

## Lista de TAD Pilha

- 1) Implemente o TAD Pilha na forma de um *wrapper* para uma lista sequencial estática (ver dica abaixo)
- 2) Implemente o TAD Pilha na forma de um *wrapper* para uma lista encadeada dinâmica (ver dica abaixo)

Dica para os exercícios 1 e 2

- Crie um arquivo .h com a definição da struct aluno (aluno.h)
- Altere seus códigos de listas de forma a retirar a definição da struct aluno do arquivo .h
- Inclua no .h de suas listas um comando `#include aluno.h`
- Inclua no .h de sua pilha um comando `#include aluno.h`
- No arquivo .c da pilha faça um `include` do .h da lista. Não o inclua no .h pois dessa forma as funções da lista ficariam visíveis no main.
- No main faça o `include` do .h da pilha e do .h do aluno

- 3) Aplicação: parênteses e colchetes

Considere o problema de decidir se uma dada sequência de parênteses e colchetes está bem-formada (ou seja, parênteses e colchetes são fechados na ordem inversa àquela em que foram abertos). Por exemplo, a sequência

`(( )) [ ( ) ]`

está bem-formada, enquanto `( [ ] )` está malformada. Suponha que a sequência de parênteses e colchetes está armazenada em uma string ASCII `s`. (Como é hábito em C, o último caractere da string é `\0`.)

Esse problema foi retirado do link abaixo, que também contém uma algoritmo com a solução

Fonte: <https://www.ime.usp.br/~pf/algoritmos/aulas/pilha.html>

- 4) Implemente o TAD Pilha utilizando alocação dinâmica. Dica: muito código será semelhante ao de lista encadeada. Não coloque a definição da struct aluno no .h da pilha.
- 5) Implemente o algoritmo que converte uma expressão matemática infix para postfix. Teste com os exemplos apresentados em sala.

Utilizar o algoritmo apresentado em

<https://www.ime.usp.br/~pf/algoritmos/aulas/pilha.html>