Pontifícia Universidade Católica de Goiás
Escola de Ciências Exatas e da Computação
CMP1048 - Técnicas de Programação I
Max Gontijo de Oliveira
Trabalho 2

1 Descrição do trabalho

Construir um programa em C++ que implemente o jogo chamado **Forca**.

1.1 Jogo Forca

O jogo Forca consiste na seleção aleatória de uma palavra qualquer que deverá ser adivinhada pelo jogador. Para que o jogador não simplesmente "chute"qual é a palavra, ele terá algumas chances de "chutar"possíveis letras que compõem a palavra. A cada letra que ele acertar, essa letra será revelada na palavra. Após esgotar todas as chances, ele será obrigado a tentar adivinhar a palavra. Se acertar qual é a palavra, o jogador vence o jogo. Caso contrário, perderá o jogo.

1.2 Construção do programa

1.2.1 Classes

O jogo deverá ser construído com o paradigma de orientação a objeto utilizando a linguagem C++. Portanto, será fundamental a definição de classes. Para auxiliar, seguem algumas classes que poderiam ser consideradas para melhor organizar o código.

- `Jogador`
  - Armazena o nome do jogador e o tempo de seu jogo (se vencer).
- `Jogo`
  - Armazena em atributos internos, informações sobre o jogo e possui métodos necessários para a partida, tais como, método para atualizar a tela, selecionar uma palavra na base de palavras, etc.

Evidentemente tratam-se apenas de sugestões. É mandatório que o programa utilize a implementação de classes, contudo, o aluno tem liberdade para escolher as classes que deseja implementar e que achar mais pertinente.

1.2.2 Uma partida

Segue um fluxo de como deve ser uma partida.

1. O programa abre e lê um arquivo que contém diversas palavras. Essas palavras devem ser carregadas na memória em um vetor de strings.
2. O programa sorteia uma palavra dentre todas as disponíveis no vetor.
3. O programa exibe na tela o caractere `"_"` N vezes, onde N é a quantidade de caracteres da palavra sorteada. Assim o jogador saberá a quantidade de letras que tem a palavra e a disposição das letras que adivinhar.
4. O jogador irá tentar adivinhar 5 letras que componham a palavra, informando uma por vez.
   - A cada letra sugerida pelo usuário, o programa irá checar se existe a letra digitada na palavra sorteada. Todos as letras da palavra sorteada que forem iguais à letra informada deverão ser exibidas ao jogador.
5. Quando esgotarem as tentativas de adivinhar algumas letras, o jogador deverá tentar acertar a palavra inteira. Assim, o programa irá ler a string completa e comparar com a palavra sorteada.
6. Se o jogador adivinhar a palavra, o programa deverá apresentar uma mensagem indicando a vitória do jogador. Caso contrário, deverá apresentar uma mensagem indicado a derrota do jogador.
7. Se o jogador vencer, o programa irá calcular o tempo que durou a partida, do momento em que o jogo começou ao momento em que o jogador venceu.
8. Em seguida, o programa irá verificar se o tempo foi menor do que o melhor tempo até então. Caso positivo, deverá ser considerado que o jogador quebrou o recorde.
9. Se o jogador quebrou o recorde, o nome e tempo do jogador deverá ser armazenado no arquivo de recorde (não é o mesmo arquivo da base de palavras).
10. Por fim, o programa deverá mostrar na tela o recorde do melhor jogo (nome do jogador e tempo em SEGUNDOS). Caso não exista, não deverá mostrar nada. Além disso, deverá mostrar o tempo gasto no jogo recém jogado, caso o jogador tenha vencido.

1.2.3 Detalhes do arquivo de palavras

Seguem algumas considerações sobre o arquivo de palavras:

- O arquivo será fornecido pelo professor e terá o nome `palavras.txt`.
- Cada linha do arquivo terá uma única palavra, passível de ser selecionada.
- O arquivo terá exatas 50 palavras, portanto, 50 linhas.
- Todas as palavras estarão em caixa alta (MAIÚSCULAS) e não haverão caracteres especiais ou acentuados.

2 Dicas de programação

Para auxiliar na construção do jogo, seguem algumas dicas.

2.1 Arquivos

Utilize as classes `ifstream` para leitura de arquivo e `ofstream` para escrita.

Para ler o arquivo de palavras, utilize a função `getline`. Essa função recebe dois parâmetros: uma *stream* e uma string. Já foi utilizada em sala para a leitura de uma string a partir do teclado. Nesse caso de ler do teclado, a chamada foi realizada assim:

```
string s;
getline(cin,s);
```

Tal como o `cin` é uma *stream*, um objeto `ifstream` também é. Assim, a função `getline` também pode ser utilizada em um arquivo. Ficaria assim:

```
ifstream arq_in_palavras("palavras.txt");
getline(arq_in_palavras,s);
```

Após a chamada da função, o programa terá lido a próxima linha no arquivo e armazenado na string `s`. Portanto, utilizando essa função será mais fácil ler linhas inteiras em um arquivo de texto.

2.2 Arquivo de recorde

Sempre que for atualizar o arquivo, recrie-o e armazene o nome em uma linha e o tempo em outra linha. Assim, recuperar as duas informações posteriormente será algo mais simples.

2.3 Tempo de jogo

Para visualizar o tempo de jogo, utilize a função `time`. Essa função irá retornar o tempo atual em segundos. Chame-a antes e depois do jogo e tire a diferença para saber quantos segundos durou a partida. Veja como ficaria:

```
long long tempoInicio = time(0);
// Realização da partida
.
.
.
long long tempoFim = time(0);
long long tempoPartida = tempoFim - tempoInicio;
```

Para fazer uso dessa função, será necessário incluir a biblioteca `ctime`.

2.4 Seleção de uma palavra aleatória

Para escolher uma palavra de forma aleatória, utilize a função `rand` associada às funções `srand` e `time`. A função rand gera um número inteiro aleatório a partir de um número semente. A função `srand` configura essa semente. Utilizaremos o tempo atual como semente. O número gerado pela função `rand` é, por vezes, muito grande. O fato é que, como haverão 50 palavras passíveis de serem sorteadas, deseja-se que o número aleatório seja entre 0 e 49. Portanto, a geração de um número aleatório entre 0 e 49 pode ser feita da seguinte maneira:

```
srand(time(0));
int indicePalavra = rand() % 50;
```

Após esse trecho de código, `indicePalavra` será o índice da palavra sorteada. Considerando que o vetor de palavras se chama `palavras`, a palavra sorteada será `palavras[indicePalavra]`.

Para fazer uso das funções `rand` e `srand`, será necessário incluir a biblioteca `cstdlib`.

3 Entrega e Avaliação

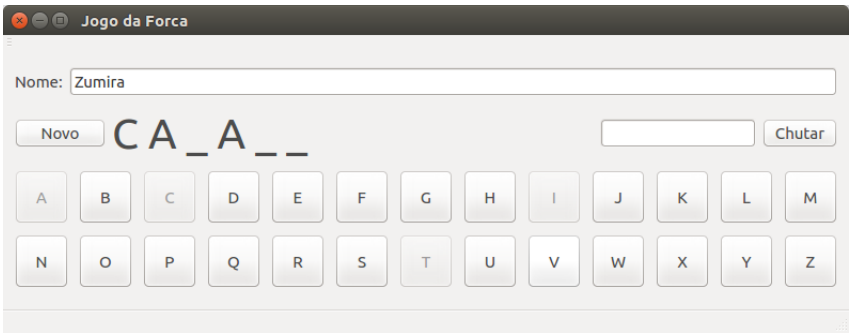
Os arquivos de código fonte deverão ser submetidos no Google Classroom em uma atividade específica para esse trabalho.

A avaliação será feita por meio de apresentação individual em sala de aula. Os critérios para avaliação serão: respeito ao prazo de entrega, quantidade de requisitos atendidos, ORGANIZAÇÃO DO CÓDIGO, uso de boas práticas de orientação a objetos mencionadas em sala de aula, compilação e execução correta, explicação do aluno e possíveis alterações no código demandadas pelo professor.

Sobre as boas práticas de orientação a objetos, algumas foram mencionadas em sala, como buscar o encapsulamento, nomes de métodos que façam sentido (sobretudo os *getters* e *setters*), modificadores de acesso utilizados corretamente, integridade das classes, etc.

3.1 Bônus

Será atribuída uma nota adicional aos trabalhos feitos com interface gráfica. Segue uma sugestão de interface para o programa:



Note que se trata apenas de uma sugestão. Você pode fazer de forma mais simples ou mais complexa. Note, ainda, que nessa sugestão não são mostradas todas as informações solicitadas no trabalho (como, por exemplo, tempo total da partida ou nome e tempo do jogador detentor do recorde). Se essa sugestão fosse a interface do jogo, certamente haveria uma outra janela que mostraria as demais informações no momento oportuno.

Mas vale ressaltar que o uso de interface gráfica não é obrigatório e nem deve ser considerado prioritário. Um trabalho completo sem interface gráfica vale mais do que um trabalho incompleto com uma linda interface gráfica.