

## TT05 – Sophia Carrazza



APÓS CONCLUIR ESTA SESSÃO DE ESTUDO, RESPONDA:

As pilhas e as filas são estruturas de dados que armazenam coleções de itens. Considere uma pilha P1 e duas filas F1 e F2 e suponha a seguinte situação. Alguns elementos são inseridos em F1 e, em seguida, retirados de F1 e inseridos em P1. Depois, eles são retirados de P1 e inseridos em F2. Finalmente, eles são retirados de F2 e impressos na tela. Sabendo que essa situação utiliza a sequência de estruturas "Fila - Pilha - Fila", responda qual das sequências de estruturas abaixo pode ser usada para imprimir a mesma saída na tela.

Fila - Fila - Fila.

✓ Pilha - Pilha - Pilha.

O enunciado coloca uma sequência resultante de inserções e remoções em uma "Fila - Pilha - Fila". Nesse caso, a primeira fila mantém a ordem de inserção e a pilha inverte a ordem. A segunda fila recebe os elementos na ordem inversa e os mantém na ordem inversa.

A alternativa usa três pilhas. Logo, P1 inverte a ordem, P2 volta para a ordem inicial e P3 inverte a ordem novamente. Isso é a mesma coisa que o apresentado no enunciado.

Pilha - Pilha - Fila.

Pilha - Fila - Pilha.

Fila - Pilha - Pilha

Parabéns!



### Exercício de criação da class Lista

```
class Lista {
    int array[], n;
    Lista(){
        array = new int[6];
        n=0;
    }
    Lista(int tamanho){
        array = new int[tamanho];
        n=0;
    }
    void InserirInicio(int x) {
        if (n>= array.Length){
            Environment.Exit(0); //como se fosse um break
        }
        for (int i=n; i>0; i--){ //leva os elementos pro fim do array
            array[i] = array[i-1];
        }
        array[0] = x;
        n++;
    }
    void InserirFim(int x) {
        if (n>= array.Length){
```

```

        Environment.Exit(0);
    }
    array[n] = x;
    n++;
}
void Inserir(int x, int pos) {
    if (n >= array.Length || pos < 0 || pos > n){
        Environment.Exit(0);
    }
    for (int i=n; i>0; i--){
        array[i] = array[i-1];
    }
    array[pos]=x;
    n++;
}
void RemoverInicio(int x) {
    if (n == 0){
        Environment.Exit(0);
    }
    int novo = array[0];
    n--;
    for (int i=0; i<n; i++){ //leva os elementos pro inicio do array
        array[i] = array[i+1]; }
    return novo;
}

void RemoverFim(int x) {
    if (n==0){
        Environment.Exit(0);
    }
    return array[--n]; //retorna o array ja com menos um espaco (tirou o ultimo)
}
void Inserir(int pos) { //nao precisa inserir o x, pois ele ja esta la, so precisa informar sua posicao p retiralo

    if (n==0 || pos < 0 || pos >= n){ //se pos=n, vao ter n-1 elementos disponiveis
        Environment.Exit(0); }
    int novo = array[pos];
    n--;

    for (int i=pos; i<n; i++){
        array[i] = array[i+1];
    }
    return novo; }
void Mostrar(){
    Console.WriteLine("[");
    for (int i=0; i<n; i++){
        Console.WriteLine(array[i] + " ");
    }
    Console.WriteLine("]");
}
}

```

### Questões teóricas da Sessão de Lista Linear

**q1-** O nove não foi removido, mas não faz parte mais do array logicamente, logo, fica na memória como lixo.

**q2-** Um valor é removido logicamente quando ele é inativo (não mais considerado como parte do analisado) de forma lógica, mas não eliminado fisicamente da estrutura de dados. Enquanto isso, na remoção física, há uma remoção permanente que libera o espaço de memória na estrutura.

**q3-** A formatação lógica rápida cria uma estrutura nova sem verificar os setores físicos em busca de erros, o que não garante a remoção dos dados antigos (lixo). Já a formatação física, realiza o preparo dos setores de armazenamento e os "conserta" para funcionar o novo sistema de arquivos, o que é mais demorado mas mais seguro.

**q4-** Quando um arquivo é inserido na lixeira, não é excluído permanentemente, mas sim funciona como uma pasta temporária para os arquivos que são excluídos, mas ainda ocupa espaço no disco. Enquanto isso, quando um arquivo é deletado permanentemente, o SO marca os espaços alocados pelo arquivo como disponíveis novamente para reuso. Ou seja, os dados não são apagados completamente no disco, mas são considerados vazios e prontos para serem sobrescritos com novos dados.

**q5-** Será mostrado na tela: "[3 7 1]"

IV) O método inserir abaixo mantém a ordem crescente dos elementos.

```
void Inserir(int x) {  
    if(n <= vet.length){  
        vet[n++] = x;  
    }  
}
```

E correto apenas o que se afirma em

✓ I, III e IV.

A afirmação I é verdadeira porque o método apresentado apenas efetua a remoção no início que, consequentemente, desloca os elementos restantes para a esquerda do **array**.

A afirmação II é falsa porque o método apresentado é o inserir no início que insere o novo elemento na primeira posição independente desse valor ser menor que os existentes no **array**.

A afirmação III é verdadeira porque o método apresentado apenas efetua a remoção no fim.

A afirmação IV é verdadeira porque o método apresentado é o inserir no fim que insere o novo elemento na última posição independente desse valor ser maior que os existentes no **array**.

I e III.

II, III e IV.

II e IV.

I e II.

Parabéns!

