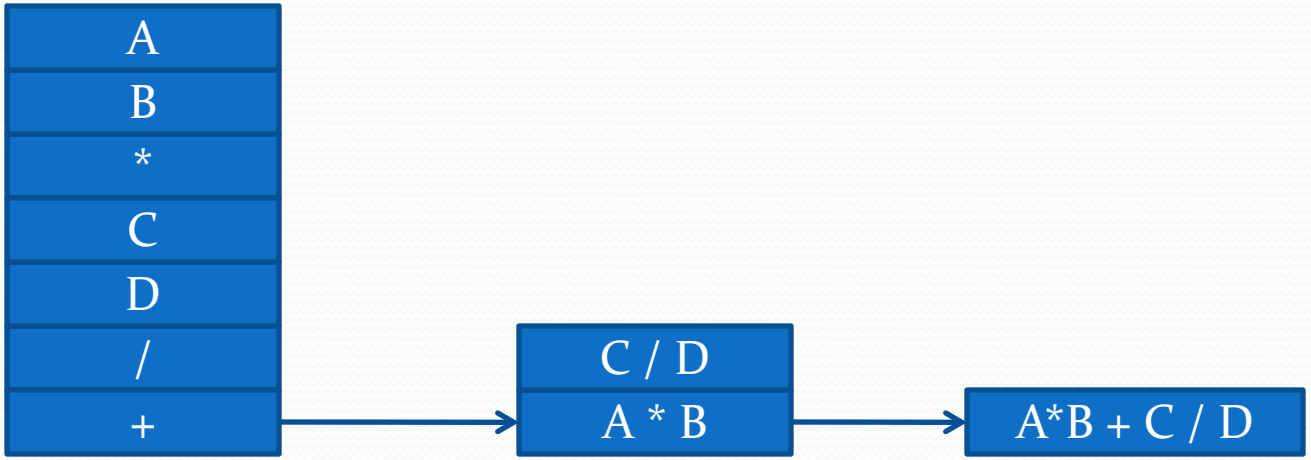


Pilhas e formas infixa e posfixa

- A expressão infixa A*B + C/D:
  - Infixa é fácil de visualizar.
  - Qual a ordem computacional?
    - 1º - A*B. (pegue A e B e multiplique)
    - 2º - C/D. (pegue C e D e divida)
    - 3º Soma dos resultados.
  - Forma posfixa:
    - A B * C D / +
    - Não existe operador matemático entre os fatores e parcelas da expressão.
    - Ao se colocar em uma pilha esta expressão, toda vez que ocorrer um *pop* e houver um operador matemático, deve se realizar a operação matemática com os elementos já desempilhados.

Pilhas e formas infixa e posfixa

- Exemplo:
  - Push + -> Push / -> Push D -> Push C -> Push * -> Push B-> Push A

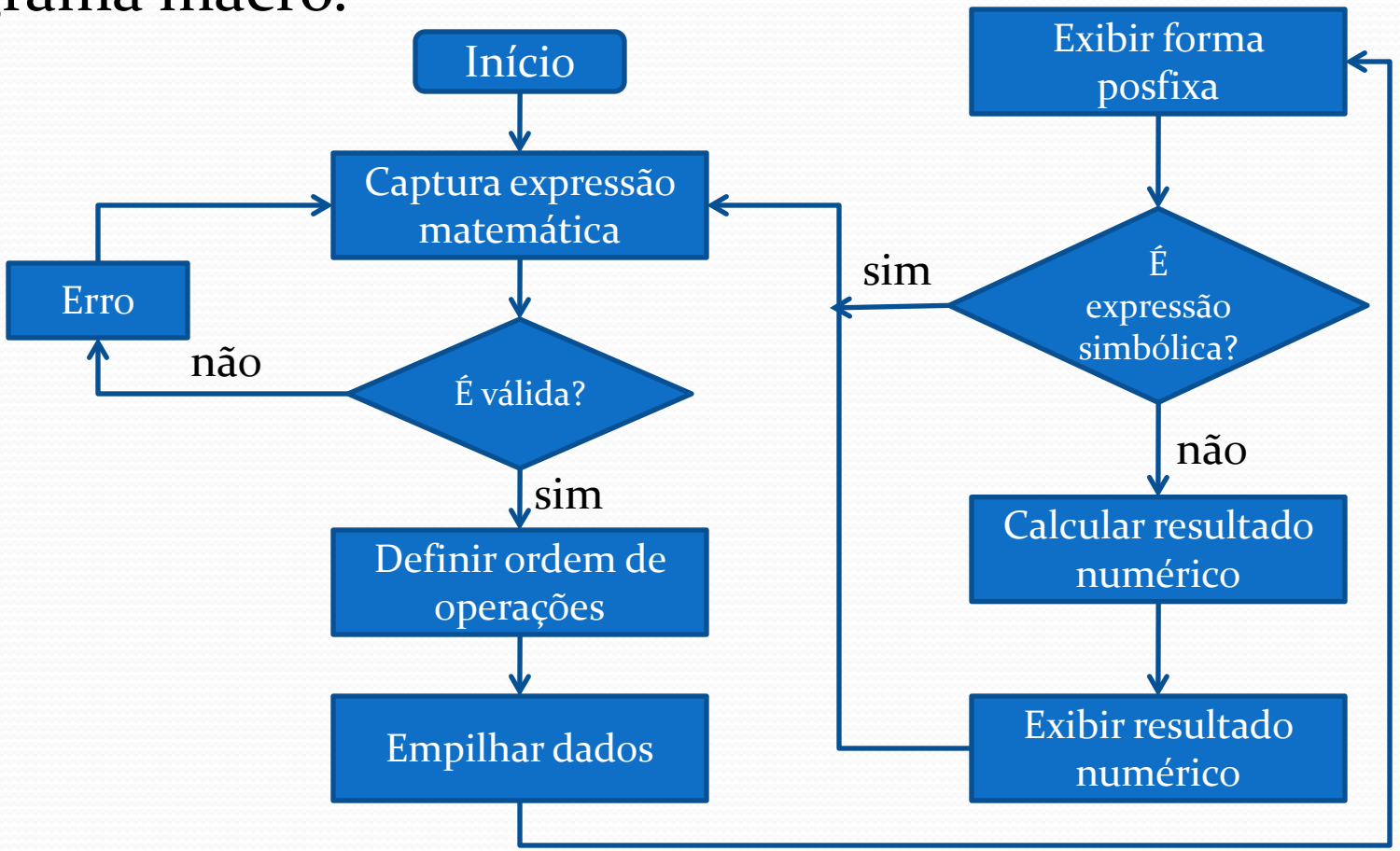


Implementação do Trabalho

- Implementação em JAVA, javac 1.6.0_37
- IDE utilizada: Netbeans 7.0.1
  - As seguintes classes foram criadas:
    - Elemento.java
    - Pilha.java
    - PilhaVaziaException.java
    - ExpressaoAritmetica.java
    - ExpressaoGUI.java
    - ProcessadorException.java

Implementação do Trabalho

- Diagrama macro.



Implementação do Trabalho

- No cálculo de complexidade, foi encontrado que:
  - O(n) = n, onde n é o número de elementos da pilha.
  - Não existem loops aninhados pelo código.

- Avaliação das expressões aritméticas numéricas, se dá dois-a-dois, após encontrado o primeiro operador matemático, por exemplo:
  - A B C *
    - Valor nº 1 = A
    - Valor nº 2 = B
      - Valor nº 1 = A*B
      - Valor nº 2 = C
        - Retorna Valor nº 1 * Valor nº 2

Conclusão

- A utilização de pilhas é recomendável para processar expressões aritméticas, realizando a conversão de sua forma infixa para posfixa.
- A forma posfixa é computacionalmente mais eficaz.
- A forma infixa é didadicamente melhor.