| INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA – CAMPUS CAMPINA GRANDE | | | |
|---|---|---|---|
| INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PARAÍBA Campus Campina Grande | CURSO: | CURSO DE ENGENHARIA DA COMPUTAÇÃO | |
| | PERÍODO: | | TURMA: |
| | DISCIPLINA: | PROGRAMAÇÃO E ESTRUTURAS DE DADOS | |
| | PROFESSOR: | CÉSAR ROCHA VASCONCELOS | SEMESTRE LETIVO |

# Prática Laboratório – Pilhas Encadeadas

1) Utilizando técnicas de modularização de sistemas vistas em sala, implemente o tipo abstrato de dados **TPilhaEnc** (de implementação encadeada e tipo base inteiros). Implemente este tipo numa biblioteca chamada **pilhaenc.h** Esta biblioteca deve seguir o mesmo padrão da biblioteca *pilhaseq.h* desenvolvida no exercício de Pilhas Sequenciais. Ou seja, contendo uma interface bem definida (apenas definição de tipos e protótipos de funções) em um arquivo com extensão.h . A biblioteca deve possuir as todas as funções básicas vistas em sala.

    a. **Lembre-se:** em cada uma das operações, identifique possíveis situações de erros do usuário e exiba mensagens para ele nestas situações. (Ex. o programa deve exibir mensagens no caso do usuário tentar desempilhar um item numa pilha que está vazia etc.). Isto é um sinal de maturidade na programação e torna seu TAD mais robusto (menos suscetível a travamentos).

2) Modifique o menu de operações do editor da pilha sequencial da prática anterior. Ou seja, reutilize o seu código para geração de um novo menu de operações contendo todas as novas operações deste TAD na questão acima.

3) Uma vez que as operações básicas foram implementadas na questão 1, insira agora na biblioteca **pilhaenc** as seguintes **novas operações** (crie novos protótipos! Não use funções existentes):

    a. Uma rotina para apenas mostrar o sub-topo da pilha: `int subtopo(TPilhaEnc p, int *valor)`

    b. Uma rotina para desempilhar N elementos a partir do topo: `void desempilhaN (TPilhaEnc *p, int n )`

    c. Uma rotina para esvaziar a pilha (desempilhando todos elementos): `void esvaziar(TPilhaEnc *p)`

    d. Uma rotina especial para consultar (sem remover) a base da pilha: `int base(TPilhaEnc p)`

4) Depois, insira novas opções ao menu principal da questão anterior. Ele deve ficar como o da figura abaixo:

```
Editor de Pilha v2.0
================
1-  Inicializar
2-  Empilhar
3-  Desempilhar
4-  Elemento do topo
5-  Imprimir pilha
6-  Esvaziar a pilha
7-  Base da pilha
8-  Tamanho da pilha
9-  Desempilha N elementos
10- Mostrar sub-topo

Digite sua opção: [   ]
```

5) Implemente duas funções na sua biblioteca: uma primeira função que receba duas pilhas encadeadas como parâmetro e empilhe todos os elementos da pilha 2 para a pilha 1. Ao final desta função, a pilha2 deve ficar vazia e pilha1 com todos os elementos das duas pilhas. A lógica de funcionamento é mostrada na figura à esquerda abaixo. O protótipo da função: `void concatena_pilhas(TPilhaEnc *p1, TPilhaEnc *p2)`. A segunda função deve receber uma pilha como parâmetros e retirar todos os números ímpares. A lógica de funcionamento desta segunda função é mostrada na figura à direita abaixo. O protótipo: `void retira_impares(TPilhaEnc *pilha)`