Trabalho prático 3 – ferramenta de verificação ortográfica

1) Informação Geral

O objetivo do trabalho prático 3 é avaliar a capacidade do estudante para analisar um problema algorítmico, utilizando estruturas derivadas às apresentadas na unidade curricular, e implementar em C uma solução correta e eficiente. Este trabalho deverá ser feito de forma autónoma por cada grupo nas aulas práticas e completando-o fora das aulas até à data limite estabelecida. A consulta de informação nas diversas fontes disponíveis é aceitável. No entanto, o código submetido deverá ser apenas da autoria dos elementos do grupo, e quaisquer cópias detetadas serão devidamente penalizadas. A eventual incapacidade de explicar o código submetido por todos os elementos do grupo também implicará uma penalização.

O prazo de submissão na página de Programação 2 do moodle é 23 de maio às 13:00.

2) Descrição do Trabalho

Foi-lhe solicitado pela empresa *minisoft* ajuda na implementação de uma ferramenta de verificação ortográfica (vulgo "spell checking") de elevado desempenho para uma plataforma com capacidades de processamento e memória limitadas. A *minisoft* é uma *startup* que procura desenvolver a primeira ferramenta ortográfica útil para ser embebida em formulários Web.

Muitas aplicações – como processadores e editores de texto, clientes de e-mail – incluem ferramentas de verificação ortográfica. A tarefa de uma ferramenta destas é relativamente simples. Dada uma lista de palavras com ortografia correta (que vamos designar por lista de palavras) e um texto de entrada, a ferramenta de verificação identifica todas as palavras com ortografia incorreta (isto é, todas as palavras que não se encontram na lista de palavras) no texto de entrada. Quando uma palavra com ortografia incorreta é identificada no texto de entrada, uma boa ferramenta ortográfica tem a capacidade de sugerir palavras com ortografia correta que de alguma forma estão relacionadas com as palavras incorretas.

A título de exemplo, considere que a lista de palavras contém as seguintes palavras: *bake, cake, main, rain, vase*.

Se o texto de entrada contém a palavra *vake*, uma boa ferramenta de verificação vai identificar *vake* como sendo uma palavra com ortografia incorreta, uma vez que não consta na lista de palavras, e sugere como alternativa as palavras *bake*, *cake* ou *vase* como sendo a palavra pretendida.

Como se depreende, a tarefa principal de uma ferramenta de verificação ortográfica é centrada na pesquisa de palavras numa lista de palavras, potencialmente bastante extensa. Uma pesquisa eficiente é assim crítica para o desempenho de uma ferramenta ortográfica: uma vez que a pesquisa não é apenas para verificar se as palavras no texto de entrada existem na lista de palavras, mas também efetuar sugestões de alternativas para as palavras com ortografia

incorreta. O que implica que uma técnica de pesquisa pouco eficiente torna a ferramenta ortográfica inútil.

A forma mais simples de sugerir correções é através de um método de tentativa-erro. Se assumirmos que uma palavra com ortografia incorreta contém apenas um erro, podemos testar todas as correções possíveis e verificar quais as que têm a ortografia correta. Habitualmente um bom corretor ortográfico pesquisa por quatro erros possíveis (exemplos para a palavra "word"): uma letra errada ("wofd"), uma letra a mais ("worrd"), uma letra a menos ("wod"), ou um par de letras trocadas ("wodr"). Nesta fase, e para simplificar, na sua ferramenta de verificação a *minisoft* apenas está a lidar com a última classe de erros (par de letras trocadas).

A primeira abordagem da *minisoft* na implementação da ferramenta ortográfica passou por armazenar a lista de palavras numa lista ligada. No entanto, à medida que a quantidade de palavras aumentava, esta estrutura apresentou "alguns" problemas.

Após discutir com os engenheiros da *minisoft* envolvidos no projeto, o seu grupo relembrou o que aprendeu em Programação 2 de modo a escolhe a(s) estrutura(s) de dados mais adequadas para a implementação da ferramenta de verificação ortográfica. Para a demonstrar a sua eficiência deve utilizar um caso de teste fornecido pela *minisoft*.

3) Implementação do trabalho

O ficheiro *PROG2_1516_T3.zip* contém alguns dos ficheiros necessários para a realização do trabalho, nomeadamente:

- *spellChecker.c* que inclui o processamento da linha de comando e indicação de como deve estruturar a saída do seu programa (fundamentalmente na medição do tempo de execução).
- *wordlist.txt* ficheiro que contém a lista de palavras ortograficamente corretas (dicionário de inglês).
- *twoMistakes.txt*, *smallMistakes.txt* e *hugeMistakes.txt* são 3 ficheiros de teste que deverá utilizar na validação da sua solução (detalhes abaixo).

Para a realização do trabalho deve incluir as bibliotecas que considerar necessárias e selecionar as estruturas de dado mais adequadas a cada tarefa. Se assim o entender pode incluir outros ficheiros de modo a organizar o seu código, mas deve manter a função *main* no ficheiro *spellChecker.c*.

Deve começar por implementar as rotinas de leitura da lista de palavras (contidas no ficheiro *wordlist.txt*, uma palavra por linha) e o seu armazenamento na estrutura de dados adequada. Deve depois implementar as rotinas necessárias à leitura dos ficheiros de entrada e o seu processamento frase-a-frase (a cada linha do ficheiro corresponde um parágrafo, que pode conter uma ou mais frases). O processamento de cada frase implica verificar se cada palavra que a constitui está (ou não) com a ortografia correta (deve ter atenção às questões de maiúsculas/minúsculas na primeira palavra da frase, carateres de pontuação, aspas e parêntesis, valores numéricos, etc.). Caso encontre uma palavra que não tenha a ortografia correta deve utilizar uma rotina que permita sugerir alternativas ortograficamente corretas (relembre o tipo de erros que deve considerar).

Para efeitos de verificação da eficiência da sua solução, a *minisoft* pretende que o seu programa deve contabilizar o tempo de processamento (em segundos), o número de palavras com

ortografia incorreta que encontra e o total de sugestões. O ficheiro spellChecker.c já tem os extratos de código que suportam (parcialmente) esta funcionalidade. Utilize como referência a a função *main* que é fornecida, onde deverá implementar as funcionalidades principais. NÃO altere o output do programa nem o nome do ficheiro.

4) Utilização do programa, dados de Entrada e de saída

Os dados de entrada para o seu programa são a lista de palavras disponível no ficheiro *wordlist.txt* e os ficheiros de entrada *twoMistakes.txt*, *smallMistakes.txt* e *hugeMistakes.txt*.

O seu programa deverá receber o nome do ficheiro a verificar e o modo de funcionamento (*verboso* com detalhes ou *silencioso* sem detalhes) através da linha de comando:

```
./spellChecker ficheiro_com_texto_a_verificar modo
```

O seu programa deve ler as palavras que constam no ficheiro de entrada (ficheiro de texto), lendo uma linha de cada vez (cada linha corresponde a um parágrafo), processando uma frase de cada vez e pesquisando cada palavra da frase na lista de palavras. Se a palavra não é encontrada o programa deve apresentar a frase completa onde se encontra a palavra com a ortografia incorreta, com uma indicação da palavra e uma lista de sugestões. Por exemplo, para o ficheiro de entrada *twoMistakes.txt* (que contém apenas a frase "`This is a lnie of text that has a mispselling in it.`") e executando no modo "verboso" (`>./spellChecker twoMistakes.txt verboso`) o seu programa deve apresentar a seguinte saída:

```
This is a lnie of text that has a mispselling in it.
      Word not found: lnie
      Perhaps you meant: line, lien

This is a lnie of text that has a mispeslling in it.
      Word not found: mispselling
      Perhaps you meant: misspelling

Tempo gasto na verificação (em segundos): 0
Número de palavras com ortografia errada encontradas: 2
Número de sugestões efetuadas: 3
```

Deste modo vai poder testar se o seu programa tem o comportamento esperado. No entanto, não será esta a melhor de avaliar o desempenho da ferramenta.

Em modo "silencioso" o seu programa realiza as mesmas tarefas, não apresenta detalhes na consola, apresenta apenas o resumo da verificação. Executando no modo "silencioso" (`./spellChecker twoMistakes.txt silencioso`) o seu programa deve apresentar a seguinte saída:

```
Tempo gasto na verificação (em segundos): 0
Número de palavras com ortografia errada encontradas: 2
Número de sugestões efetuadas: 3
```

Valide a eficiência da sua solução de modo a verificar se pode a vir a ser adotada pela *minisoft* testando o seu programa com o ficheiro de entrada *hugeMistakes.txt*.

5) Ferramenta de desenvolvimento

A utilização de um IDE, por exemplo o Eclipse, é aconselhável no desenvolvimento deste trabalho. Para além gerir o processo de compilação, o IDE permite fazer debugging de uma forma mais eficaz. Poderá encontrar informações sobre a utilização do Eclipse num breve tutorial disponibilizado no moodle.

6) Avaliação

A classificação do trabalho é dada pela avaliação feita à implementação submetida pelos estudantes mas também pelo desempenho dos estudantes na aula dedicada a este trabalho. A classificação final do trabalho (T3) é dada por:

T3 = 0.5 Implementação + 0.1 Complexidade + 0.1 Memória + 0.2 Desempenho

A classificação da implementação é essencialmente determinada por testes automáticos adicionais. Caso a implementação submetida não compilar, esta componente será de 0%.

A Complexidade também será avaliada, sendo considerados 3 patamares: 100% escolha das estruturas de dados e algoritmos mais adequadas, 50% escolha das estruturas de dados e algoritmos adequadas, 0% escolha das estruturas de dados e algoritmos desadequadas.

A gestão de memória também será avaliada, sendo considerados 3 patamares: 100% nenhum memory leak, 50% alguns memory leaks mas pouco significativos, 0% muitos memory leaks.

O desempenho será avaliado durante a aula e está dependente da entrega do formulário "Preparação do trabalho" que se encontra disponível no moodle. A classificação de desempenho poderá ser diferente para cada elemento do grupo.

7) Submissão da resolução

A submissão é apenas possível através do moodle e até à data indicada no início do documento. Deverá ser submetido um ficheiro zip contendo:

- o ficheiro spellChecker.c e outros (incluindo bibliotecas que utilizou) com as funções devidamente implementadas de forma a produzir a solução correta ao problema apresentado.
- um ficheiro autores.txt indicando o nome e número dos elementos do grupo

Nota importante: apenas as submissões com o seguinte nome serão aceites: T3_G<numero_do_grupo>.zip. Por exemplo, T3_G999.zip

Deverá entregar o conjunto de ficheiros .c e .h fornecidos com as funções devidamente implementadas de forma a produzir a solução correta ao problema apresentado.