Universidade Federal de São Carlos - Campus Sorocaba

Trabalho Integrado

Disciplinas: Programação Orientada a Objetos e Estrutura de Dados 1
Professoras responsáveis: Katti Faceli e Tiemi C. Sakata

Regras para entrega

- Data de entrega: 10/06 até às 8:00 da manhã
- Apresentação/Entrevista: a marcar, a partir de 10/06. Integrantes ausentes no momento da avaliação do seu grupo receberão nota 0.
- Use o horário de atendimento para tirar suas dúvidas
    - Inclusive com o monitor de ED
- Notas poderão ser individuais.
- Deve ser feito em grupos de 4 alunos (2 grupos de 5). Não serão aceitos mais do que 13 grupos.
    - Grupos com menos de 4 alunos terão nota máxima 6.0
    - Preencher integrantes do grupo na planilha https://docs.google.com/spreadsheets/d/1vSfzXfuTSC3buT6wvee8RQ54IRL76PJ4REK1mKldM0k/edit#gid=0
- Forma de entrega:
    - Eletronicamente (via Moodle): um arquivo compactado contendo todos os arquivos .h, .cpp, os arquivos para inicialização do sistema e um arquivo com o relatório no formato PDF;
    - Não incluir outros arquivos (.o, .exe, etc...);
    - O nome deste arquivo deverá ser composto dos primeiros nomes dos integrantes da equipe (alunos com colegas de mesmo nome, incluir as iniciais do sobrenome).

Descrição

Considerando o cenário a seguir, foi elaborado o diagrama de classes simplificado em anexo. A sua equipe foi contratada para fazer a implementação do sistema projetado. Para isso, as seguintes regras devem ser seguidas:

- Todo o tratamento de erros deve ser feito pelo mecanismo de tratamento de exceção;
- O código deve estar bem documentado, e deve ser feita também uma documentação externa na forma de um relatório (usar seus conhecimentos de disciplinas anteriores e material disponibilizado);
- Acrescentar os atributos e métodos que forem necessários. O diagrama completo deve ser apresentado na documentação externa;
- Ao iniciar e finalizar a execução, o sistema deve recuperar e armazenar em arquivo todos os dados (dicionário e texto). Empregar os operadores << e >>. Para isso,

estudar os capítulos 23 e 24 da “Apostila de Programação Orientada a Objeto em C++“. Esses capítulos falam sobre entrada e saída de dados em C++ e manipulação de arquivos. Dar atenção especial ao capítulo 24.

- Com exceção da AVL exigida, você pode usar as estruturas que desejar para as implementações, incluindo as classes da STL.

Cenário

A sua equipe foi contratada para desenvolver um corretor ortográfico. O corretor deverá permitir a verificação de erros num texto comparando as palavras no arquivo de texto com as palavras num dicionário. Palavras que não fazem parte do dicionário são potenciais erros de escrita. Assim, caso o corretor encontre uma palavra que não faça parte do dicionário, ele deverá permitir ao usuário: corrigir a palavra, ignorar o erro, selecionar uma palavra a partir de uma lista de palavras semelhantes ou adicionar a palavra no dicionário. Ao apresentar um erro para ser corrigido, o corretor deverá apresentar também ao usuário, o contexto em que o erro ocorreu (a palavra anterior e a palavra seguinte ao erro). O corretor deverá também manter uma lista dos erros encontrados no texto (corrigidos ou não). Essa lista deverá conter apenas uma entrada para cada palavra errada e deverá manter o número de vezes que o mesmo erro ocorreu no texto. O Corretor deverá ser uma classe que contenha um dicionário, um texto e a lista das palavras erradas.

A classe Texto deverá conter uma lista de palavras, o nome do arquivo original do texto e permitir carregar um texto a partir de um arquivo, percorrer o texto, palavra por palavra, alterar uma palavra e gravar o texto em um arquivo (não necessariamente o original).

A classe Dicionário deverá armazenar as palavras obtidas do arquivo "dict.txt" e deverá permitir consulta e inclusão de palavras e fornecer uma lista de palavras semelhantes à uma determinada palavra (considerar palavras semelhantes aquelas que começam com as mesmas 2 letras). O relacionamento entre as classes Dicionário e Palavra deverá ser implementado por meio de uma árvore AVL.

A classe Palavra deverá conter ao menos a palavra, um método que verifique se duas palavras são semelhantes e ter o operador == sobrecarregado para verificar se duas palavras são idênticas.

O sistema deverá conter pelo menos as classes Corretor, Dicionário, Texto, Palavra e AVL. A classe corretor deverá conter métodos para a interação com o usuário e também para manipular as classes dicionário e texto.

A classe AVL deve ter na interface apenas os métodos: vazia, insere, remove e busca.

Diagrama simplificado