UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE - UFRN

Instituto Metrópole Digital

IMD0029 – Estrutura de Dados Básicas I

IMD0030 – Linguagens de Programação I

Data de divulgação: 06/05/2016

Data de entrega: 07/06/2016

1. Descrição do trabalho:

O objetivo deste trabalho é avaliar, de forma conjunta, as habilidades desenvolvidas nas disciplinas IMD0029 e IMD0030. O trabalho consiste em implementar um programa para linha de comandos que permita realizar buscas no conteúdo de arquivos de texto. De um modo geral, o programa deverá receber como entrada, através de linha de comandos, palavras-chave e deverá buscar quais linhas dos arquivos contém aquelas palavras-chave. Ao término da sua execução, o programa deverá exibir no console as linhas que contém as palavras-chave. Ademais, o programa a ser implementado deverá ser capaz de registrar o tempo de execução de suas buscas e de controlar sua base de busca, isto é, deverá controlar sobre quais arquivos de texto serão realizadas as buscas. Neste contexto, o mesmo programa deverá funcionar em dois modos básicos distintos: (i) modo gerenciamento da base de buscas e (ii) modo de buscas. A seguir, estes modos são descritos em mais detalhes.

1.1. Modo gerenciamento da base de buscas

O programa deverá manter uma base de buscas, isto é, deverá manter um conjunto de arquivos de texto sobre os quais serão realizadas as buscas, devendo ser possível inserir e remover arquivos texto a esta base. O objetivo principal do modo de gerenciamento da base de buscas é otimizar a eficiência das buscas que serão realizadas. Para isto, será necessário pré-processar os arquivos mantidos na base de buscas de modo a manter uma estrutura de dados que apoie a realização de buscas eficientes. É item obrigatório neste trabalho o pré-processamento dos arquivos texto e a manutenção de uma estrutura de dados para manter a base de buscas já pré-processada.[1]

Para gerenciar a base de buscas, o programa deverá ser invocado da seguinte forma:

```
> ./buscaIMD <opção> <nome_arquivo>
```

Na sintaxe acima, o arquivo executável "`buscaIMD`" é invocado passando pela linha de comandos dois argumentos: `<opção>` e `<nome_arquivo>`. A seguir, estes argumentos são descritos:

[1] Uma dica para a implementação desta estrutura de dados é pesquisar por estratégias de indexação de dados (ex.: indexação invertida).

- `<opção>` : o primeiro argumento indica qual operação sobre a base de dados se deseja realizar. Devem ser disponibilizadas as seguintes opções:
    - `-i` : Insere um novo arquivo na base de buscas
    - `-r` : Remove um arquivo da base de buscas
    - `-li` : Lista todos os arquivos contidos na base de buscas na ordem em que foram inseridos à base
    - `-la` : Lista todos os arquivos contidos na base de buscas em ordem alfabética
    - `-lt` : Lista todos os arquivos contidos na base de buscas em ordem decrescente da quantidade de palavras em cada arquivo

- `<nome_arquivo>` : o segundo argumento indica um ou mais arquivos texto que se deseja inserir ou remover da base de buscas.

É importante ressaltar que apenas as opções `-i` e `-r` requerem o segundo argumento `<nome_arquivo>`; a opção `-l` não requer qualquer argumento extra. A seguir, são mostrados alguns exemplos de como o programa deverá ser invocado para gerenciar sua base de buscas.

Inserir arquivos na base de buscas

Exemplo 1: Inserindo só um arquivo por vez

```
> ./buscaIMD -i libertadores.txt
     >> Arquivo "libertadores.txt" inserido na base de buscas.
```

No exemplo acima, apenas um arquivo é inserido na base de buscas. O programa deverá verificar se o argumento passado pela linha de comando realmente aponta para um arquivo texto existente. Após a execução do programa, deverá ser exibida no console uma mensagem indicando se houve sucesso ou não na inserção do arquivo à base de dados.

Exemplo 2: Inserindo mais de um arquivo por vez

```
> ./buscaIMD -i libertadores.txt granja.txt
     >> Arquivo "libertadores.txt" inserido na base de buscas.
     >> Arquivo "granja.txt" inserido na base de buscas.
```

No exemplo acima, mais de um arquivo é inserido na base de buscas ao mesmo tempo. O programa deverá ser capaz de receber pela linha de comando um ou mais argumentos indicando quais arquivos se deseja inserir na base de dados. Similar ao exemplo anterior, deverá ser verificado se os argumentos passados realmente apontam para arquivos texto existentes e deverá ser exibida no console uma mensagem indicando se houve sucesso ou não na inserção do arquivo à base de dados.

Exemplo 3: Inserindo arquivo que já existe na base

```
> ./buscaIMD -i libertadores.txt granja.txt
     >> Arquivo "libertadores.txt" já estava na base de buscas.
     Seu registro foi atualizado.
     >> Arquivo "granja.txt" inserido na base de buscas.
```

No exemplo acima, é mostrado o caso em que se tenta inserir na base de dados um arquivo texto que já foi inserido anteriormente. O programa deverá ser capaz de identificar que um arquivo já foi inserido anteriormente e, neste caso, atualizar o seu registro.

Exemplo 4: Inserindo arquivo que não existe no sistema

```
> ./buscaIMD -i libertadores.txt granja.txt
     >> Arquivo "libertadores.txt" não foi encontrado no caminho
     encontrado.
     >> Arquivo "granja.txt" inserido na base de buscas.
```

No exemplo acima, é mostrado o caso em que se tenta inserir na base de dados um arquivo texto que não é encontrado no caminho passado como argumento ao programa. O programa deverá ser capaz de identificar se um arquivo de fato existe no sistema.

Remover arquivos da base de buscas

Exemplo 1: Removendo só um arquivo por vez

```
> ./buscaIMD -r libertadores.txt
     >> Arquivo "libertadores.txt" removido da base de buscas.
```

No exemplo acima, apenas um arquivo é removido da base de buscas. O programa deverá verificar se o argumento passado pela linha de comando realmente aponta para um arquivo que já foi inserido anteriormente na base de buscas. Após a execução do programa, deverá ser exibida no console uma mensagem indicando se houve sucesso ou não na remoção do arquivo da base de dados.

Exemplo 2: Removendo mais de um arquivo por vez

```
> ./buscaIMD -r libertadores.txt granja.txt
     >> Arquivo "libertadores.txt" removido da base de buscas.
     >> Arquivo "granja.txt" removido da base de buscas.
```

No exemplo acima, mais de um arquivo é removido da base de buscas ao mesmo tempo. O programa deverá ser capaz de receber pela linha de comando um ou mais argumentos indicando quais arquivos se deseja remover da base de dados. Similar ao exemplo anterior, deverá ser verificado se os argumentos passados realmente apontam para arquivos que já estão na base de buscas e deverá ser exibida no console uma mensagem indicando se houve sucesso ou não na inserção do arquivo à base de dados.

Exemplo 3: Removendo arquivo que não existe na base

```
> ./buscaIMD -r libertadores.txt granja.txt
     >> Arquivo "libertadores.txt" não estava na base de buscas.
     >> Arquivo "granja.txt" removido da base de buscas.
```

No exemplo acima, é mostrado o caso em que se tenta remover da base de dados um arquivo texto que não constava na base de buscas. O programa deverá ser capaz de identificar se um arquivo consta ou não na base de buscas.

Listar arquivos da base de buscas

Exemplo 1: Listando arquivos da base de buscas em ordem de inserção

```
> ./buscaIMD -li
     >> Arquivos contidos na base de buscas:
            -  "libertadores.txt"
            -  "granja.txt"
```

No exemplo acima, são listados todos os arquivos na ordem em que os arquivos foram inseridos.

Exemplo 2: Listando arquivos da base de buscas em ordem alfabética

```
> ./buscaIMD -la
     >> Arquivos contidos na base de buscas:
            -  "granja.txt"
            -  "libertadores.txt"
```

No exemplo acima, são listados todos os arquivos em ordem alfabética.

Exemplo 3: Listando arquivos da base de buscas em ordem decrescente de quantidade de palavras

```
> ./buscaIMD -lt
     >> Arquivos contidos na base de buscas:
            -  "granja.txt"
            -  "libertadores.txt"
```

No exemplo acima, são listados todos os arquivos em ordem decrescente da quantidade de palavras.

1.2. Modo de buscas

Após construída a base de buscas, o programa deverá ser capaz de realizar buscas por palavras-chave nesta base. Particularmente, o programa deverá buscar em todos os arquivos da base de buscas, exibindo em quais linhas de cada arquivo as palavras-chave ocorrem. A exibição dos resultados sempre deverá ser agrupada por arquivo, isto é, para cada arquivo na base de dados, deverão ser exibidas todas as linhas onde as palavras-chave ocorreram. Dos resultados encontrados num mesmo arquivo, as ocorrências devem sempre ser exibidas em ordem crescente das linhas. Mais adiante serão exibidos alguns exemplos para que fique mais claro como devem ser exibidos os resultados.

Para realizar as buscas, o programa deverá ser invocado da seguinte forma:

> ./buscaIMD <opção_busca> <opção_impressão> <tempo> <palavras_chave>

Na sintaxe acima, o arquivo executável "buscaIMD" é invocado passando pela linha de comandos quatro argumentos: <opção_busca>, <opção_impressão>, <tempo> e <palavras_chave>. A seguir, estes argumentos são descritos:

- <opção_busca> : o primeiro argumento indica de que forma a busca deve ser realizada. Devem ser disponibilizadas as seguintes opções:
    - -bAND : A busca é feita por linhas dos arquivos texto que contém todas as palavras-chave passadas como argumento de entrada
    - -bOR : A busca é feita por linhas dos arquivos texto que contém ao menos uma das palavras-chave passadas como argumento de entrada

- <opção_impressão> : o segundo argumento indica de que forma o resultado das buscas devem ser exibidos. Devem ser disponibilizadas as seguintes opções:
    - -pA : A impressão é feita exibindo os resultados em ordem alfabética do nome do arquivo.
    - -pC : A impressão é feita exibindo os resultados em ordem decrescente do número de vezes que as palavras-chave ocorreram em cada arquivo.
    - -pI : A impressão é feita exibindo os resultados na ordem em que cada arquivo foi inserido na base.

- <tempo> : o terceiro argumento indica se é necessário registrar o tempo de execução das buscas. Devem ser disponibilizadas as seguintes opções:
    - -tT : O programa deve registrar e exibir o tempo total de execução de da busca.
    - -tF : O programa não deve registrar nem exibir o tempo total de execução da busca
- <palavras_chave> : o quarto argumento indica uma ou mais palavras-chaves que devem ser buscadas na base de buscas.

É importante ressaltar que no modo de buscas é obrigatório que se definam apenas os argumentos <opção_busca> e <palavras_chave>; a definição dos outros argumentos é opcional. Caso não se definam os argumentos <tempo> e <opção_impressão>, devem ser assumidos como valor *default* as opções "-tF" e "-pI", respectivamente. A seguir, são mostrados alguns exemplos de como o programa deverá ser invocado para realizar buscas.

Realizar buscas por palavras-chaves

Exemplo 1: Buscando só uma palavra-chave

```
> ./buscaIMD -bAND galo
   >> Foram encontradas 2 linhas no arquivo "libertadores.txt":
     - linha 23: "o galo mais uma vez se salvou na libertadores"
     - linha 50: "na próxima fase, o galo joga em casa contra o"
   >> Foram encontradas 3 linhas no arquivo "granja.txt":
     - linha 1: "o galo é o macho da galinha, comumente tratado"
     - linha 4: "Algumas especies de galo são criadas como aves"
     - linha 8: "O galo é extremamente territorialista, sempre"
```

```
   >> Foram encontradas 0 linhas no arquivo "clima.txt":
```

No exemplo acima, para cada arquivo da base de dados, são listadas as linhas do arquivo em que a palavra-chave ocorreu. Além disso, como não foi especificado o argumento `<opção_impressão>`, os resultados foram exibidos na ordem em que os arquivos foram inseridos na base de buscas.

Exemplo 2: Buscando mais de uma palavra-chave com opção AND

```
> ./buscaIMD -bAND galo macho
   >> Foram encontradas 0 linhas no arquivo "libertadores.txt":
   >> Foram encontradas 1 linhas no arquivo "granja.txt":
      - linha 1: "o galo é o macho da galinha, comumente tratado"
   >> Foram encontradas 0 linhas no arquivo "clima.txt":
```

No exemplo acima, para cada arquivo da base de dados, são listadas as linhas do arquivo em que todas palavras-chave ocorreram na mesma linha.

Exemplo 3: Buscando mais de uma palavra-chave com opção OR

```
> ./buscaIMD -bOR galo macho
   >> Foram encontradas 2 linhas no arquivo "libertadores.txt":
      - linha 23: "o galo mais uma vez se salvou na libertadores"
      - linha 50: "na próxima fase, o galo joga em casa contra o"
   >> Foram encontradas 5 linhas no arquivo "granja.txt":
      - linha 1: "o galo é o macho da galinha, comumente tratado"
      - linha 4: "Algumas especies de galo são criadas como aves"
      - linha 8: "O galo é extremamente territorialista, sempre"
      - linha 15: "a relação é restrita com relação a outro macho"
      - linha 45: "o macho é ligeiramente maior que a fêmea"
   >> Foram encontradas 0 linhas no arquivo "clima.txt":
```

No exemplo acima, para cada arquivo da base de dados, são listadas as linhas do arquivo em que ao menos uma das palavras-chave ocorreram na mesma linha.

Configurar impressão das buscas

Exemplo 1: Exibindo resultados em ordem alfabética dos nomes dos arquivos

```
> ./buscaIMD -bAND -pA galo
   >> Foram encontradas 0 linhas no arquivo "clima.txt":
   >> Foram encontradas 3 linhas no arquivo "granja.txt":
      - linha 1: "o galo é o macho da galinha, comumente tratado"
      - linha 4: "Algumas especies de galo são criadas como aves"
      - linha 8: "O galo é extremamente territorialista, sempre"
   >> Foram encontradas 2 linhas no arquivo "libertadores.txt":
      - linha 23: "o galo mais uma vez se salvou na libertadores"
      - linha 50: "na próxima fase, o galo joga em casa contra o"
```

No exemplo acima, os resultados das buscas são exibidos em ordem alfabética dos nomes dos arquivos.

Exemplo 2: Exibindo resultados pelo número de ocorrência nos arquivos

```
> ./buscaIMD -bAND -pC galo
   >> Foram encontradas 3 linhas no arquivo "granja.txt":
     - linha 1: "o galo é o macho da galinha, comumente tratado"
     - linha 4: "Algumas especies de galo são criadas como aves"
     - linha 8: "O galo é extremamente territorialista, sempre"
   >> Foram encontradas 2 linhas no arquivo "libertadores.txt":
     - linha 23: "o galo mais uma vez se salvou na libertadores"
     - linha 50: "na próxima fase, o galo joga em casa contra o"
   >> Foram encontradas 0 linhas no arquivo "clima.txt":
```

No exemplo acima, os resultados das buscas são exibidos em ordem decrescente do número de ocorrência das palavras-chave nos arquivos.

Exemplo 2: Exibindo resultados em ordem de inserção dos arquivos

```
> ./buscaIMD -bAND -pI galo
   >> Foram encontradas 2 linhas no arquivo "libertadores.txt":
     - linha 23: "o galo mais uma vez se salvou na libertadores"
     - linha 50: "na próxima fase, o galo joga em casa contra o"
   >> Foram encontradas 3 linhas no arquivo "granja.txt":
     - linha 1: "o galo é o macho da galinha, comumente tratado"
     - linha 4: "Algumas especies de galo são criadas como aves"
     - linha 8: "O galo é extremamente territorialista, sempre"
   >> Foram encontradas 0 linhas no arquivo "clima.txt":
```

No exemplo acima, os resultados das buscas são exibidos na ordem de inserção dos arquivos texto à base de dados. Esta é a opção de exibição *default* do programa.

Registrar tempo de execução das buscas

Exemplo 1: Exibindo tempo de execução das buscas

```
> ./buscaIMD -bAND -pA -tT galo
   >> Foram encontradas 0 linhas no arquivo "clima.txt":
   >> Foram encontradas 3 linhas no arquivo "granja.txt":
     - linha 1: "o galo é o macho da galinha, comumente tratado"
     - linha 4: "Algumas especies de galo são criadas como aves"
     - linha 8: "O galo é extremamente territorialista, sempre"
   >> Foram encontradas 2 linhas no arquivo "libertadores.txt":
     - linha 23: "o galo mais uma vez se salvou na libertadores"
     - linha 50: "na próxima fase, o galo joga em casa contra o"
   >> Tempo total de execução: 3021 ms
```

No exemplo acima, além dos resultados das buscas, também é exibido o tempo total de execução das buscas realizadas em todos os arquivos da base de buscas.

2. Dicas para a realização do trabalho

Lembre-se de tratar casos excepcionais e de erro no uso do programa pela linha de comando. Isto é, o programa deverá ser capaz de identificar quando o usuário passou argumentos errados pela linha de comando. Caso identifique que foram passados argumentos errados pela linha de comando, o programa deverá encerrar a sua execução e exibir uma mensagem de erro explicando ao usuário como o programa deve ser usado.

O programa deverá ser bem documentado. Todo arquivo fonte deverá conter um cabeçalho descrevendo o propósito daquele arquivo e todas as funções deverão ter documentação descrevendo brevemente a função, seus parâmetros de entrada, seu retorno e variáveis globais que manipule.

O seu programa deverá ser modular, isto é, deverá ser adequadamente dividido em diferentes arquivos. Lembre-se de proteger cada arquivo de cabeçalho com uso de diretivas de compilação **#ifdef** e **#ifndef**, evitando, com isso, problemas de redefinições.

Gerencie bem as versões dos artefatos produzidos para este trabalho, evitando perdas de conteúdo devido a conflitos entre versões dos documentos. É importante ressaltar que, de maneira geral, o trabalho é bastante trabalhoso e sua realização deve ser iniciada imediatamente ao receber o enunciado. Por este motivo, também é importante que as tarefas deste trabalho sejam bem divididas entre os membros da dupla. A divisão das tarefas e o trabalho em equipe é um motivo a mais para preocupar-se com o gerenciamento disciplinado das versões dos documentos. Por fim, o enunciado poderá deixar margens para dúvidas. Sempre consulte o professor para solucioná-las.

3. Entrega do trabalho

O trabalho deve ser feito em duplas, não sendo permitido duplas de alunos de turmas diferentes.

O programa deverá obrigatoriamente ser implementado em C++. Não serão aceitos trabalhos implementados em qualquer outra linguagem de programação. **Todas as estruturas e algoritmos (busca, ordenação, etc) deverão ser implementadas pelos alunos, não sendo permitido o uso de bibliotecas prontas (STD, STL, BLAST, etc.) para estes fins**.

O entregável deste trabalho consistirá em:

- Arquivos fontes da implementação do programa especificado na Seção 1. As funções dos arquivos fontes devem ser devidamente documentadas.
- Arquivo LEIAME.txt explicando como deve ser compilado o programa e como deve ser usado pela linha de comandos.
- Relatório técnico descrevendo a solução proposta, detalhando quais estruturas e algoritmos foram usados para implementar as principais funcionalidades do programa.

4. Critérios de correção

O objetivo deste trabalho não é apenas escrever um programa que resolva o problema de busca proposto. O objetivo é desenvolver um programa de boa qualidade e que comprovadamente implementem, empreguem e ajustem os algoritmos e estruturas de dados vistos em sala de aula para resolver o problema proposto de maneira eficaz e eficiente.

Independentemente da dificuldade, do esforço e do tempo gasto pelo aluno, o que vai ser examinado ao se corrigir o trabalho é a qualidade deste, independentemente do número de horas que o aluno possivelmente tenha levado para desenvolver os programas. Além disso, o que será corrigido é o que foi entregue. Ou seja, caso sejam entregues arquivos obsoletos ou errados, ou sejam esquecidos arquivos, o aluno perderá os pontos correspondentes. Desta forma, é responsabilidade dos alunos tomar cuidado ao compor a versão final a ser entregue.

São itens essenciais a serem implementados neste trabalho:

- Pré-processar os arquivos texto mantidos na base de buscas
- Implementar uma estrutura de dados que mantenha a base de buscas já pré-processada
- Implementar todas as operações de gerenciamento da base de dados
- Implementar as buscas seguindo as estratégias *AND* e *OR*
- Implementar todas as formas de exibir os resultados

Obrigatoriamente, deverão ser usados nestes trabalho:

- Algoritmo de busca
- Algoritmo de ordenação
- Estrutura lista encadeada (com encadeamento simples ou duplo, tanto faz)
- Estrutura tabela de dispersão
- Leitura e escrita em arquivos
- Sobrecarga de funções e/ou templates
- Ponteiro para funções
- Alocação dinâmica de memória

Cabe ressaltar que estes itens serão considerados apenas no caso de terem sido aplicados corretamente e em situações adequadas. Em outras palavras, não basta apenas inserir sem critérios estes elementos para pontuar, pois, neste caso, estes serão ignorados!

5. Itens que contarão como pontuação extra

Além dos itens obrigatórios descritos nas seções passadas, também será possível acumular pontos extras caso os itens a seguir sejam implementados corretamente. Cabe ressaltar que a nota final do trabalho será dada pela soma dos pontos obtidos nos itens obrigatórios e dos pontos extra, num total máximo de 10 (dez) pontos, sendo os pontos excedentes descartados. Para implementar estes itens, não será permitido o uso de funções prontas de

quaisquer bibliotecas. Os alunos deverão implementar as estruturas e os algoritmos necessários.

Chaves de busca booleanas compostas:
Esta funcionalidade deverá permitir a construção de chaves de busca compostas por operadores booleanos AND-OR. Por exemplo: deverá ser possível fazer buscas com base em chaves de busca da forma "Documento OR Item AND Chave", em que deverão ser retornadas todas as linhas em que ocorrem a palavra-chave "Documento", ou todas as linhas em que ocorre ao mesmo tempo as palavras-chave "Item" e "Chave". Para esta funcionalidade, considere que o AND tem precedência sobre o OR.

Busca por substring:
Esta funcionalidade deverá permitir a busca por palavras-chave que são substrings de palavras contidas nos arquivos texto. Por exemplo, considere o seguinte exemplo:

```
> ./buscaIMD -bOR galo macho
   >> Foram encontradas 2 linhas no arquivo "libertadores.txt":
      - linha 23: "o galo mais uma vez se salvou na libertadores"
      - linha 50: "na próxima fase, o galo joga em casa contra o"
   >> Foram encontradas 5 linhas no arquivo "granja.txt":
      - linha 1: "o galo é o macho da galinha, comumente tratado"
      - linha 4: "Algumas especies de galo são criadas como aves"
      - linha 8: "O galo é extremamente territorialista, sempre"
      - linha 15: "a relação é restrita com relação a outros machos"
      - linha 45: "o macho é ligeiramente maior que a fêmea"
   >> Foram encontradas 0 linhas no arquivo "clima.txt":
```

Perceba nos resultados do arquivo `"granja.txt"` que a palavra-chave `"macho"` ocorreu na linha 15, embora a palavra contida nesta linha do arquivo texto não seja a string `"macho"`, mas sim a string `"machos"`. Ou seja, a palavra-chave passada como argumento da busca é uma substring de uma palavra contida no arquivo texto.

Implementação usando Classes
Caso o projeto seja corretamente implementado em classes (utilizando conceitos de Orientação a Objetos), isto contará como elemento de pontuação extra. Cabe ressaltar que esta pontuação extra aplica-se somente a implementação consistente usando classes, não bastando apenas a criação de uma ou outra classe.

6. Acompanhamento e dúvidas

Estão previstas aulas de acompanhamento do projeto na disciplina IMD0029, onde os alunos terão condições de esclarecer suas dúvidas, bem como mostrar o andamento de seus projetos. Embora a disciplina IMD0030 siga com conteúdo do programa de curso e que também poderá ser utilizado neste projeto, a cada aula, sempre que for requerido pelos alunos, os últimos 30min poderão ser dedicados a este mesmo fim. Para além disso,

dúvidas poderão ser esclarecidas através de comunicação no SIGAA ou nos horários de atendimentos dos professores e monitor (quando for o caso).