DCC013 - Estrutura de Dados / DCC107 - Laboratório de Programação II

Lista contígua

Você pode utilizar qualquer ambiente de programação para desenvolver sua atividade. Ao final, **copie e cole o seu código-fonte com a resposta aqui mesmo neste documento**, dentro dos espaços indicados para isso e **preservando a identação do código**. **Depois que terminar sua avaliação, não se esqueça de entregar sua atividade!** Fique atento ao relógio, pois as atividades entregues com atraso não serão aceitas.

Para resolver esta atividade, **clique aqui para baixar o projeto da aula 6 de laboratório de programação 2, que contém a implementação do TAD ListaCont para lista contígua de números inteiros.** Na sua solução para a questão abaixo, você pode utilizar/chamar qualquer uma das operações que estejam disponíveis no projeto (exatamente do jeito que ele se encontra no site da disciplina). Outras operações que você eventualmente deseje utilizar devem ser copiadas para sua resposta neste documento.

Implementar a operação `void ListaCont::removeMultiplos(int val);` que remove todas as ocorrências de valores múltiplos de `val` de uma lista contígua. Para isso, deve ser usado um vetor auxiliar para que não seja necessário ficar "puxando" elemento a elemento. Se a lista não possuir nenhum valor múltiplo de `val`, a mesma não deve ser alterada. Não é necessário alterar o valor de `max`. A operação deve percorrer a lista somente uma vez.

Exemplo: Considere uma lista com os valores `[3, 2, 5, 4, 9]` e `val = 3`. A lista resultante é `[2, 5, 4]`.

```
void ListaCont::removeMultiplos(int val)
{
    int *vetAux = new int[max]; //Vetor auxiliar
    int j = 0;
    bool foiAlterado = false;
    int numRemocoes = 0; // Número de remoções para diminuir em n

    for (int i = 0; i < n; i++)
    {
          if (get(i) % val != 0) //Se o resto da divisão por val
for diferente 0, mantemos
        {
            foiAlterado = true;
            vetAux[j] = get(i);
```

```
            j++;
        }
        else
        {
            numRemocoes++; //Se o resto for igual de 0, removemos
        }
    }

      if (foiAlterado) //Se ele foi alterado em algum momento,
apontamos vet para vetAux e limpamos a memória
    {
        delete[] vet;
        vet = vetAux;
        n = n - numRemocoes;
    }
}
```