

Aplicações com Pilha

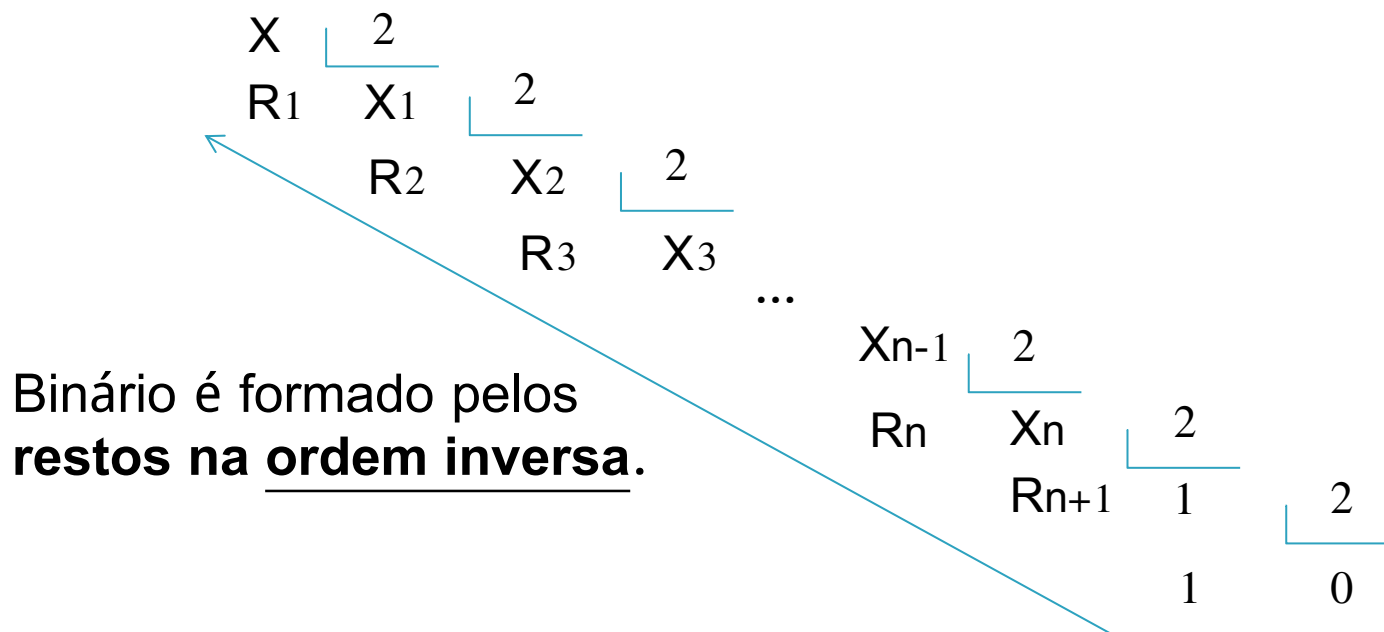
Algoritmos e Estrutura de Dados 1
Prof. Luiz Gustavo Almeida Martins

1ª Aplicação Clássica de Pilha

Conversão decimal para binário:

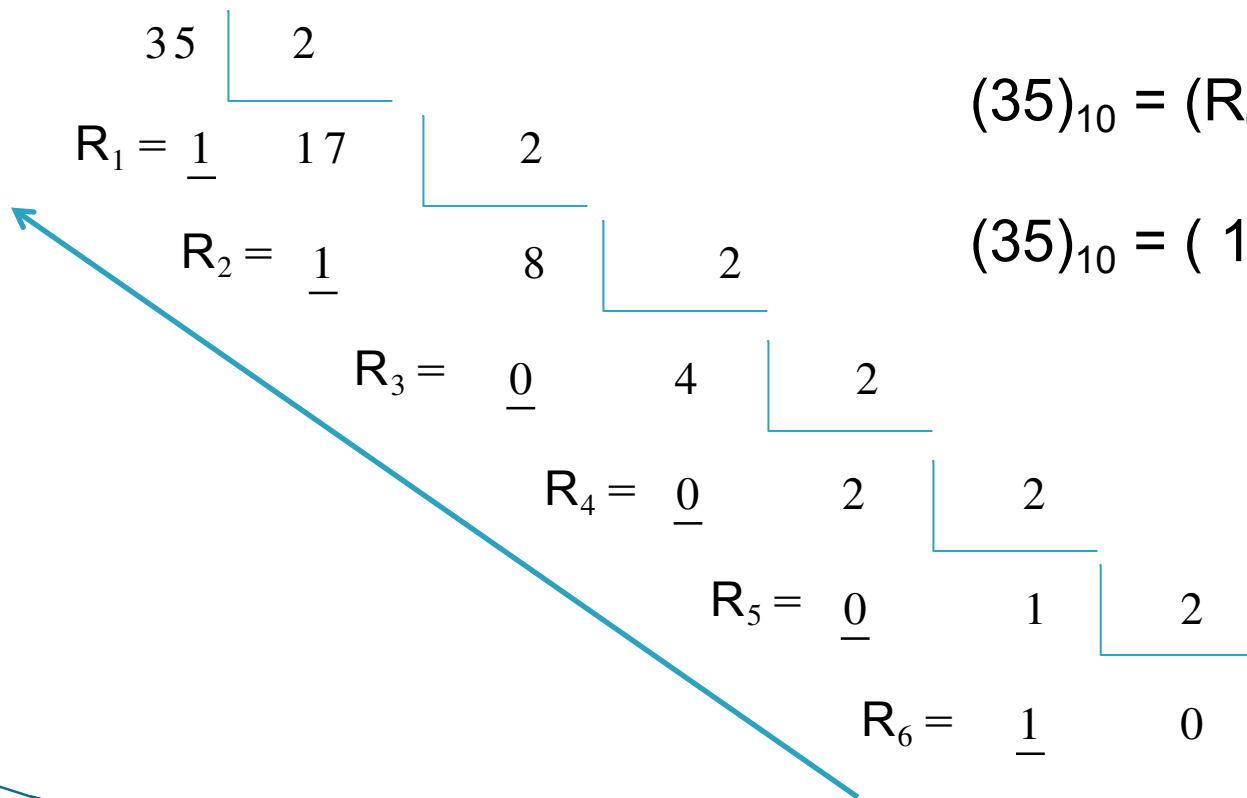
Seja ***X*** um número na base 10 (**decimal**), como obter sua representação na base 2 (**binário**)?

R: Através da aplicação de sucessivas divisões por 2.



Conversão de Decimal para Binário

Exemplo: $(35)_{10} = (?)_2$



$$(35)_{10} = (R_6 R_5 R_4 R_3 R_2 R_1)_2$$

$$(35)_{10} = (1\ 0\ 0\ 0\ 1\ 1)_2$$

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

Pseudo-Código

2ª 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 válida 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 <i>a</i> e <i>b</i>
$) A / D (+E$	Não obedece <i>b</i>
$(A + B * (C / D)$	Não obedece <i>a</i>

Validação de Escopo

Conceitos e definições:

- **Profundidade do agrupamento:** n° de escopos abertos que não foram fechados.
- **Diferença de escopo:** n° escopos abertos subtraído do n° 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 **pseudocódigo baseado na diferença de escopo** 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) $6 / (A + (B - 2)) / (C / (D + 5))$

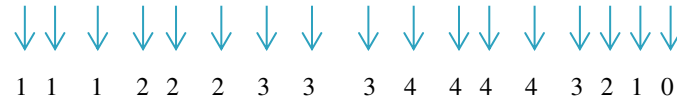
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

0 0 1 1 1 2 2 2 2 1 0 0 1 1 1 2 2 2 2 1 0

Válida

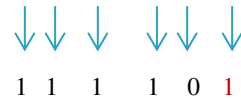
Validação de Escopo

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



Válida

c) (D - E))



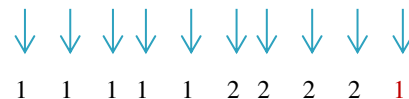
Inválida

d)) A / D (+ E



Inválida

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



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 (caracter a caracter).
- 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 \neq 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

T →

{

$2 + \{ [$

↑
Empilha

T →

[
{

$2 + \{ [F * 4]$

↑ ↑ ↑ ↑
Desempilha

T →

{

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

↑ ↑
Empilha

T →

[
{

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

$T \rightarrow$

(
[
{

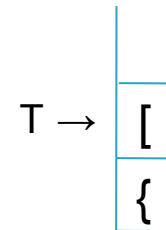
Validação de Escopo

Exemplo (continuação):

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



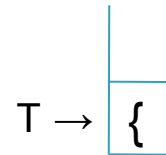
Desempilha



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



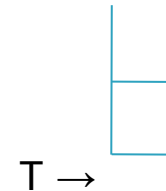
Desempilha



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



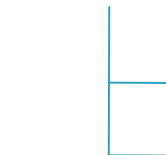
Desempilha



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



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.
Precisa do tamanho da string?
- **Saída:** “1” se a expressão é válida e “0” caso contrário.

Pseudo-Código

Formas de Representação de Expressões Matemáticas

∃ 3 tipos de representação de expressões:

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

3ª 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 (1º nível).

$(((((\textcolor{red}{A}\textcolor{blue}{B} \wedge) * C) - D) + ((\textcolor{red}{E}\textcolor{blue}{F}) / (\textcolor{red}{G}\textcolor{blue}{H} -))))$

3. Fazer a conversão dos parênteses do 2º nível.

$((((\textcolor{red}{A}\textcolor{blue}{B} \wedge) \textcolor{blue}{C} *) - D) + ((\textcolor{red}{E}\textcolor{blue}{F}) (\textcolor{red}{G}\textcolor{blue}{H} -) /))$

Conversão da Notação Infixa para Pós-Fixa

Processo manual (continuação):

4. Fazer a conversão dos parênteses do 3º nível.

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

5. Fazer a conversão dos parênteses do último nível

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

6. Tirar os parênteses.

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

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.

$$+ - * ^ ABCD // EF - GH$$

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
$A*(B-C)$	$*A-BC$	$ABC-*$
$(A-B)/(C+D)$	$/-AB+CD$	$AB-CD+ /$
$A+B(C*D^E)$	$+A/B*C^DE$	$ABCDE^*/+$

Conversão da notação infixa para pós-fixa

Notação infixa pré-manipulada:

- Colocação **manual** de parênteses.
Expressão infixa 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 0: Colocar os parênteses manualmente conforme as precedências das operações e inicializar a pilha

Passo 1: Varrer a expressão da esquerda para direita e para cada símbolo **s** lido:

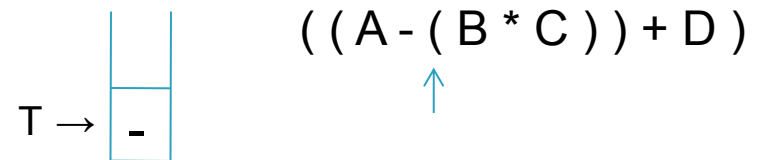
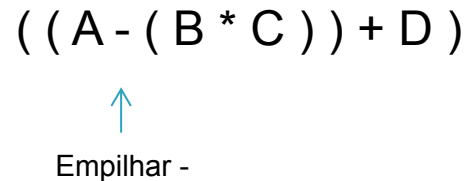
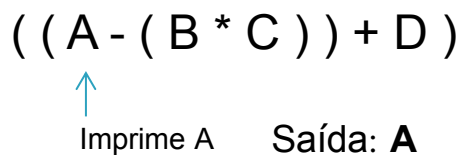
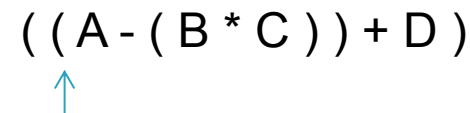
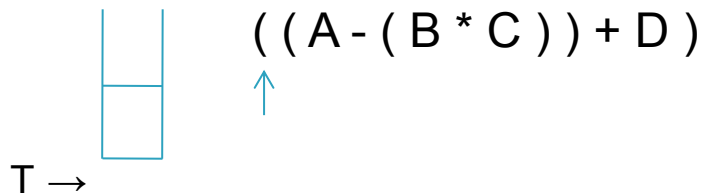
- SE '(' ENTÃO **ignorar**.
- SE **operando** ENTÃO **imprimir ou copiar** para uma estrutura de saída.
- SE **operador** ENTÃO colocar na pilha (**empilhar**).
- SE ')' ENTÃO **desempilhar** o operador no topo da pilha e **imprimir** o mesmo (ou copiar)

Conversão da notação infixa para pós-fixa

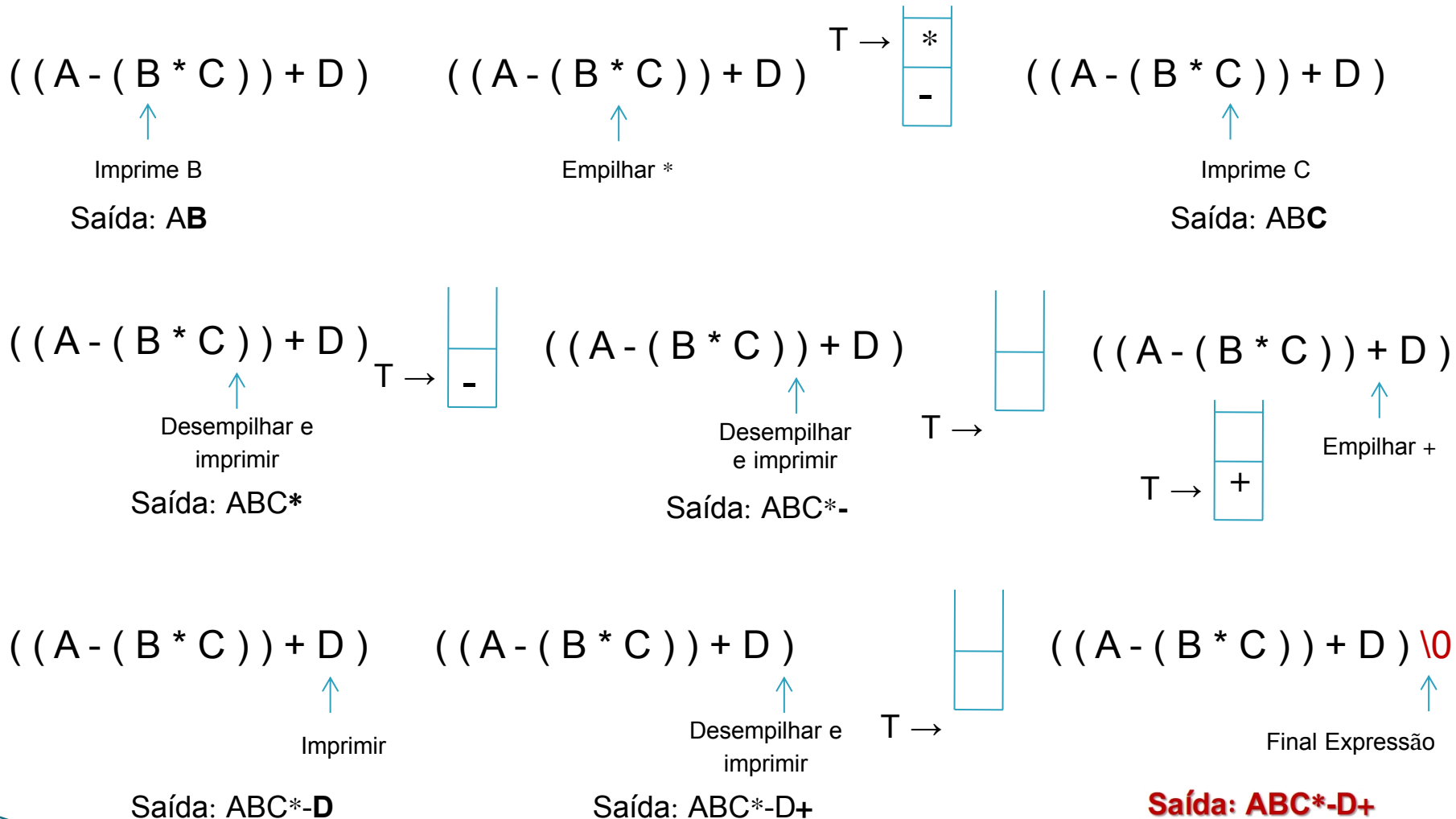
Passo 2: Ao final da expressão, se a mesma for válida, a pilha deve estar vazia.

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

Início



Conversão da notação infixa para pós-fixa



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+

- Pseudo-código para a conversão pós-fixa a partir de uma infixa com parênteses colocados previamente na expressão.

Pseudo-código

- 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

Expressões de entrada com parênteses somente quando necessário.

Conversão da notação infixa para pós-fixa

1º caso (sem parênteses):

- **Passo 0:** Inicializar a pilha
- **Passo 1:** 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 2:** 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 infixa para pós-fixa

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

s: A Ação: Imprime "A"

s: \wedge Ação: Empilha " \wedge "

s: B Ação: Imprime "B"

s: * Ação: Desempilha e imprime " \wedge " e empilha "*"

Pilha \geq s \wedge

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

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: $AB \wedge C * D - E F / G / +$

Conversão da notação infixa para pós-fixa

1º Caso - Conversão SEM Parênteses:

Pseudo-código

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 0:** Inicializar a pilha
- **Passo 1:** 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** (diferente de parênteses):
 - Desempilha e imprime **operadores** com prioridade **maior ou igual a s**.
MUDANÇA: nesse caso, deve **considerar o parêntese de abertura como um operador de menor prioridade**.
 - Empilha **s**.

Conversão da notação infixa para pós-fixa

2º caso (com parênteses eventuais):

- **Passo 1 (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 2:** Ao final da expressão, **desempilha e imprime TODOS os operadores** existentes na pilha.

Conversão da notação infixa para pós-fixa

Ex2: $(A + B) / (C * D) ^ E$

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: $(A + B) / (C * D) ^ E$

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 ^ /**

Conversão da notação infixa para pós-fixa

2º Caso - Conversão COM Parênteses EVENTUAIS:

Pseudo-código

4ª Aplicação Clássica de Pilha

Avaliação de expressões matemáticas na forma pós-fixa:

- É relativamente fácil construir uma função que utiliza uma pilha para avaliar expressões na forma pós-fixa:
- **Passo 1:** Iniciar 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: X_1 (2º POP) e X_2 (1º POP)
 - Efetuar a operação $X_1 \text{ S } X_2$ 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

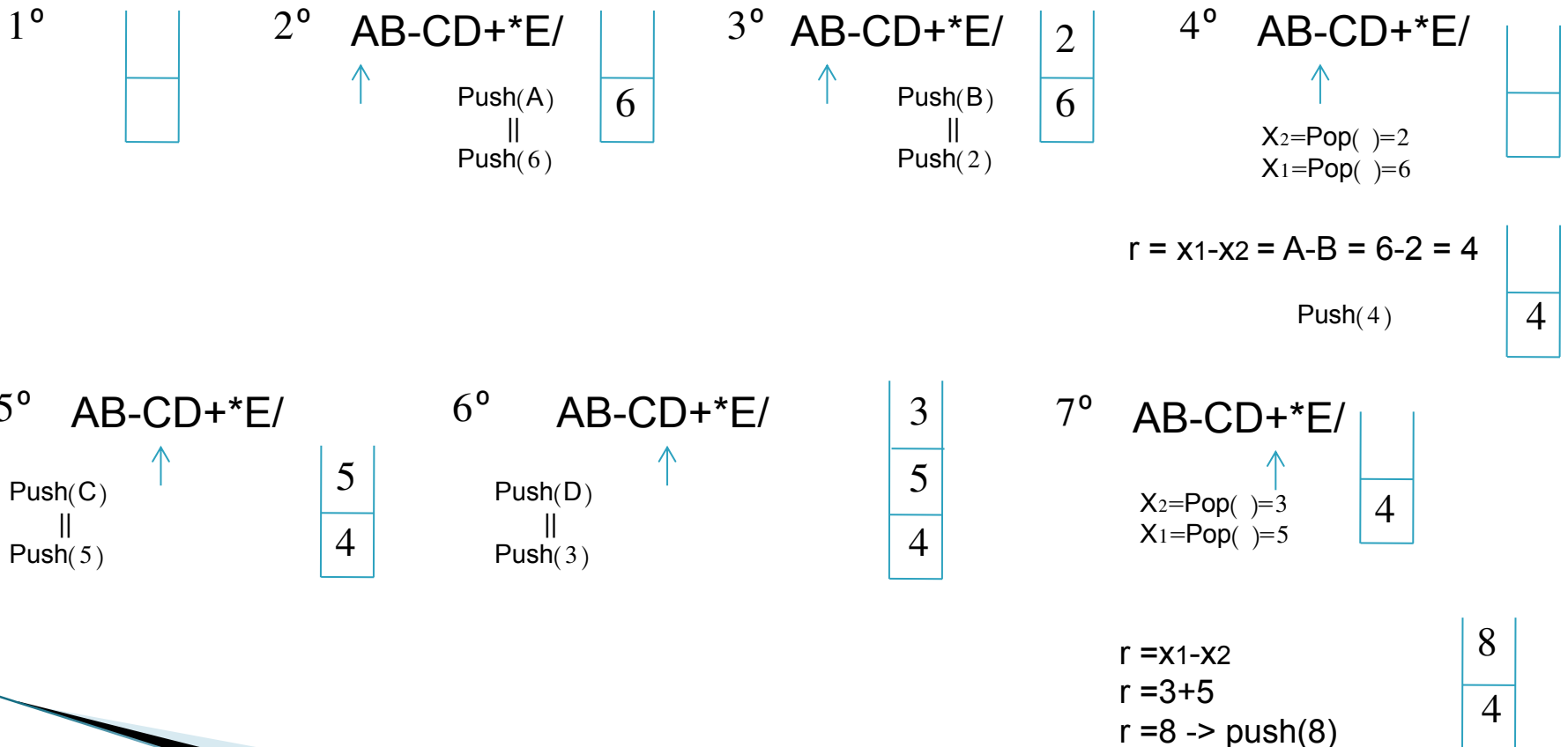
Exemplo:

Avaliar a expressão $AB-CD+^*E/$ sendo:

A	B	C	D	E
6	2	5	3	8

Avaliação de Expressões Matemáticas na Forma Pós-Fixa

A	B	C	D	E
6	2	5	3	8




Avaliação de Expressões Matemáticas na Forma Pós-Fixa

A	B	C	D	E
6	2	5	3	8

8º AB/ CD+ * E/


↑

$X_2 = \text{Pop}() = 8$
 $X_1 = \text{Pop}() = 4$



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

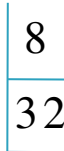
Push(32)



9º AB/ CD+*E/

↑


Push(E)
 \parallel
 Push(8)



10º AB/ CD+*E /


↑

$X_2 = \text{Pop}() = 8$
 $X_1 = \text{Pop}() = 32$



$r = 32 / 8 = 4$

Push(4)



Fim da expressão: $\text{pop}() \rightarrow \text{Resultado} = 4$

Avaliação de Expressões Matemáticas na Forma Pós-Fixa

O pseudo-código a seguir apresenta a solução para avaliar uma **expressão na notação pós-fixa** através de uma pilha.

Pseudo-código