
Pilhas

Fabrício J. Barth

BandTec - Faculdade de Tecnologia Bandeirantes

Fevereiro de 2011

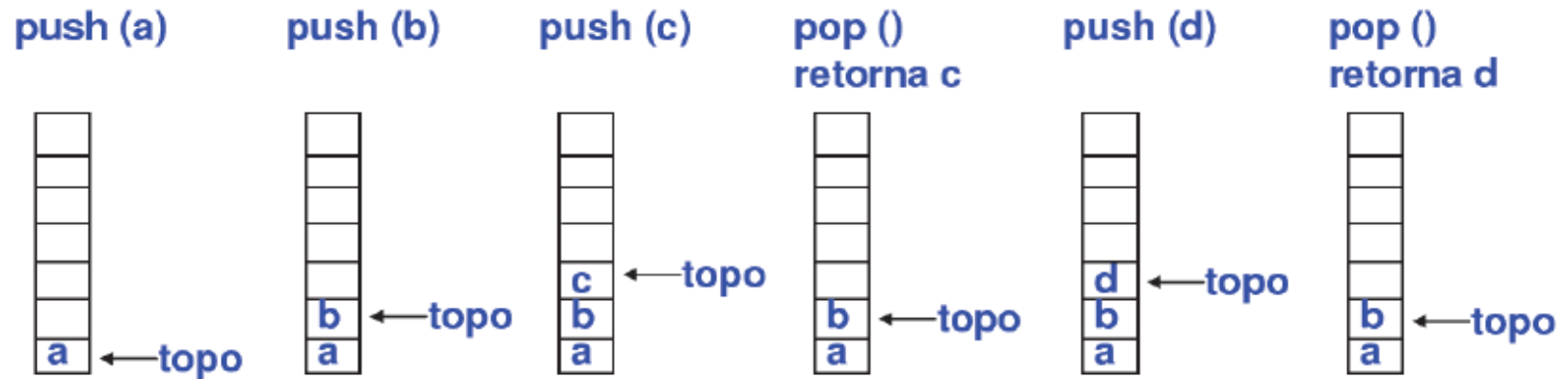
Tópicos Principais

- Introdução
- Interface do tipo pilha
- Exemplo de uso: verificação de expressões
- Implementação de pilha com lista encadeada

Introdução - Pilha

- novo elemento é inserido no topo e acesso é apenas ao topo
 - ★ o primeiro que sai é o último que entrou (LIFO - *last in, first out*)
- operações básicas:
 - ★ empilhar (*push*) um novo elemento, inserindo-o no topo
 - ★ desempilhar (*pop*) um elemento, removendo-o do topo

Introdução - Pilha



Interface do tipo Pilha

public interface Pilha

- **cria()** : aloca dinamicamente a estrutura da pilha; inicializa seus campos e retorna seu objeto
- **push(v)** e **pop()**: inserem e retiram, respectivamente um valor na pilha
- **vazia()**: informa se a pilha está ou não vazia
- **libera()**: destrói a pilha, liberando toda a memória usada pela estrutura.

Interface do tipo pilha

```
1 public interface Pilha {  
2     public Pilha cria();  
3     public void push(float v);  
4     public float pop();  
5     public boolean vazia();  
6     public void libera();  
7 }
```

Implementação de pilha com vetor

```
1  public class PilhaVetor implements Pilha{
2
3      /*numero maximo de elementos*/
4      private final static int N = 50;
5      private int count = -1;
6      private float[] elementos;
7
8      public Pilha cria() {
9          this.elementos = new float[N];
10         return this;
11     }
12
13     public void push(float v) {
14         if(this.count >= N-1){
15             System.out.println("Capacidade da pilha estorou - " +
16                 "finalizando programa");
17             System.exit(1);
18         }
19         /*inseri elemento na proxima posicao livre*/
20         this.elementos[count+1] = v;
21         this.count++;
22     }
```

```
1
2     public float pop() {
3         float v;
4         if(this.vazia()){
5             System.out.println("Pilha vazia - " +
6                                 "finalizando o programa");
7             System.exit(1);
8         }
9         /*retira elemento do topo*/
10        v = this.elementos[count];
11        this.count--;
12        return v;
13    }
14
15    public boolean vazia() {
16        return (this.count <= -1);
17    }
18
19    public void libera() {
20        /*nao libera de fato, apenas reinicializa*/
21        this.elementos = new float[N];
22        this.count = -1;
23    }
24 }
```


Exemplo de uso

- Verificação de expressões matemáticas
 - ★ Considerando cadeias de caracteres com expressões matemáticas que podem conter termos entre parênteses, colchetes ou chaves, ou seja, entre os caracteres (e), ou [e], ou { e };
 - ★ função que retorna 1, se os parênteses, colchetes e chaves de uma expressão aritmética *exp* são abertos e fechados corretamente, ou 0 caso contrário;

- ★ Para a expressão $2 * \{3 + 4 * (2 + 5[2 + 3])\}$ retornaria 1;
- ★ Para a expressão $2 * \{3 + 4 * (2 + 5[2 + 3])\}$ retornaria 0;
- Protótipo do método: **verificaFechamento(char[] s)**

Exemplo de uso

Verificação de expressões matemáticas: a estratégia é percorrer a expressão da esquerda para a direita:

- Se encontra (, [ou {, empilha;
- Se encontra),] ou }, desempilha e verifica o elemento no topo da pilha, que deve ser o caractere correspondente;
- Ao final, a pilha deve estar vazia.

Exemplo de uso

```
1  private char fecho(char c){
2      if(c==']') return '[';
3      if(c=='}') return '{';
4      if(c=='') return '(';
5      return 'x';
6  }
7
8  private boolean verificaFechamento(char[] s){
9      Pilha p = new PilhaVetorChar(); p.cria();
10     for(int i=0; i<s.length; i++){
11         if(s[i]=='(' || s[i]=='{' || s[i]=='['){
12             p.push(s[i]);
13         }else if(s[i]==')' || s[i]=='}' || s[i]==']'){
14             if(p.vazia()) return false;
15             if(p.pop()!=fecho(s[i])) return false;
16         }
17     }
18     if(!p.vazia()) return false;
19     p.libera();
20     return true;
21 }
```

Implementação de pilha com lista

```
1 public class PilhaLista implements Pilha{
2     private NodoPilha prim;
3     public Pilha cria() {return null;}
4     public void push(float v) {}
5     public float pop() {return 0;}
6     public boolean vazia() {return false;}
7     public void libera() {}
8 }
9 class NodoPilha{
10     private float v;
11     private Nodo prox;
12 }
```

Implementação de pilha com lista

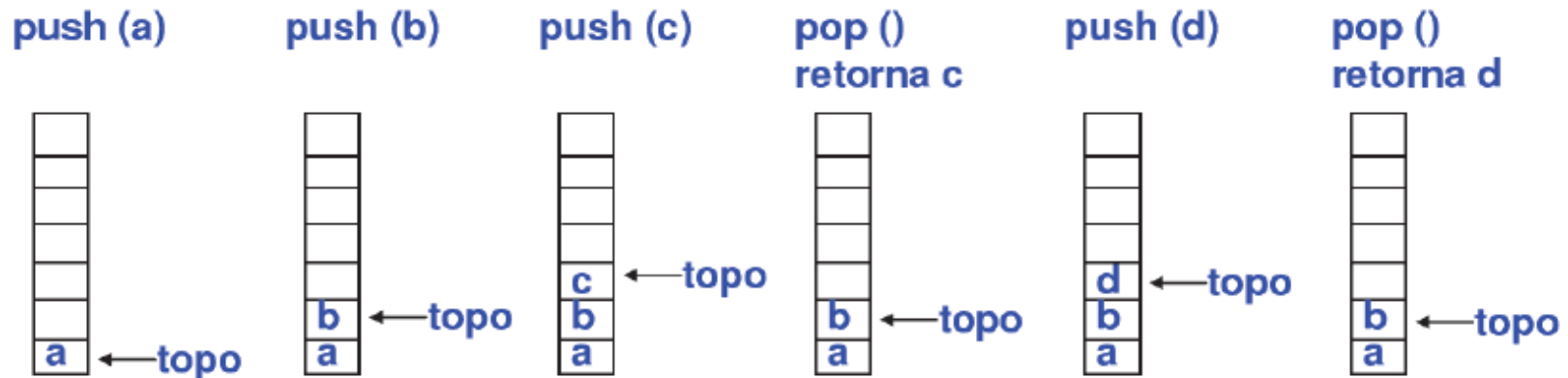
```
1  public class PilhaLista implements Pilha{
2      private NodoPilha prim;
3
4      public Pilha cria() {
5          this.prim = null;
6          return this;
7      }
8
9      public boolean vazia() {
10         return (this.prim == null);
11     }
12
13     public void libera() {
14         this.prim = null;
15     }
16 }
```

Implementação de pilha com lista (push e pop)

```
1
2     public void push(float v) {
3         NodoPilha novo = new NodoPilha();
4         novo.setInfo(v);
5         novo.setProx(prim);
6         prim = novo;
7     }
8
9     public float pop() {
10        NodoPilha n;
11        if(vazia()){
12            System.out.println("Pilha vazia");
13            System.exit(1);
14        }
15        n = this.prim;
16        prim = n.getProx();
17        return n.getInfo();
18    }
19 }
```

Resumo - Pilha

- **push**: insere novo elemento no topo da pilha
- **pop**: remove o elemento do topo da pilha



Material de **consulta**

- Capítulo 11 do livro: “Introdução a Estruturas de Dados” do Waldemar Celes, Renato Cerqueira e José Lucas Rangel.

Material de referência

- Capítulo 11 do livro: “Introdução a Estruturas de Dados” do Waldemar Celes, Renato Cerqueira e José Lucas Rangel.
- Imagens retiradas do site da disciplina de Programação II da PUC do Rio de Janeiro.
<http://www.inf.puc-rio.br/inf1007/>.