Definição do trabalho da M2

Data de entrega: 20/05/2025 (até 23:59)

Modalidade: grupos de 3 integrantes.

Visão Geral:

O jogo da forca é um jogo em que o jogador tem que acertar qual é a palavra proposta, tendo como dica o número de letras. A cada letra errada, é desenhado uma parte do corpo do enforcado. O jogo termina ou com o acerto da palavra ou com o término do preenchimento das partes corpóreas do enforcado (Wikipedia, 2021).

Descrição:

Para começar o jogo se desenha uma base e um risco correspondente ao lugar de cada letra.

Por exemplo, para a palavra "MERCADO", se escreve: M E R C A D O ------> _ _ _ _ _ _ _

O jogador que tenta adivinhar a palavra deve ir dizendo as letras que podem existir na palavra. Cada letra que ele acerta é escrita no espaço correspondente: M E R C A D O → M _ _ C A _ _

Caso a letra não exista nessa palavra, ele perde uma tentativa (desenha-se uma parte do corpo, iniciando pela cabeça, tronco, braços…)

O jogo é ganho se a palavra é adivinhada. Caso o jogador não descubra qual palavra até esgotar o número de tentativas ele perde.

REGRAS PARA O DESENVOLVIMENTO

O jogo deverá possuir um menu onde será possível escolher:

- Jogar
- Sobre
- Fim

A seguir serão listadas, de trás para frente, o que deve ser implementado em cada parte do menu.

Fim:

Essa é a opção para finalizar o programa. Observe que seu jogo só deve ser encerrado ao selecionar essa opção, caso qualquer outra opção seja escolhida ela deve retornar ao menu no fim de sua execução.

Sobre:

Quando essa opção for selecionada, deverão ser exibidas a equipe de desenvolvimento (o nome de cada membro da equipe), o mês/ano (exemplo: maio/2025) e o nome do professor/disciplina. Também deve ser exibida a regra do uso das letras, por exemplo: usar somente letras maiúsculas nas tentativas.

Jogar:

Essa é a parte onde o jogo acontece de verdade!

Inicie o jogo escolhendo aleatoriamente entre 10 palavras que serão o código que o jogador tentará quebrar. Esse código deverá respeitar as seguintes regras:

- A cada nova partida uma palavra diferente deve ser sorteada; e
- As palavras devem conter 6 caracteres representados por 6 variáveis do tipo char
- Uma vez escolhida a palavra que deverá ser descoberta pelo jogador, você deverá solicitar que o jogador digite uma letra
- O jogador tem 10 tentativas para acertar a palavra completa (todos os seis caracteres)
- Após cada tentativa o jogador deverá ser informado:
    - O número de tentativas restantes
    - O número acertos
    - Os acertos realizados nas posições corretas (Ex: Se a palavra sorteada for PENSAR, ao digitar a letra A, deve ser exibido: _ _ _ _ A _ .

O jogo deve acabar quando o jogador acertar todas as letras ou quando o número de tentativas zerar.

Se o jogador vencer, informe que ele venceu e retorne ao menu.

Se o jogador perder, informe que ele perdeu e retorne ao menu.

Dicas de desenvolvimento:

O código, a seguir, exemplifica o uso das funções `rand()` e `srand()`;

- `rand()` gera um número pseudo-aleatório entre `0` e `RAND_MAX`, mas essa faixa pode ser facilmente alterada com o operador de resto da divisão inteira.
- `srand()` gera uma nova semente aleatória baseada no parâmetro passado entre os parênteses da função. É comum utilizar a função `time()`, pois ela pega o horário do sistema que muda a cada milésimo de segundo. Note que se a função `srand()` não for utilizada a sequência de números pseudo-aleatórios gerados pela função `rand()` será sempre a mesma.

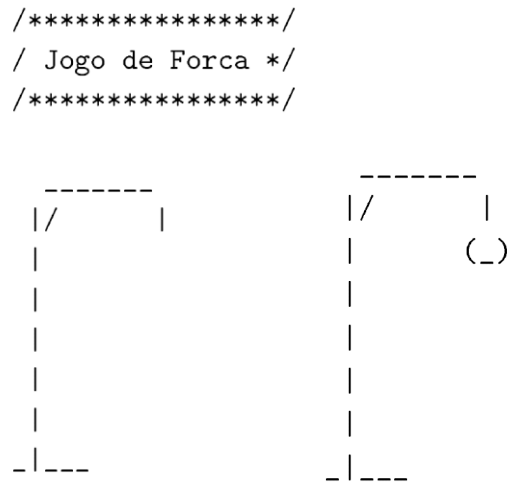
```cpp
#include <iostream>
#include <time.h> ///para habilitar a função time()
using namespace std;

int main()
{
    int r, i;
    ///os números gerados serão sempre iguais, pois usam sempre a mesma semente randomica
    cout << "\nGerando 10 valores randomicos entre 0 e 99 (sempre iguais): \n";
    for(i=0;i<10;i++){
        r=rand()%100; ///o % coloca os números gerados entre 0 e o resto da divisão-1
        cout<<r<<" ";
    }
    ///os números gerados mudarão a cada execução, pois estamos criando uma nova semente
    cout << "\nGerando 10 valores randomicos entre 0 e 99: \n";
    srand(time(NULL));///semente randomica gerada a partir da hora do sistema
    for(i=0;i<10;i++){
        r=rand()%100; ///o % coloca os números gerados entre 0 e o resto da divisão-1
        cout<<r<<" ";
    }
    return 0;
}
```

Outros dois comandos bastante úteis no desenvolvimento de programas no console, são os comandos `system("cls")` e `system("pause")`. Para habilitar a estas funções inclua a biblioteca <stdlib.h>.

- `system("cls")` é um comando que limpa a tela do console (**cl**ear **s**creen). Esse comando é bastante útil, pois em uma tela limpa é mais fácil dar destaque aquilo que se está mostrando no momento.
- `system("pause")` é um comando útil, principalmente quando usado em conjunto com o `system("cls")`, pois ele pausa a execução da aplicação até que o usuário aperte qualquer tecla, bastante útil quando se quer exibir algo antes de limpar a tela para iniciar uma nova execução.

Observação: Outras funcionalidades extras podem ser adicionadas além das obrigatórias, como a representação das partes do corpo do enforcado a cada letra errada; imagens de início do jogo e de representação de Vitória ou Derrota ao final de cada jogo.

Exemplo:



Defesa (Obrigatória e individual):

A defesa é obrigatória e individual, sendo realizada durante o horário da aula na data estipulada. Se algum integrante não estiver presente durante a aula de defesa realizará a defesa em outra data.

***Entregas**:*

- *Postar no Material didático da intranet: Código desenvolvido em C++ no CodeBlocks ou outra IDE, arquivo main.cpp, renomeado com o nome dos integrantes do grupo.*

Critérios de Avaliação:

---

1. Organização e clareza do código = 5% da nota.
2. Identificação dos autores e comentários pertinentes no código = 5% da nota.*
3. Funcionamento correto conforme a especificação = 30% da nota.
4. Uso de recursos da linguagem = 10% da nota.
5. Apresentação e defesa do código = 40% da nota.

** comentários excessivos terão descontos na nota.*

Observações Importantes:

- **Evidências de cópia de outras fontes (colegas, Internet, assistente de IA)** no código fonte resultarão em nota ZERO como nota final deste trabalho, sem possibilidade de recuperação da nota.
- Para o desenvolvimento do código **não poderão ser utilizadas** variáveis compostas (*arrays*), *structs* e funções, com exceção das apresentadas nas Dicas de desenvolvimento.