UNIVERSIDADE FEDERAL DE LAVRAS

CIÊNCIA DA COMPUTAÇÃO (BACHARELADO)

INTRODUÇÃO AOS ALGORÍTMOS

JOÃO PEDRO TEODORO DE ABREU

KARIM SOARES LENTZ

PAULO SÉRGIO MENDES TACIANO

AS MÚSICAS MAIS TOCADAS DO SPOTIFY

Projeto Prático

LAVRAS

2025

# INTRODUÇÃO

O desenvolvimento deste projeto prático tem como foco a criação de um sistema de cadastro com armazenamento de dados em arquivos binários, utilizando a linguagem de programação C++.

O tema escolhido pelo grupo foi “As músicas mais tocadas do Spotify”, por ser um assunto popular e de fácil identificação. A base de dados criada inclui os seguintes campos: Artista (string com espaço), nome da música (string com espaço), ano de lançamento (inteiro), gênero musical (string), número de streams em bilhões (ponto flutuante double) e volume médio em decibéis (ponto flutuante double).

O sistema tem o objetivo de reforçar conceitos da disciplina, com funcionalidades como importação de dados a partir de um arquivo CSV, inserção e remoção lógica de registros, ordenação por diferentes campos e busca binária. Além disso, o programa possibilita a exportação dos dados atualizados novamente para um arquivo CSV, garantindo portabilidade e organização.

Em síntese, o projeto não apenas cumpre os objetivos acadêmicos propostos, mas também oferece uma aplicação prática e interativa, capaz de manipular e apresentar informações de forma eficiente.

# DESCRIÇÃO DE ALTO NÍVEL

O

# ORDEM DOS DADOS ARMAZENADOS NO ARQUIVO

O arquivo de entrada utilizado no projeto é o “spotify100.csv”, que possui os seguintes campos: artista (string com espaço), nome da música (string com espaço), ano de lançamento (inteiro), gênero musical (string), número de streams em bilhões (double) e volume médio em decibéis (double), e que originalmente armazena as músicas em ordem decrescente de streams, ou seja, as músicas mais reproduzidas aparecem primeiro.

Durante a execução do programa, o vetor de objetos com as músicas é carregado mantendo essa ordem inicial. Ao longo das funcionalidades, o usuário pode realizar diferentes operações: buscas, ordenações temporárias em memória (por artista, por nome da música ou por ano), e adicionar ou apagar músicas.

No momento de salvar as alterações, o programa permite que o usuário escolha em que ordem os dados devem ser gravados de volta no arquivo original. As opções disponíveis são:

* Ordem padrão (decrescente por streams);
* Ordem alfabética por artista;
* Ordem alfabética por nome da música;
* Ordem crescente por ano de lançamento.

Após a escolha, o programa reordena os registros apenas no momento da gravação, e sobrescreve o arquivo “spotify100.csv” com os dados na nova sequência escolhida.

Dessa forma, centralizamos o armazenamento em um único arquivo, mantendo a consistência das informações, mas permitindo ao usuário definir a forma como os dados serão organizados ao final do processo.

# DIFICULDADES E SOLUÇÕES AO LONGO DA DO DESENVOLVIMENTO DO TRABALHO

Durante o desenvolvimento do projeto, algumas dificuldades e soluções para elas surgiram em uma ordem:

Ao analisar nosso arquivo CSV original percebemos alguns campos incompatíveis além de não ser fiel com dados atualizados das músicas mais ouvidas na plataforma Spotify. O primeiro desafio foi gerar um novo arquivo CSV com os campos atualizados, o que exigiu pesquisar individualmente cada campo e reescrever o arquivo do zero. Depois, foi necessário adaptar o código inicial ensinado em sala para funcionar com o novo formato do arquivo.

Ao tentar ordenar as músicas por ano, identificou-se que, sempre que o vetor era reordenado, ele perdia a ordem original por streams. A solução foi manter o vetor de músicas sempre na ordem original e usar vetores auxiliares de índices para fazer ordenações específicas. Na busca binária por ano, foi preciso ordenar previamente os anos.

Ao lidar com strings, descobriu-se que a comparação string1 < string2 faz a ordenação alfabética. Durante a implementação da busca binária por música, o código inicialmente não considerava músicas com nomes repetidos, e a busca exigia que o usuário digitasse exatamente o mesmo nome que estava no arquivo.

O menu do programa travava em loop infinito quando o usuário digitava uma string inválida ao invés de um número. O problema foi corrigido mudando a variável de controle para string, evitando falha de leitura.

Depois, foi implementada a função de adicionar músicas, o que trouxe um novo problema: as músicas adicionadas eram impressas com espaçamento maior que as demais. Esse erro foi causado por um arquivo.ignore() extra na leitura, o qual foi removido para normalizar a formatação.

Outro ponto importante foi garantir que o programa não soubesse o tamanho fixo do arquivo. Para isso, foi implementado um redimensionamento dinâmico, começando com 40 músicas e aumentando de 10 em 10, sempre que o limite era atingido.

Na busca por artista, percebeu-se que um cantor pode ter várias músicas, exigindo ajuste na lógica para buscar e listar todas as músicas de um mesmo artista.

Houve também um problema em que o programa não reconhecia as duas últimas músicas do arquivo. Para resolução, foi necessário converter todos os nomes de artistas e músicas para minúsculas antes das buscas, para evitar problemas com maiúsculas/minúsculas. Também foi implementada uma funcionalidade que, ao adicionar uma música com o mesmo nome de outra já existente, ambas aparecem na impressão.

Situação atual: O programa já permite leitura, busca, adição e impressão.

Ainda falta: implementar salvamento em arquivo tipado, permitir exportação com ordenações diferentes, salvar as alterações feitas no vetor e padronizar a tabulação final no arquivo de saída.

# CONCLUSÕES

É