Técnicas e desenvolvimento de algoritmos (Turma D)

Relatório

Projeto: Jogo da Forca

Repositorio: https://github.com/Fernando-Nazario/JogoDaForca_C

Integrantes:

1. Fernando Nazário de Oliveira (RGM: 38726181)
2. Alik Breno Dones Chacon (RGM: 38536781)
3. Luy Neves Fernandes de Oliveira (RGM: 38445034)
4. Marcelo Alencar Oliveira de Assis (RGM: 37726056)
5. Italo Monteiro Leite (RGM: 39638812)

Descrição:
O programa seleciona aleatoriamente um tema e uma palavra, exibindo uma representação gráfica da forca e linhas que indicam as letras da palavra oculta. O jogador deve tentar adivinhar as letras da palavra. A cada erro, uma parte do boneco na forca é desenhada. Se o desenho da forca for completado, o jogador perde. Caso consiga descobrir a palavra antes disso, o jogador vence a rodada.

Ao vencer, o jogador ganha 10 pontos e pode optar por continuar jogando. A cada rodada vencida consecutivamente, são acumulados mais 10 pontos. Quando o jogador decide desistir ou perde uma rodada, os pontos acumulados são salvos em um ranking.

Dificuldades encontradas:

Banco de palavras: Para permitir o cadastro de novas palavras no jogo da forca, foi necessário usar alocação dinâmica de memória, o que permite expandir o banco de palavras sem limites fixos. Além disso, para organizar as palavras por temas, foi usado o struct, que facilita o agrupamento das palavras relacionadas a cada tema e mantém o sistema mais organizado.
Solução:

```c
//Estrutura do banco de palavras
typedef struct BancoDePalavras {
    char *tema;
    char **palavras;
    int qtd_palavras;
} bdPlvr;
```

Criamos uma struct chamada BancoDePalavras, que contém um char tema para armazenar o nome do tema, um array de palavras para guardar as palavras relacionadas ao tema e um int qtd_palavras para indicar a quantidade de palavras em cada tema.

Adicionar o tema ao Banco de Palavras: Para adicionar temas ao banco de palavras, utilizamos a struct já criada. Era necessário inserir os temas para, em seguida, associar as palavras a eles. Para isso, foi preciso alocar vários structs dinamicamente, com cada struct representando um tema.

Main():

```c
//Obtendo o quantidade de temas, pelos temas predefinido.
int num_temas = sizeof(lista_temas_pred) / sizeof(lista_temas_pred[0]);

//Criar o banco de palavras com base em num_temas
bdPlvr *banco = criarBancoPlvr(num_temas);
```

A main recebe o banco criado em criarBancoPlvr.

criarBancoPlvr():

```c
//Função para criar um banco de palavras
bdPlvr *criarBancoPlvr(int num_temas) {
    bdPlvr *banco = (bdPlvr *)malloc(num_temas * sizeof(bdPlvr)); //Alocando as struct bdPlvr
    //Verificando se a alocação foi feita
    if (!banco) {
        printf("Falha ao alocar memoria para o banco de palavras.\n");
        exit(1);
    }
    //Inicializando tema/palavras/qtd_palavras dos bancos de palavras criados, que são baseado em num_temas.
    for (int i = 0; i < num_temas; i++) {
        banco[i].tema = NULL;
        banco[i].palavras = NULL;
        banco[i].qtd_palavras = 0;
    }
    //Retorna o banco de dados criado.
    return banco;
}
```

Primeiro, é realizada a alocação do banco de palavras com base no número de temas, que até o momento é 10. Em seguida, verifica-se se a memória foi alocada corretamente. Depois, um loop inicializa as variáveis de cada banco, configurando o tema e as palavras como NULL e a quantidade de palavras como 0. Por fim, a função retorna o ponteiro banco.

Adicionar as palavras aos temas:
Com os structs alocados e inicializados, e cada um com seu tema definido, chegou o momento de adicionar as palavras aos respectivos temas.

Main():

```c
//Lista de palavras predefinidas.
char *lista_palavras_pred[][5] = {
    {"Carro", "Moto", "Barco", "Aviao", "Helicoptero"},          // Veiculos
    {"Cachorro", "Gato", "Leao", "Macaco", "Flamingo"},           // Animais
    {"Maca", "Banana", "Laranja", "Mango", "Uva"},                // Frutas
    {"Brasil", "EstadosUnidos", "China", "Franca", "Japao"},     // Paises
    {"Cadeira", "Mesa", "Televisao", "Computador", "Lampada"},    // Objetos
    {"Coracao", "Pulmao", "Cerebro", "Estomago", "Rim"},          // Corpo Humano
    {"Vermelho", "Azul", "Verde", "Amarelo", "Preto"},            // Cores
    {"Violao", "Piano", "Guitarra", "Bateria", "Flauta"},         // Instrumentos Musicais
    {"Futebol", "Basketball", "Volei", "Natacao", "Ciclismo"},    // Esportes
    {"Computador", "Smartphone", "Tablet", "Router", "Impressora"} // Tecnologia
};
```

Lista de palavras pré-definidas.

```c
//Adicionando os temas no banco de palavras
for (int i = 0; i < num_temas; i++) {
    adicionarTema(banco, i, lista_temas_pred[i]); //Tema
    for (int j = 0; j < qtd_palavras; j++) {
        adicionarPalavraAoTema(banco, i, lista_palavras_pred[i][j]); //Adicionando as palavras.
    }
}
```

Foi criado um loop para adicionar as palavras aos seus respectivos bancos de palavras.

adicionarPalavraAoTema():

```c
// Função para adicionar uma palavra a um tema no banco.
void adicionarPalavraAoTema(bdPLvr *banco, int indice_tema, char *palavra) {
    banco[indice_tema].palavras = (char **)realloc(banco[indice_tema].palavras, (banco[indice_tema].qtd_palavras + 1) * sizeof(char
    *)); //Realocando o array dentro de banco de palavras para adicionar mais 1 palavra;

    //Verificar se a realocação foi feita com sucesso
    if (!banco[indice_tema].palavras) {
        printf("Erro ao realocar memoria para palavras.\n");
        exit(1);
    }
    //Copiando a palavra para dneotro do array palavra dentro do banco[indice]
    banco[indice_tema].palavras[banco[indice_tema].qtd_palavras] = strdup(palavra);

    //Verificando se foi copiado com sucesso
    if (!banco[indice_tema].palavras[banco[indice_tema].qtd_palavras]) {
        printf("Falha ao duplicar a palavra.\n");
        exit(1);
    }

    //Aumentando o contador de qtd_palavras dentro do banco[indice]
    banco[indice_tema].qtd_palavras++;
}
```

A função adicionarPalavraAoTema recebe três parâmetros: o ponteiro do banco, o índice do tema e a palavra, que é fornecida uma por vez. Em seguida, é feita a realocação de memória para o array de palavras do banco no índice correspondente, permitindo armazenar mais uma palavra. Após isso, verifica-se se a realocação foi bem-sucedida. A

palavra recebida é então atribuída ao banco, por exemplo, banco[0].palavras[0] recebe o resultado de strdup(palavra), que cria uma cópia da palavra recebida. A seguir, é realizada uma verificação para garantir que o strdup foi executado corretamente. Por fim, a variável de contagem de palavras, como banco[0].qtd_palavras, é incrementada em 1 a cada nova palavra adicionada.

Adicionar tema e palavra do usuário:

Main()

```c
//Loop do menu
while(1){
menu(&banco, &num_temas);
}
```

Loop para o menu

Menu()

```c
//Adicionar novo tema
case 4: {
    system("cls");
    char novo_tema[50]; //Nome do tema.
    int qtd_novas_palavras; //Quantidade de palavras que terá dentro do tema.

    // Verificar se a entrada do usuário é válida.
    while (1) {
        printf("\nQuantas palavras deseja adicionar ao novo tema? "); //Obtendo a quantidade de palavras que o novo tema terá.
        if (scanf(" %d", &qtd_novas_palavras) == 1 && qtd_novas_palavras > 0) {
            while (getchar() != '\n'); // Limpa o buffer de entrada
            break; // Entrada válida, sair do loop
        } else {
            printf("Entrada invalida. Tente novamente: ");
            while (getchar() != '\n'); // Limpa o buffer de entrada
        }
    }

    printf("\nInsira o nome do novo tema: ");   //Obtendo o nome do novo tema

    //Verifica se o que foi digitado pelo usuário não é nulo
    if (fgets(novo_tema, sizeof(novo_tema), stdin) != NULL) {
        novo_tema[strcspn(novo_tema, "\n")] = '\0';

        if (strlen(novo_tema) > 0) {
            //Realocando memória para adicionar um novo tema.
            bdPlvr *novo_banco = realloc(*banco, (*num_temas + 1) * sizeof(bdPlvr));
            //Verificando se a realocação foi feita com sucesso.
            if (!novo_banco) {
                printf("Erro ao realocar banco de palavras.\n");
                return;
            }
            //Incrementando em *banco
            *banco = novo_banco;
```

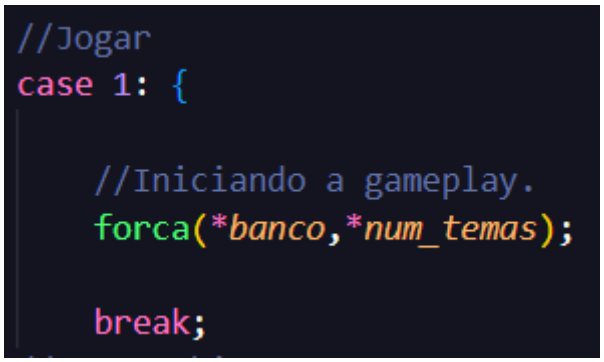
```
            //Adicionando o tema
            adicionarTema(*banco, *num_temas, novo_tema);

            //Loop para adicionar as palavras ao tema
            for (int i = 0; i < qtd_novas_palavras; i++) {
                char palavra[50];
                printf("%dª Palavra: ", i + 1);
                //Verificando se o palavra do usuário não é nulo
                if (fgets(palavra, sizeof(palavra), stdin) != NULL) {
                    palavra[strcspn(palavra, "\n")] = '\0';
                    if (strlen(palavra) > 0) {
                        //Adicionando as palavras ao tema que o usuário criou.
                        adicionarPalavraAoTema(*banco, *num_temas, palavra);
                    } else {
                        printf("Palavra invalida. Tente novamente.\n");
                        i--;
                    }
                }
            }
            (*num_temas)++;
            printf("\nTema '%s' adicionado com sucesso!\n", novo_tema);
        } else {
            printf("O tema nao pode estar vazio.\n");
        }
    }
    break;
```

Depois de escolher a opção correta e entrar no **case 4** do switch-case, o código inicializa a variável novo_tema[50] (para armazenar o nome do novo tema) e solicita a quantidade de palavras que o novo tema terá. Um loop while é usado para garantir que a entrada do usuário seja válida, já que a quantidade de palavras pode ser digitada de forma incorreta, como por exemplo, com caracteres não numéricos. Após obter a quantidade de palavras, o programa solicita o nome do novo tema, verificando se o nome possui mais de 0 caracteres. Em seguida, um novo banco de palavras é criado, usando o ponteiro *banco como referência para alocar dinamicamente um novo banco com o novo tema. É feita uma verificação para garantir que a realocação de memória foi bem-sucedida. Após isso, o novo banco é adicionado ao banco original (incrementando *banco). Finalmente, é chamada a função adicionarTema() para adicionar o novo tema ao banco de dados.

```
//Adicionar um tema ao banco de palavras
void adicionarTema(bdPlvr *banco, int indice_tema, char *tema) {

    banco[indice_tema].tema = strdup(tema); //Adicionando o tema no banco[indice]

    //Verificando se o tema foi copiado para o banco.
    if (!banco[indice_tema].tema) {
        printf("Falha ao duplicar o tema.\n");
        exit(1);
    }

    banco[indice_tema].palavras = NULL; //Ainda não tem palavra
    banco[indice_tema].qtd_palavras = 0; //Ainda não tem palavra
}
```

Após o tema ser adicionado, é hora de adicionar as palavras ao novo tema. Para isso, um loop for é usado, iterando o número de vezes correspondente à quantidade de palavras informada anteriormente (qtd_palavras). Dentro do loop, a palavra é capturada usando fgets, e em seguida, a função adicionarPalavraAoTema() é chamada para adicionar a palavra ao tema. O referencial *num_temas é incrementado para refletir o aumento no número de temas.

Gameplay:



```
//Jogar
case 1: {

    //Iniciando a gameplay.
    forca(*banco,*num_temas);

    break;
```

```
//Função principal para a forca.

void forca(bdPlvr *banco, int num_temas){
    system("cls"); //Limpando terminal

    int erros = 0; //Acumulador de erros.
    int quantAcertos = 0; //Acumulador de acertos.

    char *palavra;
    char *tema;

    int jogar_novamente_escolha = 0; //Varíavel que vai salvar a
escolha do usuário lá na frente.

    int i = 0; //Contador i

    sortearTemaEPalavra(banco, num_temas, &tema, &palavra);
//Sortenado a palavra e o tema atual.

    //Loop para deixar a palavra atual totalmente minúscula.
    while(palavra[i]){
        palavra[i] = tolower(palavra[i]);
        i++;
    }

    // Calculando o tamanho da palavra usando a função strlen
    int tamanho = strlen(palavra);

    // Inicializa o array de tracinhos
    char *tracinhos = calloc(tamanho + 1, sizeof(char));
    for (int i = 0; i < tamanho; i++) {
```

```
        tracinhos[i] = '_';
    }

    // Mostrar o tema
    printf("TEMA: %s\n\n", tema);

    // Exibe a forca inicial (sem corpo)
    escreveBonequinho(erros);

    // Exibe os tracinhos
    for (int i = 0; i < tamanho; i++) {
        printf("%c ", tracinhos[i]);
    }
    printf("\n");

    //Loop principal da gameplay
    while(erros < 8){

        char letra;
        int quantAcertosTemp = 0;

        // Pegando a letra do usuário
        printf("\n\nDigite uma letra: ");
        scanf(" %c", &letra);
        getchar();

        // Verificando se tem a letra na palavra
        for(int i = 0; palavra[i] != '\0'; i++){
            if(palavra[i] == letra){
                quantAcertosTemp++;
                if(tracinhos[i] == '_'){
                    tracinhos[i] = letra;
                    quantAcertos++;
                }
            }
        }

        // Se errar, adiciona mais uma etapa do corpo na forca
        if(quantAcertosTemp == 0){
            erros++;
        }

        // Sai do loop caso acerte
        if(quantAcertos == tamanho) {
            pontos_usuario_atual+=10; //Adicionando 10 pontos a
variável que vai guardar o valor temporariamente para essas
rodadas.
            printf("\nVoce venceu! A palavra era: %s\n\n",
palavra);
            printf("---(+10 pontos)---\n\n");
            printf("Deseja continuar?\n");
            printf("1.Sim\t2.Nao\n\n");
```

```
            printf("---->");

            //Verificando a validade da resposta do usuário
            while(1){
                if(scanf(" %d",&jogar_novamente_escolha) == 1 &&
jogar_novamente_escolha > 0 && jogar_novamente_escolha <=2){
                    break;
                }else{
                    system("cls");
                    printf("Comando invalido. Tente
novamente!\n");
                    printf("Deseja continuar?\n\n");
                    printf("1.Sim\t2.Nao\n\n");
                    printf("---->");
                    while(getchar() != '\n');
                }
            }

            //Verificando o que o usuário escolheu.
            if(jogar_novamente_escolha == 1){
                forca(banco,num_temas); //Reinicia a forca.
                jogar_novamente_escolha = 0; //Reseta a escolha.
            }else{
                //Se a pontuação atual temporaria for maior que
a pontuação do usuário ele substitui.
                if(pontos_usuario_atual > pontuacao_usuario){
                    pontuacao_usuario = pontos_usuario_atual;
                }
                pontos_usuario_atual = 0; //Zera a variavel
temporaria.
                jogar_novamente_escolha = 0; //Reseta a escolha.
            }
            break;
        }

        // Limpa o terminal
        system("cls");

        // Mostrar o tema
        printf("TEMA: %s\n\n", tema);

        // Exibe a forca
        escreveBonequinho(erros);

        // Exibe os tracinhos e letras, caso tenha
        for (int i = 0; i < tamanho; i++) {
            printf("%c ", tracinhos[i]);
        }
        printf("\n");
    }
```

```
    // Caso saia do loop com 8 erros, será imprimido a mensagem
de derrora
    if(erros == 8){
        printf("\nVoce perdeu! A palavra era: %s\n\n", palavra);
        printf("Deseja jogar novamente?\n\n");
        printf("1.Sim\t2.Nao\n\n");
        printf("---->");

        //Verificando validade da resposta do usuário.
        while(1){
            if(scanf(" %d",&jogar_novamente_escolha) == 1 &&
jogar_novamente_escolha > 0 && jogar_novamente_escolha <= 2){
                break;
            }else{
                while(getchar() != '\n');
                system("cls");
                printf("Comando invalido. Tente novamente!\n");
                printf("Deseja jogar novamente?\n\n");
                printf("1.Sim\t2.Nao\n\n");
                printf("---->");
            }
        }

        //Verificando qual foi a resposta do usuário
        if(jogar_novamente_escolha == 1){
            forca(banco,num_temas); //Reinicia a forca.

            //Substitui os pontos caso seja maior para formar um
novo recorde.
            if(pontos_usuario_atual > pontuacao_usuario){
                pontuacao_usuario = pontos_usuario_atual;
                pontos_usuario_atual = 0;
            }

            pontos_usuario_atual = 0;//Zera os pontos
temporarios.
        }else{ //Volta para o menu

            //Caso ele tenha feito mais pontos substitui aqui
também.
            if(pontos_usuario_atual > pontuacao_usuario){
                pontuacao_usuario = pontos_usuario_atual;
                pontos_usuario_atual = 0;
            }

            pontos_usuario_atual = 0; //Zera a pontuação
temporaria.
            jogar_novamente_escolha = 0; //Reseta a escolha.
        }

    }
```

```
    // Limpa o termminal quando acabar
    system("cls");

    // Libera a memória alocada
    free(tracinhos);
}
```

A função forca é a principal função que controla o jogo da forca. Primeiro, o terminal é limpo e são inicializados os acumuladores de erros e acertos, além de ponteiros para armazenar a palavra e o tema. A variável jogar_novamente_escolha é inicializada com 0 para armazenar a escolha do usuário sobre continuar ou não, e o índice i é inicializado com 0. Em seguida, a função sortearTemaEPalavra() é chamada para sortear o tema e a palavra a ser adivinhada. A palavra sorteada é convertida para minúsculas para evitar diferenças entre letras maiúsculas e minúsculas. O tamanho da palavra é calculado e o array tracinhos é alocado dinamicamente com o tamanho da palavra mais 1. O array é preenchido com underscores, representando as letras ainda não descobertas. O tema sorteado é exibido, e a função escreveBonequinho() é chamada para mostrar o boneco da forca com o número de erros. Os tracinhos são exibidos em seguida.
O loop principal do jogo continua enquanto o número de erros for menor que 8. O jogador é solicitado a digitar uma letra, que é verificada na palavra. Se a letra for encontrada, ela substitui o traço correspondente no array tracinhos e o contador de acertos é incrementado. Se a letra não for encontrada, o número de erros aumenta. Se o número de acertos for igual ao tamanho da palavra, o jogador vence e ganha 10 pontos. O jogo pergunta se o jogador deseja continuar jogando. Se a resposta for "Sim", o jogo reinicia; se for "Não", a pontuação é registrada e o jogo termina. Se o jogador fez mais pontos que o recorde, a pontuação é atualizada.
Se o número de erros atingir 8, o jogador perde. A palavra correta é exibida e o jogador é perguntado se deseja jogar novamente. Se escolher "Sim", o jogo reinicia; se escolher "Não", o jogo termina. No final, a memória alocada para os tracinhos é liberada.

escreveBonequinho():

```
// Uma função que, de acordo com os erros, vai mostrar a etapa
que o bonequinho está
void escreveBonequinho(int erros){
    switch(erros) {
        case 0:
            printf("\n +----+\n");
            printf(" |    |\n");
            printf(" |    |\n");
            printf(" |     \n");
            printf(" |     \n");
            printf(" |     \n");
            printf(" |     \n");
            printf(" |     \n");
            printf(" |     \n");
            printf("_|_    \n\n");
            break;
        case 1:
            printf("\n +----+\n");
```

```
    printf(" |    |\n");
    printf(" |    |\n");
    printf(" |  (*_*)\n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf("_|_    \n\n");
    break;
case 2:
    printf("\n +----+\n");
    printf(" |    |\n");
    printf(" |    |\n");
    printf(" |  (*_*)\n");
    printf(" |    |\n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf("_|_    \n\n");
    break;
case 3:
    printf("\n +----+\n");
    printf(" |    |\n");
    printf(" |    |\n");
    printf(" |  (*_*)\n");
    printf(" |    |\n");
    printf(" |  --|\n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf("_|_    \n\n");
    break;
case 4:
    printf("\n +----+\n");
    printf(" |    |\n");
    printf(" |    |\n");
    printf(" |  (*_*)\n");
    printf(" |    |\n");
    printf(" |  --|--\n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf("_|_    \n\n");
    break;
case 5:
    printf("\n +----+\n");
    printf(" |    |\n");
    printf(" |    |\n");
    printf(" |  (*_*)\n");
    printf(" |    |\n");
```

```
            printf(" |  --|--\n");
            printf(" |    |\n");
            printf(" |     \n");
            printf(" |     \n");
            printf("_|_    \n\n");
            break;
        case 6:
            printf("\n +----+\n");
            printf(" |    |\n");
            printf(" |    |\n");
            printf(" |  (*_*)\n");
            printf(" |    |\n");
            printf(" |  --|--\n");
            printf(" |    |\n");
            printf(" |    ^\n");
            printf(" |     \n");
            printf("_|_    \n\n");
            break;
        case 7:
            printf("\n +----+\n");
            printf(" |    |\n");
            printf(" |    |\n");
            printf(" |  (*_*)\n");
            printf(" |    |\n");
            printf(" |  --|--\n");
            printf(" |    |\n");
            printf(" |    ^\n");
            printf(" |     \\\n");
            printf("_|_    \n\n");
            break;
        case 8:
            printf("\n +----+\n");
            printf(" |    |\n");
            printf(" |    |\n");
            printf(" |  (*_*)\n");
            printf(" |    |\n");
            printf(" |  --|--\n");
            printf(" |    |\n");
            printf(" |    ^\n");
            printf(" |   / \\\n");
            printf("_|_    \n\n");
            break;
        default:
            printf("Essa quantidade de erro não é possível");
    }
}
```

sortearTemaPalavra():

```c
//Função para sortear os temas e palavras.
void sortearTemaEPalavra(bdPlvr *banco, int num_temas, char **tema, char **palavra) {

    srand(time(NULL));

    if (num_temas == 0) {
        *tema = NULL;
        *palavra = NULL;
        return;
    }

    // Sorteia um tema aleatoriamente
    int indice_tema = rand() % num_temas;

    // Verifica se o tema tem palavras associadas
    if (banco[indice_tema].qtd_palavras == 0) {
        *tema = NULL;
        *palavra = NULL;
        return;
    }

    // Sorteia uma palavra aleatoriamente dentro do tema selecionado
    int indice_palavra = rand() % banco[indice_tema].qtd_palavras;

    // Retorna os resultados
    *tema = banco[indice_tema].tema;
    *palavra = banco[indice_tema].palavras[indice_palavra];
}
```

Saindo:

```c
case 6:
    system("cls");
    printf("\nSaindo...\n");
    exit(0);
```

Apêndice:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <time.h>
#include <ctype.h>

int pontuacao_usuario = 0; //Pontuação final do usuário.
int pontos_usuario_atual = 0; //Pontuação atual do jogo.

//Estrutura do banco de palavras
typedef struct BancoDePalavras {
    char *tema;
    char **palavras;
    int qtd_palavras;
} bdPlvr;

//Inicializando função sortearTemaEPalavra
void sortearTemaEPalavra(bdPlvr *banco, int num_temas, char
**tema, char **palavra);

//Função para criar um banco de palavras
bdPlvr *criarBancoPlvr(int num_temas) {
    bdPlvr *banco = (bdPlvr *)malloc(num_temas *
sizeof(bdPlvr)); //Alocando as struct bdPlvr
    //Verificando se a alocação foi feita
    if (!banco) {
        printf("Falha ao alocar memoria para o banco de
palavras.\n");
        exit(1);
    }
    //Inicializando tema/palavras/qtd_palavras dos bancos de
palavras criados, que são baseado em num_temas.
    for (int i = 0; i < num_temas; i++) {
        banco[i].tema = NULL;
        banco[i].palavras = NULL;
        banco[i].qtd_palavras = 0;
    }
    //Retorna o banco de dados criado.
    return banco;
}

//Adicionar um tema ao banco de palavras
void adicionarTema(bdPlvr *banco, int indice_tema, char *tema) {

    banco[indice_tema].tema = strdup(tema); //Adicionando o tema
no banco[indice]

    //Verificando se o tema foi copiado para o banco.
    if (!banco[indice_tema].tema) {
```

```
        printf("Falha ao duplicar o tema.\n");
        exit(1);
    }

    banco[indice_tema].palavras = NULL; //Ainda não tem palavra
    banco[indice_tema].qtd_palavras = 0; //Ainda não tem palavra
}

// Função para adicionar uma palavra a um tema no banco.
void adicionarPalavraAoTema(bdPlvr *banco, int indice_tema, char
*palavra) {
    banco[indice_tema].palavras = (char
**)realloc(banco[indice_tema].palavras,
(banco[indice_tema].qtd_palavras + 1) * sizeof(char *));
//Realocando o array dentro de banco de palavras para adicionar
mais 1 palavra;

    //Verificar se a realocação foi feita com sucesso
    if (!banco[indice_tema].palavras) {
        printf("Erro ao realocar memoria para palavras.\n");
        exit(1);
    }
    //Copiando a palavra para dneotro do array palavra dentro do
banco[indice]
    banco[indice_tema].palavras[banco[indice_tema].qtd_palavras]
= strdup(palavra);

    //Verificando se foi copiado com sucesso
    if
(!banco[indice_tema].palavras[banco[indice_tema].qtd_palavras])
{
        printf("Falha ao duplicar a palavra.\n");
        exit(1);
    }

    //Aumentando o contador de qtd_palavras dentro do
banco[indice]
    banco[indice_tema].qtd_palavras++;
}

//Função para sortear os temas e palavras.
void sortearTemaEPalavra(bdPlvr *banco, int num_temas, char
**tema, char **palavra) {

    srand(time(NULL));

    if (num_temas == 0) {
        *tema = NULL;
        *palavra = NULL;
        return;
    }
```

```
    // Sorteia um tema aleatoriamente
    int indice_tema = rand() % num_temas;

    // Verifica se o tema tem palavras associadas
    if (banco[indice_tema].qtd_palavras == 0) {
        *tema = NULL;
        *palavra = NULL;
        return;
    }

    // Sorteia uma palavra aleatoriamente dentro do tema
selecionado
    int indice_palavra = rand() %
banco[indice_tema].qtd_palavras;

    // Retorna os resultados
    *tema = banco[indice_tema].tema;
    *palavra = banco[indice_tema].palavras[indice_palavra];
}

//Funçaõ para liberar memoria alocada para o banco de palavras.
void liberarBanco(bdPlvr *banco, int num_temas) {
    for (int i = 0; i < num_temas; i++) {
        free(banco[i].tema);
        for (int j = 0; j < banco[i].qtd_palavras; j++) {
            free(banco[i].palavras[j]);
        }
        free(banco[i].palavras);
    }
    free(banco);
}

// Uma função que, de acordo com os erros, vai mostrar a etapa
que o bonequinho está
void escreveBonequinho(int erros){
    switch(erros) {
        case 0:
            printf("\n +----+\n");
            printf(" |    |\n");
            printf(" |    |\n");
            printf(" |     \n");
            printf(" |     \n");
            printf(" |     \n");
            printf(" |     \n");
            printf(" |     \n");
            printf(" |     \n");
            printf("_|_    \n\n");
            break;
        case 1:
            printf("\n +----+\n");
            printf(" |    |\n");
            printf(" |    |\n");
```

```
    printf(" |  (*_*)\n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf("_|_    \n\n");
    break;
case 2:
    printf("\n +----+\n");
    printf(" |    |\n");
    printf(" |    |\n");
    printf(" |  (*_*)\n");
    printf(" |    |\n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf("_|_    \n\n");
    break;
case 3:
    printf("\n +----+\n");
    printf(" |    |\n");
    printf(" |    |\n");
    printf(" |  (*_*)\n");
    printf(" |    |\n");
    printf(" |  --|\n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf("_|_    \n\n");
    break;
case 4:
    printf("\n +----+\n");
    printf(" |    |\n");
    printf(" |    |\n");
    printf(" |  (*_*)\n");
    printf(" |    |\n");
    printf(" |  --|--\n");
    printf(" |     \n");
    printf(" |     \n");
    printf(" |     \n");
    printf("_|_    \n\n");
    break;
case 5:
    printf("\n +----+\n");
    printf(" |    |\n");
    printf(" |    |\n");
    printf(" |  (*_*)\n");
    printf(" |    |\n");
    printf(" |  --|--\n");
    printf(" |    |\n");
```

```
            printf(" |     \n");
            printf(" |     \n");
            printf("_|_    \n\n");
            break;
        case 6:
            printf("\n +----+\n");
            printf(" |    |\n");
            printf(" |    |\n");
            printf(" |  (*_*)\n");
            printf(" |    |\n");
            printf(" |  --|--\n");
            printf(" |    |\n");
            printf(" |    ^\n");
            printf(" |     \n");
            printf("_|_    \n\n");
            break;
        case 7:
            printf("\n +----+\n");
            printf(" |    |\n");
            printf(" |    |\n");
            printf(" |  (*_*)\n");
            printf(" |    |\n");
            printf(" |  --|--\n");
            printf(" |    |\n");
            printf(" |    ^\n");
            printf(" |     \\\n");
            printf("_|_    \n\n");
            break;
        case 8:
            printf("\n +----+\n");
            printf(" |    |\n");
            printf(" |    |\n");
            printf(" |  (*_*)\n");
            printf(" |    |\n");
            printf(" |  --|--\n");
            printf(" |    |\n");
            printf(" |    ^\n");
            printf(" |   / \\\n");
            printf("_|_    \n\n");
            break;
        default:
            printf("Essa quantidade de erro não é possível");
    }
}

//Função principal para a forca.
void forca(bdPlvr *banco, int num_temas){
    system("cls"); //Limpando terminal

    int erros = 0; //Acumulador de erros.
    int quantAcertos = 0; //Acumulador de acertos.
```

```
    char *palavra;
    char *tema;

    int jogar_novamente_escolha = 0; //Varíavel que vai salvar a
escolha do usuário lá na frente.

    int i = 0; //Contador i

    sortearTemaEPalavra(banco, num_temas, &tema, &palavra);
//Sortenado a palavra e o tema atual.

    //Loop para deixar a palavra atual totalmente minúscula.
    while(palavra[i]){
        palavra[i] = tolower(palavra[i]);
        i++;
    }

    // Calculando o tamanho da palavra usando a função strlen
    int tamanho = strlen(palavra);

    // Inicializa o array de tracinhos
    char *tracinhos = calloc(tamanho + 1, sizeof(char));
    for (int i = 0; i < tamanho; i++) {
        tracinhos[i] = '_';
    }

    // Mostrar o tema
    printf("TEMA: %s\n\n", tema);

    // Exibe a forca inicial (sem corpo)
    escreveBonequinho(erros);

    // Exibe os tracinhos
    for (int i = 0; i < tamanho; i++) {
        printf("%c ", tracinhos[i]);
    }
    printf("\n");

    //Loop principal da gameplay
    while(erros < 8){

        char letra;
        int quantAcertosTemp = 0;

        // Pegando a letra do usuário
        printf("\n\nDigite uma letra: ");
        scanf(" %c", &letra);
        getchar();

        // Verificando se tem a letra na palavra
        for(int i = 0; palavra[i] != '\0'; i++){
            if(palavra[i] == letra){
```

```
                quantAcertosTemp++;
                if(tracinhos[i] == '_'){
                    tracinhos[i] = letra;
                    quantAcertos++;
                }
            }
        }

        // Se errar, adiciona mais uma etapa do corpo na forca
        if(quantAcertosTemp == 0){
            erros++;
        }

        // Sai do loop caso acerte
        if(quantAcertos == tamanho) {
            pontos_usuario_atual+=10; //Adicionando 10 pontos a
variável que vai guardar o valor temporariamente para essas
rodadas.
            printf("\nVoce venceu! A palavra era: %s\n\n",
palavra);
            printf("---(+10 pontos)---\n\n");
            printf("Deseja continuar?\n");
            printf("1.Sim\t2.Nao\n\n");
            printf("---->");

            //Verificando a validade da resposta do usuário
            while(1){
                if(scanf(" %d",&jogar_novamente_escolha) == 1 &&
jogar_novamente_escolha > 0 && jogar_novamente_escolha <=2){
                    break;
                }else{
                    system("cls");
                    printf("Comando invalido. Tente
novamente!\n");
                    printf("Deseja continuar?\n\n");
                    printf("1.Sim\t2.Nao\n\n");
                    printf("---->");
                    while(getchar() != '\n');
                }
            }

            //Verificando o que o usuário escolheu.
            if(jogar_novamente_escolha == 1){
                forca(banco,num_temas); //Reinicia a forca.
                jogar_novamente_escolha = 0; //Reseta a escolha.
            }else{
                //Se a pontuação atual temporaria for maior que
a pontuação do usuário ele substitui.
                if(pontos_usuario_atual > pontuacao_usuario){
                    pontuacao_usuario = pontos_usuario_atual;
                }
```

```
                pontos_usuario_atual = 0; //Zera a variavel
temporaria.
                jogar_novamente_escolha = 0; //Reseta a escolha.
            }
            break;
        }

        // Limpa o terminal
        system("cls");

        // Mostrar o tema
        printf("TEMA: %s\n\n", tema);

        // Exibe a forca
        escreveBonequinho(erros);

        // Exibe os tracinhos e letras, caso tenha
        for (int i = 0; i < tamanho; i++) {
            printf("%c ", tracinhos[i]);
        }
        printf("\n");
    }

    // Caso saia do loop com 8 erros, será imprimido a mensagem
de derrora
    if(erros == 8){
        printf("\nVoce perdeu! A palavra era: %s\n\n", palavra);
        printf("Deseja jogar novamente?\n\n");
        printf("1.Sim\t2.Nao\n\n");
        printf("---->");

        //Verificando validade da resposta do usuário.
        while(1){
            if(scanf(" %d",&jogar_novamente_escolha) == 1 &&
jogar_novamente_escolha > 0 && jogar_novamente_escolha <= 2){
                break;
            }else{
                while(getchar() != '\n');
                system("cls");
                printf("Comando invalido. Tente novamente!\n");
                printf("Deseja jogar novamente?\n\n");
                printf("1.Sim\t2.Nao\n\n");
                printf("---->");
            }
        }

        //Verificando qual foi a resposta do usuário
        if(jogar_novamente_escolha == 1){
            forca(banco,num_temas); //Reinicia a forca.

            //Substitui os pontos caso seja maior para formar um
novo recorde.
```

```
            if(pontos_usuario_atual > pontuacao_usuario){
                pontuacao_usuario = pontos_usuario_atual;
                pontos_usuario_atual = 0;
            }

            pontos_usuario_atual = 0;//Zera os pontos
temporarios.
        }else{ //Volta para o menu

            //Caso ele tenha feito mais pontos substitui aqui
também.
            if(pontos_usuario_atual > pontuacao_usuario){
                pontuacao_usuario = pontos_usuario_atual;
                pontos_usuario_atual = 0;
            }

            pontos_usuario_atual = 0; //Zera a pontuação
temporaria.
            jogar_novamente_escolha = 0; //Reseta a escolha.
        }

    }

    // Limpa o termminal quando acabar
    system("cls");

    // Libera a memória alocada
    free(tracinhos);
}

//Funcao para exibir o menu principal.
void menu(bdPlvr **banco, int *num_temas) {

    int opcao; //Opção do menu que o usuário vai escolhar logo
em frente.

    char *criadores[] = {"Fernando", "Italo", "Marcelo", "Alik",
"Luy"}; //Nome dos criadores.

    //Enquanto o comando do usuário não for válida ele
continuará exibindo o menu e pedindo o comando, quando o comando
for válido ele dá break.
    while (1) {
        printf("\n=== MENU ===\n");
        printf("1. Jogar\n");
        printf("2. Ver ranking\n");
        printf("3. Listar Temas\n");
        printf("4. Adicionar Novo Tema e Palavras\n");
        printf("5. Creditos\n");
        printf("6. Sair\n");
        printf("Escolha uma opcao: \n");
        printf("---> ");
```

```
        //Verificando se o comando do usuário é um número e
menor igual a 6.
        if (scanf(" %d", &opcao) == 1 && opcao >= 1 && opcao <=
6) {
            while (getchar() != '\n'); // Limpa o buffer
            break;
        } else {
            system("cls");
            printf("\nComando invalido. Tente novamente.\n");
            while (getchar() != '\n'); // Limpa o buffer
        }
    }

    //Switch case da escolha do usuário
    switch (opcao) {

        //Jogar
        case 1: {

            //Iniciando a gameplay.
            forca(*banco,*num_temas);

            break;
        //Ver ranking
        case 2:
            //Mostrando o ranking.
            system("cls");
            printf("\n=== Record ===\n");
            printf("---%dpts---\n",pontuacao_usuario);
            break;

        //Exibir temas
        case 3:
            system("cls");
            //Loop para mostrar os temas.
            printf("\n=== Temas Disponiveis ===\n");
            for (int i = 0; i < *num_temas; i++) {
                printf("%d. %s\n", i + 1, (*banco)[i].tema);
            }
            break;
        }

        //Adicionar novo tema
        case 4: {
            system("cls");
            char novo_tema[50]; //Nome do tema.
            int qtd_novas_palavras; //Quantidade de palavras que
terá dentro do tema.

            // Verificar se a entrada do usuário é válida.
            while (1) {
```

```
                printf("\nQuantas palavras deseja adicionar ao
novo tema? "); //Obtendo a quantidade de palavras que o novo
tema terá.
                if (scanf(" %d", &qtd_novas_palavras) == 1 &&
qtd_novas_palavras > 0) {
                    while (getchar() != '\n'); // Limpa o buffer
de entrada
                    break; // Entrada válida, sair do loop
                } else {
                    printf("Entrada invalida. Tente novamente:
");
                    while (getchar() != '\n'); // Limpa o buffer
de entrada
                }
            }

            printf("\nInsira o nome do novo tema: ");
//Obtendo o nome do novo tema

            //Verifica se o que foi digitado pelo usuário não é
nulo
            if (fgets(novo_tema, sizeof(novo_tema), stdin) !=
NULL) {
                novo_tema[strcspn(novo_tema, "\n")] = '\0';

                if (strlen(novo_tema) > 0) {
                    //Realocando memória para adicionar um novo
tema.
                    bdPlvr *novo_banco = realloc(*banco,
(*num_temas + 1) * sizeof(bdPlvr));
                    //Verificando se a realocação foi feita com
sucesso.
                    if (!novo_banco) {
                        printf("Erro ao realocar banco de
palavras.\n");
                        return;
                    }
                    //Incrementando em *banco
                    *banco = novo_banco;

                    //Adicionando o tema
                    adicionarTema(*banco, *num_temas,
novo_tema);

                    //Loop para adicionar as palavras ao tema
                    for (int i = 0; i < qtd_novas_palavras; i++)
{
                        char palavra[50];
                        printf("%dª Palavra: ", i + 1);
                        //Verificando se o palavra do usuário
não é nulo
```

```
                        if (fgets(palavra, sizeof(palavra),
stdin) != NULL) {
                            palavra[strcspn(palavra, "\n")] =
'\0';
                            if (strlen(palavra) > 0) {
                                //Adicionando as palavras ao
tema que o usuário criou.
                                adicionarPalavraAoTema(*banco,
*num_temas, palavra);
                            } else {
                                printf("Palavra invalida. Tente
novamente.\n");
                                i--;
                            }
                        }
                    }
                    (*num_temas)++;
                    printf("\nTema '%s' adicionado com
sucesso!\n", novo_tema);
                } else {
                    printf("O tema nao pode estar vazio.\n");
                }
            }
            break;
        }

        case 5:
            system("cls");
            printf("\n=== Creditos ===\n");
            for (int i = 0; i < 5; i++) {
                printf("- %s\n", criadores[i]);
            }
            break;

        case 6:
            system("cls");
            printf("\nSaindo...\n");
            exit(0);
    }
}

int main() {
    setlocale(LC_ALL, "");

    int qtd_palavras = 5;

    //Lista de temas predefinidos
    char *lista_temas_pred[] = {
    "Veiculos", "Animais", "Frutas", "Paises", "Objetos", "Corpo
Humano", "Cores", "Instrumentos Musicais", "Esportes",
"Tecnologia"
    };
```

```
    //Lista de palavras predefinidas.
    char *lista_palavras_pred[][5] = {
        {"Carro", "Moto", "Barco", "Aviao", "Helicoptero"},
      // Veiculos
        {"Cachorro", "Gato", "Leao", "Macaco", "Flamingo"},
       // Animais
        {"Maca", "Banana", "Laranja", "Mango", "Uva"},
      // Frutas
        {"Brasil", "EstadosUnidos", "China", "Franca", "Japao"},
     // Paises
        {"Cadeira", "Mesa", "Televisao", "Computador",
"Lampada"},     // Objetos
        {"Coracao", "Pulmao", "Cerebro", "Estomago", "Rim"},
      // Corpo Humano
        {"Vermelho", "Azul", "Verde", "Amarelo", "Preto"},
      // Cores
        {"Violao", "Piano", "Guitarra", "Bateria", "Flauta"},
       // Instrumentos Musicais
        {"Futebol", "Basketball", "Volei", "Natacao",
"Ciclismo"},     // Esportes
        {"Computador", "Smartphone", "Tablet", "Router",
"Impressora"} // Tecnologia
    };

    //Obtendo o quantidade de temas, pelos temas predefinido.
    int num_temas = sizeof(lista_temas_pred) /
sizeof(lista_temas_pred[0]);

    //Criar o banco de palavras com base em num_temas
    bdPlvr *banco = criarBancoPlvr(num_temas);

    //Adicionando os temas no banco de palavras
    for (int i = 0; i < num_temas; i++) {
        adicionarTema(banco, i, lista_temas_pred[i]); //Tema
        for (int j = 0; j < qtd_palavras; j++) {
            adicionarPalavraAoTema(banco, i,
lista_palavras_pred[i][j]); //Adicionando as palavras.
        }
    }

    //Loop do menu
    while(1){
    menu(&banco, &num_temas);
    }

    //Liberando o espaço de memória do banco de palavras.
    liberarBanco(banco, num_temas);
    return 0;
}
```