**ESTRUTURAS DE DADOS**

**Exercícios de Fixação: Utilizando Pilhas**

1. Ao receber duas Pilhas, retorne a concatenação das mesmas na primeira Pilha.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | Pilha 1 |  | Pilha 2 |  | Resultante |
|  |  |  |  |  | Pilha 1 |
|  |  |  |  |  |  | |
|  |  |  |  | Topo ->8 | 23 | |
|  |  |  |  | 7 | 41 | |
|  |  |  |  | 6 | 56 | |
|  |  |  |  | 5 | 74 | |
|  |  | Topo ->4 | 23 | 4 | 28 | |
| Topo ->3 | 45 | 3 | 41 | 3 | 45 | |
| 2 | 32 | 2 | 56 | 2 | 32 | |
| 1 | 87 | 1 | 74 | 1 | 87 | |
| 0 | 34 | 0 | 28 | 0 | 34 | |

1. Faça uma função que possibilite a exclusão de um elemento qualquer da Pilha escolhido pelo Usuário.

*Entrada:*

Elemento=> 87

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Saída |
|  |  |  |  | Resultante |
|  | Pilha |  |  | Pilha |
|  |  |  |  |  | |
| Topo ->8 | 23 |  | 8 |  | |
| 7 | 41 |  | Topo ->7 | 23 | |
| 6 | 56 |  | 6 | 41 | |
| 5 | 74 |  | 5 | 56 | |
| 4 | 28 |  | 4 | 74 | |
| 3 | 45 |  | 3 | 28 | |
| 2 | 32 |  | 2 | 45 | |
| 1 | 87 |  | 1 | 32 | |
| 0 | 34 |  | 0 | 34 | |

1. Implemente uma função que retorne o resultado de qualquer expressão pós-fixada (Polonesa) recebida por parâmetro em uma string. Considerando que os operandos terão apenas 1 dígito. A expressão abaixo na forma infixa é a seguinte: (1+2)\*(4-5).

*Entrada (na forma pós fixada):*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | + | 4 | 5 | - | \* | ‘\0’ |

#### Saída: -3

Exemplos para teste:

|  |  |
| --- | --- |
| **Pós-fixada** | **Infixada** |
| 5 7 \* 8 + | 5\*7+8 |
| 3 4 9 + \* | 3\*(4+9) |
| 8 2 + 6 \* | (8+2)\*6 |
| 4 1 + 7 4 + \* | (4+1)\*(7+4) |

1. Simule a utilização de ‘n’ Pilhas conforme a necessidade do Usuário. As operações de inserção (PUSH) e retirada (POP), e em qual Pilha deverão agir, devem ser definidas por sorteio.

*Sugestão: utilizar um vetor de Pilhas.*

1. Elaborar um módulo em C que ao receber um vetor de números inteiros e seu tamanho lógico, exiba-o de modo que seus elementos apareçam de forma inversa, conforme o exemplo:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 456 | 123 | 543 | 997 | 890 | 547 | 854 | Lixo | Lixo | Lixo | |
| 1 | 2 | 3 | 4 | 5 | 6 | TL=7 | 8 | 9 | 10 |

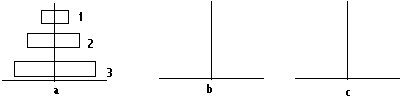
A saída produzida pelo programa seria a seguinte:

854 547 890 997 543 123 456

1. Faça um módulo que receba, via parâmetro, um arquivo texto e retorne um novo arquivo com o conteúdo totalmente invertido. Utilize uma Pilha como estrutura para auxílio ao processo.
2. Escreva uma função que ao receber uma palavra, retorne se a mesma é palíndromo ou não.

Por exemplo: Entrada  **ARARA**

Saída  **ARARA 🡪 SÃO PALÍNDROMOS!**

1. Elabore um algoritmo que mova três discos de uma Torre de Hanói, que consiste em três hastes (a – b – c), uma das quais serve de suporte para, por exemplo, 3 discos de tamanhos diferentes (1 –2 – 3), os menores sobre os maiores. Pode-se mover um disco de cada vez para qualquer haste contanto que nunca seja colocado um disco maior sobre o menor. O objetivo é transferir os ‘n’ discos para outra haste de forma não recursiva.
2. Escrever uma função que tome ‘n’ palavras de entrada (cadeia de caracteres) e as escreva na ordem reversa. Ex.: se a chamada PalavrasOrdemReversa(char Frase[100]) fosse executada e as palavras digitadas fossem:

### O elefante bebe Água

A saída produzida pelo programa seria a seguinte:

## Água bebe elefante O