Definições

- Uma expressão aritmética é composta de operandos, operadores e delimitadores.
- As operações são definidas por seus operadores. A ordem de avaliação das operações depende dos delimitadores e da prioridade dos operadores.
- As expressões aritméticas podem ser representas por 3 tipos de notação.
  - Infixa
  - Pré-fixa
  - Pós-fixa

Avaliação de expressões

- A avaliação de expressões na forma pós-fixa é bastante simplificada por diversas razões:
  - Não há necessidade de emprego de delimitadores;
  - A prioridade dos operadores deixa de ser relevante;
  - As expressões podem ser analisadas fazendo uma varredura do início para o final armazenando os operandos em uma pilha, aplicando os operadores sobre os elementos da pilha e empilhando os resultados das operações efetuadas.

Avaliação de expressões

- A avaliação de expressões aritméticas pode ser feita transformando as expressões infixas, familiares aos seres humanos, em expressões pós-fixas e, a seguir, avaliando o resultado das expressões pós-fixas obtidas.
- Para a transformação de expressões aritméticas infixas em pós-fixas e sua avaliação é conveniente definir diversas funções auxiliares.

Transformação de expressões aritméticas infixas em pós-fixas

- Esta transformação será mostrada da seguinte maneira:
  - Inicialmente será apresentado um algoritmo com bastante abstração
  - Será apresentado um exemplo
  - Finalmente serão apresentadas as declarações de funções utilizadas em um algoritmo com pouca abstração (para implementação)

Transformação de expressões infixas em pós-fixas

Conversão da forma infixa para pós-fixa

A entrada é um vetor infixVect de tokens (strings) de uma expressão infixa

- **Quando o token for um operando**
  - Inclui-lo no final do vetor postfixVect de tokens (strings) que armazena a expressão pós-fixa correspondente
- **Quando o token for um parênteses ou um operador**
  - Se o token x for "("
    - Empilhar o token x em stackVect de tokens (strings)
  - Se o token x for ")"
    - Repetidamente desempilhar um token y de stackVect e adiciona-lo ao vetor postfixVect até encontrar "(" no final da pilha stackVect. Desempilhar então "(" de stackVect.
    - Se stackVect esvaziar antes de encontrar um "(", a expressão de entrada não é válida.

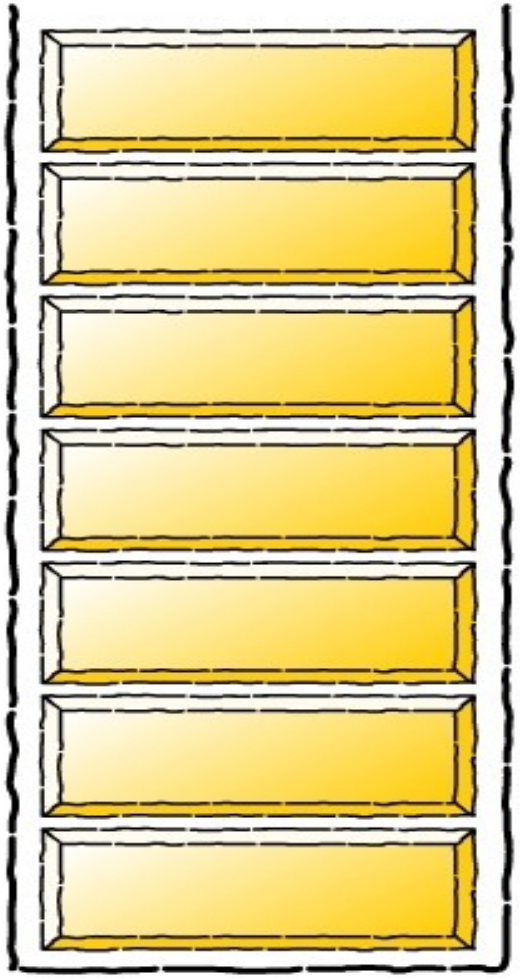
Conversão da forma infixa para pós-fixa

Se o token x for um operador

- **Passo 1**: Examinar o token y **no topo** de stackVect.
- **Passo 2**: **Se**(caso 1) stackVect **não** estiver vazia **e** (caso 2) y **não** for "(" **e** (caso 3) y for um operador de precedência **igual ou maior** do que a de x, desempilhar o token y de stackVect, adiciona-lo ao vetor postfixVect, e **ir para o Passo 1 novamente**.
- **Passo 3**: **Se** (caso 1) stackVect estiver vazia **ou** (caso 2) y for "(" **ou** (caso 3) y for um operador de **precedência menor** do que a de x , adicionar o token x ao vetor postfixVect.

Quando todos os tokens de infixVect tiverem sido processados, repetidamente desempilhar um token y de stackVect e adiciona-lo ao vetor postfixVect até esvaziar a pilha stackVect.

Conversão da forma infixa para pós-fixa



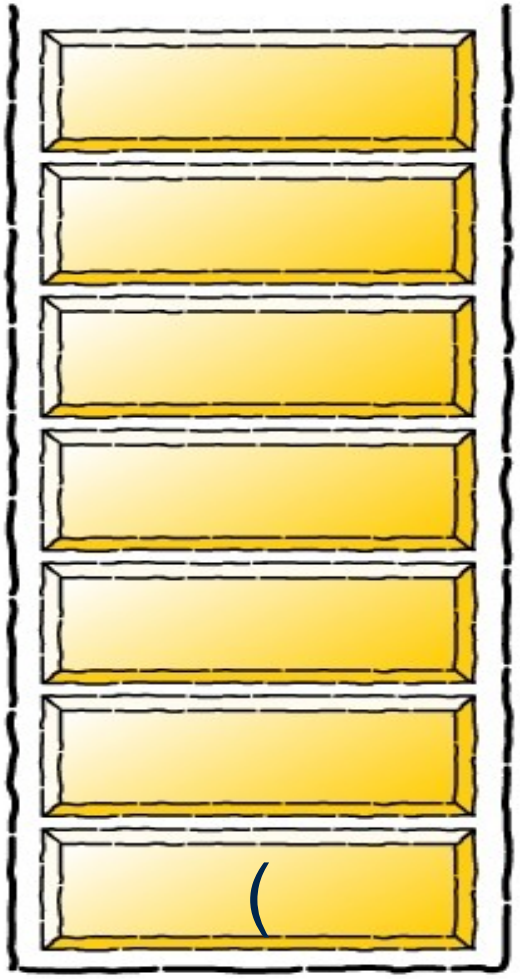
stackVect

infixVect

(a + b - c) * d – (e + f)

postfixVect

Conversão da forma infixa para pós-fixa



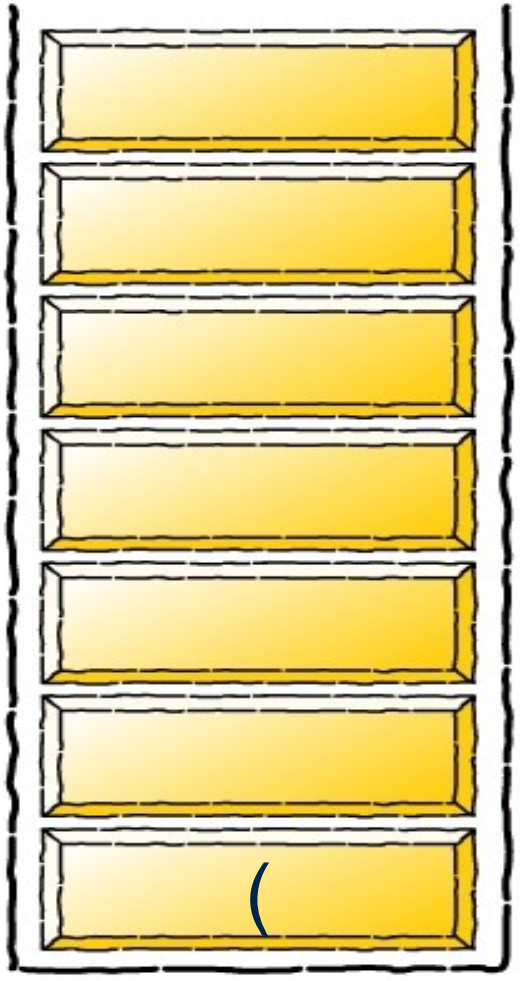
stackVect

infixVect

a + b - c) * d – (e + f)

postfixVect

Conversão da forma infixa para pós-fixa



stackVect

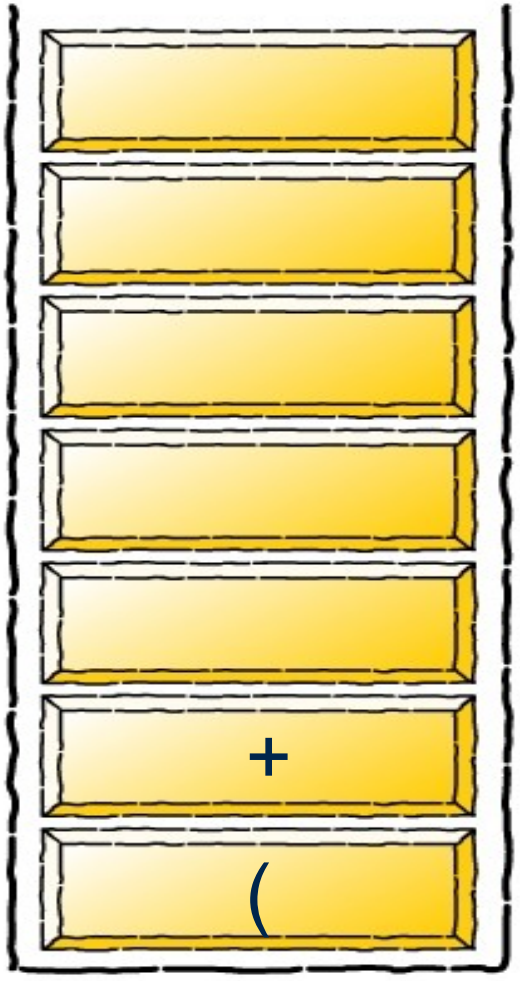
infixVect

+ b - c) * d – (e + f)

postfixVect

a

Conversão da forma infixa para pós-fixa



stackVect

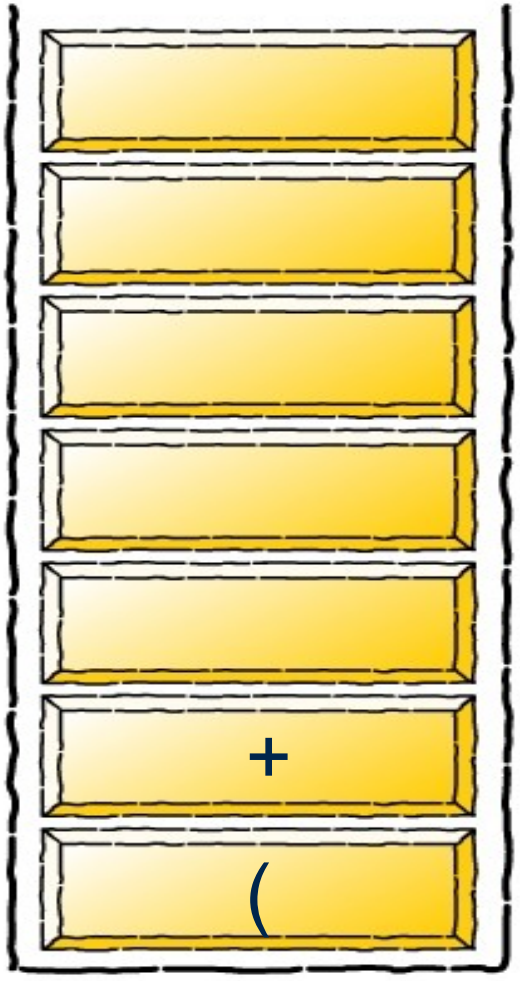
infixVect

b - c) * d – (e + f)

postfixVect

a

Conversão da forma infixa para pós-fixa



stackVect

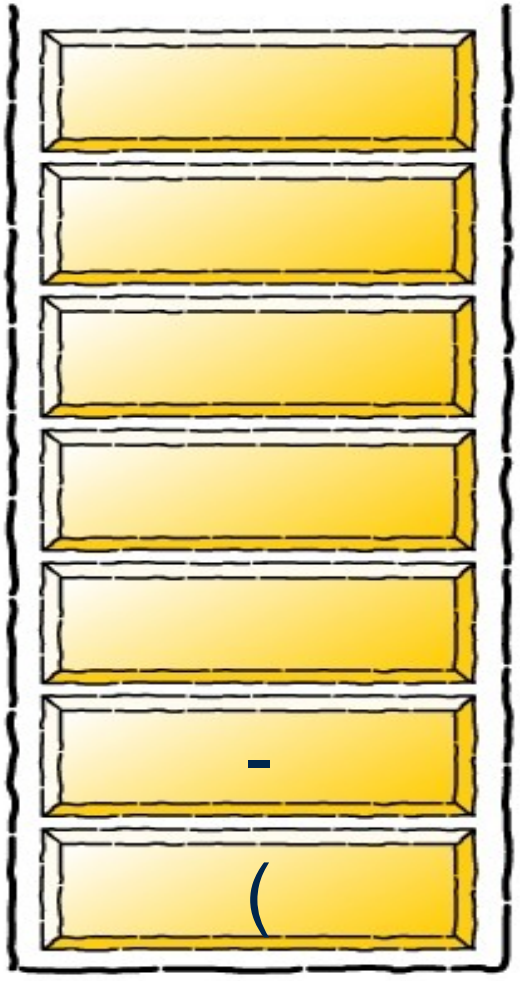
infixVect

- c) * d – (e + f)

postfixVect

a b

Conversão da forma infixa para pós-fixa



stackVect

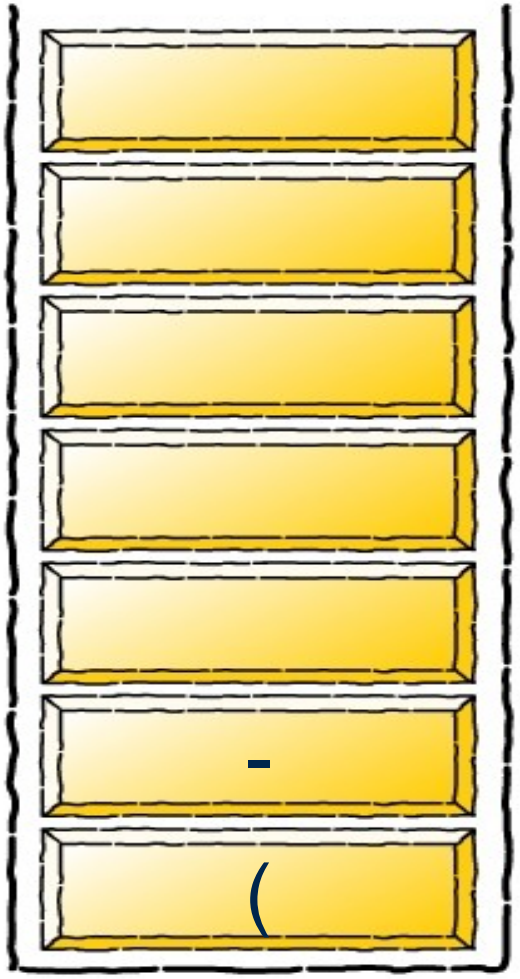
infixVect

c) * d – (e + f)

postfixVect

a b +

Conversão da forma infixa para pós-fixa



stackVect

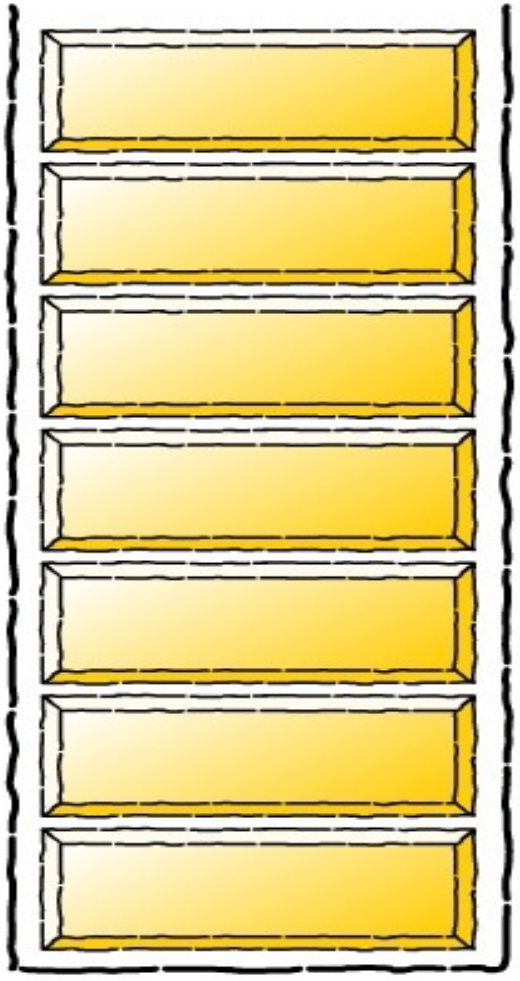
infixVect

) * d – (e + f)

postfixVect

a b + c

Conversão da forma infixa para pós-fixa



stackVect

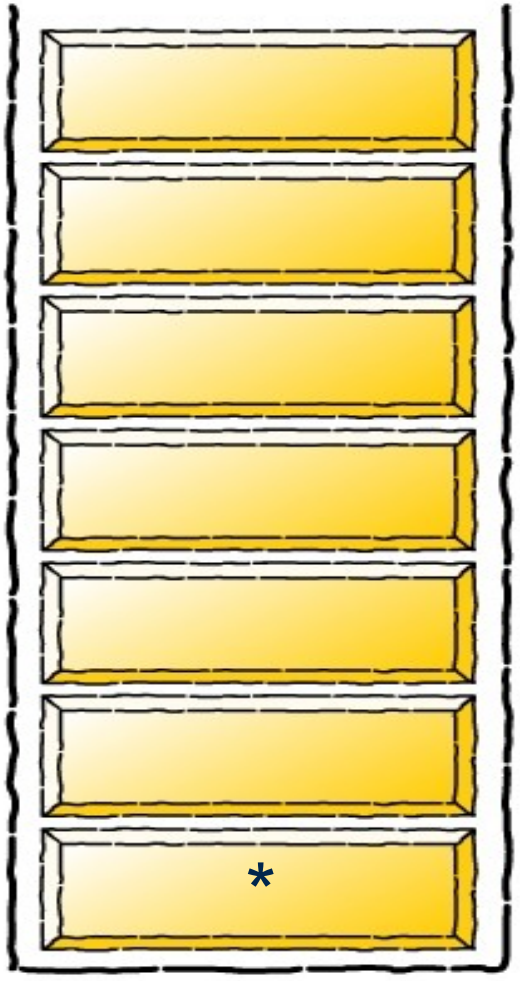
infixVect

* d – (e + f)

postfixVect

a b + c -

Conversão da forma infixa para pós-fixa



stackVect

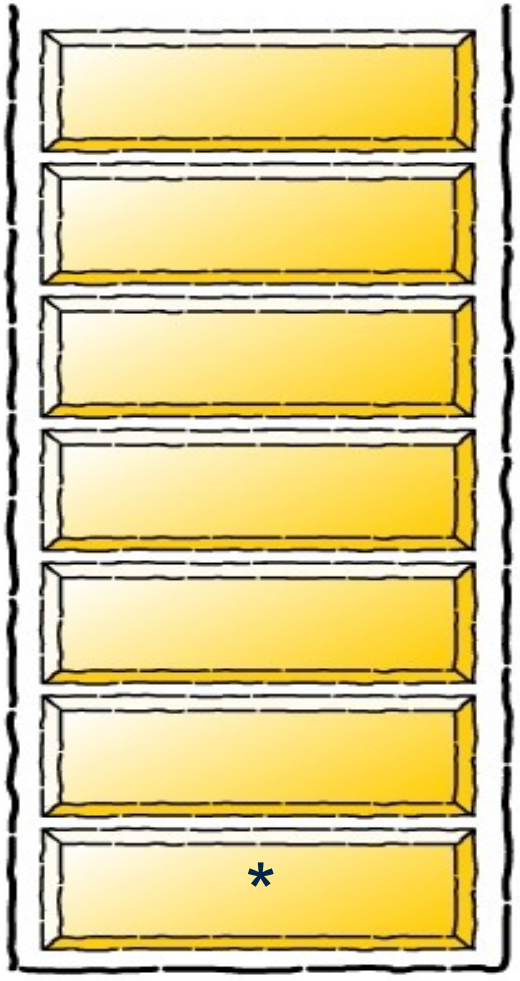
infixVect

d – (e + f)

postfixVect

a b + c -

Conversão da forma infixa para pós-fixa



stackVect

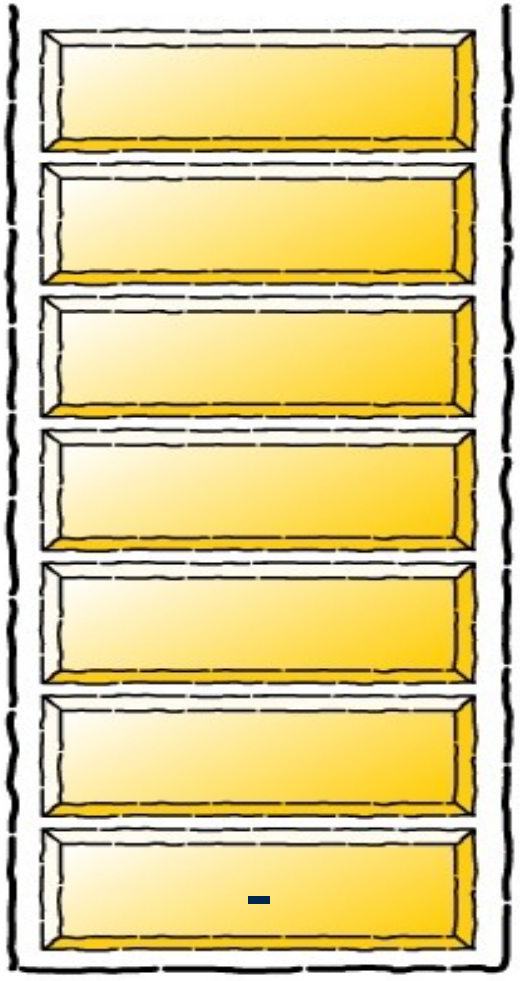
infixVect

– (e + f)

postfixVect

a b + c - d

Conversão da forma infixa para pós-fixa



stackVect

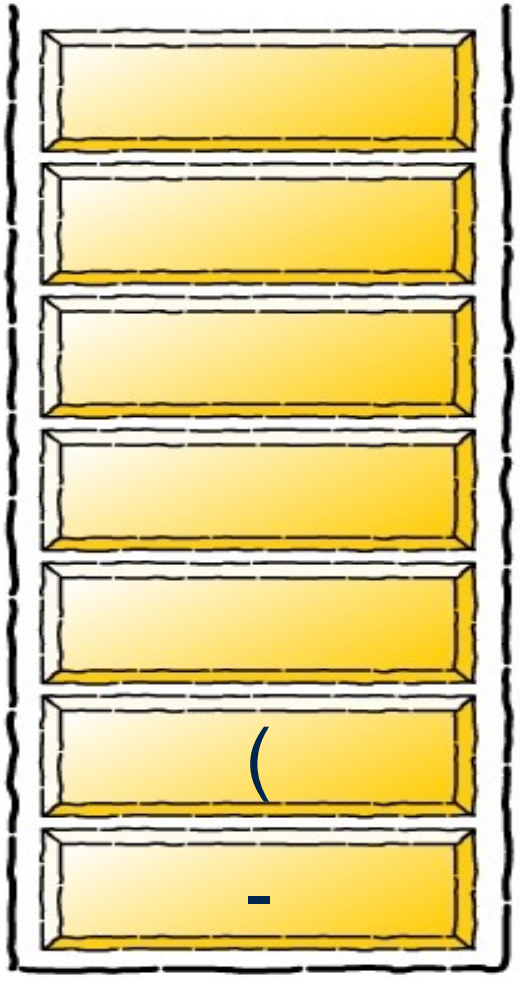
infixVect

(e + f)

postfixVect

a b + c – d *

Conversão da forma infixa para pós-fixa



stackVect

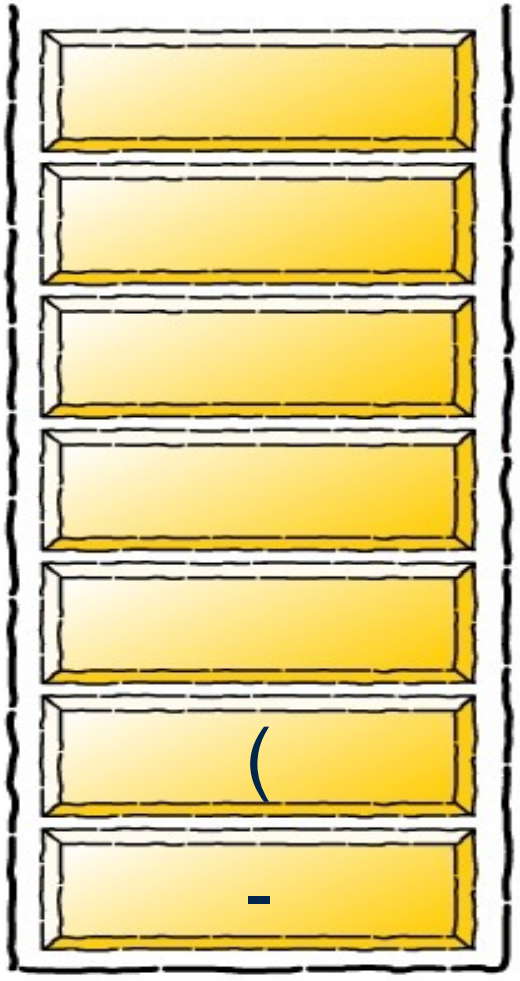
infixVect

e + f)

postfixVect

a b + c – d *

Conversão da forma infixa para pós-fixa



stackVect

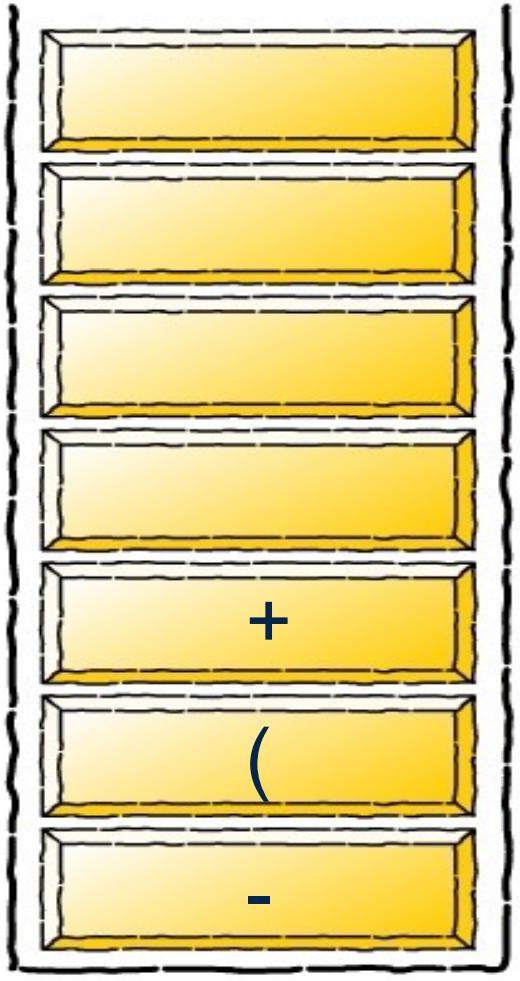
infixVect

+ f)

postfixVect

a b + c – d * e

Conversão da forma infixa para pós-fixa



stackVect

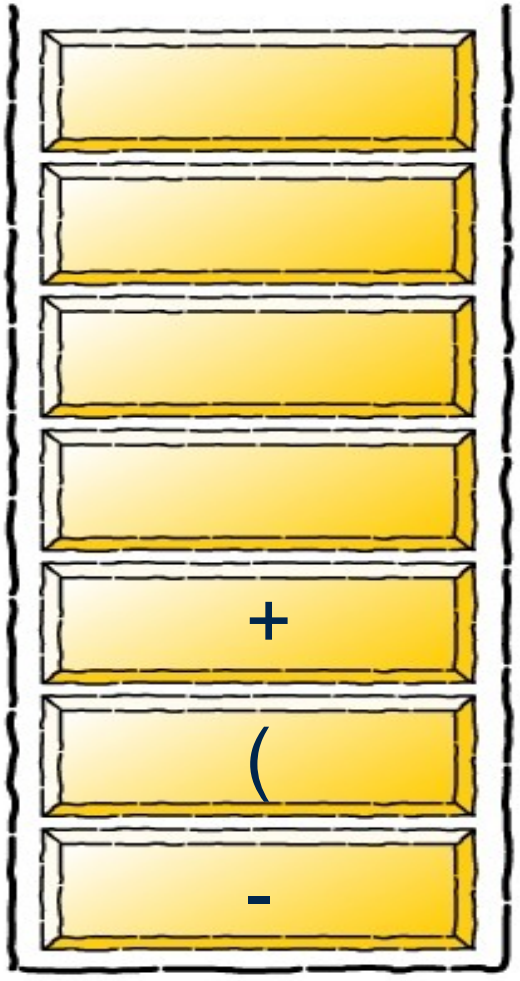
infixVect

f)

postfixVect

a b + c – d * e

Conversão da forma infixa para pós-fixa



stackVect

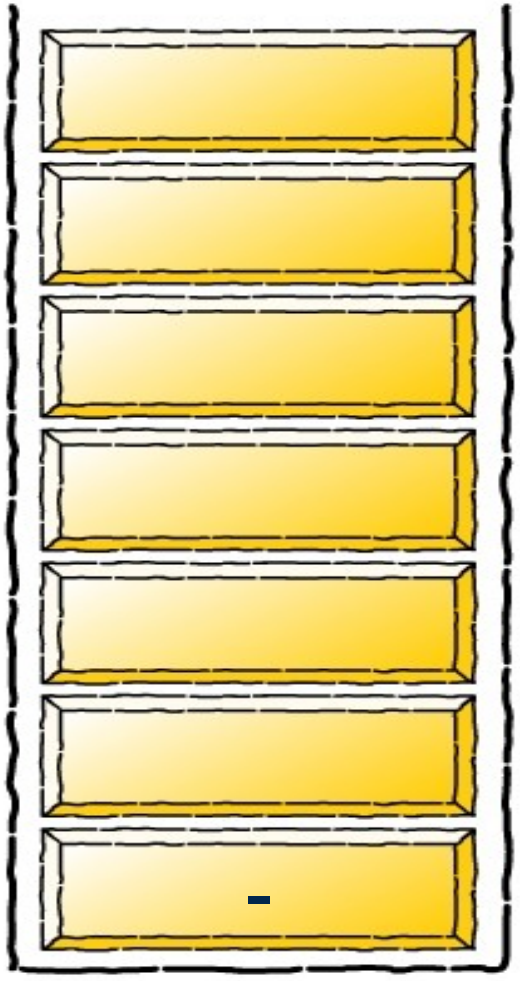
infixVect

)

postfixVect

a b + c – d * e f

Conversão da forma infixa para pós-fixa



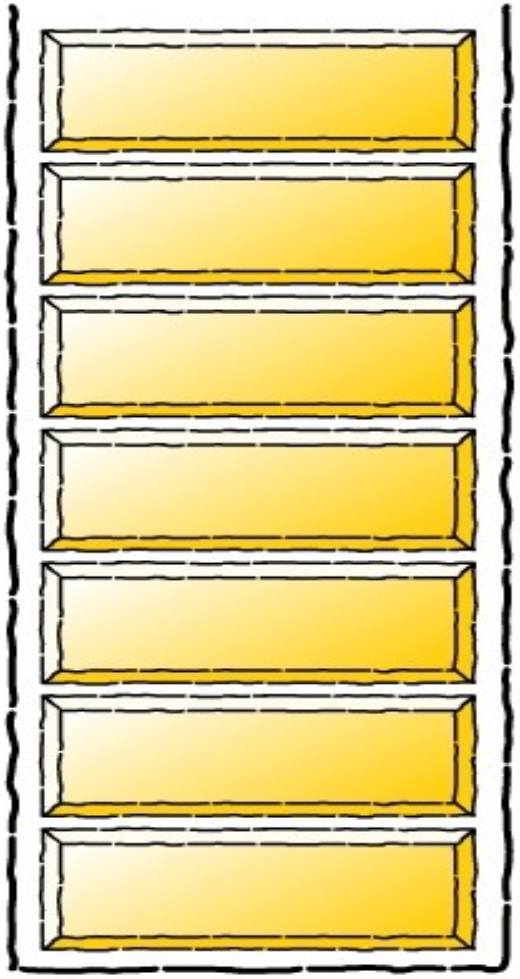
stackVect

infixVect

postfixVect

a b + c – d * e f +

Conversão da forma infixa para pós-fixa



stackVect

infixVect

postfixVect

a b + c – d * e f + -

Funções auxiliares

- Transformação de expressões aritméticas infixas em pós-fixas
  - isoperand
  - isdigit
  - prio
  - expon
  - oper
  - orcd
  - postfix
- Avaliação de expressões aritméticas pós-fixas
  - eval

Transformação de expressões aritméticas infixas em pós-fixas

- **isoperand**
  - Recebe um caractere, ou "token", e retorna verdadeiro (TRUE) quando o caractere não for

    +, -, *, /, , (,).
- **isdigit**
  - Recebe um caractere, ou "token", e retorna TRUE quando o "token" for numérico.

Transformação de expressões aritméticas infixas em pós-fixas

- **prio**
  - Recebe um "token", que seja operador e retorna
    - **1** caso o operador seja + ou **-**
    - **2** caso o operador seja * ou /
    - **3** caso o operador seja ^
    - **0** em caso contrário
- **expon**
  - Recebe dois reais e retorna o valor do primeiro real elevado à potencia igual ao segundo real
- **oper**
  - Recebe um "token" (operador) e dois reais (operandos) e retorna o resultado da aplicação do operador aos dois operandos

Transformação de expressões aritméticas infixas em pós-fixas

- **prcd**
  - Recebe dois “tokens”, representando operadores e retorna TRUE quando o primeiro operador tem precedência sobre o segundo operador ao aparecer à esquerda deste último em uma expressão infixa sem parênteses. Casos particulares ocorrem com o tratamento de parênteses.

Transformação de expressões aritméticas infixas em pós-fixas

- Regras de tratamento de precedência de parênteses

REGRA	COMENTÁRIO
1	Qualquer operador que segue '(' é empilhado
2	'(' só não seria empilhado se o topo da pilha contivesse ')'
3	Quando um ')' encontra um '(' não há desempilhamento. Ambos são descartados. Quando um ')' encontra qualquer outro operador existe desempilhamento
4	')' encontrado na pilha, o que é absurdo. Retorna valor indefinido
GERAL	Regra válida para os demais operadores

- Ao entrar um ')' saem todos os elementos até encontrar um '('. Ambos os delimitadores são descartados.

Transformação de expressões aritméticas infixas em pós-fixas

- **postfix**
  - Recebe dois strings. O primeiro string (de entrada) representa uma expressão aritmética sob a forma infixa. O segundo string serve para retornar uma expressão aritmética na forma pós-fixa.
  - O string de entrada é varrido do início até o final.
  - Se o token do string de entrada for um operando é imediatamente copiado para o string de saída. Se o token de entrada for um operador, enquanto houver operadores empilhados com precedência maior do que o operador de entrada, estes operadores que estavam empilhados são transferidos para a expressão de saída. O operador de entrada é empilhado.
  - Ao término da varredura os operadores são desempilhados e copiados para a expressão de saída. Os delimitadores não são transferidos para o string de saída.

Avaliação de expressões aritméticas pós-fixas

- **eval**
  - Recebe um string que representa uma expressão infixa e faz uma varredura do início para o final do string. Se o token corrente for um operando deve ser empilhado sob a forma de real. Se o token corrente for um operador devem ser desempilhados dois operandos sobre os quais o operador vá atuar. O resultado da aplicação do operador aos operandos é empilhado.
  - Ao término da varredura do string a função retorna o topo da pilha.

Avaliação de expressões pós-fixas

Avaliação de expressões aritméticas pós-fixas

- Esta avaliação será mostrada da seguinte maneira:
  - Inicialmente será apresentado um algoritmo com bastante abstração
  - Será apresentado um exemplo

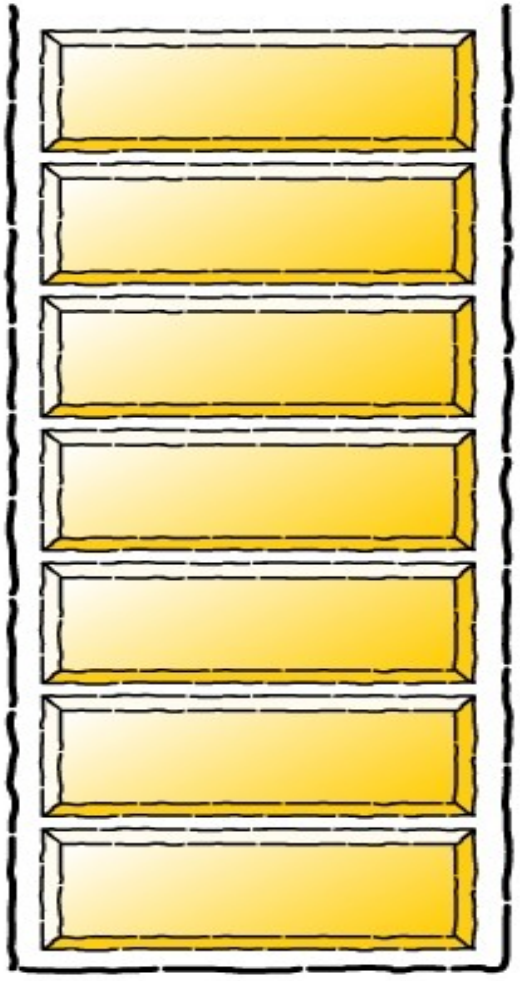
Avaliação de expressões aritméticas pós-fixas

Ler os tokens de um vetor de tokens (strings) **postfixVect** contendo uma expressão pós-fixa

- Quando o *token* for um operando
  - Empilhá-lo em **stackVect**
- Quando o *token* for um operador
  - Desempilhar de **stackVect** dois operandos
  - Aplicar o operador aos dois operandos desempilhados de **stackVect** obtendo um valor (novo operando)
  - Empilhar em **stackVect** o valor obtido da aplicação do operador aos operandos desempilhados

Ao término da varredura do string de **postfixVect** a função retorna o topo da pilha

Avaliação de expressões pós-fixas



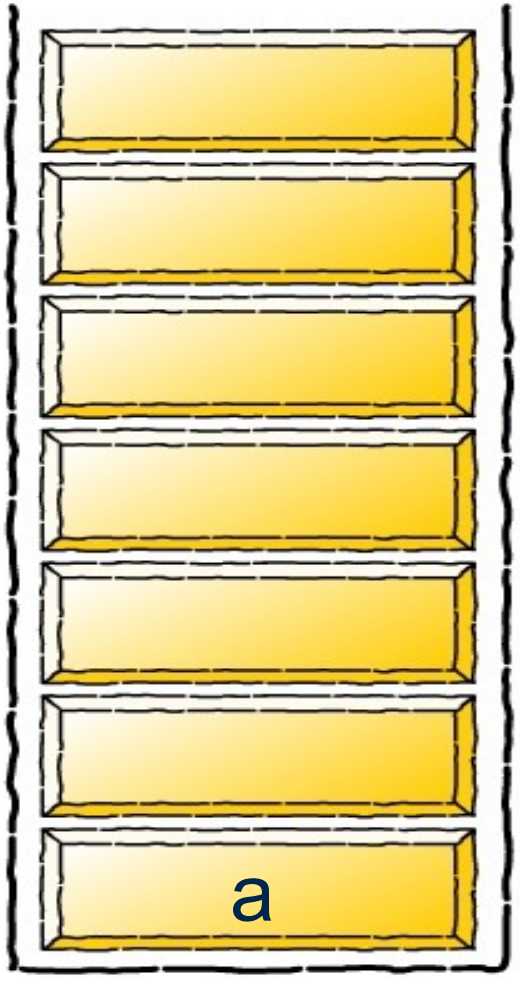
stackVect

postfixVect

a b + c – d * e f + -

Situação inicial

Avaliação de expressões pós-fixas



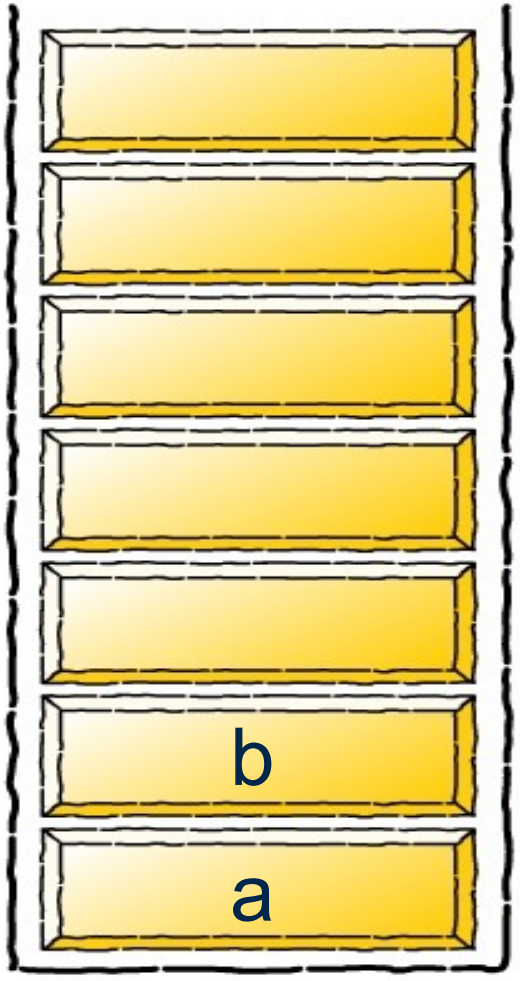
stackVect

postfixVect

a b + c – d * e f + -

Empilhado o operando a

Avaliação de expressões pós-fixas



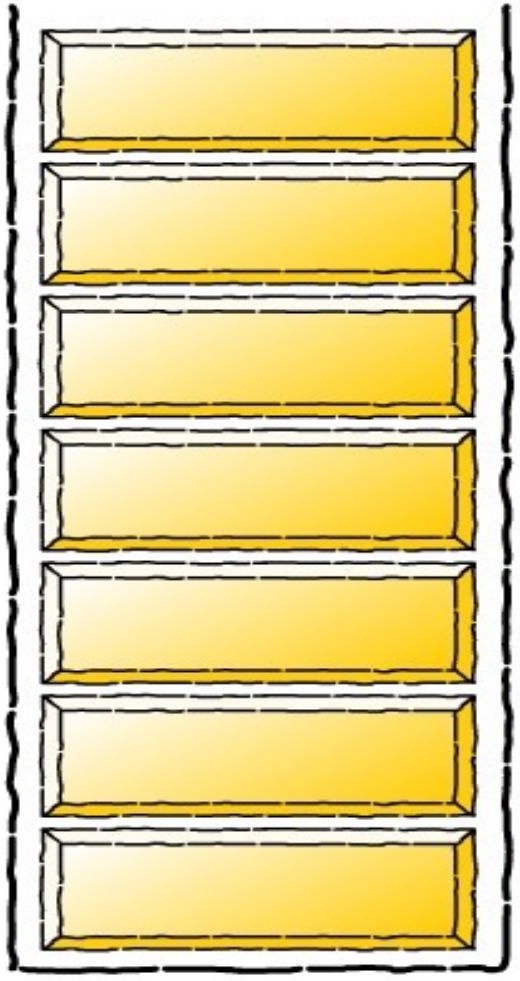
stackVect

postfixVect

a b + c – d * e f + -

Empilhado o operando b

Avaliação de expressões pós-fixas



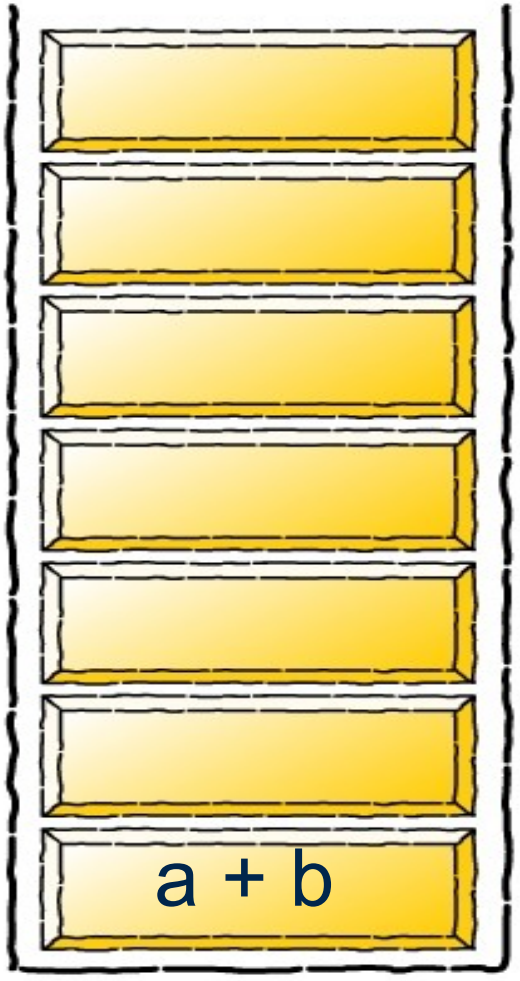
stackVect

postfixVect

a b + c – d * e f + -

Desempilhados operandos a e b
Operador + aplica-se a a e b

Avaliação de expressões pós-fixas



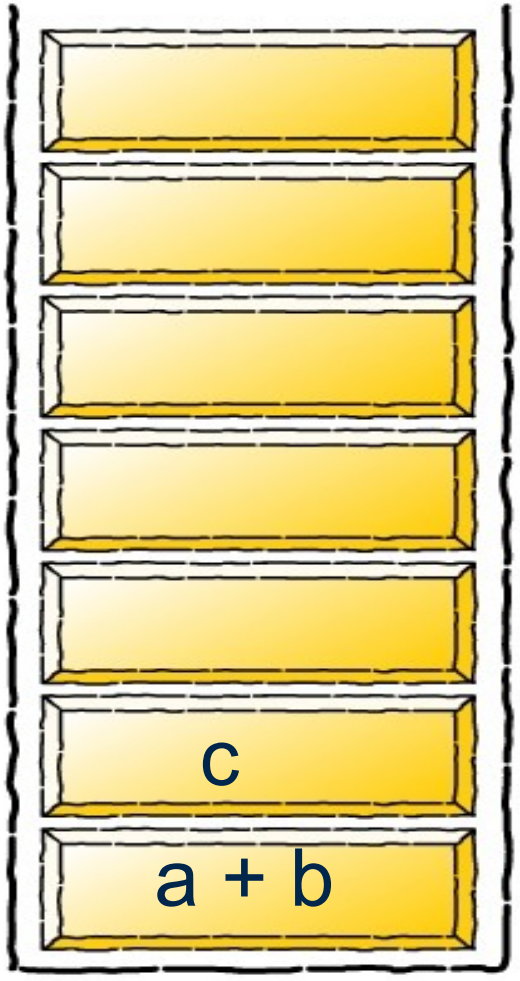
stackVect

postfixVect

a b + c – d * e f + -

Empilhado o resultado a + b

Avaliação de expressões pós-fixas



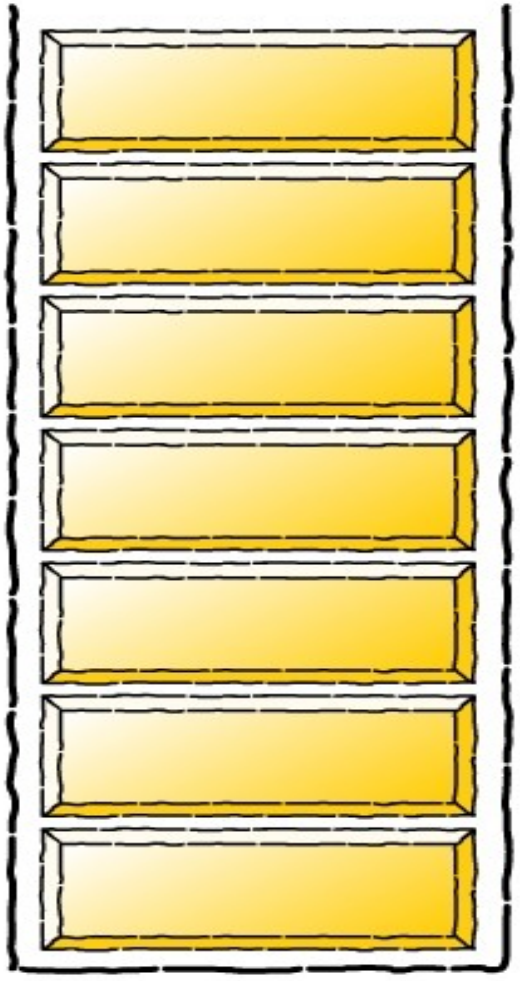
stackVect

postfixVect

a b + c – d * e f + -

Empilhado o operando c

Avaliação de expressões pós-fixas



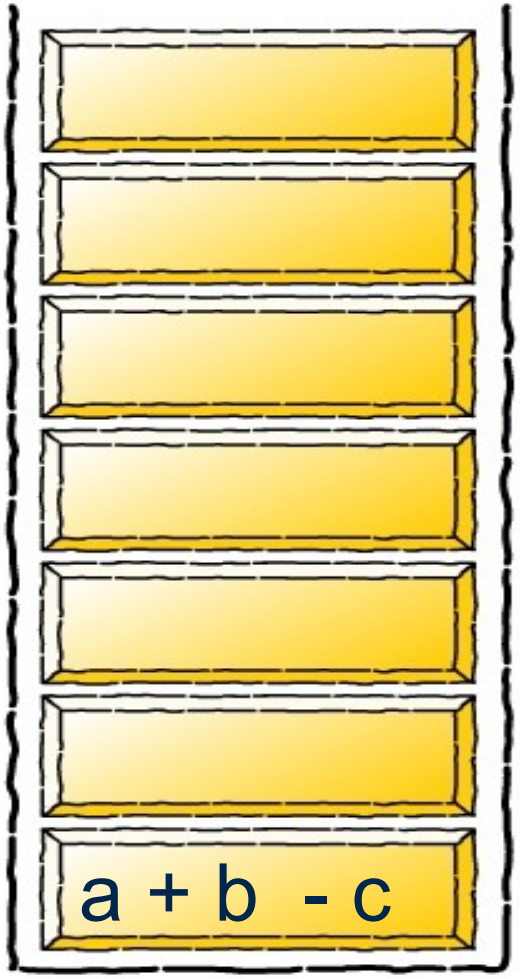
stackVect

postfixVect

a b + c - d * e f + -

Desempilhados operandos (a + b) e c
Operador - aplica-se a (a + b) e c

Avaliação de expressões pós-fixas



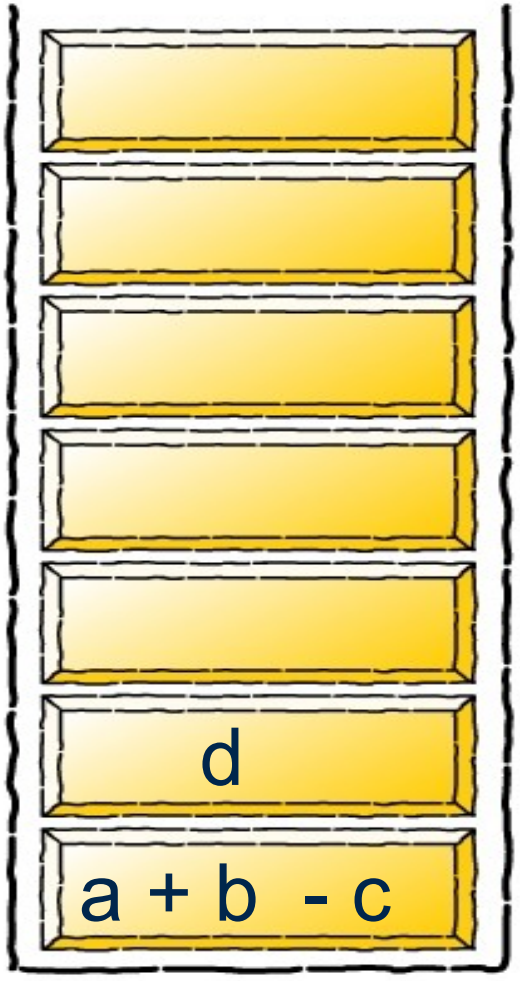
stackVect

postfixVect

a b + c - d * e f + -

Empilhado o resultado (a + b - c)

Avaliação de expressões pós-fixas



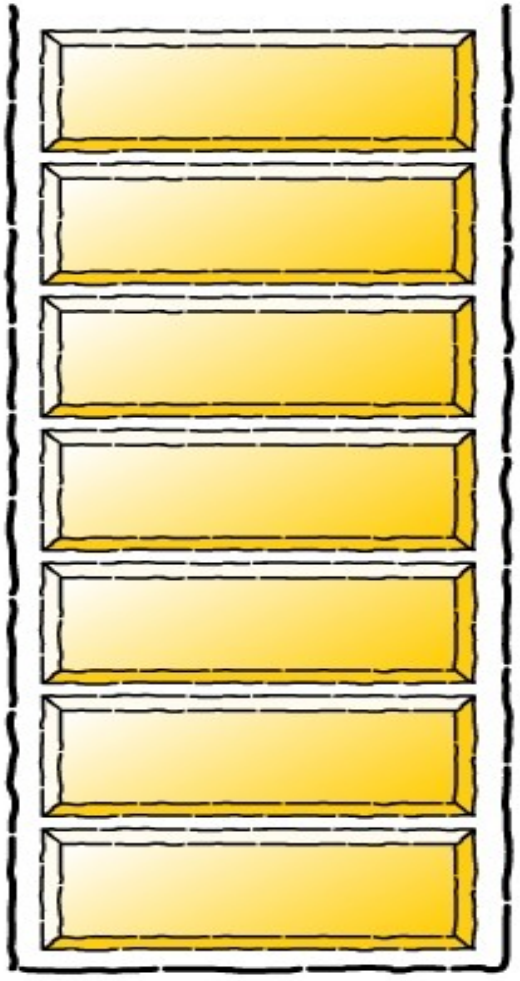
stackVect

postfixVect

a b + c – d * e f + -

Empilhado o operando d

Avaliação de expressões pós-fixas



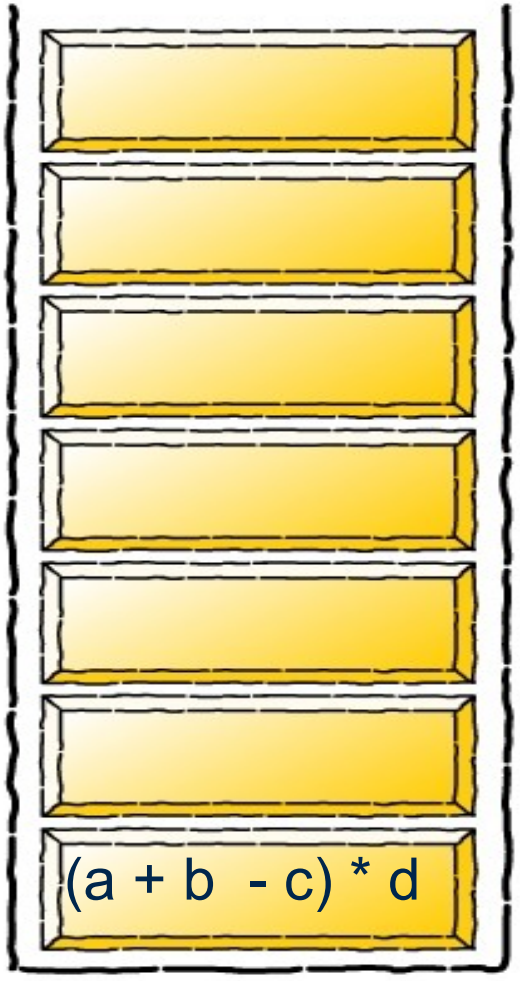
stackVect

postfixVect

a b + c – d * e f + -

Desempilhados (a+b-c) e d
Operador * aplica-se a (a+b-c) e d

Avaliação de expressões pós-fixas



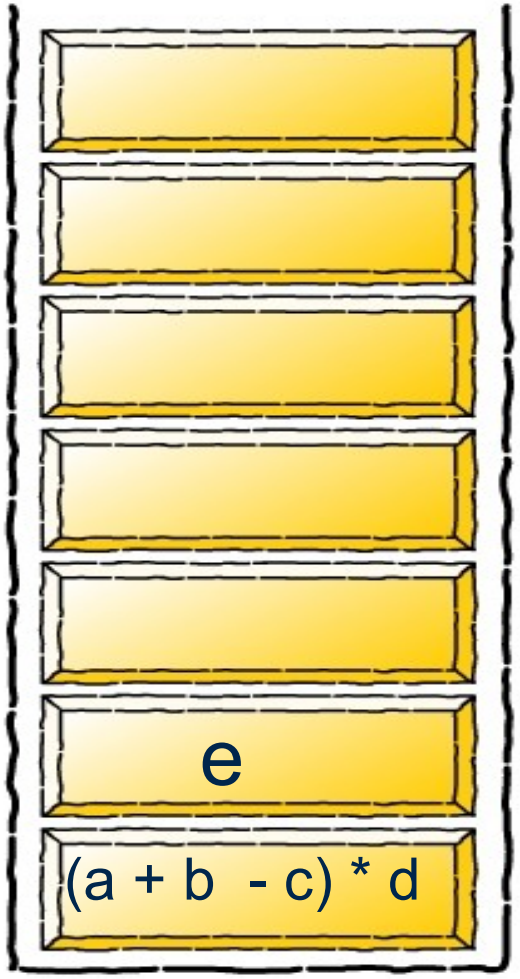
stackVect

postfixVect

a b + c – d * e f + -

Empilhado o operando (a+b-c) * d

Avaliação de expressões pós-fixas



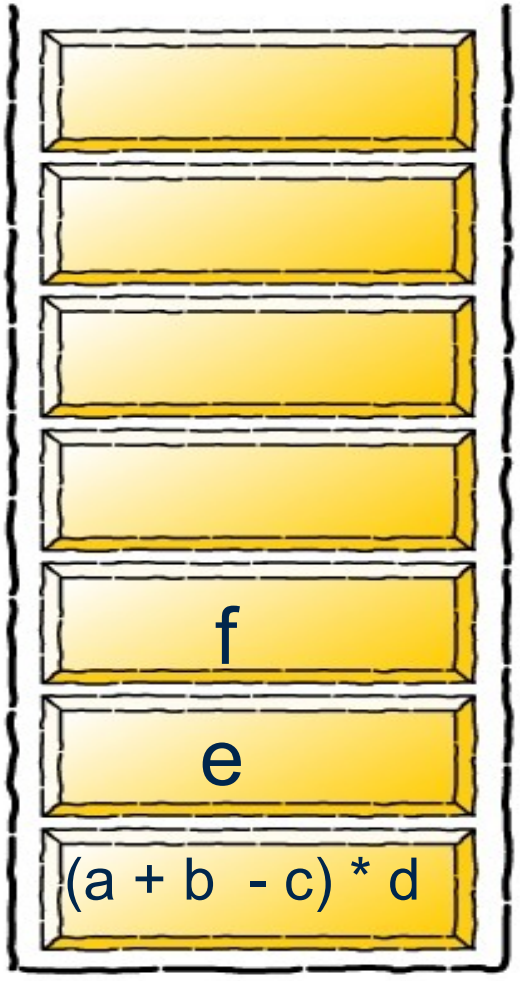
stackVect

postfixVect

a b + c – d * e f + -

Empilhado o operando e

Avaliação de expressões pós-fixas



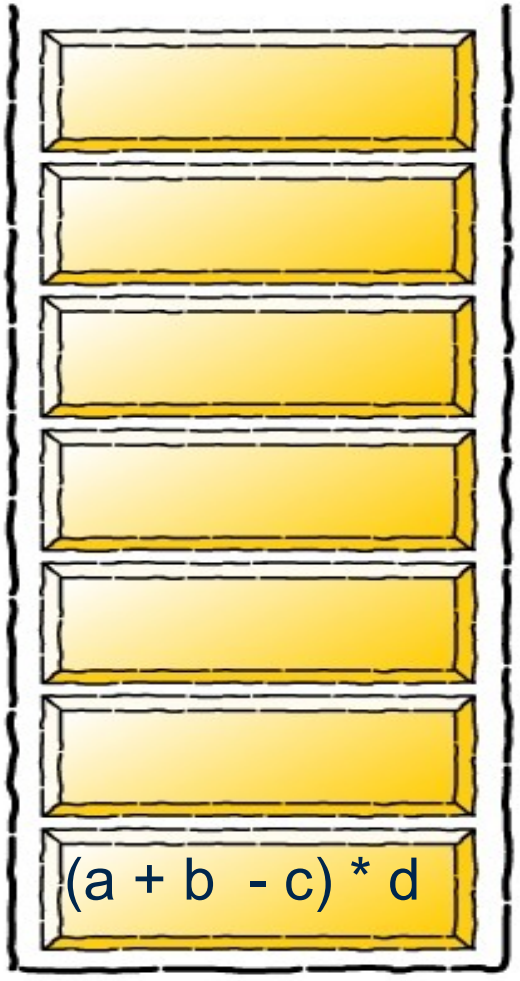
stackVect

postfixVect

a b + c – d * e f + -

Empilhado o operando f

Avaliação de expressões pós-fixas



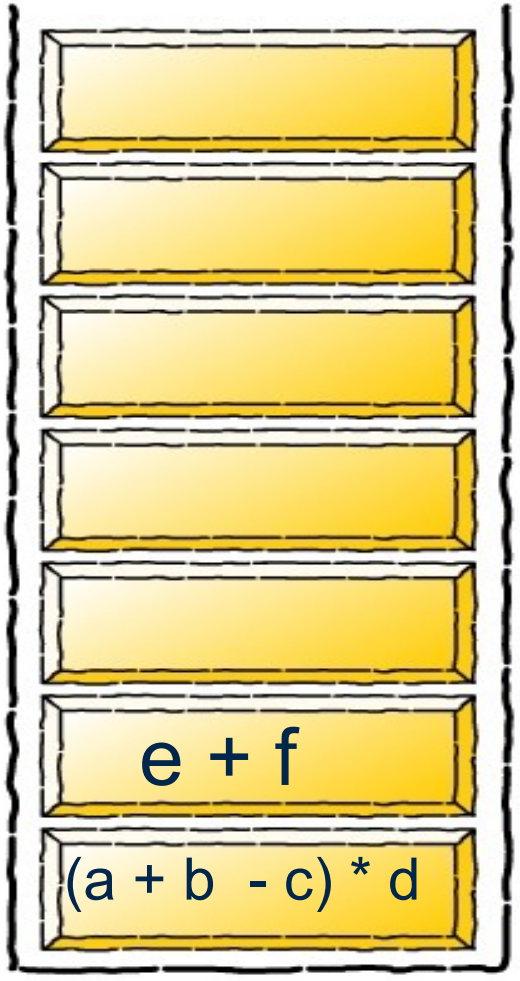
stackVect

postfixVect

a b + c – d * e f + -

Desempilhados operandos e e f
Operador + aplica-se a e e f

Avaliação de expressões pós-fixas



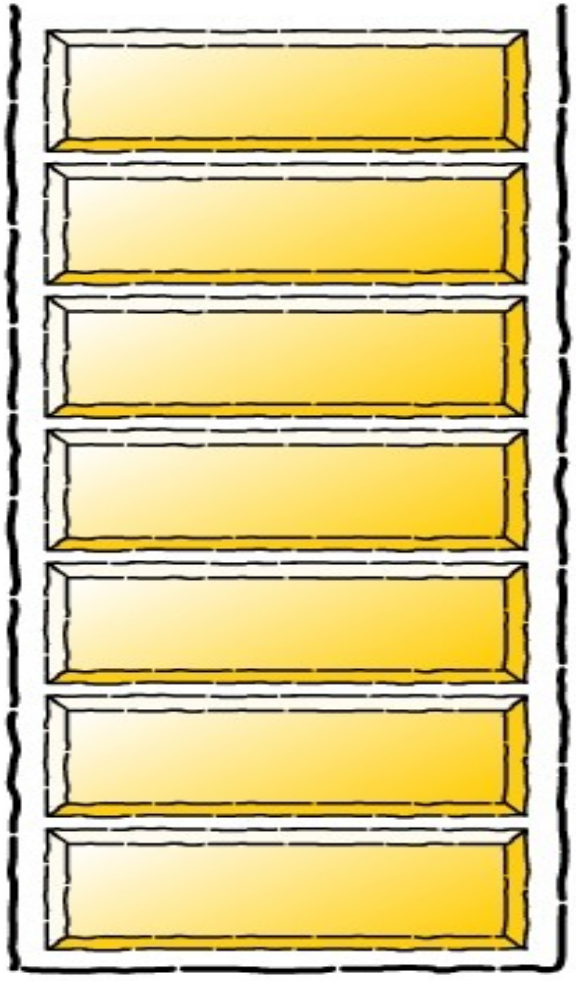
stackVect

postfixVect

a b + c – d * e f + -

Empilhado operando e + f

Avaliação de expressões pós-fixas



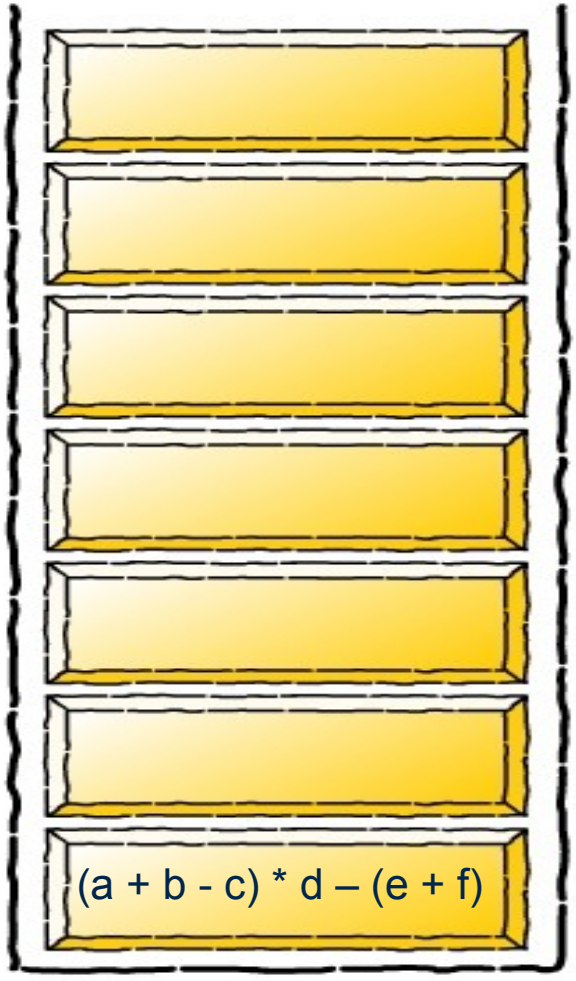
stackVect

postfixVect

a b + c – d * e f + -

Desempilhados operandos (a+b-c)*d e (e+ f)
Operador - aplica-se a (a+b-c)*d e (e+ f)

Avaliação de expressões pós-fixas



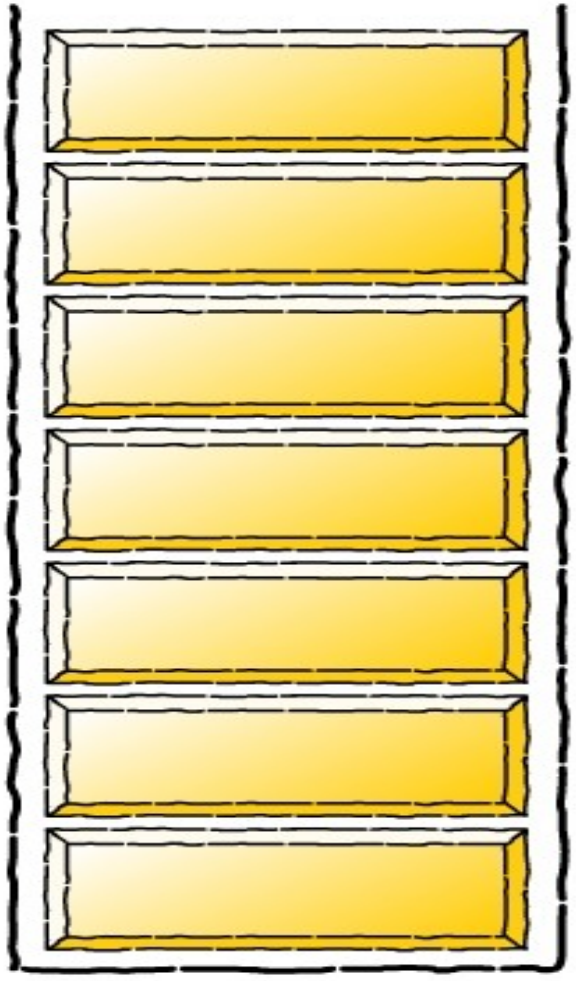
stackVect

postfixVect

a b + c – d * e f + -

Empilhado (a + b - c) * d – (e + f)

Avaliação de expressões pós-fixas



stackVect

postfixVect

a b + c – d * e f + -

Saída obtida

Desempilhar (a + b - c) * d – (e + f)