UNIPÊ – CENTRO UNIVERSITÁRIO DE JOÃO PESSOA

ADILANNE LAURA DE OLIVEIRA BRAGANÇA
FABIANO ANTERO DA SILVA
MATEUS VELOSO FELIPE
NITAI CHARAN ÁLVARES PEREIRA

DESAFIO DE PROGRAMAÇÃO
Jogo da forca

PROFESSOR EDUARDO
INTRODUÇÃO À LINGUAGEM DE PROGRAMAÇÃO
CIÊNCIA DA COMPUTAÇÃO
PERÍODO 2

JOÃO PESSOA
2016.1

DESCRIÇÃO

A aplicação consiste em um tradicional jogo da forca, em que o usuário precisa descobrir qual é a palavra, digitando letra a letra, até que esteja completa. Há um número máximo de erros para cada nível de dificuldade (fácil e difícil).

O usuário tem a opção de Iniciar uma Nova Partida, com um jogador (tendo dificuldade fácil ou difícil, podendo salvar a partida, bater um recorde e entrar no ranking), ou com dois jogadores; Continuar Partida, em que poderá continuar uma partida que foi salva; Ranking Geral, em que aparecerá os dez primeiros recordes obtidos no jogo; e Sair.

O código foi dividido em nove arquivos, sendo esses:

1.arquivamento.c

- Que contém a função `void salvarJogo(T_vetores *vetorUtilizado)`, que salva todo o jogo no arquivo chamado 'dados.dat'.
- E contém a função `void retornoJogoSalvo(T_vetores *vetorUtilizado)`, que abre o arquivo e o lê para continuar uma partida que foi salva.

2.biblioteca.h

- Contém todos os "includes" necessários para a aplicação funcionar;
- Algumas macros definidas, como a '`MAX_RECORDES`', que será usada para delimitar o número máximo de recordes que podem ser salvos no ranking;
- Definições das estruturas T_vetores (que contém vetores e variáveis que serão usadas durante o jogo, e algumas que serão alocadas dinamicamente) e T_Recorde (que possui as informações necessárias para gravar o recorde no ranking);
- Os nomes das funções que serão usadas por mais de um arquivo no programa;
- E a definição de uma macro "LIMPATELA", que realiza a função system("clear").

3.iniciarPartida.c

- Contém a declaração de algumas variáveis de controle que serão usadas no escopo do arquivo;

- A função "`void iniciarPartida(char dificuldade,int doisJogadores,char*strEscolhida, int jogoSalvo)`", que recebe como parâmetros a dificuldade do jogo, se são um ou dois jogadores, o vetor que guarda a palavra digitada pelo usuário caso sejam dois jogadores, e a variável que diz se existe um jogo salvo ou não; e é responsável por inicializar a estrutura de dados que contém os vetores que serão usados, chamar a função que inicia (aloca e limpa) os vetores, chamar a função que sorteia a palavra da

vez, verificar se existe um jogo salvo e abri-lo, rodar o jogo de acordo com a dificuldade, assegurando que não sorteie mais palavras que o número de palavras contidas no vetor da dificuldade, e chamando as funções que imprimem as mensagens para o usuário, que verificam as letras digitadas e as palavras formadas.

- A função "`void mensagens(int QUANTIDADE,T_vetores *vetorUtilizado)`", que é responsável por exibir na tela as mensagens do jogo para que o usuário saiba o que deve fazer ou se algo deu errado, por exemplo a mensagem de que a letra está contida na palavra, ou que o que foi digitado não é uma letra;

- A função "`void verificaLetra(int quantidade,T_vetores *vetorUtilizado,int doisJogadores)`", que recebe como parâmetros a quantidade de tentativas daquele nível, a estrutura de dados do jogo, e a variável que diz se são dois jogadores ou não, e valida a letra digitada, verificando se ela já foi digitada antes, se ela está contida na palavra da vez, ou se o usuário digitou 0 para sair;

- A função "`void verificaPalavras(int quantidade,T_vetores * vetorUtilizado,int doisJogadores,char dificuldade)`", que recebe como parâmetros a quantidade de tentativas daquele nível, a estrutura de dados do jogo, a variável que diz se são dois jogadores ou não, e a dificuldade da partida atual, e verifica se o usuário perdeu ou ganhou o jogo, comparando os vetores, ou verificando a variável "errou" para ver se as tentativas já foram esgotadas, e salva a pontuação que foi obtida cada vez que uma palavra foi acertada.

- E a função "`void flush()`", que limpa o buffer do teclado.

4.novaPartida.c

- Contém a função "`void novaPartida()`", que chama a função que imprime na tela o menu de iniciar uma nova partida (que pergunta se são um ou dois jogadores), e verifica qual foi a opção escolhida. Se for um jogador, chama a função que imprime o menu de dificuldade e a função de iniciar a partida. Se forem dois jogadores, pergunta qual palavra será usada na partida, salva em um vetor temporário, aloca dinamicamente um outro vetor, e copia o conteúdo do vetor temporario para ele. Em seguida, chama a função de iniciar a partida;

- A função "`char menuIniciar()`", que imprime o menu de nova partida (que pergunta se são um ou dois jogadores), lê a opção digitada pelo usuário, verifica se é uma opção válida, e retorna a escolha.

- E a função "`char menuDificuldade()`", que imprime o menu de dificuldade (que pergunta se o usuário deseja jogar no modo fácil ou difícil), ê a opção digitada pelo usuário, verifica se é uma opção válida, e retorna a escolha.

5.principal.c

- Contém a função “`int main(int argc, char const *argv[])`”, que chama as outras funções, capazes de executar a aplicação. Chama a função do menu principal, e verifica qual foi a opção digitada. Se foi a de iniciar nova partida, chama a função novaPartida(). Se foi a de Continuar partida, chama a função iniciarPartida, que verifica se existe um jogo salvo ou não. E se for a do ranking, chama as funções de obter ranking, imprimir ranking, e destruir ranking.

- A função “`char menuPrincipal()`”, que imprime o menu principal do jogo na tela, verifica se a opção digitada pelo usuário é válida, e retorna a escolha.

- A função “`void print_ranking(T_Recorde* ranking)`”, que recebe a estrutura de dados do ranking como parâmetro, e que imprime o ranking na tela, verificando se há ou não um ranking salvo. Se não houver, imprime uma mensagem de que não há registros a exibir.

- E a função “`void print_rpad(char* str, unsigned int columns)`”, que recebe como parâmetros a string a ser impressa e o número de colunas desejado, e posiciona tudo na tela, e serve para organizar graficamente o ranking.

6. ranking.c

- Contém a função “`void salvar_ranking(T_Recorde* ranking)`”, que recebe como parâmetro a estrutura de dados dos recordes, e salva em um arquivo qual foi o recorde, o nome da pessoa que bateu o recorde, o tamanho desse nome, e a pontuação. Depois, fecha o arquivo;

- A função “`int posicao_recorde(T_Recorde* ranking, unsigned int pontuacao)`”, que recebe como parâmetro a estrutura de dados dos recordes e a pontuação, e verifica em qual posição no ranking esta pontuação está, retornando-a;

- A função “`int numero_recorde(unsigned int pontuacao)`”, que recebe como parâmetro a pontuação,^obtém o ranking, obtém a posição do recorde, e retorna 0 ou um número até o número máximo de recordes previamente definido ;

- A função “`void inserir_recorde(char* nome, unsigned int pontuacao)`”, que recebe como parâmetro o nome e a pontuação, e que insere o recorde dentro do ranking, fazendo as verificações necessárias, como se a posição está dentro do número limite de recordes, e destoindo o ranking no final;

- A função “`T_Recorde* obter_ranking()`”, que vai ler de maneira ordenada os recordes

escritos no arquivo de ranking;

- E a função "`void destruir_ranking(T_Recorde* ranking)`", que recebe como parâmetro a estrutura a ser destruída, e libera com *free* todas as alocações dinâmicas que foram feitas.

7. sorteadorDePalavras.c

- Contém a função "`void palavraDaVez(char dificuldade,T_vetores vetorUtilizado,int doisJogadores,char *strEscolhida)`", que recebe como parâmetros a dificuldade da partida, a estrutura com os vetores que estão sendo usados, a variável que diz se são dois jogadores ou não, e o vetor guarda a palavra da vez temporariamente, e que chama a função que vai gerar o vetor de palavras organizado aleatoriamente.

- E a função "`void geraRand(int qtd_palavras,T_vetores *vetorUtilizado,char dificuldade,int doisJogadores,char *strEscolhida)`", que recebe como parâmetros a quantidade de palavras disponíveis no vetor de cada dificuldade, a estrutura com os vetores que estão sendo usados, a dificuldade da partida,a variável que diz se são dois jogadores ou não, e o vetor guarda a palavra da vez temporariamente. Ela vai sortear, a partir da função srand(), números inteiros aleatoriamente, que vão ser colocados em um vetor e serão os índices usados na partida para usar palavras sem se repetir e em ordens diferentes a cada vez que o jogo for rodado.

8. tratamentoDeVetores.c

- Contém a função "`void iniciavetores(T_vetores *vetorUtilizado,int *idxVerificacao)`", que recebe como parâmetro a estrutura de dados que contém os vetores utilizados no jogo, e um índice de verificação (variável de controle). Ela é responsável por alocar dinamicamente esses vetores, limpá-los, e zerar as variáveis de verificação (`*idxVerificacao` e `vetorUtilizado->errou`);

- E a função "`void finalizavetores(T_vetores *vetorUtilizado)`", que recebe como parâmetro a estrutura e que é responsável por fazer um *free* em todos os vetores que foram alocados dinamicamente.

9. verificaRanking.c

- Contém a função "`void verificaRanking(T_vetores *vetorUtilizado)`", que recebe como parâmetro a estrutura de dados que contém os vetores utilizados no jogo, e que é responsável por verificar se a pontuação obtida é um recorde ou não. Para isso, chama a função "numero_recorde" e verifica se o recorde é maior que 0. Se for, pede o nome do jogador, e chama a função "inserir_recorde" para colocá-lo no ranking.

A atividade foi desenvolvida em conjunto, ou seja, algumas das funções foram divididas entre os integrantes do grupo, e grande parte foi realizada com a ajuda uns dos outros.

Nitai ficou com a responsabilidade de desenvolver a função "`void iniciarPartida(char dificuldade,int doisJogadores,char*strEscolhida, int jogoSalvo)`", Fabiano com as funções de arquivamento.c, ou seja a "`void salvarJogo(T_vetores *vetorUtilizado)`" e a "`void retornoJogoSalvo(T_vetores *vetorUtilizado)`", Mateus ficou responsável por adaptar as funções disponibilizadas pelo professor referentes ao ranking ao nosso código, que se encontram nos arquivos ranking.c e verificaRanking.c, e Adilanne ficou responsável pelas funções de tratamentoDeVetores.c, que são a "`void iniciavetores(T_vetores *vetorUtilizado,int *idxVerificacao)`" e a "`void finalizavetores(T_vetores *vetorUtilizado)`".

As demais funções foram desenvolvidas pelo grupo como um todo, de maneira que um começava escrevendo, e os outros iam adicionando código, corrigindo e testando. Algumas funções, como as de imprimir os menus, e outras partes do código, foram aproveitadas do código do desafio da unidade anterior.

A dinâmica de troca de código foi feita na maior parte do tempo através do GitHub, quando o código não estava sendo desenvolvido presencialmente pelos integrantes do grupo.