Universidade Federal de Minas Gerais
Raul Nascimento Cunha

Documentação
Trabalho Prático

Belo Horizonte
Novembro de 2018

Introdução

Este documento resume as funcionalidades e soluções dadas ao problema de criação de uma máquina de busca com índice invertido, o índice invertido é uma estrutura contendo uma entrada para cada palavra (termo) que aparece em, pelo menos, um documento.

O problema estava separado em três partes, sendo elas a leitura de arquivos e tratamento da informação, estrutura de índice invertido e leitura simples da palavra a ser buscada e apresentação dos arquivos que contém a palavra buscada.

Desenvolvimento

Parte 1 – Leitura de arquivos

Iniciando a partir da ‘main’ e seguindo o caminho de dados: foi feita a implementação de uma função sem argumentos e que retorna o nome inserido pelo teclado em arquivo separado, nomeado ‘funcao_get_nome_arquivo.cpp’, para leitura do nome do arquivo inserido pelo usuário. Essa função somente imprime na tela a instrução e recebe o nome do arquivo em forma de ‘string’. A função ‘get_nome_arquivo’ se encontra dentro de um ‘loop’ ‘while’ e é seguida de um ‘if’ que tem o objetivo de pular para o fim do programa caso não seja inserido nenhum arquivo. Após o primeiro ‘if’ há outro que faz sair do ‘loop’ caso seja lido o valor de sentinela, definido como ‘FIM’. Caso não seja digitado FIM, o programa chama uma função nomeada ‘le_arquivo’ que recebe como parâmetros o nome do arquivo, o endereço para o ‘map’, para o ‘iterator’ do ‘map’, para o ‘set’ e para o ‘iterator’ do ‘set’ utilizados no armazenamento de arquivos.

A função ‘le_arquivo’ é a mais extensa, sua chamada está ainda presente dentro do primeiro ‘loop’ da função ‘main’. Nela foram usadas as bibliotecas ‘fstream’, ‘map’, ‘set’, ‘sstream’, ‘algorithm’ e ‘cctype’, além da biblioteca padrão ‘iostream’ e de uma classe criada para tratamento de erros de abertura de arquivos chamada de ‘openFileException’. Todo o código dessa função foi envolvido por um ‘try’, que testa se está tudo correto com a abertura do arquivo, e caso não esteja, lança um ‘throw’ com a exceção e nome do arquivo com erro, que é coletado por um ‘catch’ ao fim da função e imprime na tela o erro e o nome do arquivo incorreto. O programa continua após esse teste de abertura correta para um ‘if’, que testa se o arquivo está aberto para uso. A partir desse ponto, foram feitos tantos ‘whiles’ quantos foram os caracteres a serem removidos, tornando difícil a leitura dessa parte do código, porém foi a única forma encontrada de remover todos os caracteres previamente selecionados para remoção.

Parte 2 – Estrutura de dados do índice invertido

Ao fim do filtro de caracteres de ‘whiles’ da função ‘le_arquivo’, a palavra é toda convertida para minúsculo usando a função ‘transform’ presente na biblioteca ‘algorithm’ e também a função ‘tolower’ presente na biblioteca ‘cctype’ e adicionada ao ‘map’ com o nome do arquivo lido adicionado ao ‘set’. As estruturas de dados padrão ‘map’ e ‘set’ foram escolhidas para a implementação por armazenarem uma chave e um valor que são relacionados, e por armazenar um conjunto de informações, respectivamente, ambas guardando os dados já de forma ordenada, por padrão, em ordem crescente, reduzindo o tempo de busca posteriormente.

Parte 3 – Consultas simples

Para a consulta de palavras foi feito um outro ‘while’ externo ao primeiro, com a mesma hierarquia deste, que entra e continua no ‘loop’ enquanto o contador de arquivos implementado no primeiro ‘while’, com a intenção de garantir que pelo menos um arquivo seja inserido antes de continuar a execução do programa, seja maior que zero. Caso o contador tenha valor maior que zero, é impressa na tela a instrução para essa parte do programa, informando ao usuário que ele deve inserir o nome de uma palavra para realizar a busca ou ‘FIM’ para finalizar a busca.

A palavra é lida a partir da chamada de uma função chamada ‘get_palavra’, que, assim como a função ‘get_nome_arquivo’, informa ao usuário que ele insira a palavra e a recebe em forma de ‘string’, retornando-a ao fim da função para o programa que a chamou. Caso a palavra seja diferente de ‘FIM’, o programa prossegue para a próxima parte.

A partir da leitura da palavra a ser buscada, é feita a busca no ‘map’ usando a função ‘find’ da biblioteca ‘map’ que retorna o ‘iterator’ apontado para a posição daquela palavra ou apontando para o fim da estrutura, caso a palavra não se encontre no ‘map’, ou seja, caso não tenha sido lida dos arquivos inseridos no começo do programa. Há um ‘if’ para essa parte que testa se a palavra existe no ‘map’ e imprime para o usuário a mensagem de que a palavre não existe nos arquivos. Após buscar a palavra, é feita a iteração da estrutura de dados ‘set’ referente àquela palavra, e são impressos na tela os nomes dos arquivos nos quais aquela palavra se encontra.

Fim do programa principal.

A classe de tratamento de exceção possui apenas o construtor e uma função ‘what’, que retorna a mensagem definida no construtor da classe.

Parte 4 – Documentação e entrega

O desenvolvimento desse programa foi deito usando a IDE Codelite e suas versões controladas periodicamente usando o Git local e posteriormente, após resolução de alguns problemas com a chave ssh, o GitHub.

Parte 5 – Testes de unidades

Os testes de unidades foram conduzidos ao longo do desenvolvimento do programa, fazendo-se uso de funções ‘cout’, de inserção de parâmetros com resultados previsiveis nas funções para testar o bom funcionamento de todas elas e a capacidade de detectar erros nas entradas dos parâmetros em alguns casos.

Não foram utilizados quaisquer ‘frameworks’ de automação de testes por falta de compatibilidade com a IDE utilizada, e dada a limitação de ‘hardware’ do computador no qual foi desenvolvido o programa, não havia a possibilidade de instalar o Visual Studio, que segundo muitas fontes, seria a IDE mais apropriada para instalação e uso dos ‘frameworks’ de teste.