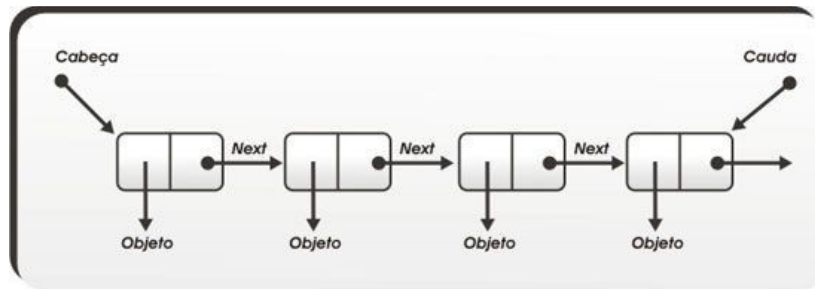


# DIM0120 - 2023.1 – Projeto 1

## Introdução

Uma lista ligada (ou encadeada) simples é uma estrutura de dados linear, formada por células (ou nós) que são compostas por dados (ou objeto) do mesmo tipo e um ponteiro (endereço) para o próximo elemento. O ponteiro para a primeira célula da lista é conhecido como cabeça (*head*) da lista; é comum representar esse ponteiro como uma célula cujo conteúdo (objeto) é ignorado e cujo endereço indica a primeira célula com conteúdo da lista. A última célula da lista (cauda ou *tail*) possui como campo de endereço um ponteiro para *Null*, indicando o final da lista.



Fonte: Wikipédia

Listas ligadas são estruturas de dados bastante utilizadas e serão tema recorrente de diversas disciplinas do curso. Antes de iniciar a atividade, faça uma revisão sobre lista ligada.

## Tarefa

Como parte do time de desenvolvimento de um software para reprodução de músicas, você foi solicitado a criar um programa para gerenciar uma sequência de músicas (*playlist*). Neste cenário, faça as seguintes atividades:

- 1) Modele uma classe capaz de representar uma musica do sistema: musicas são representadas pelo seu título e o nome do artista;
  - a) Crie os construtores e destrutores correspondentes à classe
- 2) Modele uma classe capaz de representar uma lista ligada, onde cada elemento armazena um objeto do tipo Música.
  - a) Crie métodos de inserção e remoção de elementos na lista
  - b) Crie métodos de busca e acesso à elementos na lista
  - c) Crie os construtores e destrutores correspondentes
- 3) Modele uma classe capaz de representar uma playlist que contém músicas do sistema: playlists são representadas por um nome apenas.

- a) Crie os construtores e destrutores correspondentes à classe
- b) Crie métodos de adição e remoção de músicas à uma playlist (músicas devem ser armazenadas em uma lista ligada)
- c) Crie métodos de retornar a próxima música a ser tocada
  - i) Para cada chamada deste método a música retornada deve ser a próxima da lista de músicas na playlist (começando pela primeira, depois a segunda e assim por diante). Caso não haja mais músicas para ser tocadas, o método deve retornar NULL.
- d) Crie um método, **usando recursão**, para imprimir todas as músicas que formam a *playlist*.

4) Implemente uma forma de gerenciar músicas do sistema (adicionar, remover e listar), músicas cadastradas devem ser armazenadas usando uma lista ligada.

- a) Caso uma música seja removida, ela também deve ser removida de qualquer playlist que ela tenha sido adicionada.

5) Implemente uma forma de gerenciar playlists do sistema (adicionar, remover e listar), playlists podem ser armazenadas no formato de arrays ou usando outra lista ligada(precisa ser implementada).

6) Implemente uma forma de gerenciar músicas em uma playlist (adicionar, remover, mover) Para

a sua solução, você deve, obrigatoriamente:

- Utilizar modularização nas classes e no main: cada classe deve ser um arquivo separado do main. Modularize funções se achar necessário.
- Toda a compilação deve ser feita utilizando-se makefiles, utilize o cmake para .

**Importante:** não é permitido utilizar bibliotecas da STL que implementem listas ligadas ou outras estruturas de dados genéricas.

## Autoria e política de colaboração

Esta atividade pode ser feita no máximo em dupla. A cooperação entre estudantes da turma é estimulada, sendo aceitável a discussão de ideias e estratégias. Contudo, tal interação não deve ser entendida como permissão para utilização de (parte de) código fonte de outros colegas, o que pode caracterizar situação de plágio. Trabalhos copiados em todo ou em parte de outros colegas ou da Internet serão rejeitados.

## Entrega

Você deverá submeter um único arquivo compactado no formato .zip contendo, de forma organizada, todos os códigos fonte e o arquivo makefile, sem erros de compilação e documentados, através da opção *Tarefas* na Turma Virtual do SIGAA. Seu arquivo compactado deverá incluir também um arquivo texto README contendo: a identificação completa do grupo; a descrição de como compilar e rodar o programa, **incluindo um roteiro (exemplo) de entradas e comandos que destaquem as funcionalidades**; a descrição das limitações (caso existam) do programa e quaisquer dificuldades encontradas.

## Orientações gerais

- 1) Utilize estritamente recursos da linguagem C++.
- 2) Aplique boas práticas de programação. Codifique o programa de maneira legível (com indentação de código fonte, nomes consistentes, etc.). **Modularize** e comente seu código.

## Avaliação

O trabalho será avaliado sob os seguintes critérios: (i) utilização correta e abrangente dos conteúdos vistos na disciplina; (ii) a corretude da execução do programa implementado, que deve apresentar saída em conformidade com a especificação e as entradas de dados fornecidas, e; (iii) a aplicação correta de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte. A presença de mensagens de aviso (*warnings*) ou de erros de compilação e/ou de execução, a modularização inapropriada e a ausência de documentação são faltas que serão penalizadas; (iv) apresentação do trabalho em sala de aula para o professor.