Lista de exercícios de programação em linguagem C: Pilhas

**Parte 1- Questões Teóricas**

**Exercício 1 - Conceito de Pilha**

Explique o que é uma pilha em Estrutura de Dados. Dê dois exemplos do cotidiano em que o comportamento de uma pilha pode ser observado.

Uma pilha é uma estrutura de dados do tipo LIFO (*Last In, First Out*), ou seja, o último elemento a entrar é o primeiro a sair.  
Podemos imaginar uma pilha como uma pilha de pratos: você sempre coloca (empilha) um novo prato por cima e só pode retirar (desempilhar) o prato que está no topo.

Exemplos do cotidiano:

1- Pilha de pratos em um restaurante — o último prato colocado em cima é o primeiro a ser retirado.  
2- Histórico de páginas no navegador — a última página acessada é a primeira a ser removida ao clicar em “voltar”.

**Exercício 2 - Estruturas Estática X Dinâmica**

a) Diferencie pilha **estática** e **pilha dinâmica**

Estática - Implementada com vetor (array) de tamanho fixo definido na compilação.

Dinâmica - Implementada com lista encadeada, crescendo ou diminuindo conforme a necessidade durante a execução.

b) Cite vantagens e desvantagens de cada abordagem.

Estática   
- Vantagem - Implementação simples e acesso rápido aos elementos.  
- Desvantagem - Tamanho fixo — pode desperdiçar memória ou não comportar todos os elementos necessários.

Dinâmica

- Vantagem – Usa memória de forma flexível, sem limite fixo.

- Desvantagem - Acesso mais lento e maior uso de ponteiros, tornando o código mais complexo.

**Exercício 3 -** Considere uma estrutura de dados em uma tabela denominada TAB. A posição 1 armazena a próxima posição livre da tabela. Em dado o momento, a tabela apresenta os dados preenchidos, conforme consta abaixo.

**tabela TAB**

[ 7; 25; 12; 21; 18; 23; - ; - ; - ; - ]

1 2 3 4 5 6 7 8 9 10

A inserção do número 15 na tabela, resultou no exposto abaixo.

tabela TAB

[ 8; 15; 25; 12; 21; 18 ;23; - ; - ; - ]

1 2 3 4 5 6 7 8 9 10

Ao serem retirados dois elementos, a tabela apresenta a seguinte configuração.

tabela TAB

[ 6; 12; 21; 18; 23; - ; - ; - ; - ; - ]

1 2 3 4 5 6 7 8 9 10

Tal Estrutura é uma?   
a) fila com início na posição 2   
b) fila com início na posição TAB(1)  
c) fila com fim na posição 2  
d) pilha com topo na posição TAB(1)   
e) pilha com topo na posição 2  
  
**Exercício 4 – Operações Fundamentais**

Liste e explique as operações básicas de uma pilha

* Push - Insere (empilha) um novo elemento no topo da pilha. Se for pilha estática, verifica antes se não está cheia.
* Pop - Remove (desempilha) o elemento do topo da pilha. Se estiver vazia, a operação não é permitida.
* Topo (ou peek) - Retorna o valor do elemento do topo, sem removê-lo da pilha.
* isEmpty - Verifica se a pilha está vazia (sem elementos). Retorna verdadeiro ou falso.
* isFull - Verifica se a pilha está cheia (no caso da pilha estática com tamanho fixo).

**Parte 2 – Implementações Básicas**

**Exercicio 5 – Pilha Estática**

Implemente uma pilha estática utilizando vetor em C. O programa deve com ter as funções básicas (push, pop, topo, isEmpty, isFull) e um menu para o usuário interagir.

#include <stdio.h>

#include <stdlib.h>

#define TAM 5

struct Pilha {

int dados[TAM];

int topo;

};

void inicializar(struct Pilha \*p) {

p->topo = -1;

}

int isEmpty(struct Pilha \*p) {

return (p->topo == -1);

}

int isFull(struct Pilha \*p) {

return (p->topo == TAM - 1);

}

void push(struct Pilha \*p, int valor) {

if (isFull(p)) {

printf("A pilha está cheia! Não é possível empilhar.\n");

} else {

p->topo++;

p->dados[p->topo] = valor;

printf("Valor %d empilhado com sucesso!\n", valor);

}

}

void pop(struct Pilha \*p) {

if (isEmpty(p)) {

printf("A pilha está vazia! Nada para desempilhar.\n");

} else {

int removido = p->dados[p->topo];

p->topo--;

printf("Valor %d removido da pilha.\n", removido);

}

}

void topo(struct Pilha \*p) {

if (isEmpty(p)) {

printf("A pilha está vazia!\n");

} else {

printf("Topo da pilha: %d\n", p->dados[p->topo]);

}

}

void mostrar(struct Pilha \*p) {

if (isEmpty(p)) {

printf("A pilha está vazia!\n");

} else {

printf("Elementos da pilha:\n");

for (int i = p->topo; i >= 0; i--) {

printf("| %d |\n", p->dados[i]);

}

}

}

int main() {

struct Pilha p;

inicializar(&p);

int opcao, valor;

do {

printf("\n=== MENU PILHA ===\n");

printf("1 - Empilhar (push)\n");

printf("2 - Desempilhar (pop)\n");

printf("3 - Mostrar topo\n");

printf("4 - Mostrar pilha\n");

printf("5 - Sair\n");

printf("Escolha uma opcao: ");

scanf("%d", &opcao);

switch (opcao) {

case 1:

printf("Digite um valor: ");

scanf("%d", &valor);

push(&p, valor);

break;

case 2:

pop(&p);

break;

case 3:

topo(&p);

break;

case 4:

mostrar(&p);

break;

case 5:

printf("Saindo do programa...\n");

break;

default:

printf("Opcao invalida!\n");

}

} while (opcao != 5);

return 0;

}