

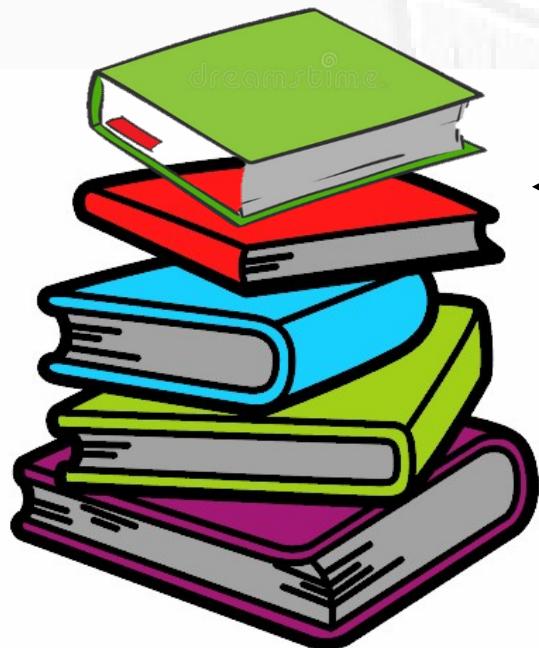
Pilha

Slides: Regina Célia Coelho (com alterações)
Daniela Leal Musa (musa@unifesp.br)

Definição de Pilha

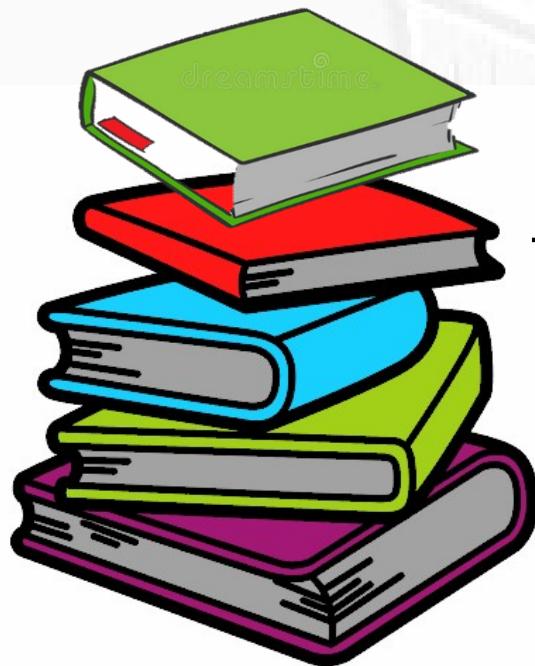


Definição de Pilha



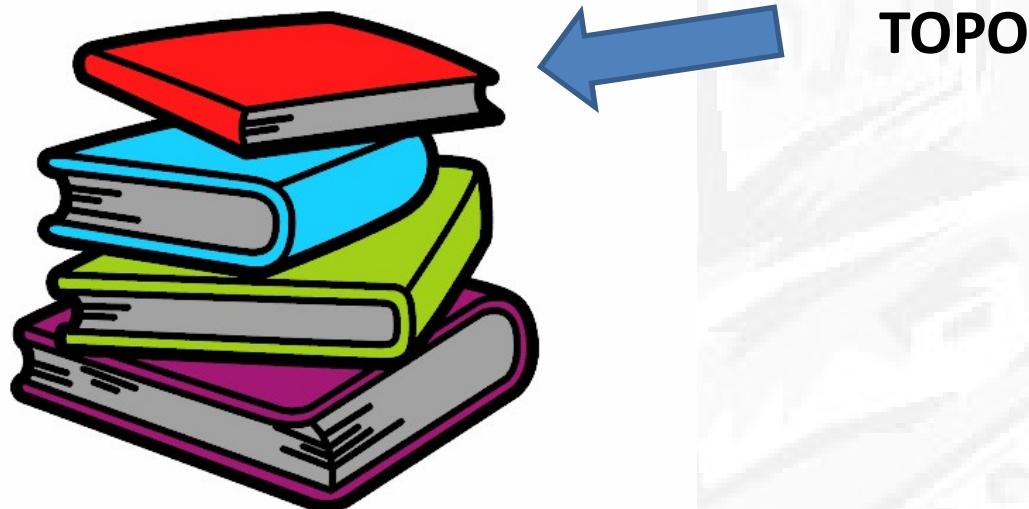
EMPILHAR (PUSH)

Definição de Pilha



DESEMPLHAR (POP)

Definição de Pilha



Definição de Pilha



BASE

Definição de Pilha



Está Vazia?

Está Cheia?

Definição de Pilha

- Conjunto de dados em que:
 - ✓ O novo elemento sempre é inserido no topo da pilha (ordem de chegada);
 - ✓ O elemento removido é sempre o que chegou a menos tempo na pilha (o último inserido);
 - ✓ A consulta normalmente retorna o elemento no topo da pilha.

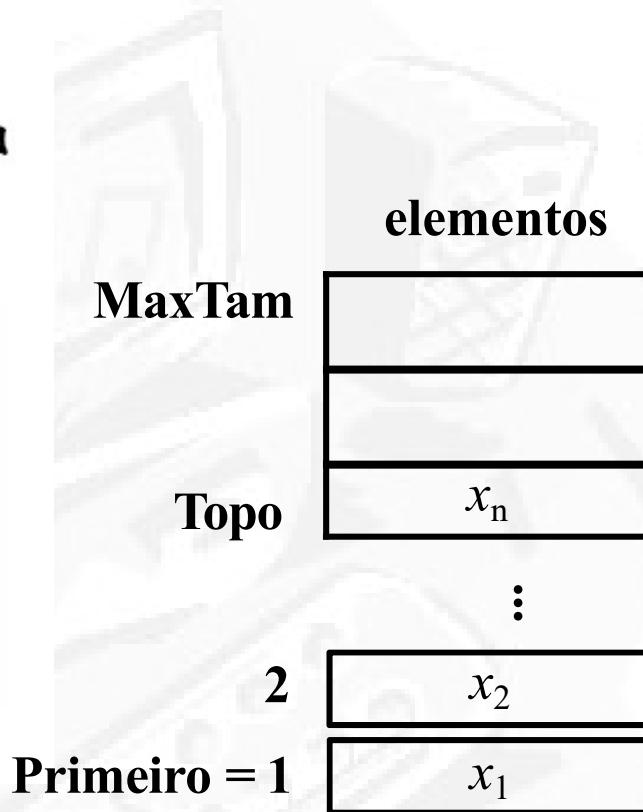
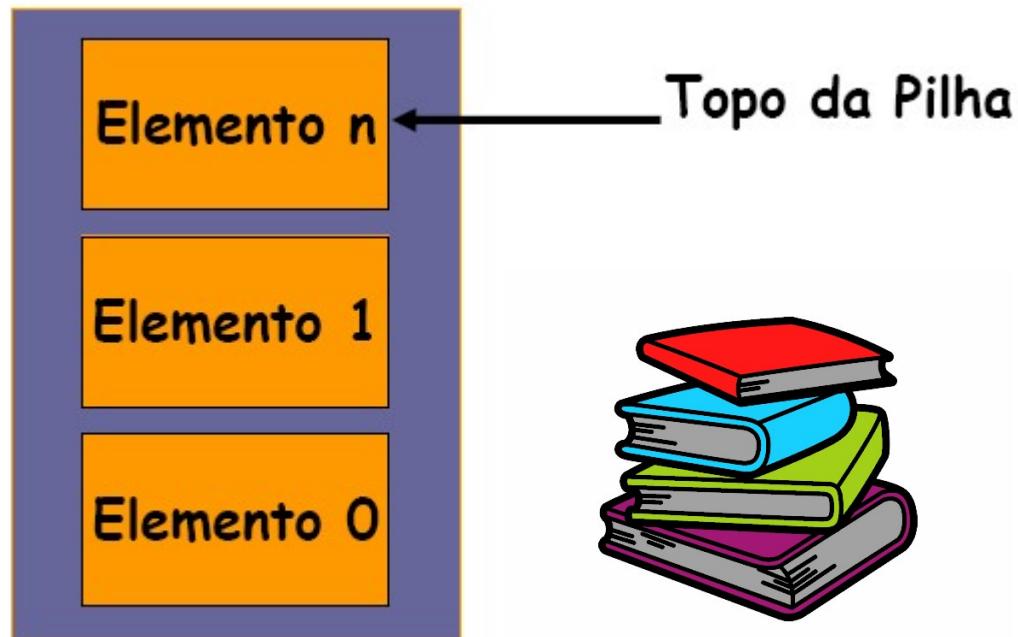
Definição de Pilha

- Estrutura de dados comumente chamada de LIFO = *Last In, First Out.*
- O último elemento a entrar na pilha será o primeiro elemento a sair da pilha

Aplicações de Pilha

- Verificar se um código fonte está bem estruturado;
- Parser de expressões aritméticas;
- Controle da sequência de chamadas de funções;
- Recursividade;
- Processamento de quaisquer estruturas aninhadas de profundidade imprevisível:
 - ✓ Sintaxe de expressões aritméticas;
 - ✓ Controle da sequência de chamadas de expressões aritméticas.

Representação Gráfica de Pilhas



Operações Básicas sobre Pilhas

- Criar a Pilha
- Empilhar elemento (*Push*)
- Desempilhar elemento (*Pop*)
- Consultar elemento no topo da pilha (*Peek*)
- Listar os elementos

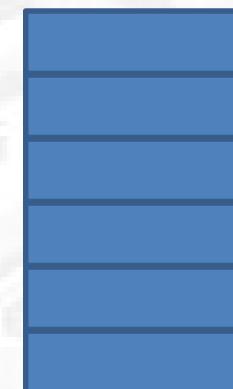
Implementando uma Pilha

- **Contiguidade Física**
- Encadeada (alocação dinâmica)

Criando uma pilha

- Dada um pilha $P=(a(1), a(2), \dots, a(n))$, dizemos que $a(1)$ é o elemento da base da pilha; $a(n)$ é o elemento topo da pilha; e $a(i+1)$ está acima de $a(i)$.

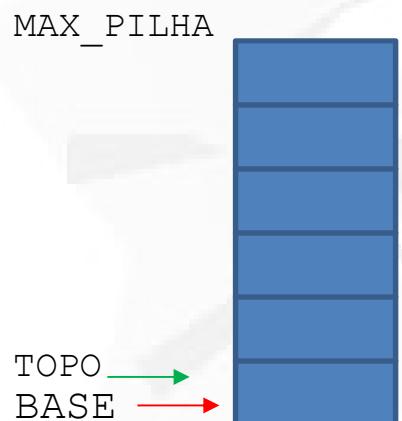
```
MAX_PILHA  
Typedef struct {  
    int vet[MAX_PILHA];    a(n)  
    int topo;  
} TPilha;
```



Contigüidade física

Exemplo de Manipulação

1. Inicializar pilha com 6 posições, de números inteiros.
2. Inserir elemento com valor 5
3. Inserir elemento com valor 11
4. Inserir elemento com valor 8



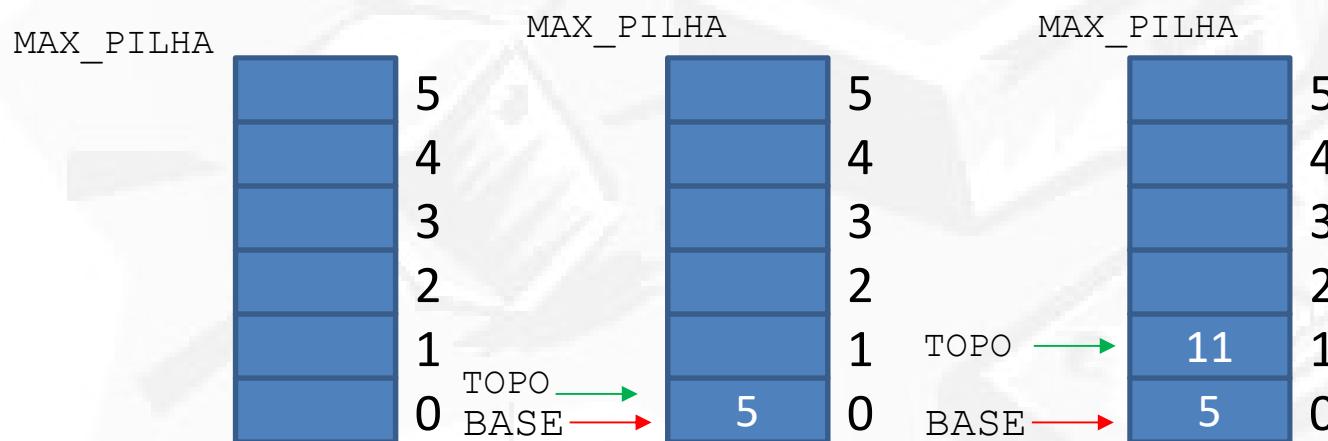
Exemplo de Manipulação

1. Inicializar pilha com 6 posições, de números inteiros.
2. Inserir elemento com valor 5
3. Inserir elemento com valor 11
4. Inserir elemento com valor 8



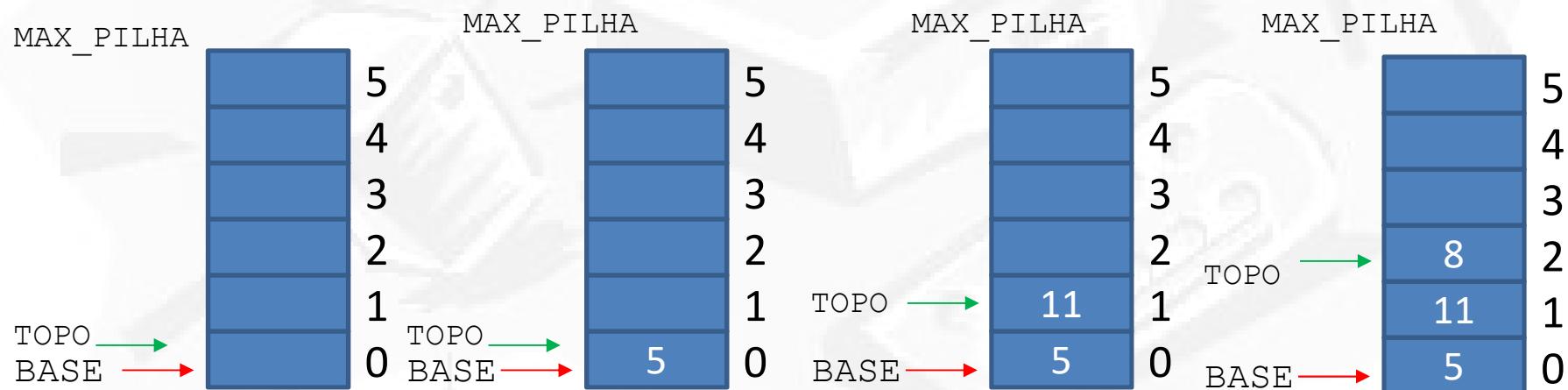
Exemplo de Manipulação

1. Inicializar pilha com 6 posições, de números inteiros.
2. Inserir elemento com valor 5
3. Inserir elemento com valor 11
4. Inserir elemento com valor 8



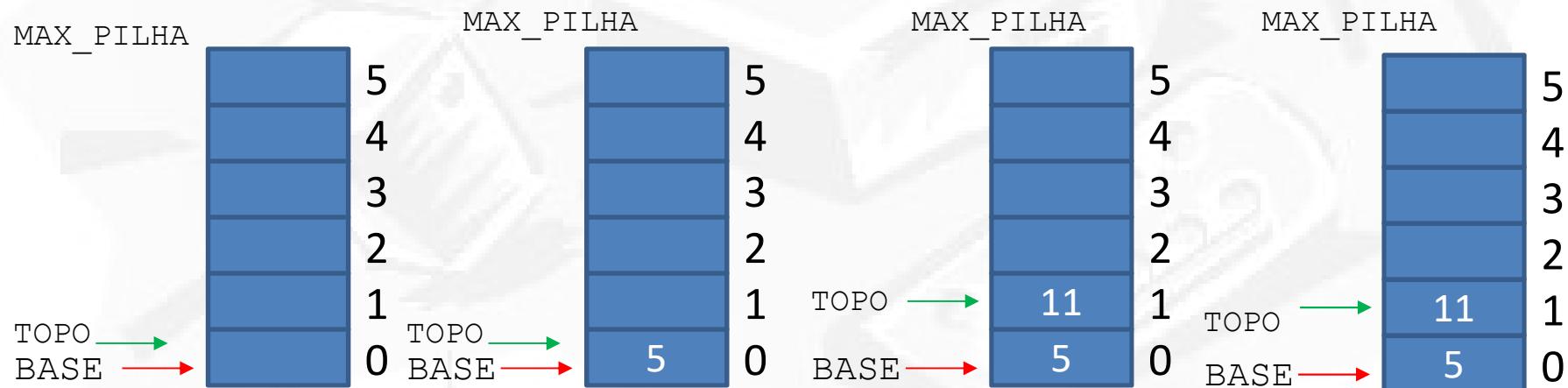
Exemplo de Manipulação

1. Inicializar pilha com 6 posições, de números inteiros.
2. Inserir elemento com valor 5
3. Inserir elemento com valor 11
4. Inserir elemento com valor 8



Exemplo de Manipulação

1. Inicializar pilha com 6 posições, de números inteiros.
2. Inserir elemento com valor 5
3. Inserir elemento com valor 11
4. Inserir elemento com valor 8
5. Remover elemento



Criando uma pilha

- Primeiro inicializamos uma pilha fazendo o campo topo = -1:

Criando uma pilha

- Primeiro inicializamos uma pilha fazendo o campo topo = -1:

```
TPilha* nova () {  
    TPilha* np = (TPilha *) malloc (sizeof (TPilha));  
    np->topo = -1;  
    return np;  
}
```

Empilhando dados na pilha

- Para empilhar é necessário considerar se a pilha já foi criada e a possibilidade de estouro de pilha, uma vez que estamos trabalhando com vetor de tamanho fixo.

```
int push (TPilha* p, int val) {  
    if (p->topo >= MAX_PILHA-1) // Pilha cheia  
        return -1;  
    (p->topo)++;  
    p->vet[p->topo] = val;  
    return 0;  
}
```

Desempilhando dados na Pilha

- Para desempilhar é necessário considerar se a pilha foi criada e a possibilidade de *underflow*.

```
int pop (TPilha* p, int* val) {  
    if (p->topo < 0)  
        return -1; // Pilha vazia  
    *val = p->vet[p->topo];  
    p->topo--;  
    return 0;  
}
```

Liberando espaços alocados

- Para liberar os espaços alocados basta apenas liberar o espaço alocado para a estrutura criada. Faça a função libera.

```
TPilha *remover (TPilha* p) {  
    free (p);  
    return (NULL);  
}
```

Exercícios

- Faça uma função para listar todos os elementos da pilha.

Exercícios

- Faça uma função para listar todos os elementos da pilha.

```
void ConsultaPilha (TPilha *p) {  
  
    if (p->topo < 0)  
        return -1  
    for (int i=0;i<=p->topo;i++)  
        printf ("\n Elemento: %d", p->vet[p->topo-i]);  
}
```

Exercícios

- Faça um programa que use uma pilha para inverter a ordem das letras da frase. Por exemplo, dado o texto “ESTE EXERCICIO E MUITO FACIL” a saída deve ser “LICAF OTIUM E OICICREXE ETSE”. Use as funções push e pop.

- Faça uma programa que use uma pilha para inverter a ordem das letras da frase. Por exemplo, dado o texto “ESTE EXERCICIO E MUITO FACIL” a saída deve ser “LICAF OTIUM E OICICREXE ETSE”. Use as funções push e pop.

```
int main() {
    TPilha *minhapilha;
    char string[40];
    printf ("\n Insira frase para empilhar: ");
    scanf ("%[^\\n]s", string);
    minhapilha = nova();
    if (minhapilha != NULL) {
        for (int i=0; i<(strlen(string)); i++) {
            if (push(minhapilha, string[i])==0)
                printf ("\n Empilhado");
        }
    }
    ConsultaPilha(minhapilha);
    return 0;
}
```

Exercícios

- Refaça o programa anterior de forma que agora seja invertido a ordem das letras de cada palavra de uma cadeia de caracteres, preservando a ordem das palavras. Por exemplo, dado o texto “ESTE EXERCICIO E MUITO FACIL” a saída deve ser “ETSE OICICREXE E OTIUM LICAF”.

Use as funções push e pop.

Exercícios

- Digamos que nosso alfabeto seja formado pelas letras a, b e c. Considere o seguinte conjunto de cadeias de caracteres sobre nosso alfabeto: c, aca, bcb, abcba, bacab, aacaa, bbcbb, . . . Qualquer cadeia deste conjunto tem a forma WcM, sendo que W é uma sequência de letras que só contém a e b e M é o inverso de W, ou seja, M é W lido de trás para frente. Escreva um programa que determina se uma cadeia X pertence ou não ao nosso conjunto, ou seja, determina se X é da forma WcM.

Aplicação de Pilha

- Uma das aplicações clássicas de pilhas é a avaliação de expressões aritméticas
- Representações na forma tradicional são ambíguas e por isso obriga o pré-estabelecimento de “regras de prioridade”. Estas representações não são convenientes para o computador
- A ordem de execução das operações na expressão “ $A + B * C - 4 / 2$ ” só torna-se clara se as regras de prioridade são conhecidas:
 - ✓ A mesma expressão na forma “ $A + (B * C) - (4 / 2)$ ” seria mais facilmente interpretada numa rápida leitura.

Outras Notações

- Seja a expressão na notação tradicional $A * B - C / D$
- **Notação completamente Parentizada:** acrescenta-se sempre um parênteses a cada par de operandos e seu operador.

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

- **Notação Polonesa (ou prefix):** os operadores aparecem imediatamente antes dos operandos. Esta notação especifica quais operadores, e em que ordem, devem ser calculados. Por isso dispensa o uso de parênteses, sem ambigüidades.

$$-* A B / C D$$

- **Notação Polonesa Reversa (ou posfix):** é como a polonesa na qual os operadores aparecem após os operandos.

$$A B * C D / -$$

Exercícios

Infix	PréFix	PósFix
$A+B*C$		
$A+B-C$		
$A+(B-C)$		
$(A+B)/(C-D)$		

Exercícios

Infix	PréFix	PósFix
$A+B*C$	$+A*BC$	$ABC*+$
$A+B-C$	$-+ABC$	$AB+C-$
$A+(B-C)$	$+A-BC$	$ABC-+$
$(A+B)/(C-D)$	$/+AB-CD$	$AB+CD-/$

Exemplo 1 - algoritmo para avaliação de uma expressão Pós-Fixada

- Os operandos são números reais (positivos e negativos);
- Os operadores são: {+, -, x, /, ^}
- Os delimitadores são: {(,)}
- Cada operando é empilhado numa pilha de valores
- Quando se encontra um operador :
 - desempilhar o número apropriado de operandos (dois para operadores binários e um para operadores unários)
 - realizar a operação devida
 - empilhar o resultado

EXEMPLO 1

Resolver expressão em notação posfixa

1 2 – 4 5 + *

- 1) Varrer expressão da esquerda para direita

EXEMPLO 1

Avaliação da expressão $1\ 2 - 4\ 5 + *$

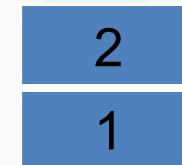
- 1) Empilhar dois operandos

2
1

EXEMPLO 1

Avaliação da expressão $1\ 2 - 4\ 5 + *$

1) Empilhar dois primeiros operandos



2) Quando aparecer operador:

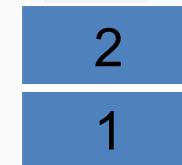
- desempilhar operandos
- realizar operação
- empilhar resultado



EXEMPLO 1

Avaliação da expressão $1\ 2 - 4\ 5 + *$

1) Empilhar dois primeiros operandos



2) Quando aparecer operador:

- desempilhar operandos
- realizar operação
- empilhar resultado



EXEMPLO 1

Avaliação da expressão $1\ 2 - 4\ 5 + *$

1) Empilhar dois primeiros operandos



2) Quando aparecer operador:

- desempilhar operandos
- **realizar operação**
- empilhar resultado

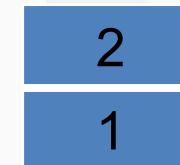
$1 - 2$



EXEMPLO 1

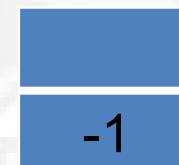
Avaliação da expressão $1\ 2 - 4\ 5 + *$

1) Empilhar dois primeiros operandos



2) Quando aparecer operador:

- desempilhar operandos
- realizar operação $1-2 = -1$
- empilhar resultado



EXEMPLO 1

Avaliação da expressão $1\ 2 - 4\ 5 + *$

Repetir passos 1 e 2

1) Empilhar dois operandos

5
4

2) Quando aparecer operador:

- desempilhar operandos
- realizar operação
- empilhar resultado

9
-1

EXEMPLO 1

Avaliação da expressão $1\ 2 - 4\ 5 + *$

Quando aparecer operador:

- desempilhar operandos
- realizar operação $- 1 * 9 = -9$
- empilhar resultado

9
-1
-9

EXEMPLO 2 – Converter expressão de InFix para PosFix

- Vetor com a expressão de entrada Infixa
- Vetor com a expressão de saída PosFix
- Pilha

O processamento consistem em uma repetição que analisa cada item de entrada e gerencia a pilha e a transferência para a saída

A repetição continua enquanto não ocorrer erro e não terminar a expressão infixa

Quando todos os valores de entrada tiverem sido processados, repetidamente desempilhar um elemento da pilha e adicioná-lo a vetorPosFix até esvaziar Pilha

EXEMPLO 2

Caso o item de entrada (elemento) seja:

- **um operando**
 - Incluí-lo no final do vetor PostFix
- **um “(“**
 - Empilhar elemento na Pilha
- **um “)”**
 - Repetidamente desempilhar um elemento da Pilha e adicioná-lo ao vetor PosFix até encontrar “(“
 - Desempilhar “(“ de Pilha
 - Se Pilha esvaziar antes de encontrar um “(“, a expressão de entrada não é válida.
- **um operador**
 - Se Pilha estiver vazia ou o elemento de entrada tiver prioridade maior do que a operador topo de pilha
 - Empilhar o elemento de entrada.
 - Caso contrário
 - desempilhar o elemento da Pilha e adiciona-lo ao vetor PosFix
 - Repetir a comparação do elemento de entrada com o novo topo de pilha

EXEMPLO 2

Operando -> vetor

infix

a + b * d

postfix

a

Pilha

Algoritmos e Estruturas de Dados

EXEMPLO 2

Operador -> pilha vazia
empilha

infix

a + b * d

posfix

a

+

Pilha

Algoritmos e Estruturas de Dados

EXEMPLO 2

Operando ->vetor

infix

a + b * d

postfix

ab

+

Pilha

Algoritmos e Estruturas de Dados

EXEMPLO 2

Operador com prioridade maior
empilhar

infix

a + b * d

postfix

ab

*

+

Pilha

Algoritmos e Estruturas de Dados

EXEMPLO 2

Operando empilhar

infix

a + b * d

posfix

abd

*

+

Pilha

Algoritmos e Estruturas de Dados

EXEMPLO 2

Avaliou toda expressão
desempilhar tudo

infix

a + b * d

posfix

abd*+

*

+

Pilha

Algoritmos e Estruturas de Dados

FIM