SCC0502 - Algoritmos e estruturas de dados I

Prof. Dr. Bruno Matheus
Estágiário PAE: Rafael Nakanishi

Lista de Exercícios
Pilhas

Exercícios teóricos

1. Explique em poucas palavras o que são *stack overflow* e *stack underflow*.
2. Explique como um programa utiliza, internamente, uma estrutura de pilhas.
3. Considerando a estrutura de pilhas e seus métodos (`push()`; `pop()`; `top()`), explique como você faria para retornar o segundo elemento do topo da pilha, sem modificá-la.
4. **Expressões aritméticas.** Considere a expressão

   $$(((5 + 4)/3) * (2 + 7)).$$

   Levando em consideração que as expressões mais internas aos parênteses são realizadas primeiro, escreva uma sequência de comandos utilizando a estrutura de pilhas (`push` e `pop`) para resolver essa expressão.

Exercícios práticos

1. Implemente o algoritmo de conversão de decimal para binário, visto em aula.
2. Implemente uma função que retorna uma cópia de uma pilha dada como entrada.
3. **Inversão de strings.** Dada uma sequência de caracteres, implemente uma função que inverta essa sequência.
4. **Parênteses balanceados.** Dois parênteses são considerados balanceados se um "abre parênteses"está ao lado esquerdo de um "fecha parênteses". Implemente uma função que verifica se uma sequência de parênteses está balanceada.

Bônus: existem três tipos de parênteses: `"()"`, `"[]"`, `"{}"`. Nesse caso, os parênteses são considerados balanceados se o "abre parêntese"está à esquerda de um "fecha parêntese"do mesmo tipo. Modifique a função anterior para checar se a sequência de parênteses está balanceada.

5. **Palíndromos.** Uma palavra é um palíndromo se a sequência de caracteres que a compõe pode ser lida da esquerda para a direita e vice-versa (Exemplo: *arara*, *hanah*). Escreva uma função que, dada uma palavra, retorne `true` caso a palavra seja um palíndromo, e `false` caso contrário. Utilize a estrutura de dados *pilha*.

6. Implemente uma função para avaliar expressões aritméticas, como as apresentadas no item 4 dos exercícios teóricos.

7. **Expressões pós-fixas.** Comumente, as expressões aritméticas são escritas usando *notação infixa*: `argumento1 operador argumento2`. Um problema dessa notação são as regras de prioridade e parênteses para indicar a ordem das operações: `5 + 7 * 3 = 26` e `(5 + 7) * 3 = 36`.

   Existe outro meio de escrever tais expressões ariméticas, denomidado *pós-fixo*: `argumento1 argumento2 operador`. Nessa notação, não se faz necessário o uso de parênteses: `5 7 3 * + = 26` e `5 7 + 3 * = 36`.

   Implemente uma função que, dado uma expressão aritmética pós-fixa, avalie-a e retorne o valor correto da expressão.