U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO	MESTRADO INTEGRADO EM ENGENHARIA BIOMÉDICA | 3º ANO EBE0116 | ESTRUTURAS DE DADOS E ALGORITMOS | 2019/2020 TRABALHO PRÁTICO

Jogo de palavras - Boggle

Introdução

O objetivo deste trabalho prático é desenvolver uma aplicação que permita jogar um jogo de palavras, conhecido por *Boggle*, que é jogado usando uma grelha de dados com letras, no qual os jogadores tentam encontrar palavras formadas por sequências de letras adjacentes na grelha.

Objetivos

Através da realização deste trabalho, pretende-se que os estudantes pratiquem o desenvolvimento de programas, em linguagem C++, seguindo o paradigma de programação "orientada a objetos", selecionando estruturas de dados e algoritmos adequados para a implementação do jogo e tirando partido das estruturas de dados e algoritmos disponíveis na Standard Template Library (STL) de C++.

Regras do jogo

No essencial, o jogo consiste em, durante **N** minutos (tipicamente, **N**=3), usando as letras visíveis nas faces de 16 cubos colocados num tabuleiro (ver figura), formar o maior número possível de palavras, constituídas por letras contíguas (3 no mínimo), de forma a que nenhuma letra seja reutilizada na formação de uma palavra.

As regras detalhadas de funcionamento do jogo podem ser consultadas nas referências [1] e [2]. Pretende-se implementar apenas a versão mais comum do jogo (não a que envolve o uso do "*Boggle challenge cube*" [1]), com cubos que tenham apenas uma letra em cada uma das faces.

Funcionamento do programa

- O programa deve ser executado em "modo consola", isto é, com uma interface simples, em modo texto. Poderá ser apresentado texto colorido recorrendo às funções disponibilizadas no Moodle para esse efeito.
- O programa deve fazer o seguinte:
  - Ler a seguinte informação relativa ao jogo, registada (pela ordem indicada) no ficheiro **BOGGLE_CONFIG.TXT** (ver exemplo fornecido no Moodle):
    - O nome do ficheiro de texto que contém as letras dos cubos; o nome deste ficheiro deve ser **BOARD_XXX.TXT**, em que **XXX** é a sigla da língua (ex: **PT** para português, **EN** para inglês, ..., ou **INT** para a versão internacional [3][4]).
    - o nome do ficheiro de texto que contém a lista de palavras válidas; o nome deste ficheiro deve ser **WORDS_XX.TXT**, em que **XX** é a sigla da língua (ver acima).
    - O tempo máximo (em segundos) que cada jogador tem para escrever as palavras da sua aposta, numa ronda.
    - O número mínimo de letras que deve ter uma palavra.
    - O número de pontos necessário para vencer o jogo.
  - Ler, do teclado, os nomes dos jogadores (terminados com **CTRL-Z**). Não há

limite para o número de jogadores.

- Preparar o jogo:
  - Ler as letras dos cubos contidas em **BOARD_XXX.TXT** e baralhar os cubos; para permitir jogar com tabuleiros de diversas dimensões, a primeira linha do ficheiro deve indicar as dimensões do tabuleiro (ver exemplo fornecido no Moodle);
  - Ler as palavras válidas, contidas em **WORDS_XXX.TXT**, para uma estrutura de dados interna ao programa; no Moodle são disponibilizados 2 ficheiros contendo uma lista de palavras portuguesas e uma lista de palavras inglesas;
- Possibilitar que os jogadores joguem o jogo:
  - Em cada ronda, pedir sequencialmente (em vez de simultaneamente, como no jogo tradicional) a cada um dos jogadores que, durante o tempo máximo estabelecido, escreva as palavras da sua aposta, as quais devem ser guardadas numa estrutura de dados adequada. O tabuleiro deve ser mostrado no ecrã, antes de solicitar que o jogador faça as suas apostas.
  - Verificar se as palavras da aposta são válidas, isto é, se fazem parte da lista de palavras da língua em causa e se podem ser formadas com as letras do tabuleiro, contabilizando a pontuação de cada jogador, na ronda e no total, tendo em conta as regras do jogo.
  - Ao calcular a pontuação de cada jogador numa ronda, o programa deve mostrar a pontuação atribuída a cada uma das palavras da aposta, acompanhada de uma explicação: se a palavra tiver uma pontuação diferente de zero, deve ser mostrada a posição no tabuleiro das letras que a constituem; se a palavra tiver uma pontuação zero, deve ser explicado o motivo: a palavra não tem o número mínimo de letras necessário; não é possível formar a palavra com as letras do tabuleiro; a palavra não faz parte da lista de palavras válidas; a palavra foi escolhida por outro(s) jogador(es).
  - Assim que um jogador atingir o número a pontuação necessária para vencer, anunciar o nome do vencedor.
- Criar um relatório do jogo, registando num ficheiro de texto, **BOGGLE_GAME_-XXX.TXT**, onde **XXX** representa o número de ordem do jogo (001, 002, …999), a instância do tabuleiro que foi usada no jogo, o nome de cada jogador, as suas apostas e a pontuação obtida e, finalmente, o nome do vencedor. Sugestão: use um ficheiro **BOGGLE_GAME_NUM.TXT** para registar (apenas) o número de jogos que foram jogados; na primeira vez que o jogo for jogado, o ficheiro não deve existir, devendo ser criado automaticamente pelo programa.

Desenvolvimento do programa e especificações adicionais

- **Este é um trabalho de grupo**. Cada grupo deve ser constituído por **3 estudantes**. **Todos os elementos do grupo devem participar no desenvolvimento do trabalho**. A constituição de cada grupo deve ser indicada, usando o 'link' disponível no Moodle, até ao fim do dia 9/dez/2019.
- O ficheiro **BOGGLE_CONFIG.TXT** deve ter um formato idêntico ao do ficheiro fornecido no Moodle, podendo o valor de cada um dos parâmetros ser diferente dos indicados. Por isso, o programa deve permitir jogar com tabuleiros de outras dimensões (no máximo 6x6), com outros tempos para cada aposta ou com outras pontuações necessárias para vencer o jogo.
- Os ficheiros **BOARD_XXX.TXT** devem um formato idêntico ao ficheiro **BOARD_INT.TXT** fornecido no Moodle: a primeira linha contém as dimensões do tabuleiro; cada uma das linhas seguintes contêm as letras das faces de cada um dos cubos (um cubo por linha).
- Os ficheiros **WORDS_XXX.TXT** devem um formato idêntico ao ficheiro **WORDS_PT.TXT** fornecido no Moodle, isto é, devem conter uma palavra por linha.

- O programa deve ser implementado com recurso algumas "classes" definidas pelo programador, por exemplo, para representar um cubo de letras (**Cube**), um tabuleiro (**Board**), a lista de palavras válidas (**Dictionary**, embora não se trate exatamente de um dicionário) e um jogador (**Player**); outras poderão revelar-se úteis ou convenientes. O código deve estar estruturado de forma a separar a definição de cada classe e a implementação dos seus métodos, em ficheiros com a extensão **.h** e **.cpp**, respetivamente. No Moodle, é fornecido um esboço inicial, parcial, da interface das classes **Cube**, **Board** e **Dictionary** e **Player**, que poderá ser usado como ponto de partida para o desenvolvimento.
- São fornecidos no Moodle:
    - O código de uma função (**findWord**) que permite verificar se é possível formar uma palavra com as letras contidas num vetor 2D, usando uma técnica conhecida por *recursive backtracking.* Este código pode ser usado como ponto de partida para o desenvolvimento do método da classe *Board* que implementa uma funcionalidade semelhante. Sugere-se que se comece por analisar o código e perceber o seu funcionamento.
    - O código de um programa (**timedRead**) que permite ler *strings* durante um certo intervalo de tempo. Este código pode ser usado como ponto de partida para o desenvolvimento do método da classe *Player* que permite ler as palavras que constituem a aposta do utilizador, numa ronda.

Entrega do trabalho

- Todos os ficheiros de código (**.h**, **.cpp** e **.pdf**, abaixo referidos) devem ter um cabeçalho com a identificação do grupo e dos elementos constituintes:

```
// GRUPO xx
// nome do estudante 1
// nome do estudante 2
// nome do estudante 3
```

- Criar um diretório com o nome **Gxx** (**xx** representa o número do grupo de trabalho, com dois dígitos; por exemplo, G01) e copiar para lá o código-fonte de cada programa (apenas os ficheiros com extensão **.cpp** e **.h**). Os ficheiros de texto usados pelo programa não devem ser submetidos, uma vez que na avaliação serão usados ficheiros semelhantes aos fornecidos no Moodle; por isso, a estrutura dos ficheiros de texto anteriormente referidos deve ser respeitada rigorosamente, caso contrário, o trabalho submetido pode não ser avaliado.
- Acrescentar ao diretório um ficheiro **pdf**, contendo um **pequeno relatório** constituído por 3 secções:
    - 1- Classes: nesta secção deve ser apresentada a declaração (métodos e atributos) de cada uma das classes usadas no desenvolvimento do programa, acompanhada de breves comentários sobre a funcionalidade dos métodos e o significado dos atributos.
    - 2- Estado de desenvolvimento: nesta secção deve ser descrito o estado de desenvolvimento do programa, indicando se foram ou não cumpridos todos os objetivos e, caso não tenham sido, quais os que ficaram por cumprir.
    - 3- Exemplo de execução: nesta secção deve(m) ser mostrada uma ou mais capturas de ecrã que ilustrem o funcionamento do jogo.
- Compactar o conteúdo desse diretório num ficheiro **Gxx.zip** e submeter este ficheiro na página da disciplina de EDA, no Moodle da FEUP.
- Apenas um dos elementos de cada grupo de trabalho deve fazer a entrega.
- **Data limite** para a **entrega** do trabalho: **3/jan/2019, às 23:59h**.
- **Apresentação e discussão** do trabalho: em local e data a indicar.

Avaliação do trabalho

Na avaliação do trabalho serão tidos em conta os seguintes aspetos:

- Estrutura e modularidade do código.
- Estruturas de dados e algoritmos utilizados.
- Apresentação do código, incluindo, indentação, comentários e nomes dos identificadores de tipos, constantes, variáveis e funções.
- Funcionamento do programa.
- Discussão do trabalho, durante a apresentação.

Referências

[1] – "Boggle", https://www.hasbro.com/common/instruct/boggle.pdf

[2] – "Boggle", https://en.wikipedia.org/wiki/Boggle

[3] – "Boggle", https://fr.wikipedia.org/wiki/Boggle

[4] – "Boggle: What is the dice configuration for Boggle in various languages?", https://boardgames.stackexchange.com/questions/29264/boggle-what-is-the-dice-configuration-for-boggle-in-various-languages

JAS