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

Lista encadeada

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 7 de laboratório de programação 2, que contém a implementação do TAD ListaEncad para lista encadeada 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). Quaisquer outras operações necessárias para a sua resposta devem ser implementadas e incluídas neste documento.

Implementar a operação `void ListaEncad::retrocedeUltimo(int n1);` para retroceder o ponteiro `ultimo` de uma **lista simplesmente encadeada com descritor** (LSED) `n1` nós em direção ao início (à esquerda). O parâmetro `n1` tem que estar entre `0` e `n`, emitir a mensagem "Valor invalido" caso `n1` não esteja nesse intervalo. Observar que o valor de `n` tem que ser alterado para ficar coerente com o novo número de nós da lista uma vez que os `n1` nós finais deverão ser liberados ao se retroceder o ponteiro `ultimo` `n1` nós da LSED. Essa operação tem que ter complexidade `O(n)`.

Exemplo: Considere a lista simplesmente encadeada com descritor `L` com os valores `L = [45 2 35 6 19]`:

- O comando `L.retrocedeUltimo(3)` torna `L = [45 2]`.
- O comando `L.retrocedeUltimo(5)` torna `L = []` (lista vazia).

```
void ListaEncad::retrocedeUltimo(int n1) {
        if (n1 >= 0 && n1 <= n) {
            No *p = primeiro, *ant = NULL, *novoUltimo = primeiro;
            for (int i = 1; p != NULL; i++) {
                ant = p;
                p = p->getProx();
                if (i < n - n1)
                    novoUltimo = p;
                else if (i > n - n1)
                    delete ant;
            }
            if (n1 == n) {
                primeiro = NULL;
```

```
            ultimo = NULL;
            n = 0;
        }
        else {
            ultimo = novoUltimo;
            novoUltimo->setProx(NULL);
            n = n - n1;
        }
    }
    else
        cout << "Valor invalido" << endl;
}
```