Trabalho de Introdução aos Algoritmos - 10A

Alunos:
Isabel Valadares Pessoa
Luís Eduardo Ramos Spinola
Pablo dos Santos Martins

1. Introdução

O desenvolvimento do sistema de gerenciamento de músicas teve como objetivo permitir a leitura, organização, busca, edição e gravação de um banco de dados contendo informações extraídas da plataforma YouTube. Desde o início, o grupo optou por padronizar todas as músicas com base em dados dessa plataforma, para garantir a consistência nos campos de duração, ano de publicação e número de visualizações.

Para isso, foi criada a struct **Musica**, que centraliza todos os dados referentes a cada registro, além de um conjunto de funções primárias e secundárias responsáveis pela leitura do arquivo, buscas, ordenações, validações e manipulação dinâmica do banco.

2. Estrutura da Struct Musica

A struct central contém sete membros, cada um destinado a armazenar uma informação específica da música:

```
string nome;
string artista;
string duracao;
int ano;
long long views;
double media_views;
string descricao;
```

Cada variável foi escolhida de acordo com as necessidades observadas durante o desenvolvimento:

2.1 Nome e artista (string)

Utilizados para armazenar o título da música e o nome do artista.
Ambos são strings devido à variedade de caracteres, acentos e tamanhos.

2.2 Duração (string)

Decidiu-se usar string em vez de valores numéricos porque:

- O formato do YouTube segue o padrão `M:SS` ou `MM:SS`
- Músicas longas podem aparecer como `H:MM:SS`
- O uso de string evita cálculos desnecessários e facilita a leitura direta do CSV

2.3 Ano (int)

Armazena o ano de lançamento no YouTube.
Passou por validações para impedir valores incompatíveis.

2.4 Views (long long)

Inicialmente o grupo utilizou um int, mas após observar que a música *Hello*, com mais de 3 bilhões de visualizações, causava erros, concluiu-se:

- int (32 bits) suporta no máximo 2.147.483.647
- Músicas populares podem ultrapassar esse limite
- O programa estava lendo incorretamente todas as linhas seguintes devido ao overflow

Solução adotada:

- Uso de long long, que suporta valores extremamente maiores
- Alternativa considerada: unsigned int

2.5 media_views (double)

A média de visualizações por ano foi calculada dividindo:

```
views / (2025 - ano_lancamento)
```

Para evitar médias absurdas ou divisão por zero:

- Se a música tem menos de um ano de publicação -> recebe seus próprios views como média.

2.6 descricao (string)

Armazena a parte favorita da letra da música.
Foi decidido:

- Quebras de linha substituídas por `;`
- Para músicas instrumentais: `"  "` (aspas com espaço)

3. Funções Primárias

As funções primárias são chamadas diretamente na função principal e representam a parte central da lógica do programa.

3.1 Leitura_csv()

Responsável por:

- Abrir o arquivo CSV
- Ler cada linha respeitando o formato com aspas
- Traduzir cada campo em sua posição correta dentro da struct

Principais decisões:

- `getline()` com aspas delimitando strings
- Operador `>>` para variáveis numéricas
- Uso de uma variável `lixo` para descartar vírgulas e aspas
- Redimensionamento automático do vetor quando a capacidade se torna insuficiente

Essa função foi essencial para identificar o erro causado pelo tipo int ao ler números muito grandes.

3.2 Exibe_banco_de_dados()

Exibe um intervalo escolhido pelo usuário, convertendo corretamente:

- Posições iniciadas em 1 (usuário)
- Índices iniciados em 0 (computador)

Utiliza:

- `fixed` e `setprecision(4)` para formatar média
- `imprime_formatado()` para seguir o padrão visual

3.3 BuscaPorNome(), BuscaPorArtista() e BuscaPorParte()

Essas funções compartilham a mesma lógica principal:

Normalização do texto com:

```
Minusculo()
Retira_acentos()
```

Uso de getline() para permitir buscas com espaços
Uso de Achar_posicao() para encontrar trechos
Exibição formatada dos resultados

O grupo trabalhou para garantir que:

A busca fosse flexível
Não dependesse de caixa alta/baixa
Ignorasse acentos
Exibisse corretamente múltiplos resultados

3.4 Ordenações

Três funções foram criadas:

```
Ordem_alfabetica_nome()
Ordem_visualizacoes()
Ordem_media_visualizacoes()
```

Todas usando insertion sort, escolhido por:

Facilidade de implementação
Bom desempenho para vetores pequenos/médios
Comportamento estável e previsível

A ordenação altera diretamente o vetor original, pois ele é passado por referência.

3.5 Salvar()

Reescreve todo o vetor em um CSV, permitindo salvar:

Ordenações
Edições
Adições e exclusões

Foi utilizado:

- `ofstream`
- formatação padronizada com aspas para strings
- `fixed` e `setprecision()` para média

4. Funções Secundárias

As funções secundárias dão suporte às funções principais.

4.1 Retira_acentos()

Remove acentos utilizando um mapa (std::map) com pares "acentuado" -> "normalizado".

4.2 Redimensionamento()

Aumenta a capacidade do vetor em +5 posições, copiando todos os elementos anteriores.

4.3 Imprime_formatado()

Uma das funções mais complexas para o grupo.
Objetivo:

- Evitar linhas longas
- Evitar quebra de palavras
- Respeitar limite de 47 caracteres por linha

Lógica principal:

- Encontrar ultimo espaço antes do limite
- Usar substr para impressão
- Recuo com " " no início de cada linha

4.4 Minusculo()

Converte caracteres ASCII para minúsculos adicionando 32.

4.5 Achar_posicao()

Simplifica o uso de find(), retornando verdadeiro quando um trecho existe dentro de um texto.

---

5. Novas Funções Desenvolvidas (Parte adicional)

5.1 Apaga_musica()

Função destinada a excluir uma música do banco de dados.

Passos principais:

1. Busca a posição da música com `Encontra_musica_para_apagar()`
2. Exibe um menu de confirmação (`Menu()`)
3. Caso o usuário confirme:

    Cria um novo vetor (`nova_playlist`)
    Copia todos os elementos, exceto o removido
    Deleta o vetor original
    Atualiza o ponteiro
    Atualiza o tamanho
4. Caso não haja música correspondente:
   Exibe erro com `Mensagem_de_erro(501)`

Essa função demonstra o domínio do grupo sobre:

Manipulação de ponteiros
Memória dinâmica
Tratamento de entrada do usuário

5.2 Validações

`Valida_ano()`

Confere se o ano está entre 0 e 2025.

`Valida_duracao()`

Uma das validações mais complexas:

Verifica se existe `:`
Impede caracteres não numéricos
Aceita formatos:

`M:SS`
`MM:SS`
`H:MM:SS`

Valida limites:

Horas ≤ 23
Minutos ≤ 59
Segundos ≤ 59

Conta segmentos separadamente (antes do `:`, entre `:`, depois do `:`)

Valida_visualizacoes()

Aceita apenas valores entre 0 e o máximo suportado por long long.

5.3 Adicionar_nova_musica()

Permite inserir uma nova música no banco.

Etapas principais:

1. Verifica se o vetor precisa ser redimensionado.
2. Garante que não existam músicas duplicadas:

   Mesma música e artista
   Comparação sem acentos e sem diferenciação de maiúsculas/minúsculas

3. Valida:

   Ano
   Duração (formato)
   Visualizações

4. Calcula a média de visualizações

5. Lê e grava a descrição
6. Atualiza o número total de músicas

A função demonstra o cuidado do grupo em evitar:

Dados inválidos
Registros repetidos
Falhas de leitura do usuário
Cálculos incorretos