Relatório de Implementação do Programa de Gerenciamento de Registros

Gabriel Ramos, Vinicius Macedo

Fevereiro 2025

1 Introdução

Este relatório descreve as estratégias e funcionalidades do programa desenvolvido para gerenciar registros de livros. O programa é capaz de criar arquivos binários a partir de arquivos CSV, construir índices binários, realizar buscas por ID e consultas baseadas em palavras-chave. As funcionalidades principais incluem a criação de registros binários, a construção de uma árvore binária de índices, e a busca de registros utilizando um índice invertido.

2 Estrutura do Programa

O programa é composto pelos seguintes arquivos principais:

- `main.cpp`: Contém a função principal que coordena a execução do programa.
- `Buffer.h` e `Buffer.cpp`: Define a classe `Buffer` responsável por criar registros binários a partir de um arquivo CSV.
- `Indice.h` e `Indice.cpp`: Define a classe `Indice` responsável por criar e gerenciar índices binários e realizar buscas.
- `Registro.h` e `Registro.cpp`: Define a classe `Registro` que representa um registro de livro.
- `BinaryTree.h` e `BinaryTree.cpp`: Define a classe `BinaryTree` que implementa uma árvore binária para armazenar os índices.
- `Makefile`: Define as regras de compilação e execução do programa.

3 Funcionalidades

3.1 Criação de Registros Binários

A classe `Buffer` é responsável por ler um arquivo CSV contendo informações sobre livros e criar um arquivo binário com esses registros. A função `criaRegistrosBin` realiza essa tarefa.

3.2 Criação de Índices Binários

A classe `Indice` é responsável por criar um arquivo de índices binários a partir do arquivo de registros binários. A função `criaIndicesBin` lê o arquivo de registros e escreve os índices (ID e posição) em um arquivo binário de índices.

3.3 Árvore Binária de Índices

A classe `BinaryTree` implementa uma árvore binária para armazenar os índices. A função `arvoreDeIndices` da classe `Indice` lê o arquivo de índices binários e insere os índices na árvore binária.

3.4 Busca por ID

A função `busca` da classe `BinaryTree` permite buscar um registro pelo ID na árvore binária. A função `registroPorPosicao` da classe `Registro` recupera o registro correspondente a partir do arquivo binário de registros.

3.5 Busca por Palavras-Chave

A classe `Indice` também implementa um índice invertido para permitir buscas por palavras-chave. A função `indiceInvertido` cria o índice invertido a partir do arquivo de registros binários. A função `buscarRegistros` permite realizar buscas por palavras-chave, retornando os IDs dos registros relevantes.

4 Estratégias de Implementação

4.1 Tratamento de Strings

Para realizar buscas por palavras-chave, as strings de entrada são tratadas para remover pontuações e stopwords, e converter para minúsculas. As constantes globais `STOPWORDS` e `PONTUACOES` são usadas para esse propósito.

4.2 Índice Invertido

O índice invertido é implementado usando um `map` que associa cada palavra relevante a um vetor de IDs de registros. A função `tratativaDeNome` é responsável por processar os nomes dos livros e atualizar o índice invertido.

4.3 Busca por Palavras-Chave

A função `buscarRegistros` realiza a busca por palavras-chave no índice invertido. Ela trata a string de consulta, busca as palavras tratadas no índice invertido e retorna os IDs dos registros que contêm todas as palavras da consulta.

5 Conclusão

O programa desenvolvido oferece uma solução eficiente para gerenciar registros de livros, permitindo a criação de arquivos binários, a construção de índices binários, e a realização de buscas por ID e palavras-chave. As estratégias de tratamento de strings e uso de índices invertidos garantem a precisão e eficiência das buscas.