# INF 213 - Roteiro da Aula Pratica 9

O objetivo desta aula e' praticar o uso e implementacao de iteradores.

Arquivos fonte e diagramas utilizados nesta aula:
https://drive.google.com/open?id=1phKGL2OZDnDPXIypdeBBRwi4N_tsLkc3

Ao final do arquivo TestaLab9.cpp ha a saida esperada para todas as etapas.

## Etapa 1

Utilizando iteradores para a nossa classe de listas duplamente encadeadas, crie uma funcao chamada "reverse". Tal funcao devera receber como argumento uma lista (disponivel no arquivo MyList2NewIterator.h) e inverter a ordem dos elementos nela. Essa operacao devera ser realizada utilizando iteradores e as funcoes begin() e end() da nossa lista.

A funcao reverse devera ser implementada no arquivo TestaLab9.cpp.

Dicas: um desafio e' conseguir um iterador para o ultimo elemento da lista.

## Etapa 2

Na nossa implementacao de iteradores para listas duplamente encadeadas (arquivo MyList2NewIterator.h), e' possivel incrementar e decrementar os iteradores. Porem, como o iterador representando o final da lista (end()) possui internamente um ponteiro apontando para NULL, não e' possivel decrementar esse iterador.

Desenhe um diagrama de lista duplamente encadeada em uma folha de papel para entender melhor o motivo de não ser possivel decrementar tal iterador.

Em C++ (na STL), por outro lado, o iterador para listas suporta a operacao de decremento mesmo se ele estiver apontando para o final da lista.

Modifique nossa implementacao de iterador (disponivel no arquivo MyList2NewIterator.h) para que o comportamento seja similar ao dos iteradores da STL.

Descomente a parte de TestaLab9.cpp referente a Etapa 2 para testar sua nova implementacao.

Obs:
- seu metodo deve funcionar mesmo se a lista for modificada após a criacao do iterador (conforme testado em TestaLab9.cpp)! Essa etapa exige um pouco de criatividade (se precisar de ajuda, não exite em postar no Piazza ou perguntar ao professor durante a aula pratica).
- faca sempre diagramas para entender melhor o comportamento das estruturas de dados!

- a possibilidade de decrementar o end() poderia ter facilitado muito a implementacao da Etapa 1 !

**Etapa 3**

Crie uma classe chamada Range, que representa um intervalo de inteiros. Crie tambem um iterador para essa classe. Mais especificamente, esse tipo de iterador não sera utilizado para percorrer uma estrutura de dados, mas sim para percorrer o intervalo representado pela classe.

Alem dos métodos para retornar o iterador (begin() e end()), o unico metodo publico necessario nessa classe sera' seu construtor, que devera esperar dois inteiros representando o intervalo (fechado).

Por simplicidade, assuma que o limite inferior do intervalo sera sempre menor ou igual ao limite superior.

Exemplo de uso:

```
Range r(1,5); //cria um intervalo [1,5]
Range::iterator it = r.begin();
while(it!=r.end()) {
  cout << *it << " "; //imprime: 1 2 3 4 5
}

//outro exemplo, utilizando a sintaxe do C++11 (pode ser necessario ativar o suporte ao C++11 no seu compilador)
// leia: "para cada inteiro i no intervalo [1,5], imprima i"
for(int i:Range(1,5)) cout << i << " " ; //imprime: 1 2 3 4 5
```

Obs: algumas linguagens de programacao (exemplo: Python) possuem um tipo "range", que e' muito utilizado para criar loops iterando por determinado intervalo (no caso de Python, o intervalo e' semi-aberto).

**Submissao da aula pratica:**

A solucao deve ser submetida ate as 18 horas da proxima Segunda-Feira utilizando o sistema submitty (submitty.dpi.ufv.br). Atualmente a submissao so pode ser realizada dentro da rede da UFV.