INF 213 - Listas duplamente encadeadas e iteradores simples

→ LEMBREM-SE DE USAR PAPEL E CANETA COMO RASCUNHO ANTES DE IMPLEMENTAR <<--

Arquivos fonte e diagramas utilizados nesta aula:
https://drive.google.com/file/d/1umoJZziHtLfawYm2IrfKviPlCujKq_7O/view?usp=sharing

Etapa 1

O arquivo MyList2.h contém a implementação parcial da lista por contiguidade vista em sala de aula. Porém, parte do código foi removido. Sua tarefa consiste em completar o codigo e testar a classe utilizando o programa TestaMyList2.cpp .

Utilize um Debugger de memória (http://valgrind.org/docs/manual/quick-start.html) para verificar se sua implementação não contem erros de memória.

O programa de testes não compilara caso você não implemente todas etapas. Para testar cada etapa individualmente (antes de passar para a proxima), recomendo que comente o código que testas funções que serao implementadas apenas nas proximas.

Etapa 2

Implemente a funcao size() sem adicionar um novo membro de dados a sua classe, sem utilizar a variavel dataFirst/dataSize e sem usar qualquer outra funcao-membro da classe MyList2.

Dica: use recursividade.

Etapa 3

Adicione uma função chamada eraseMatchingElements a sua classe (essa função não está disponível nas listas disponibilizados pela STL). Tal função deverá receber um argumento e remover da lista todos elementos iguais a esse argumento. Ao final deve-se retornar quantos elementos foram removidos (a capacidade do vetor não deve ser alterada).

Por exemplo, se o MyList2 v armazena caracteres e v=[a,b,c,b,w,z], então após a chamada v.eraseMatchingElements('b') o valor de v deverá ser [a,c,w,z] e a chamada da função deverá ter retornado o número 2 (visto que dois caracteres 'b' foram removidos de v).

Como apagar elementos é uma operação eficiente em listas duplamente encadeadas, você pode utilizar a função erase se quiser.

Etapa 4

Implemente uma função-membro chamada reverse(), que inverte a ordem dos elementos na lista. Para praticar o uso de apontadores, **NÃO** utilize iteradores ou outras funções já providas pela classe MyList2.

Dica: use recursividade!

Etapa 5

Implemente uma funcao membro publica chamada compare(it1,it2), que dados dois iteradores (representados por ponteiros para nodos nessa classe) (it1 e it2) para a lista duplamente encadeada, retorna true se it1 apontar para uma posicao anterior a posicao apontada por it2 e false caso contrario. Suponha que ambos iteradores apontem para nodos realmente presentes na lista.

Essa função deverá ser RECURSIVA e não deverá utilizar nenhum outro método da classe MyList2 (ou seja, você não poderá utilizar métodos que simplificam o uso de iteradores).

Submissao da aula pratica:

A solucao deve ser submetida utilizando o sistema submitty (submitty.dpi.ufv.br). Atualmente a submissao so pode ser realizada dentro da rede da UFV.