



Aula 15 - Pilhas (Aplicações clássicas)

Prof. Me. Claudiney R. Tinoco
profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Bacharelado em Ciência da Computação (BCC)
Bacharelado em Sistemas de Informação (BSI)

Algoritmos e Estruturas de Dados 1 (AED1)
GBC024 - GSI006

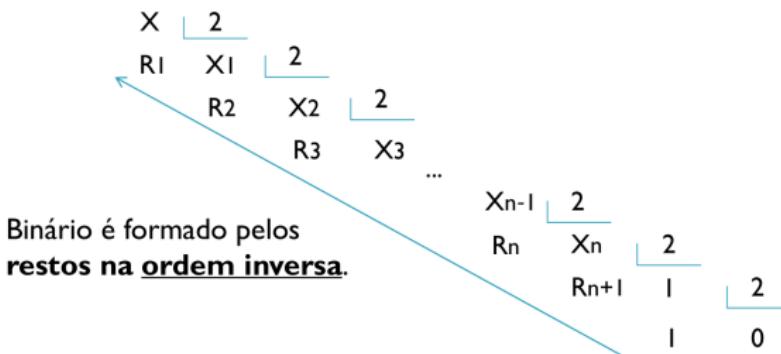


1^a Aplicação Clássica de Pilha

Conversão decimal para binário:

Seja X um número na base 10 (**decimal**), como obter sua representação na base 2 (**binário**)?

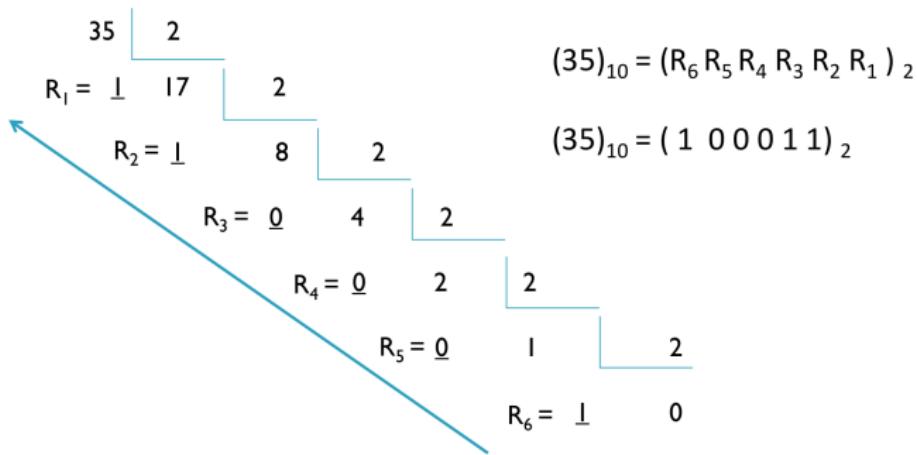
R: Através da aplicação de sucessivas divisões por 2.





Conversão de Decimal para Binário

► Exemplo: $(35)_{10} = (?)_2$



Conversão de Decimal para Binário

- ▶ Como a ordem dos restos posicionados é inversa à ordem em que os mesmos são obtidos, a estrutura Pilha pode ser utilizado para resolver
- ▶ Operação *converte_dec_bin()* : recebe um número decimal e imprime o valor correspondente na base binária



2^a Aplicação Clássica de Pilha

▶ Validação de agrupamento de escopos:

- Definição de escopo:
 - Expressões matemáticas: parênteses, chaves ou colchetes.
 - Programas: *begin-end* (Pascal) ou chaves (C e Java).
- Validação de escopo em expressões matemáticas:
 - Uma expressão matemática que utiliza agrupamento de escopos será valida se:
 - a) N° de escopos abertos = N° de escopos fechados.
 - b) Qualquer fechamento deve ser precedido pelo respectivo escopo de abertura.



Validação de Escopo

► Exemplos expressões válidas:

$6 - (A + (B-2))/(C/(D+5))$

$(A+(B+(C-(D^2))))$

► Exemplos expressões inválidas:

$(D-E))$

Não obedece **a** e **b**

$) A/D (+E$

Não obedece **b**

$(A+B^*(C-D)$

Não obedece **a**



Validação de Escopo

▶ Conceitos e definições:

- Profundidade do agrupamento: nº de escopos abertos que não foram fechados.
 - Diferença de escopo: nº escopos abertos subtraído do nº de escopos fechados.
 - Ambos podem ser calculados em qualquer ponto da expressão (**da esquerda para direita**).
-
- ▶ Se um **ÚNICO delimitador** for usado, pode-se desenvolver um **algoritmo baseado nestes conceitos** para checar validade.
 - **Forma + simples**: uso de uma variável contadora.



Validação de Escopo

- ▶ Uma expressão matemática é válida se:
 - a) No final da expressão, a diferença de escopo for zero.
 - b) A qualquer ponto da expressão, a diferença de escopo for positiva.
- ▶ Aplicação nos exemplos anteriores:

a) $6 - (A + (B - 2)) / (C / (D + 5))$

↓↓↓↓ ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
0 0 1 1 1 2 2 2 2 1 0 0 1 1 1 2 2 2 2 1 0 **Válida**



Validação de Escopo

b) $(A + (B + (C - (D * 2))))$

$\downarrow \downarrow \downarrow$
1 1 1 2 2 2 3 3 3 4 4 4 4 32 1 0

Válida

c) $(D - E))$

$\downarrow \downarrow \downarrow \downarrow$
1 1 1 1 0 -1

Inválida

d)) A / D (+ E

\downarrow
-1

Inválida

e) $(A + B * (C - D))$

$\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$
1 1 1 1 1 2 2 2 2 1

Inválida

Validação de Escopo

- ▶ Para 2 ou mais tipos de delimitadores essa estratégia de validação NÃO funciona.
 - Ex: (,), { , }, [e].
- É necessário que cada escopo aberto por um delimitador, seja fechado por um delimitador do mesmo tipo.
 - Ex: (por); { por }; [por].
 - Solução: um contador de diferença para cada tipo de delimitador.
- Deve-se garantir que a seqüência de fechamento dos delimitadores seja inversa a seqüência de abertura.
 - Contra-exemplo: {[A/2]} é uma expressão inválida.



Validação de Escopo

- ▶ Uma Pilha pode ser usada para **validar expressões com vários delimitadores**:
 - Expressão é lida da esquerda para direita (1 caractere por vez)
 - Cada iniciador de escopo é empilhado.
 - Para cada finalizador de escopo encontrado, deve-se **desempilhar o iniciador no topo da pilha para comparação**:
 - SE tipo iniciador = tipo finalizador, OK.
 - Caso contrário, **expressão INVÁLIDA**.
(pilha vazia OU tipo iniciador ≠ tipo finalizador)
- ▶ Ao final da expressão, a **pilha deve estar vazia**.



Validação de Escopo

Exemplo:

$$2 + \{ [F * 4] + [(A - B) / (C - 2)] - 3 \} + 4$$



Empilha



$$2 + \{ [$$



Empilha



$$2 + \{ [F * 4]$$



Desempilha



$$2 + \{ [F * 4] + [$$



Empilha





Validação de Escopo

► Exemplo (continuação):

$2 + \{ [F * 4] + [($

↑
Empilha

$T \rightarrow$ (

[

{

$2 + \{ [F * 4] + [(A - B)$

↑↑↑↑

Desempilha

$T \rightarrow$ [

{

$2 + \{ [F * 4] + [(A - B) / ($

↑↑

Empilha

$T \rightarrow$ (

[

{

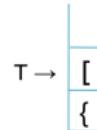


Validação de Escopo

► Exemplo (continuação):

 $2 + \{ [F * 4] + [(A - B) / (C - 2)]$ 

Desempilha

 $2 + \{ [F * 4] + [(A - B) / (C - 2)]$ 

Desempilha

 $2 + \{ [F * 4] + [(A - B) / (C - 2)] - 3 \}$ 

Desempilha

 $2 + \{ [F * 4] + [(A - B) / (C - 2)] - 3 \} + 4$ 

Fim



Pilha é vazia, então
expressão é válida

Validação de Escopo

- ▶ Operação *valida_escopo()* : verifica se uma expressão matemática é válida do ponto de vista de seus delimitadores de escopo
 - Entrada: a string que representa a expressão
 - Saída: “1” se a expressão é válida e “0” caso contrário

Validação de Escopo

- ▶ Operação *valida_escopo()* : verifica se uma expressão matemática é válida do ponto de vista de seus delimitadores de escopo
 - Entrada: a string que representa a expressão
 - Precisa passar o tamanho da string?
 - Saída: “1” se a expressão é válida e “0” caso contrário



Validação de Escopo

- ▶ Operação *valida_escopo()* : verifica se uma expressão matemática é válida do ponto de vista de seus delimitadores de escopo
 - Entrada: a string que representa a expressão
 - Precisa passar o tamanho da string? NÃO (usar o '\0')
 - Saída: “1” se a expressão é válida e “0” caso contrário



Formas de Representação de Expressões Matemáticas

► \exists 3 tipos de notação para expressões:

- **Infixa:** operador entre os operandos.
 - Ex: A+B
- **Pré-fixa:** operador precede os operandos.
 - Ex: +AB
- **Pós-fixa:** operador após os operandos.
 - Ex: AB+



3^a Aplicação Clássica de Pilha

► Conversão de notação infixa para pós-fixa:

- Processo manual:

Ex: $A \wedge B * C - D + E / F / (G-H)$

1. Colocar parênteses representando as precedências

$$(((A \wedge B) * C) - D) + ((E / F) / (G - H))$$

2. Fazer a conversão dos parênteses + internos (4º nível)

$$(((A B \wedge) * C) - D) + ((E F /) / (G H -))$$

3. Fazer a conversão dos parênteses do próximo nível (3º nível)

$$(((A B \wedge) C *) - D) + ((E F /) (G H -) /)$$



Conversão da Notação Infixa para Pós-Fixa

▶ Processo manual (continuação):

4. Fazer a conversão dos parênteses do 2º nível

$$(((\mathbf{AB} \wedge \mathbf{C}^*) \mathbf{D}-) + ((\mathbf{E} \mathbf{F}) / (\mathbf{G} \mathbf{H}-)) /)$$

5. Fazer a conversão dos parênteses do 1º nível

$$(((\mathbf{AB} \wedge \mathbf{C}^*) \mathbf{D}-) ((\mathbf{E} \mathbf{F}) / (\mathbf{G} \mathbf{H}-)) /) +$$

6. Tirar os parênteses

$$\mathbf{AB} \wedge \mathbf{C}^* \mathbf{D}- \mathbf{EF} / \mathbf{GH}- / +$$

Conversão da Notação Infixa para Pré-Fixa

- ▶ Conversão manual de **infixa para pré-fixa** é feita de forma **SIMILAR**:

Exemplo:

$$A \wedge B^* C - D + E / F / (G - H)$$

1. Colocar parênteses indicando precedências.

$$(((A \wedge B)^* C) - D) + ((E / F) / (G - H)))$$

2. Fazer a conversão nível a nível.

$$(+(-(*(\wedge AB)C)D)(//(/EF)(-GH)))$$

3. Tirar os parênteses.

$$+ - * \wedge A B C D // E F - G H$$



Conversão da Notação Infixa para Pós-Fixa e Pré-Fixa

- Notações pré-fixa e pós-fixa **NÃO SÃO imagens refletivas**

EXEMPLOS		
INFIXA	PRÉ-FIXA	PÓS-FIXA
$A^*(B-C)$	$*A-BC$	$ABC-*$
$(A-B)/(C+D)$	$/-AB+CD$	$AB-CD+/-$
$A+B/(C*D\wedge E)$	$+A/B*C\wedge DE$	$ABCDE\wedge*/+$

Conversão da notação infixa para pós-fixa

▶ Notação infixa pré-manipulada:

- Colocação **manual** de parênteses.
 - Expressão infixa deve possuir parênteses para determinar a precedência de todas as operações
 - A conversão para a notação pós-fixa pode ser feita a partir da conversão dos parênteses mais internos até chegar no parêntese mais externo
-
- ▶ Um algoritmo que usa **pilha** pode ser elaborado para esse problema

Conversão da notação infixa para pós-fixa

- ▶ **Passo 1:** Colocar os parênteses manualmente conforme as precedências das operações e inicializar a pilha
- ▶ **Passo 2:** Varrer a expressão da esquerda para direita e para cada símbolo **s** lido:
 - SE '(' ENTÃO **ignorar**
 - SE **operando** ENTÃO **imprimir ou copiar** para uma estrutura de saída
 - SE **operador** ENTÃO colocar na pilha (**empilhar**)
 - SE ')' ENTÃO **desempilhar** o operador no topo da pilha **e imprimir-lo** (ou copiá-lo para a saída)

Conversão da notação infixa para pós-fixa

- ▶ **Passo 3:** Ao final da expressão, se a mesma for válida, a pilha deve estar vazia



Conversão da notação infixa para pós-fixa

- ▶ **Passo 3:** Ao final da expressão, se a mesma for válida, a pilha deve estar vazia

Ex: A-B*C+D → ((A-(B*C))+D)

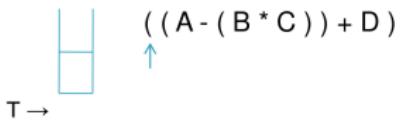


Conversão da notação infixa para pós-fixa

- ▶ **Passo 3:** Ao final da expressão, se a mesma for válida, a pilha deve estar vazia

Ex: A-B*C+D → ((A-(B*C))+D)

Início



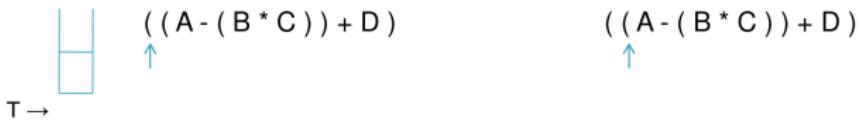


Conversão da notação infixada para pós-fixa

- Passo 3: Ao final da expressão, se a mesma for válida, a pilha deve estar vazia

Ex: $A-B^*C+D \rightarrow ((A-(B^*C))+D)$

Início

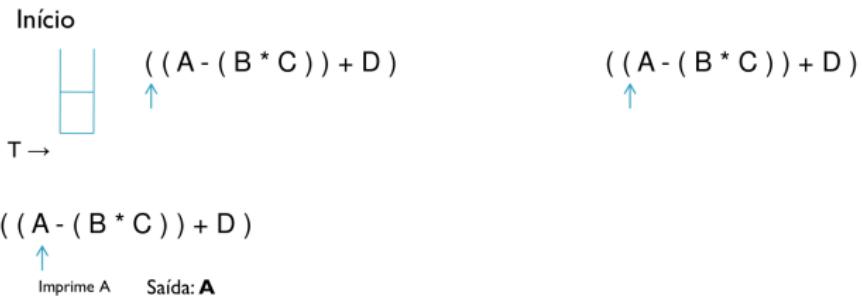




Conversão da notação infixa para pós-fixa

- ▶ **Passo 3:** Ao final da expressão, se a mesma for válida, a pilha deve estar vazia

Ex: $A - B^*C + D \rightarrow ((A - (B^*C)) + D)$

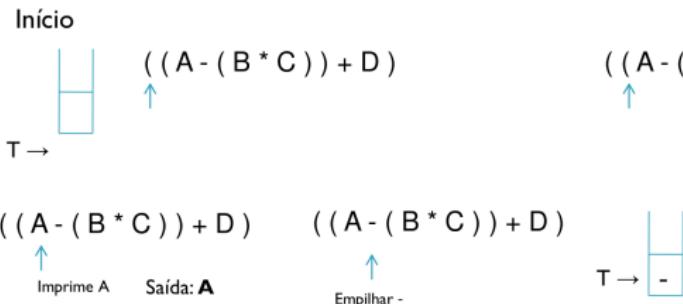




Conversão da notação infixada para pós-fixa

- Passo 3: Ao final da expressão, se a mesma for válida, a pilha deve estar vazia

Ex: $A-B^*C+D \rightarrow ((A-(B^*C))+D)$

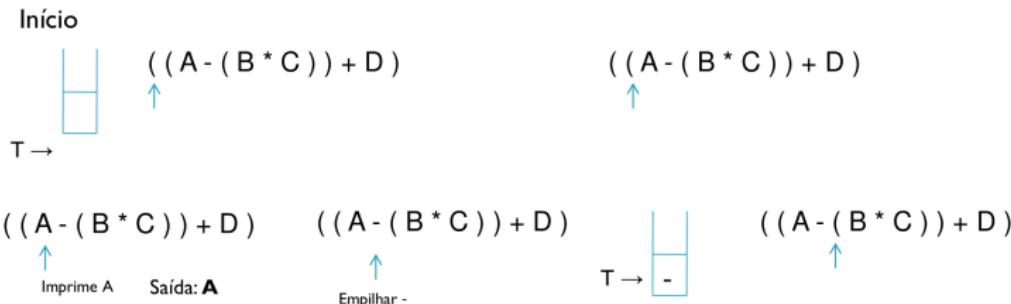




Conversão da notação infixa para pós-fixa

- ▶ **Passo 3:** Ao final da expressão, se a mesma for válida, a pilha deve estar vazia

Ex: $A - B^*C + D \rightarrow ((A - (B^*C)) + D)$





Conversão da notação infixa para pós-fixa

$$((A - (B * C)) + D)$$


Imprime B

Saída:**A****B**



Conversão da notação infixa para pós-fixa

$((A - (B * C)) + D)$ $((A - (B * C)) + D)$ $T \rightarrow$

*
-

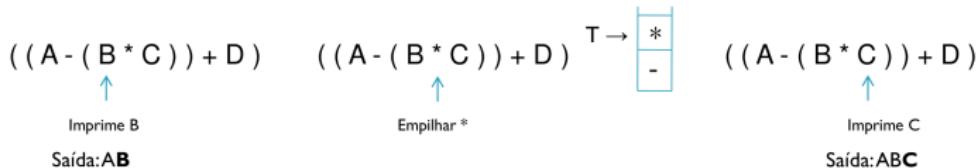
↑ ↑

Imprime B Empilhar *

Saída: **AB**

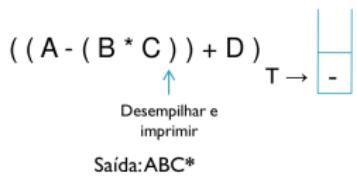
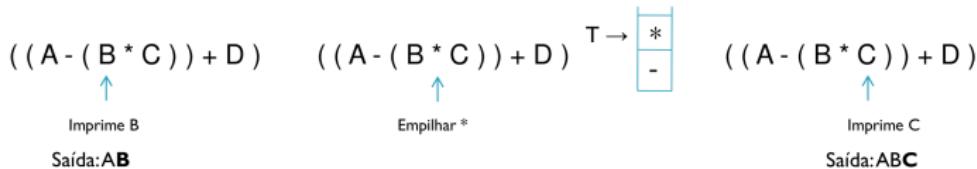


Conversão da notação infixada para pós-fixa



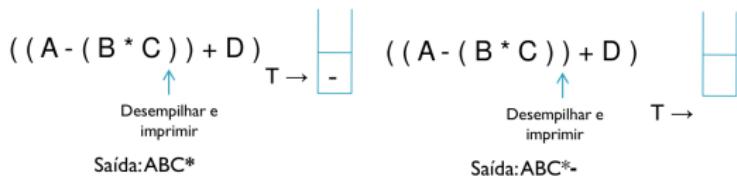
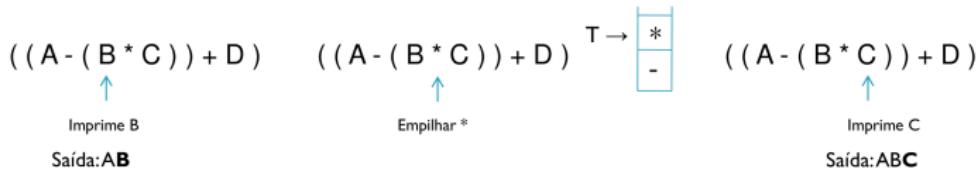


Conversão da notação infixada para pós-fixa



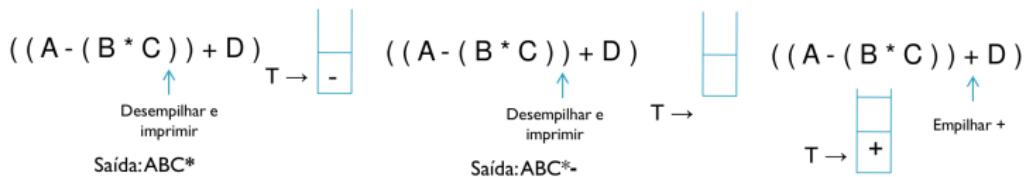
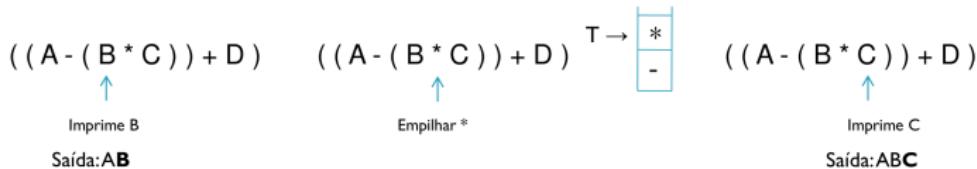


Conversão da notação infixada para pós-fixa



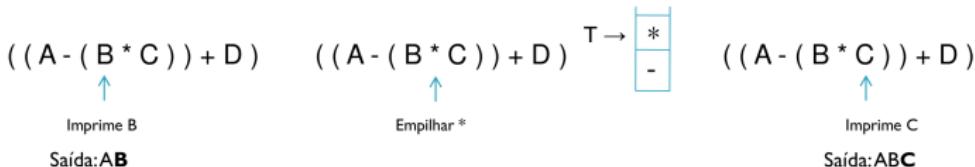


Conversão da notação infixada para pós-fixa



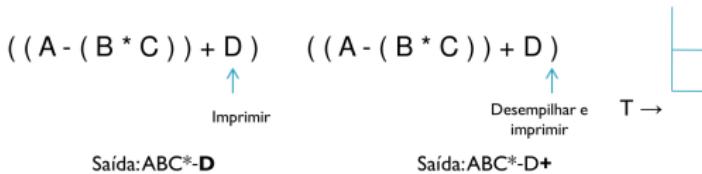
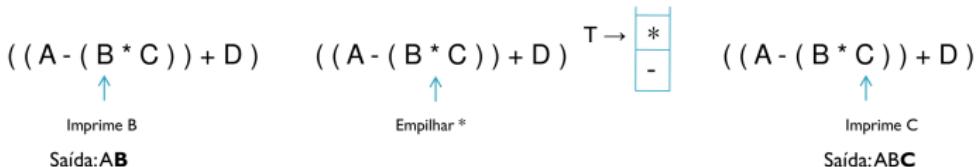


Conversão da notação infixada para pós-fixa



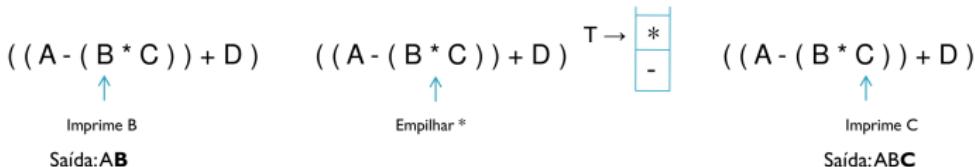


Conversão da notação infixada para pós-fixa





Conversão da notação infixa para pós-fixa





Conversão da notação infixa para pós-fixa

- Como a **pilha está vazia**, a **expressão é válida** e a conversão está na saída

Saída: ABC*-D+



Conversão da notação infixa para pós-fixa

- Como a **pilha está vazia**, a **expressão é válida** e a conversão está na saída

Saída: ABC*-D+

► **Desvantagem:**

- Depende da correta **colocação manual dos parênteses** na expressão infixa.



Conversão da notação infixa para pós-fixa

▶ Notação infixa qualquer:

- 1º Caso: sem parênteses
- 2º Caso: com parênteses eventuais
 - Usa parênteses somente quando necessário



Conversão da notação infixa para pós-fixa

► 1º caso (sem parênteses):

- **Passo 1:** Inicializar a pilha
- **Passo 2:** Varrer a expressão da esquerda para a direita e para cada símbolo lido s :
 - SE s é operando: imprimir s
 - SE s é um operador:
 - Desempilha e imprime operadores com prioridade maior ou igual a s
 - Empilha s
- **Passo 3:** Ao final da expressão, desempilha e imprime TODOS os operadores existentes na pilha



Conversão da notação infixa para pós-fixa

Ex: A + B / C * D ^ E

s: A Ação: Imprime "A"

s: + Ação:

s: B Ação: Imprime "B"

s: / Ação:

s: C Ação: Imprime "C"

s: * Ação: a) Desempilhar todos
 de prioridade > ou = Imprime "/"

b) Empilha *





Conversão da notação infixada para pós-fixada

Ex: A + B / C * D ^ E

s: D Ação: Imprime "D"

S: ^ Ação: ^



s: E Ação: Imprime "E"

s: '\0' Ação: Desempilhar e imprimir **todos** os elementos da pilha

SAÍDA: ABC / DE ^*+

Conversão da notação infixa para pós-fixa

Ex2: $A ^ B * C - D + E / F / G$

- s: A Ação: Imprime "A"
- s: ^ Ação: Empilha "^"
- s: B Ação: Imprime "B"
- s: * Ação: Desempilha e imprime "^" e empilha "*"
 Pilha $\geq s$ ^
- s: C Ação: Imprime "C"
- s: - Ação: Desempilha e imprime "*" e empilha "-"
 Pilha $\geq s$ *
- s: D Ação: Imprime "D"
- s: + Ação: Desempilha e imprime "-" e empilha "+"
 Pilha $\geq s$ -



Conversão da notação infixa para pós-fixa

Ex2: A ^ B * C - D + E / F / G

s: E Ação: Imprime "E"

s: / Ação: Prioridade Empilha "/"

Pilha < s

s: F Ação: Imprime "F"

s: / Ação: Prioridade
 Pilha >= s Desempilha e imprime "/"

Prioridade

Pilha < s Empilha "/"

s: G Ação: Imprime "G"

s: final Ação: Desempilha e imprime "/"
 Desempilha e imprime "+"

SAÍDA: AB ^ C * D-E F / G /+



Conversão da notação infixa para pós-fixa

► 2º caso (com parênteses eventuais):

- O que acontece se a expressão infixa permite a utilização de parêntese para modificar a precedência dos operadores?
- EX: $(A+B)/(C*D) ^E$
- Mudança em relação ao caso anterior (sem parênteses) ocorre principalmente em relação ao passo 2.



Conversão da notação infixa para pós-fixa

► 2º caso (com parênteses eventuais):

- Passo 1: Inicializar a pilha
- Passo 2: Varrer a expressão da esquerda para a direita e para cada símbolo lido s :
 - SE s é operando: imprimir s
 - SE s é um operador:
 - Desempilha e imprime operadores com prioridade maior ou igual a s .
 - **MUDANÇA:** nesse caso, deve considerar a abertura de parêntese como o operador de menor prioridade.
 - Empilha s .



Conversão da notação infixa para pós-fixa

► 2º caso (com parênteses eventuais):

- Passo 2 (continuação):
 - SE s é um parêntese de abertura, empilhar s
 - SE s é um parêntese de fechamento:
 - Desempilha e imprime **operadores** enquanto o topo da pilha for diferente de parêntese de abertura.
 - O parêntese de abertura é desempilhado e ignorado (não impresso).
 - Passo 3: Ao final da expressão, **desempilha e imprime TODOS os operadores** existentes na pilha.



Conversão da notação infixa para pós-fixa

► Ex2: $(A + B) / (C * D) \wedge E$

- s: (Ação: Empilhar (
- s: A Ação: Imprimir "A"
- s: + Ação: Topo Pilha Prioridade < s Empilhar +
- s: B Ação: Imprimir "B"
- s:) Ação: Desempilhar e imprimir "+"
 Desempilhar e ignorar "("
- s: / Ação: Empilhar /
- s: (Ação: Empilhar (
- s: C Ação: Imprimir "C"



Conversão da notação infixa para pós-fixa

► Ex2: $(A + B) / (C * D) \wedge E$

s: * Ação: Pilha < s Empilhar *

s: D Imprimir "D"

s:) Ação: Desempilhar e imprimir "*"
 Desempilhar e ignorar "("

s: ^ Ação: Pilha < s Empilhar ^

s: E Ação: Imprimir "E"

s: final Ação: Desempilhar e imprimir "^"
 Desempilhar e imprimir "/"

SAÍDA: AB + CD * E ^ /

4^a Aplicação Clássica de Pilha

► Avaliação de expressões matemáticas pós-fixadas:

- Algoritmo usa uma pilha para avaliar expressões na forma pós-fixa:
- **Passo 1:** Criar uma instância de pilha vazia
- **Passo 2:** Varrer a expressão da esquerda para a direita e para cada símbolo **S** lido:
 - SE **S** for um **operando**: empilhar **S**
 - SE **S** for um **operador**:
 - Desempilhar os 2 últimos operandos: **OP**₂ (1º POP) e **OP**₁ (2º POP)
 - Efetuar a operação **OP**₁ **S** **OP**₂ e empilhar o resultado da operação.
- **Passo 3:** Ao final, o resultado estará no topo da pilha

Avaliação de Expressões Matemáticas na Forma Pós-Fixa

► Exemplo:

Avaliar a expressão AB-CD+*E/ sendo:

A	B	C	D	E
6	2	5	3	8

OBS: operandos são representados por letras cujo os valores são fornecidos pelo usuário e armazenados em uma tabela



Avaliação de Expressões Matemáticas na Forma Pós-Fixa

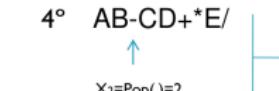
A	B	C	D	E
6	2	5	3	8



2º AB-CD+*E/
Push(A)
||
Push(6)



4º AB-CD+*E/
X2=Pop()=2
X1=Pop()=6



$$r = X_1 - X_2 = A - B = 6 - 2 = 4$$

Push(4)

5º AB-CD+*E/
Push(C)
||
Push(5)

6º AB-CD+*E/
Push(D)
||
Push(3)



7º AB-CD+*E/
X2=Pop()=3
X1=Pop()=5



$$\begin{aligned} r &= X_1 - X_2 \\ r &= 5 - 3 \\ r &= 2 \end{aligned}$$

Push(8)



Avaliação de Expressões Matemáticas na Forma Pós-Fixa

A	B	C	D	E
6	2	5	3	8

8º AB-CD+ * E/

$$\begin{array}{l} X_2 = \text{Pop}() = 8 \\ X_1 = \text{Pop}() = 4 \end{array}$$



$$r = x_1 * x_2 = 4 * 8 = 32$$

Push(32)



9º AB-CD+*E/

$$\begin{array}{l} \text{Push}(E) \\ \parallel \\ \text{Push}(8) \end{array}$$


10º AB-CD+*E /

$$\begin{array}{l} X_2 = \text{Pop}() = 8 \\ X_1 = \text{Pop}() = 32 \end{array}$$

$$r = 32 / 8 = 4$$

Push(4)



4

- Fim da expressão: `pop()` → Resultado = 4

AB-CD+*E / '\0'



Pilha vazia → Expressão válida



Referências

✓ Básica

- CELES, W., CERQUEIRA, R. e RANGEL, J. L. “Introdução a estruturas de dados”. Campus Elsevier, 2004.
- TENENBAUM, A. M., LANGSAM, Y. e AUGENSTEIN, M.J. “Estrutura de Dados Usando C”. Makron Books.

✓ Extra

- BACKES, André. “Programação Descomplicada Linguagem C”. Projeto de extensão que disponibiliza vídeo-aulas de C e Estruturas de Dados. Disponível em: <https://www.youtube.com/user/progdescomplicada>. Acessado em: 25/04/2022.

✓ Baseado nos materiais dos seguintes professores:

- Prof. André Backes (UFU)
- Prof. Bruno Travençolo (UFU)
- Prof. Luiz Gustavo de Almeida Martins (UFU)



Dúvidas?

Prof. Me. Claudiney R. Tinoco
profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU)