Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
DCC205 - Estrutura de Dados

Trabalho Prático 3 – Dicionário

Valor: 15 pontos

Data de entrega: 13/12/2022

Objetivo

O objetivo deste trabalho é projetar, codificar e avaliar diferentes implementações de um dicionário de palavras. O dicionário deve permitir consultas, inserções, retiradas, impressão ordenada e atualização de uma entrada. Cada entrada tem como chave uma palavra, seu tipo e zero ou mais significados da palavra (a palavra pode aparecer no dicionário, mas ainda sem algum significado associado). Você deve comparar duas implementações, uma baseada em árvore de pesquisa balanceada e outra baseada em hash, as quais devem ser comparadas considerando uma carga de trabalho típica do dicionário (faz parte do trabalho definir cargas de trabalho típicas).
O foco da avaliação é verificar o funcionamento correto do sistema ao executar cargas de trabalho típicas e mesmo partes dela, caracterizando diferentes situações. A análise de complexidade das funções implementadas será avaliada na documentação, e é necessário fazer uma comparação descrevendo as vantagens e desvantagens de cada implementação.

Descrição

O software a ser implementado é baseado num tipo abstrato de dados dicionário, que provê pelo menos as seguintes funções (os parâmetros são apenas para fins de exemplo, esteja à vontade para acrescentar ou remover parâmetros e mesmo funções que não sejam adequados ao seu projeto):

```
Dicionario * criaDic(int N);
int pesquisaDic(Dicionario * dic, Verbete * it);
int insereDic(Dicionario * dic, Verbete * it);
int imprimeDic(Dicionario * dic);
int atualizaDic(Dicionario * dic, Verbete * it);
int removeDic(Dicionario * dic, Verbete * it);
int destroiDic(Dicionario * dic);
```

A entrada do programa é um arquivo contendo palavras e seus significados. Cada linha do arquivo tem o seguinte formato:

```
a [applied] concerned with concrete problems or data
```

onde:
`a` - indica o tipo da verbete (adjetivo, nome ou verbo).
`[applied]` - é verbete (sem os colchetes).
`concerned with concrete problems or data` - é o significado do verbete.

Note que um verbete pode conter mais de um significado, como no exemplo a seguir:

```
a [bad] immoral, evil
a [bad] below average in quality or performance, defective
a [bad] spoiled, spoilt, capable of harming
```

Ou nenhum significado até o momento:

```
a [artistic]
a [ashamed]
a [asleep]
```

O programa principal deve ser idêntico, independente da implementação interna do TAD Dicionario.

O programa vai realizar a seguinte sequência de operações:

1. Criar o dicionário, indicando o número máximo de verbetes a ser inseridos (que pode ser o número de linhas do arquivo de entrada).
2. Inserir no dicionário criado todos os verbetes e seus significados a partir do arquivo de entrada. Note que um verbete pode ter mais de um significado e a estrutura de dados deve permitir o armazenamento dos potencialmente vários significados de um verbete.
3. Imprimir o dicionário resultante em ordem alfabética dos verbetes e ordem de inserção dos significados dos verbetes. Por exemplo, os verbetes `artistic` e `bad`, cujos significados foram apresentados acima, seriam impressos como:

```
artistic(a)
bad (a)
1. immoral, evil
2. below average in quality or performance, defective
3. spoiled, spoilt, capable of harming
```

4. Remover todos os verbetes que tem pelo menos um significado, pois a empresa que está construindo o dicionário quer saber quais verbetes ainda tem que ser complementados com os seus significados.
5. Imprimir novamente o dicionário, que deve conter apenas os verbetes que não tem significado registrado ainda.
6. Destruir o dicionário.

Quais TADs implementar?

Você deverá implementar pelo menos três Tipos Abstratos de Dados (TADs): o primeiro para o dicionário, o segundo para verbete e o terceiro para o significado do verbete. Detalhes de implementação como o tamanho do hash e a função hash são parte do projeto.

Parâmetros

O seu programa deve aceitar três parâmetros:

-i: informa o nome do arquivo de entrada
-o: informa o nome do arquivo de saída
-t: informa a implementação a ser utilizada: hash ou arv

Exemplo:

```
bin/tp3 -i entrada.txt -o saida.txt -t arv
```

Avaliação Experimental

Você deve considerar pelo menos as seguintes dimensões na sua avaliação experimental:

- Número de verbetes
- Número máximo de significados por verbete

Entregáveis

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é **terminantemente vetado**. Caso seja necessário, use as estruturas que **você** implementou nos Trabalhos Práticos anteriores para criar **suas próprias implementações** para todas as classes, estruturas de dados, e algoritmos.

Você **DEVE utilizar** a estrutura de projeto abaixo junto ao *Makefile* :

- TP
  |- src
  |- bin
  |- obj
  |- include
  Makefile

A pasta **TP** é a raiz do projeto; a pasta **bin** deve estar vazia; src deve armazenar arquivos de código (*.c, *.cpp ou *.cc); a pasta include, os cabeçalhos (*headers*) do projeto, com extensão *.h, por fim a pasta **obj** deve estar vazia. O **Makefile** deve estar na **raiz do projeto**. A execução do **Makefile** deve gerar os códigos objeto *.o no diretório **obj,** e o executável do TP no diretório **bin**.

Documentação

A documentação do trabalho deve ser entregue em formato **pdf**. A documentação deve conter os itens descritos a seguir :

Título, nome, e matrícula.
Introdução: Contém a apresentação do contexto, problema, e qual solução será empregada.
Método: Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
Análise de Complexidade: Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
Estratégias de Robustez: Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
Análise Experimental: Apresenta os experimentos realizados em termos de desempenho computacional e localidade de referência, assim como as análises dos resultados.
Conclusões: A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
Bibliografia: Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
Instruções para compilação e execução: Esta seção deve ser colocada em um apêndice ao fim do documento e em uma página exclusiva (não divide página com outras seções).
***Número máximo de páginas:** 20

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

Submissão

Todos os arquivos relacionados ao trabalho devem ser submetidos na atividade designada para tal no Moodle. A entrega deve ser feita **em um único arquivo** com extensão **.zip**, com nomenclatura nome_sobrenome_matricula.zip}, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais. O arquivo **.zip** deve conter a sua documentação no formato **.pdf** e a estrutura de projeto descrita.

Avaliação

O trabalho será avaliado de acordo com:

- Corretude na execução dos casos de teste - (50% da nota total)
- Apresentação da análise de complexidade das implementações - (15% da nota total)
- Estrutura e conteúdo exigidos para a documentação - (20% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- Cumprimento total da especificação - (5% da nota total)

Se o programa submetido não compilar[1], seu trabalho não será avaliado e sua nota será 0. **Não serão aceitos trabalhos entregues com atraso.**

Comentários gerais:

- Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.

- Leia **atentamente** o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.

- Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.

- Plágio é CRIME. Trabalho onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na sessão de bibliografia da

[1] Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

documentação. **Cópia e compartilhamento de código não são permitidos**.

- Penalização por atraso: considerando o fim do semestre, **não** será permitida a entrega com atraso.

FaQ (Frequently asked Questions)

1. **Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO**
2. Posso utilizar o tipo String? SIM.
3. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO
4. Posso utilizar alguma biblioteca para tratar exceções? SIM.
5. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
6. As análises e apresentação dos resultados são importantes na documentação? SIM.
7. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO
8. Posso fazer o trabalho em dupla ou em grupo? NÃO
9. Posso trocar informações com os colegas sobre a teoria? SIM.
10. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM.
11. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.