UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II

Máquina de Busca

Trabalho Prático

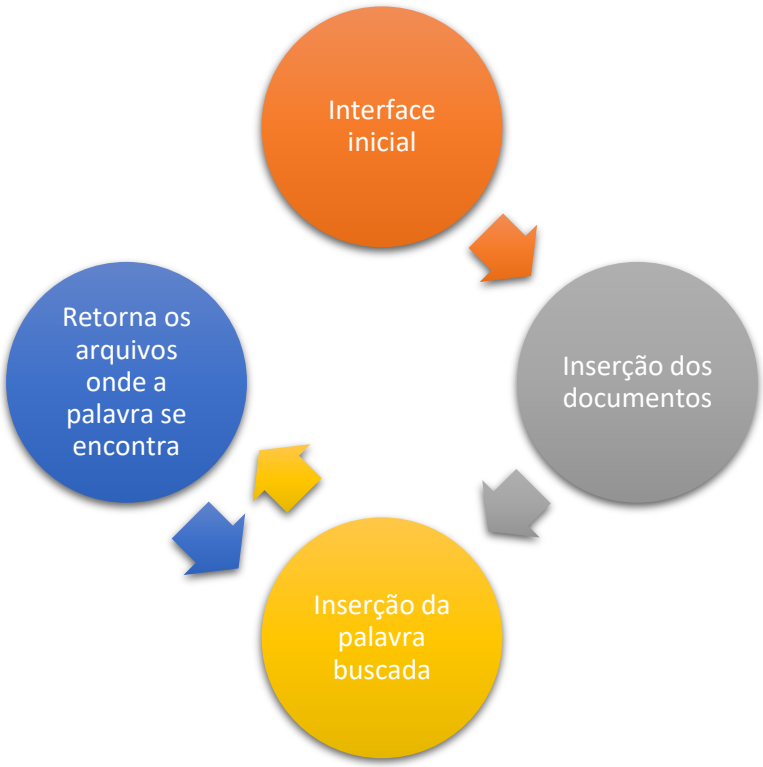
Alunos:

Hugo Ferreira Marques - *2018014573*

Thallys Felipe Gonçalves Barbosa – *2018080622*

1. Introdução

Nosso TP (Trabalho Prático) foi feito conforme as instruções passadas, uma máquina de busca com índice invertido onde o usuário insere os arquivos nos quais deseja efetuar a busca, após isso insere-se uma palavra e o programa retorna o nome dos documentos onde a palavra se encontra.



2. Funcionamento

Como não era um dos quesitos de avaliação, e não encontramos uma aplicação eficaz no contexto do software desenvolvido, optamos por não utilizar POO (Programação Orientada a Objetos), visto que não vislumbramos nenhum objeto real no qual pudesse ser aplicada e desenvolvida uma “identidade”, já que também se trata de um sistema relativamente pequeno.

Para o funcionamento, dividimos o código em 5 funções, sendo elas:

- void AbrirArquivo(set <string> &words,string filename);
- void SubstituirString(string &Palavra);
- void SubstituirSet(set <string> &words);
- void SetToMap(set <string> set, map <string,string> &map,string chave);
- void PrintMap(map <string,string> map,string chave);

E uma main(), onde continha a “interface” básica do programa e que chamava as funções acima.

1. int main(){...}

Função principal, responsável pela interface do programa por meio de “cout” e “cin”, onde foi declarada algumas variáveis globais para o funcionamento da máquina e onde também são chamadas as outras funções.

2. void AbrirArquivo(set <string> &words,string filename);

A função AbriArquivo tem como objetivo abrir um arquivo e passer todas as palavras dele para um set, para isso ela recebe um set e o nome do arquivo, previamente declarados na função main.

3. void SubstituirString(string &Palavra);

A função SubstituirString visa alterar as palavras que passarem por ela, removendo números, caracteres especiais e transformando todas as letras em minúsculas, para tal ela recebe uma string Palavra como parâmetro, e dentro dela realiza as substituições.

4. void SubstituirSet(set <string> &words);

A função SubstituirSet transforma o set constituído na função AbrirArquivo em um set com todas as palavras em letras minúsculas, sem caracteres especiais e sem números, para isso ela recebe o set a ser alterado como parâmetro e utiliza da função SubstituirString para realizar as transformações com cada palavra do Set.

5. void SetToMap(set <string> set, map <string,string> &map,string chave);

A função SetToMap tem como objetivo passer o set de palavras encontradas no arquivo para um container do tipo map, que atrela essa palavra ao nome do arquivo, para isso ela recebe o set com as palavras encontradas no arquivo, recebe o map que será preenchido e o nome do arquivo, ou seja, ela cria o índice invertido, atrelando cada palavra do arquivo ao nome do mesmo.

6. void PrintMap(map <string,string> map,string chave);

A função mais básica, simplesmente responsável por imprimir na tela do usuário os documentos (value) que contém a palavra (key) buscada.

Utilizamos de um while na função main, onde as 5 funções são executadas para cada arquivo criado, justificando a criação de variáveis globais (set e map), visto que as funções são executadas separadamente para cada arquivo e é necessário um armazenamento das palavras que transcenda um único arquivo.

3. Justificativas dos tipos e headers usados

Para o funcionamento de uma máquina de buscas é essencial o uso de contêineres, para isso utilizamos dois tipos, contêiner do tipo std::set e std::map.

O contêiner std::set é de fundamental importância porque ele armazena elementos únicos que seguem uma ordem específica, em um set o valor do elemento também o identifica. O fato de seguir uma ordem específica não tem importância no nosso contexto, mas o armazenamento de elementos únicos é fundamental para que não exista duplicidade nas palavras e consequentemente resultados de busca incorretos.

O contêiner std::map é a base do índice invertido, o map é um contêiner associativo que associa uma chave a um valor em uma ordem específica, de igual forma no set, a ordem não possui importância nesse caso, mas o armazenamento do tipo key value é importante no desenvolvimento do índice, onde a key se trata de cada palavra diferente contida no(s) arquivo(s) e o value é o valor, no caso, o nome dos documentos que o contém.

Utilizamos também de um header chamado <regex>, Regular Expressions, que tem como objetivo “padronizar padrões” a serem buscados em determinada sequência de caracteres. No nosso caso utilizamos o header para fazer a busca dos caracteres especiais e numéricos, por meio de um determinado padrão e de um método chamado regex_replace, ele encontrava o padrão e o removia.

Ademais fizemos uso apenas dos mais conhecidos, como por exemplo strings e fstream, que dispensam explicações.

4. Testes de Unidade

Para realizar os testes de unidade fizemos uso do framework CppUnit conforme recomendado pelo estagiário docente. O CppUnit roda testes em suites, e de forma simples exibe na tela se os testes passaram, ou caso tenha alguma falha, quantos passaram, quantos falharam e onde especificamente ocorreu a falha, facilitando assim o aprimoramento do código.

Nas linhas de código dos testes construímos várias assertivas para testar se cada uma das 5 funções listadas acima estava realizando sua atividade corretamente, de forma a forçar erros e verificar se até mesmo os tratamentos de exceções estavam devidamente feitos.

Visto que grande parte da função main é formada por “cout’” na construção da interface, consideramos apenas o arquivo .cpp, que basicamente possui toda a

parte realmente lógica e passível de corrupção, das funções, que possuem uma cobertura de 93% do código, a única função não testada é a que imprime na tela o nome dos arquivos porque o framework utilizado não oferece suporte para testar “cout”.

5. Conclusão

De uma maneira geral, procuramos alinhar o conteúdo aprendido em sala com boas práticas de programação, que vão desde a “identação” do código até uma tentativa de realizar o TDD (Test-Driven Development). Acreditamos ter atingido o objetivo do TP ao desenvolver a máquina de busca, com a certeza de que foi fundamental na fixação do conteúdo trabalhado durante o semestre.