Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Bacharelado em Engenharia de Automação e Controle

MATA40 - Estrutura de Dados | Período: 2015.2 | Data: 31/03/2016.
Prof. Antonio L. Apolinário Junior
Estagiário Docente: Alan Santos

Roteiro do Laboratório 4 - Pilhas

Objetivos:

- Compreender de forma prática o conceito de pilha, suas operações e aplicação;
- Implementar, em linguagem C, um TAD Pilha, baseado em arranjo dinâmico;
- Aplicar o TAD Pilha em um exemplo pratico.

Conceitos básicos:

Pilha:

Como visto em sala de aula, pilhas são estruturas de dados lineares e sequências que possuem um política de acesso bem definida: ***LIFO - Last In, First Out***. Portanto, sua implementação pode ser feita utilizando qualquer estrutura de dados linear e sequencial, como listas encadeadas ou arranjos. Nesse laboratório utilizaremos um arranjo dinâmico para definir o TAD Pilha, tal que:

```
typedef struct {   char *pilha;       // arranjo dinamico
                   int maxElems;      // numero maximo de elementos no arranjo
                   int topo;          // indice do topo da pilha
                   } tPilha;
```

Roteiro:

1. Baixe do repositório da disciplina (http://homes.dcc.ufba.br/~apolinario/Disciplinas/2015.2/MATA40/) os códigos fonte base para esse Laboratório.

2. Analise a estrutura de arquivos que compõe esse Laboratório..

3. Compile os programas executando na linha do console o comando **`make`**.

4. Não havendo erros, execute o programa **`testePilha`**. Verifique que, tal como nos outros roteiros de laboratório, as funções relativas ao TAD **`tPilha`** não estão todas

implementadas, mas garantem um valor de retorno que permite que o programa principal seja executado.

5. Abra o arquivo **`pilha.c`** e analise as funções a serem implementadas.

6. Codifique a função **`initPilha()`** para que ela, crie uma pilha com **`n`** posições.

7. Implemente na sequencia as funções **`pilhaEVazi()`** e **`PilhaECheia()`**. Compile e teste suas rotinas usando o programa de **`testePilha`**.

8. Codifique as funções **`empilha()`** e **`desempilha()`**. Compile e teste suas rotinas usando o programa de **`testePilha`**;

9. A essa altura seu TAD Pilha já esta funcional. Agora é a hora de testa-lo para resolver um problema prático. Construa um novo programa de teste, chamado **`validaExpressão`** que pede ao usuário que digite uma expressão matemática com parênteses, colchetes e chaves. A seguir seu programa deve ser capaz de validar a expressão, considerando que o numero e a posição relativa dos parênteses, colchetes e chaves estão corretas. Como discutido em sala de aula, esse problema pode ser resolvido utilizando-se uma estrutura de pilha.

10. Teste seu programa com vários tipos de expressões, com erros e corretas para verificar se seu funcionamento esta ok.

Para ler a expressão voce deve utilizar um vetor de caracteres para armazenar a sequencia de elementos da expressão. Esse vetor é tratado pela linguagem C como uma string e possui uma biblioteca de funções, string.h, suportada pela linguagem C. Para mais detalhes consulte:

http://www.ime.usp.br/~pf/algoritmos/aulas/string.html

ou

http://www.cplusplus.com/reference/cstring/