Trabalho Prático I - INF16153 Programação II - Jogo do Termo Multiplayer

Prof. Vinicíus Mota
DI/UFES

Especificação do trabalho e Monitoria: Abraão Santos

Data de entrega: 20/06/2022

Entrevista: 27/06/2022

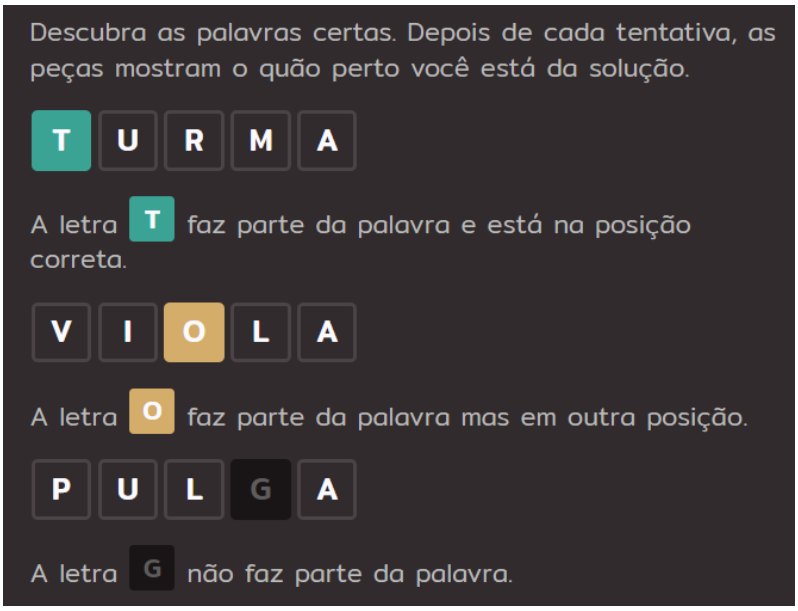
Local de entrega:

- AVA.

1. Objetivos

O objetivo geral deste trabalho é aprimorar as habilidades de programação sobre tipos abstratos de dados, ponteiros, gerenciamento de memória e bibliotecas.

O tema proposto para este trabalho prático é o desenvolvimento de uma variação do jogo *Wordle*, que em português se popularizou como o **jogo Termo**. Na versão original, o jogador deve acertar uma palavra de 5 letras em até 6 chances. A cada tentativa é apresentado se a letra não existe na palavra, se existe e está na posição errada ou se ela está na posição certa. A Figura abaixo ilustra as regras do Jogo do Termo.



Nesta versão, ao abrir o jogo, é solicitado o modo de jogo e o nome do jogador, que funcionará como um **login único**. Toda vez que abrir o programa, o usuário poderá iniciar um jogo.
A versão deste trabalho permitirá que mais de um jogador batalhe a mesma palavra. Neste modo, os jogadores batalham pela mesma palavra. Sendo que cada um joga uma vez e passa a vez para o outro.

As informações de estatísticas de um jogador devem persistir, isto é, ser salvas em um arquivo e abertas todas vez que o jogo iniciar. As informações serão similares às que existem no Jogo do Termo Original, ilustrado na figura abaixo.

A descrição e funcionalidades do sistema serão detalhadas nas próximas seções.

2. Regras do Jogo

As regras abaixo funcionam independentemente do modo escolhido pelo jogador.

- O jogo sorteia uma palavra de 5 letras.
    - Uma palavra só pode ser sorteada uma única vez, independentemente de quantas vezes o jogo é aberto.
    - As palavras não possuem caracteres especiais, como acentos e cedilhas;
    - A palavra pode ter letras repetidas;
- O jogo inicia com uma tabela vazia com 5 colunas.
- O jogador deve tentar inserir uma palavra válida, isto é, a palavra deve existir na lista de palavras.
- Caso não exista, informa a mensagem de erro e volta ao estado anterior;
- Caso a palavra tenha menos ou mais do que 5 caracteres informa a mensagem de erro;
- Caso a palavra inserida pelo jogador exista, o jogo analisa letra por letra e gera uma resposta através das cores ou alguma outra indicação visual:

    - VERDE ou CAIXA ALTA indica que aquela letra está no lugar certo;
    - AMARELO ou ENTRE PARÊNTESES indica que aquela palavra tem essa letra, porém está no lugar errado;
    - CINZA OU COM UMA EXCLAMAÇÃO NA FRENTE indica que aquela palavra não tem aquela letra;
- O jogador vence se acertar a palavra em até 6 tentativas e perde, caso contrário.
- Após o final do jogo deve ser exibido a estatística do jogador. Apresentando nome, quantidade de jogos, porcentagem de vitória, vitórias seguidas, número máximo de vitórias seguidas. Além disso, uma tabela contendo quantos jogos venceu com cada número de tentativas.
    - Os dados dos jogadores deverão ser armazenados em um arquivo binário.

OBS: Pode-se utilizar a biblioteca `conio.h` para a impressão das cores.

IMPORTANTE:

- O sistema de sorteio da palavra tem que usar como semente a data do sistema (pegar o *timestamp* do sistema).
- O programa terá que gerar um arquivo de palavras sorteadas, para que não repita uma palavra já jogada. O Programa somente poderá repetir palavras caso todas as palavras já tenham sido jogadas.
- A estrutura do(s) jogador(es) tem que ser salva em modo binário.
- Não vai ser limitado a uma palavra por dia, mas a *semente geradora* será igual ao longo do dia. Isto fará com que a sequência de palavras sorteadas seja igual ao longo do dia.
- O código deve ser devidamente separado em funções com as devidas implementações, sempre levando em consideração as boas práticas.
- É seu dever como desenvolvedor filtrar erros de entrada dos usuários. Mesmo que não estejam listados aqui. Sempre comente essas verificações.
- O Código deve estar bem comentado.

3.1 Funcionalidades

O programa funcionará com uma interface do jogo. Isso deve ser feito no terminal, inicialmente será feita a leitura do nome do jogador é solicitado que ele escolha o modo de jogo, depois é sorteado uma ou mais palavras aleatórias da lista, e então, o jogador terá a chance de tentar uma palavra aleatória sucessivamente.

Ao executar o programa, deve ser carregado em uma TAD as palavras de um arquivo. Para facilitar, será entregue o arquivo contendo todo o dicionário e outro contendo apenas as palavras de tamanho igual a 5.

3.1.1 Menu

A primeira "tela" que o usuário deverá visualizar é a tela do menu que pede o nome do jogador, como se fosse:

```
Informe o nome do jogador

Nome: <Entrada><str>
```

Ao ser lido o nome do jogador, deve ser carregado em uma TAD as estatísticas daquele jogador do arquivo `jogadores.bin` (caso seja um new-player deve criar uma nova linha ao final do arquivo para o jogador)

3.1.2 Escolha de modo

Uma nova interface aparecerá solicitando o modo que o jogador deseja jogar:

```
Jogador <str_name> escolha o modo de jogo:

1- Termo
2- 2-Player
8- Ranking
9- Estatisticas
0- Sair do Jogo
```

OPÇÃO 1 - Termo

Termo: Será sorteada 1 palavra e o jogador terá 6 tentativas de acertá-la.

O MENU ABAIXO CONSISTE EM UM EXEMPLO DO MODO TERMO (**Lembrando que você pode alterar a interface do jogo da maneira que você bem entender, mas mantendo TODOS os elementos contidos no exemplo, seja criativo(a) :)**):

```
################## TERMO ##################

                   | | | | | |


Tentativas Restantes: 6

                                Q W E R T Y U I O P
                                 A S D F G H J K L
                                   Z X C V B N M

Jogador <str_name> digite uma palavra a sua escolha: <entrada><str>
```

Supondo que a palavra sorteada seja PROVA e o usuário entrou PRATO

```
################## TERMO ##################

                   | P | R | (A) | !T | (O) |


Tentativas Restantes: 5

                                Q W E R _ Y U I O P
                                 A S D F G H J K L
                                   Z X C V B N M

Jogador <str_name> digite uma palavra a sua escolha: <entrada><str>
```

Observações:

- AS LETRAS QUE NÃO PERTENCEM A NENHUMA DAS PALAVRAS SERÃO EXCLUÍDAS DO TECLADINHO DA TELA (no exemplo acima a letra 't' não existe).
- Nesta versão, o jogador poderá continuar tentando palavras com letras que não existam. O objetivo de remover a letra do teclado virtual é apenas facilitar a visualização do jogador.
- Bônus track: Você pode implementar o modo difícil, que obriga o jogador a usar as letras já descobertas da palavra.

OPÇÃO 4 - 2-Player

Na opção 2-Player será solicitado o nome do segundo jogador como entrada

```
Informe o nome do jogador 2

Nome: <Entrada><str>
```

Que também terá suas estatísticas carregadas do arquivo (nome não pode ser igual do jogador 1)

O MENU ABAIXO CONSISTE EM UM EXEMPLO DO MODO 2-Player:

```
###################### TERMO ######################
    <str1_name> Vs <str2_name>

                   | | | | | |


Tentativas Restantes: 6

                                Q W E R T Y U I O P
                                 A S D F G H J K L
                                   Z X C V B N M

Jogador <str1_name> digite uma palavra a sua escolha:
```

Como pode ser observado, o modo 2-Player tem a mesma quantidade de tentativas do modo Termo porém:

- O Jogo segue as mesmas regras do jogo Termo;
- Cada jogador tem 50% de chance de ser o primeiro a jogar.
- Caso o jogador não acerte a palavra, a vez de jogar passa para o outro jogador;
- Ganha o jogador que acertar a palavra primeiro;
- Caso nenhum dos dois acerte é considerado um **Empate** (e uma derrota na estatística de ambos os jogadores)!

Erros e Acertos

- Se a letra está no lugar certo: cor Verde (ou caixa alta);

- Se a letra existe naquela palavra, porém está no lugar errado: cor Amarela (ou destacada entre parênteses);
- Se a letra não existe naquela palavra: cor Cinza (ou precedida por uma exclamação).

OPÇÃO 9 - ESTATÍSTICAS

A opção de `estatística` deve carregar os dados do arquivo `jogadores.bin` e imprimir na tela:

```
str_name, played, percent_win, current_streak, max_streak,
1_t
2_t
3_t
4_t
5_t
6_t
dead
```

Onde str_name é o nome do jogador, played é quantas vezes ele jogou, percent_win é a porcentagem de vitórias, current_streak é a sequência atual de vitórias, max_streak é o maior valor de vitórias sequenciais, 1_t até 6_t é a quantidade de vitórias por tentativa (EX: 2_t é a quantidade de vitórias onde ele acertou com 2 tentativas) e dead é a quantidade de derrotas.

OPÇÃO 8 - RANKING

A opção `Ranking` exibe a lista de jogadores naquele PC ordenado por porcentagem de vitórias e em caso de empate, pelo número de vitórias sequenciais.

OPÇÃO 0 - SAIR DO JOGO

Opção usada para sair do jogo, porém ao ser utilizada SEM QUE O JOGADOR TENHA JOGADO o seu current_streak é zerado, a partir do momento que o jogador entra no jogo com seu nome ele tem que jogar e ganhar ao menos 1 modo de jogo para não ter seu current_streak zerado.

Deve aparecer então um aviso na tela caso o jogador tente sair sem ter jogado

```
Você ainda não jogou nenhum modo, se sair agora terá seu current_streak zerado,
deseja continuar?

1-Continuar
2-Sair do Jogo
```

Ganhando o Jogo

Ganha-se o jogo quando o jogador acerta a palavra antes do número máximo de tentativas ou antes que seu adversário (modo 2-Players).

Ao final do jogo é exibido a tela de Estatísticas do jogador atualizada:

```
<str_name>: <played>, <percent_win>%, <current_streak>, <max_streak>
         1: <1_t>
         2: <2_t>
         3: <3_t>
         4: <4_t>
         5: <5_t>
         6: <6_t>
```

```
 7: <7_t>
 8: <8_t>
 9: <9_t>
dd: <dead>
```

Uso de TADs

O trabalho também tem como objetivo o uso correto de TADs e funções. Portanto, você deverá organizar seu código corretamente.

Exemplo: Um TAD `tJogador` que armazena diversas informações que precisarão ser acessadas pelo programa, str_name, played, percent_win, etc. Logo, é um candidato a TAD.

Teste ao decorrer do desenvolvimento do código o comando valgrind para gerenciar a memória e evitar vazamentos (NÃO DEIXE PARA TESTAR O COMANDO COM O CÓDIGO PRONTO)

Organize bem o seu código.

3. Requisitos funcionais de implementação

Este trabalho visa simular um jogo do termo com o modo 2-players (ou multi-jogador). Para isto, o desenvolvedor deve garantir que:

1. Ao inicializar o programa, deverá ser carregado de modo dinâmico na memória o arquivo contendo as palavras (modo texto/leitura, o arquivo de palavras sorteadas (modo texto/leitura/escrita) e o arquivo contendo os jogadores que já jogaram naquele computador (modo binário/leitura/escrita).
2. Cada jogador terá seu nome, que será único, pois será utilizado para armazenar dados estatísticos.
3. O arquivo `jogadores.bin` deve conter as informações de todos os jogadores. É responsabilidade dos desenvolvedores definir o melhor modo de manter este arquivo. Lembre-se, o número de jogadores pode crescer.
4. A palavra sorteada do dia será sempre a mesma enquanto o jogador não acertar ou perder em todas as tentativas. Para fins de debug, essa palavra pode ser exibida na tela, mas na versão final ela deve ser mantida apenas em memória enquanto não for acertada.

4. Arquivos de entrada

O sistema realiza a leitura de um arquivo contendo as palavras, um arquivo contendo as palavras já sorteadas e jogadas e um `.bin` contendo as informações dos jogadores.

Para o arquivo de palavras, são dispostas as informações

```
palavra1
palavra2
palavra3
```

onde a palavra é uma string 5 caracteres.

O arquivo `jogadores.bin` deverá conter todas as informações estatísticas de todos os jogadores.

5. Regras Gerais

O trabalho deverá ser feito **em dupla** e pelos próprios alunos, isto é, os alunos deverão necessariamente conhecer e dominar todos os trechos do código implementados.

Cada dupla deverá trabalhar independente das outras, não sendo permitido a cópia ou compartilhamento de código. Será realizada a verificação automática de plágio. Trabalhos identificados como iguais, em termos de programação, serão penalizados com a nota zero. Isso também inclui a dupla que forneceu o trabalho, sendo, portanto, de sua obrigação a proteção de seu trabalho contra cópias não autorizadas.

6. Entrega do Trabalho

O trabalho deverá ser submetido pelo AVA até as 23:59 do dia 20/06/2022.

O arquivo enviado pelo AVA deve seguir o padrão: PROG2_2022_MATRICULA1_MATRICULA2.zip, substituindo a matrícula pelos respectivos números de matrícula da dupla.

O arquivo zip deve conter:

- `README.md` contendo nome da dupla e com considerações de projeto que julgue necessário, incluindo as decisões de projeto tomadas.
- O código fonte bem comentado, isto é, todos os *structs* e funções devem ter descrições claras do que faz, o que representa cada parâmetro e o que ela retorna (se houver). Pode-se adicionar também comentários que ajude a entender a lógica da questão.
- `Makefile` que permita compilar o código de forma ágil

7. Avaliação

O trabalho será corrigido em dois turnos: correção do professor e entrevista. A correção do professor (CP) gerará uma nota entre 0 e 10 e as entrevistas (E) serão realizadas posteriormente à data de entrega do trabalho e será atribuída uma nota entre 0 e 1.

Alunos que fizeram o trabalho em dupla realizarão a entrevista juntos, porém terão seus desempenhos na entrevista avaliados de forma individual. Pequenas e pontuais correções de código serão permitidas no momento da entrevista, o que poderá acarretar uma CP maior.

A nota final (NF) do trabalho será dada pela seguinte fórmula: NF = E * CP

O trabalho será pontuado de acordo com sua funcionalidade e outros aspectos de implementação, conforme as listas abaixo.

Regra geral

O programa será avaliado em aspectos funcionais e de desenvolvimento. No entanto, a nota total do trabalho CP será baseada no fator de completude do trabalho seguindo seguintes premissas:

1. Programa completo: **1**
2. Apenas o jogo termo (1 jogador). **0,6**
3. Jogo incompleto e aspectos isolados funcionando. **0,3**

Portanto, a CP obtida dos itens abaixo será multiplicada pelo fator de completude.

Aspectos funcionais

1. Inicialização. Capacidade de carregar os arquivos. (5%)
2. Sorteio da palavra correto. (5%)
3. Salvar e carregar informações de um jogador em um arquivo no modo binário. (10%)
4. Uso correto e eficiente de funções.(10%)
5. O jogo segue os fluxos descritos anteriormente corretamente em todos os modos. (30%)

   5.1 Modo termo normal

   5.2 Modo 2-player
6. Exibir estatística de um jogador corretamente;
7. Exibir ranking de jogadores por número de vitórias;
8. Liberar toda memória corretamente ao sair. (10%)

Aspectos de desenvolvimento:

1. Uso correto de TADs para abstração dos elementos envolvidos no trabalho. Menos *main* mais TADs! (10%)
2. Executar o programa com o valgrind, sair corretamente e não ter vazamento de memória. (10%)
3. Qualidade da documentação do código.(10%)