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

1 INTRODUÇÃO

O desenvolvimento deste projeto prático tem como foco a criação de um sistema de cadastro com armazenamento de dados em arquivos binários e tipados, 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 ou binário, 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 ou binário, garantindo portabilidade e organização.

2 DESCRIÇÃO DE ALTO NÍVEL

Inclusão de Bibliotecas

As bibliotecas fornecem recursos para:

- Entrada e saída de dados (iostream)
- Manipulação de arquivos (fstream)
- Manipulação de strings (string, cstring)

Estrutura Musicas

A estrutura Musicas armazena os dados de uma música, incluindo:

- Nome do artista
- Nome da música
- Duração em milissegundos
- Gênero musical
- Ano de lançamento
- Quantidade de streams em bilhões
- Volume médio da música
- bool apagada - indica se a música foi apagada logicamente

Métodos da estrutura:

- void imprimir - Exibe as informações da música no console, se ela não estiver marcada como apagada.
- void leitura - Lê os dados de uma linha formatada do arquivo e preenche a estrutura. Ignora separadores e converte strings para char[].

Função toLowerCustom

Converte uma string para letras minúsculas manualmente. Isso facilita comparações case-insensitive (ignorando maiúsculas/minúsculas).

Função apagarMusica

Recebe o nome da música digitado pelo usuário, percorre o vetor e marca como apagada (sem excluir fisicamente) todas as músicas que coincidirem com esse nome.

Função ordenaVetorStrings

Ordena um vetor de strings em ordem alfabética e ajusta o vetor auxiliar que contém os índices das músicas no vetor original. É usada para ordenação estável sem alterar a posição real no vetor principal.

Função exportarDados

Permite ao usuário exportar para um arquivo externo os dados de todas as músicas não apagadas. Solicita o nome do arquivo, escreve os campos com formatação legível.

Função buscarPorArtista

Solicita o nome de um artista, percorre o vetor e exibe todas as músicas (não apagadas) que correspondam ao nome informado.

Função buscarPorAno

Recebe um ano do usuário e exibe todas as músicas (não apagadas) cujo ano de lançamento coincida com o informado.

Função buscarPorNome

Procura no vetor músicas que tenham o nome informado, exibindo as que não foram apagadas.

Função visualizarMusicas

Percorre e chama imprimir() para todas as músicas não apagadas, exibindo os dados ao usuário.

Função adicionarMusica

Adiciona uma nova música, pedindo os dados ao usuário (nome, artista, ano, gênero, etc.) e armazenando-os no vetor na posição numMsc.

Função redimensionarVetor

Caso o número de músicas ultrapasse a capacidade do vetor, esta função: Cria um vetor novo com o dobro do tamanho, copia os dados antigos para o novo vetor, libera a memória anterior e atualiza a capacidade

Função main()

A função principal organiza toda a lógica do programa.

Inicialização:

- Define capacidade inicial e contador de músicas;
- Tenta abrir arquivo binário para carregar dados existentes;
- Leitura do Arquivo (se existir);
- Lê objetos do arquivo binário até o fim, preenchendo o vetor.

Menu de opções interativas:

- Visualizar todas as músicas;
- Adicionar nova música;
- Apagar música;
- Buscar por nome, artista ou ano;
- Exportar músicas para arquivo;
- Sair e salvar.

Laço de interação com o usuário:

- Executa as funções correspondentes de acordo com a escolha;
- Redimensiona o vetor automaticamente se necessário;
- Salvamento dos dados ao final: Ao sair, grava todas as músicas (inclusive apagadas) em arquivo binário, garantindo persistência entre execuções.

3 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;
- 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.

4 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, identificamos 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, descobrimos 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.

Incorporamos 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, implementamos um redimensionamento dinâmico, começando com 40 músicas e aumentando de 10 em 10, sempre que o limite era atingido.

Na busca por artista, percebemos e 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.

Adicionamos a “função.peek()” da biblioteca “fstream” que ajuda a conferir a próxima linha do arquivo de leitura sem mudar o ponteiro de leitura.

Por fim, analisamos que há conflito na execução do programa entre o sistema operacional Windows e Linux. Quando executado em Windows o programa corta a primeira letra dos artistas e músicas na impressão e salvamento das informações, além de não imprimir palavras com acentuação corretamente. Para resolver o problema foi necessário retirar um “arquivo.ignore” da função de leitura, porém em Linux o programa funciona normalmente.

5 CONCLUSÕES

Neste projeto, conseguimos criar um sistema em C++ que organiza e gerencia informações sobre as músicas mais tocadas do Spotify, aplicando de forma prática os conceitos de algoritmos que aprendemos. O programa lê e trata os dados com cuidado, realiza buscas rápidas, ordena as informações conforme a necessidade, e permite até “excluir” músicas sem apagá-las de verdade — tudo isso de maneira flexível, mesmo quando a quantidade de dados varia.

Além disso, conseguimos exportar os resultados em diferentes formatos, facilitando o uso futuro. Durante o desenvolvimento, tivemos a oportunidade de lidar com desafios reais, como trabalhar com arquivos e strings, e entender as particularidades de diferentes sistemas operacionais.

No final, o projeto não só alcançou os objetivos técnicos, mas também trouxe um aprendizado valioso sobre organização, lógica e programação, resultando em uma aplicação útil, clara e de fácil uso.