RELATÓRIO

**INTEGRANTES:**

**1° Integrante: | 2° Integrante:**

**Nome:** João Vitor Fernandesde Souza **| Nome:** Brenno José Vaz Cavalcanti Lacerda

**RGM:** 31719660 **| RGM:** 29407583

**3° Integrante: | 4° Integrante:**

**Nome:** Robert Emanuel **| Nome:** Nicolas Matias Dantas

**RGM:** 31162509 **|** **RGM:** 29984106

INTRODUÇÃO

**DESCRIÇÃO DO JOGO:** O jogo da velha é um jogo e passatempo popular. Um Jogo com regras extremamente simples, que não traz grandes dificuldades para seus jogadores e bem fácil de ser aprendido.

**REGRAS:** No jogo da velha, participam duas pessoas, que jogam um de cada vez se alternando, preenchendo os espaços vazios de um tabuleiro 3 por 3. Cada jogador poderá usar um Símbolo X ou O. Vence o jogador que formar primeiro uma linha de três símbolos iguais, seja ela na vertical, horizontal ou diagonal. No Caso de nenhum jogador formar a linha, significa que empatou ou como é mais conhecido, deu velha.

RESULTADOS

O jogo da velha é um jogo simples, jogado por dois jogadores que escolhem entre dois símbolos (X ou O), que para ganhar, um jogador terá de formar uma linha com três símbolos iguais antes do seu adversário. Caso dê empate, se é dito que o jogo deu velha.

Tivemos dificuldade na implementação de um robô ao jogo. Na criação de um ranking e como associar ele aos modos de jogo. E a solução foi encontrada através de vídeos demonstrativos e através do estudo geral da linguagem.

APÊNDICE

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Jogo da velha (3X3 OU 4X4);

a) Apenas um tabuleiro 3x3 (ou 5x5)

b) Caso não exista o segundo jogador, o computador deve assumir a posição do

oponente;

c) Ranking em arquivo;

d) Deve haver menu com opções de Jogar, Ver Ranking, Créditos e Sair;

e) Utilize matriz com alocação dinâmica;

f) Utilize struct para armazenar dados em memória e depois transferir para arquivo;

g) Controle de jogada deve ser feito utilizando entrada de dados pelo teclado

contendo informação da posição e usuário;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define POSICOES\_PLACAR 10

//inicio

//struct para os movimentos do pc, dados do placar e dos dados do usuario.

struct Move{

int linha, coluna;

};

struct Placar{

int vitorias, derrotas, velhas;

};

struct Jogador {

char nome[20];

int pontuacao;

int peca;

};

//inicio do MENU do jogo

int main(void) {

int single();

int multi();

int Ranking();

int creditos();

int valor;

do{

system("clear");

printf("\n.......... Jogo da Velha ..........\n\n");

printf("1. Jogar \n");

printf("2. Ver Ranking \n");

printf("3. Créditos\n");

printf("4. Sair\n");

printf("\nEscolha uma opção: ");

scanf("%d", &valor);

//abrir uma das opções do usuário

switch(valor){

case 1: printf("\n.......... Escolha o modo de jogo ..........\n"); submenu1(); break;

case 2: printf("ranking"); Ranking(); break;

case 3: printf("creditos"); creditos(); break;

}

}while(valor != 4);

system("clear");

return 0;

}

//abrir o submenu pra escolher o modo de jogo

int submenu1(){

int tipo2;

printf("\n1- Singleplayer\n2- Multiplayer\n\n");

printf("\nEscolha uma opção: ");

scanf("%d", &tipo2);

switch(tipo2){

case 1: printf("\n\n.......... Você escolheu o modo Singleplayer ..........\n"); single(); break;

case 2: printf("\n\n.......... Você escolheu o modo Multiplayer ..........\n"); multi(); break;

default: printf("coloque o valor correto respectivo ao menu que quer selecionar!\n"); submenu1();

}

}

//modo singleplayer x bot pc

int single(){

void dadosUsuario(struct Jogador \*, struct Placar \*);

void mostrarUsuario(struct Jogador \*);

void redefinirTabuleiro(int a[3][3]);

void rodadaInit(int (\*tabuleiro)[3], int, struct Placar \*);

void inserirRanking(struct Jogador);

int calculaPontuacao(struct Placar);

int tabuleiro[3][3]; //Variavel que cria o tabuleiro

int valor; //Variavel dados do usuário

struct Jogador jogador;

struct Placar placar;

//preencher dados do usuario

dadosUsuario(&jogador, &placar);

//Loop das rodadas até jogador desejar sair

do{

system("clear");

redefinirTabuleiro(tabuleiro);

//inicio o loop das rodadas

rodadaInit(tabuleiro, jogador.peca, &placar);

jogador.pontuacao = calculaPontuacao(placar);

mostrarUsuario(&jogador);

printf("\n\n1. Continuar \n\n");

printf("2. Sair\n");

printf("\nEscolha uma opção: ");

scanf("%d", & valor);

}while(valor != 2);

inserirRanking(jogador);

return 0;

}

//Pegar os dados do usuario

void dadosUsuario(struct Jogador \*jogador, struct Placar \*placar){

int valor;

system("clear");

printf("\n.......... Bem vindo ao jogo da velha em C ..........\n");

printf("\nQual seu nome: ");

scanf("%s", (\*jogador).nome);

do{

printf("\nEscolha entre: ");

printf("\n1. X \n");

printf("2. O\n");

printf("\nEscolha uma opção:");

scanf("%d", & valor);

//Analisar a opção do usuario

switch(valor){

case 1:

(\*jogador).peca = 1;

break;

case 2:

(\*jogador).peca = -1;

break;

}

}while(valor > 2 || valor <=0);

//inicializar os placares em 0

(\*placar).velhas = 0;

(\*placar).derrotas = 0;

(\*placar).vitorias = 0;

}

//executar o loop das jogadas no tabuleiro

void rodadaInit(int tabuleiro[3][3], int jogador, struct Placar \*placar){

void desenharTabuleiro(int (\*tabuleiro)[3]);

void rodadaJogador(int (\*tabuleiro)[3], int);

void mostrarPlacar(struct Placar placar);

int existemPossibilidades(int (\*tabuleiro)[3]);

int verificaVencedor(int (\*tabuleiro)[3], int);

void jogadaBot(int (\*tabuleiro)[3], int);

int bot = jogador == 1?-1:1; //Define qual será a peca do bot a partir da escolha do jogador

int vez = rand()%3 + jogador; //Define quem irá começar a jogar

int vencedor = 0;

//Loop para as jogadas

while(existemPossibilidades(tabuleiro)){

system("clear");

mostrarPlacar(\*placar);

desenharTabuleiro(tabuleiro);

int rodada = 0;

if(vez % 2 == 0){

//jogada do usuario

printf("\n\nSua Vez!");

rodadaJogador(tabuleiro, jogador);

rodada = jogador;

}else{

// jogada do bot

printf("\n");

jogadaBot(tabuleiro, bot);

rodada = bot;

}

vez++;

//se alguém vencer

if(verificaVencedor(tabuleiro, rodada)){

system("clear");

if(rodada == jogador){

printf("\nParabéns!!! a vitória é sua \n\n");

vencedor = 1;

(\*placar).vitorias++;

}else{

printf("Você perdeu!!! mais sorte na próxima vez\n");

vencedor = 1;

(\*placar).derrotas++;

}

break;

}

}

//caso dê empate = velha

if(!vencedor){

system("clear");

printf("\n Deu Velha \n\n");

(\*placar).velhas++;

}

desenharTabuleiro(tabuleiro);

mostrarPlacar(\*placar);

}

void rodadaJogador(int tabuleiro[3][3], int jogador){

int verificaJogada(int (\*tab)[3], int, int);

void efetuarJogada(int (\*tab)[3],int, int, int);

int linha, coluna;

//Loop para o usuario definir uma jogada correta

do{

printf("\nInsira a Linha que deseja jogar:");

scanf("%d", &linha);

printf("\nInsira a Coluna que deseja jogar:");

scanf("%d", &coluna);

}while(!verificaJogada(tabuleiro, linha-1, coluna-1));

efetuarJogada(tabuleiro, linha-1, coluna-1, jogador);

}

//Desenhar o tabuleiro

void desenharTabuleiro(int tabuleiro[3][3]){

char converteNumero(int);

int tabuleiroChar[3][3];

for(int i = 0; i <3; i++){

printf("\n %d", i+1);

printf("\t ");

for(int j = 0; j<3; j++){

tabuleiroChar[i][j] = converteNumero(tabuleiro[i][j]);

if(j < 2){

printf(" %c |", tabuleiroChar[i][j]);

}else{

printf(" %c ", tabuleiroChar[i][j]);

}

}

if(i<2){

printf("\n\t ------------");

}

}

printf("\n\n\t 1 2 3 \n");

}

//Recebe tabuleiro, e altera seus valores em nulo

void redefinirTabuleiro(int tab[3][3]){

for(int i = 0; i <3; i++){

for(int j = 0; j<3; j++){

tab[i][j] = 0;

}

}

}

//Recebe um numero e converte

char converteNumero(int valor){

if(valor == 0){

return ' ';

}else{

if(valor == 1){

return 'X';

}else{

if(valor == -1){

return 'O';

}

}

}

return ' ';

}

//Efetuar jogada no tabuleiro

void efetuarJogada(int tabuleiro[3][3], int linha, int coluna, int jogador){

tabuleiro[linha][coluna] = jogador;

}

//verificar se esta disponivel, sendo 1 ou 0

int verificaJogada(int tab[3][3], int x, int y) {

if(x >= 0 && y >= 0 && x <= 3 && y<= 3){

if(tab[x][y] == 0){

return 1;

}

}

return 0;

}

//Verifica se existe um vencedor

int verificaVencedor(int tabuleiro[3][3], int jogador){

// Verificando as linhas para decidir se há um vencedor.

if (tabuleiro[0][0]==jogador

&& tabuleiro[0][1]==jogador

&& tabuleiro[0][2]==jogador) {

printf("\nLinha 1\n");

return 1 ;

}

if (tabuleiro[1][0]==jogador

&& tabuleiro[1][1]==jogador

&& tabuleiro[1][2]==jogador) {

printf("\nLinha 2\n");

return 1 ;

}

if (tabuleiro[2][0]==jogador

&& tabuleiro[2][1]==jogador

&& tabuleiro[2][2]==jogador) {

printf("\nLinha 3\n");

return 1 ;

}

if (tabuleiro[0][0]==jogador

&& tabuleiro[1][0]==jogador

&& tabuleiro[2][0]==jogador) {

printf("\nColuna 1\n");

return 1 ;

}

// Verificando as colunas para decidir se há um vencedor.

if (tabuleiro[0][1]==jogador

&& tabuleiro[1][1]==jogador

&& tabuleiro[2][1]==jogador) {

printf("\nColuna 2\n");

return 1 ;

}

if (tabuleiro[0][2]==jogador

&& tabuleiro[1][2]==jogador

&& tabuleiro[2][2]==jogador) {

printf("\nColuna 3\n");

return 1 ;

}

// Verificando as diagonais decidir se há um vencedor.

if (tabuleiro[0][0]==jogador

&& tabuleiro[1][1]==jogador

&& tabuleiro[2][2]==jogador) {

printf("\nDiagonal principal\n");

return 1 ;

}

if (tabuleiro[0][2]==jogador

&& tabuleiro[1][1]==jogador

&& tabuleiro[2][0]==jogador) {

printf("\nDiagonal secundaria\n");

return 1 ;

}

return 0; //Caso não tenha vencedor, retorna 0

}

//Esta função retorna verdadeiro se houver movimentos no tabuleiro.

//Retorna 0 se não há mais movimentos para jogar.

int existemPossibilidades(int tabuleiro[3][3]){

for (int i = 0; i<3; i++)

for (int j = 0; j<3; j++)

if (tabuleiro[i][j]== 0)

return 1;

return 0;

}

//Codigo referente a analise do Bot

//Analisar e efetuar a jogada do bot

void jogadaBot(int tabuleiro[3][3], int bot){

struct Move movimentoBot(int a[3][3], int);

void efetuarJogada(int (\*tabuleiro)[3], int linha, int coluna, int jogador);

struct Move jogadaBot = movimentoBot(tabuleiro, bot);

efetuarJogada(tabuleiro, jogadaBot.linha, jogadaBot.coluna, bot);

}

//escolhe um caminho possivel para seguir

int minimax(int tabuleiro[3][3], int isMax, int jogador){

int avaliaJogada(int (\*tabuleiro)[3], int);

int pontuacaoJogada = avaliaJogada(tabuleiro, jogador);

// Se o Maximizer ganhou o jogo, devolver a pontuação avaliada

if (pontuacaoJogada == 10){

return pontuacaoJogada;

}

// Se o Minimizer ganhou o jogo, devolver pontuação avaliada

if (pontuacaoJogada == -10){

return pontuacaoJogada;

}

if (existemPossibilidades(tabuleiro)){

return 0;

}

// Se não houver mais movimentos e nenhum vencedor, é velha

if (isMax == 1)

{

int melhor = -1000;

// Percorre todos os caminhos

for (int i = 0; i<3; i++)

{

for (int j = 0; j<3; j++)

{

// Verifique se o caminho está vazio

if (tabuleiro[i][j]== 0)

{

// Fazer o movimento

tabuleiro[i][j] = jogador;

melhor = minimax(tabuleiro, !isMax, jogador);

// Desfaz o movimento

tabuleiro[i][j] = 0;

}

}

}

return melhor;

}

//Se o minimizador se mover

else

{

int melhor = 1000;

// Percorre todos os caminhos

for (int i = 0; i<3; i++)

{

for (int j = 0; j<3; j++)

{

// Verifique se o caminho está vazio

if (tabuleiro[i][j]== 0)

{

// fazer o movimento

tabuleiro[i][j] = jogador == 1 ? -1: 1;

melhor = minimax(tabuleiro, !isMax, jogador);

tabuleiro[i][j] = 0;

}

}

}

return melhor;

}

}

//Esse função avalia se a jogada irá resultar chances de vitoria ou derrota

int avaliaJogada(int tabuleiro[3][3], int jogador){

// Verificando as linhas para vitória

for (int linha = 0; linha<3; linha++)

{

if (tabuleiro[linha][0]==tabuleiro[linha][1] &&

tabuleiro[linha][1]==tabuleiro[linha][2] && tabuleiro[linha][2]!=0)

{

if (tabuleiro[linha][0]==jogador)

return +10;

else

return -10;

}

}

// Verificando colunas para vitória

for (int coluna = 0; coluna<3; coluna++)

{

if (tabuleiro[0][coluna]==tabuleiro[1][coluna] &&

tabuleiro[1][coluna]==tabuleiro[2][coluna] && tabuleiro[2][coluna]!=0)

{

if (tabuleiro[0][coluna]==jogador)

return +10;

else

return -10;

}

}

// Verificando as diagonais para vitória

if (tabuleiro[0][0]==tabuleiro[1][1] && tabuleiro[1][1]==tabuleiro[2][2] && tabuleiro[2][2]!=0)

{

if (tabuleiro[0][0]==jogador)

return +10;

else

return -10;

}

if (tabuleiro[0][2]==tabuleiro[1][1] && tabuleiro[1][1]==tabuleiro[2][0] !=0)

{

if (tabuleiro[0][2]==jogador)

return +10;

else

return -10;

}

//se nenhum ganhou, retorna 0

return 0;

}

//struct para escolher o melhor movimento para o bot

struct Move movimentoBot(int tabuleiro[3][3], int jogador){

int verificaJogada(int (\*tab)[3], int, int);

int pontuacaoJogada = -1000;

struct Move melhorMovimento;

melhorMovimento.linha = -1;

melhorMovimento.coluna = -1;

//Percorrer todas os caminhos, avalia a função minimax para todas os caminhos vazios.

//em conjunto retornar o melhor caminho

for (int i = 0; i<3; i++)

{

for (int j = 0; j<3; j++)

{

// Verifique se o caminho está vazio

if (tabuleiro[i][j]== 0)

{

//fazer omovimento

tabuleiro[i][j] = jogador;

// computar função de avaliação para mover

int moveVal = minimax(tabuleiro, 0, jogador);

// Desfaz o movimento

tabuleiro[i][j] = 0;

// Se o caminho atual for melhor que o avaliado, refaça o movimento

if (moveVal > pontuacaoJogada)

{

melhorMovimento.linha = i;

melhorMovimento.coluna = j;

pontuacaoJogada = moveVal;

}

}

}

}

//Caso não tenha uma melhor jogada, sera definido uma posição aleatoria

if(pontuacaoJogada == 0){

do{

melhorMovimento.linha = rand()%3;

melhorMovimento.coluna = rand()%3;

}while(!verificaJogada(tabuleiro, melhorMovimento.linha,melhorMovimento.coluna));

}

return melhorMovimento;

}

int multi(){

// inicio

int l, c, linha, coluna, jogador, ganhou, jogadas, opcao, sair;

char jogo[3][3];

do{

jogador = 1;

ganhou = 0;

jogadas = 0;

for(l = 0; l < 3; l++){

for(c = 0; c < 3; c++){

jogo[l][c] = ' ';

}

}

do{

printf("\n.......... Bem vindo ao jogo da velha em C ..........\n");

printf("\nJogador 1 = 0\nJogador 2 = X\n\n");

printf("\n\n\t 0 1 2\n\n");

for(l = 0; l < 3; l++){

for(c = 0; c < 3; c++){

if(c == 0)

printf("\t");

printf(" %c ", jogo[l][c]);

if(c < 2)

printf("|");

if(c == 2)

printf(" %d", l);

}

if(l < 2)

printf("\n\t-----------");

printf("\n");

}

// processamento

do{

printf("\nJogador %dº, Digite a linha que deseja jogar: ", jogador);

scanf("%d", &linha);

printf("\nJogador %dº, Digite a coluna que deseja jogar: ", jogador);

scanf("%d", &coluna);

}while(linha < 0 || linha > 2 || coluna < 0 || coluna > 2 || jogo[linha][coluna] != ' ');

if(jogador == 1){

jogo[linha][coluna] = '0';

jogador++;

}

else{

jogo[linha][coluna] = 'X';

jogador = 1;

}

jogadas++;

//saida

if(jogo[0][0] == '0' && jogo[0][1] == '0' && jogo[0][2] == '0' ||

jogo[1][0] == '0' && jogo[1][1] == '0' && jogo[1][2] == '0' ||

jogo[2][0] == '0' && jogo[2][1] == '0' && jogo[2][2] == '0'){

printf("\nParabéns!!! a vitória é sua jogador 1\n");

ganhou = 1;

}

if(jogo[0][0] == 'X' && jogo[0][1] == 'X' && jogo[0][2] == 'X' ||

jogo[1][0] == 'X' && jogo[1][1] == 'X' && jogo[1][2] == 'X' ||

jogo[2][0] == 'X' && jogo[2][1] == 'X' && jogo[2][2] == 'X'){

printf("\nParabéns!!! a vitória é sua jogador 2\n");

ganhou = 1;

}

if(jogo[0][0] == '0' && jogo[1][0] == '0' && jogo[2][0] == '0' ||

jogo[0][1] == '0' && jogo[1][1] == '0' && jogo[2][1] == '0' ||

jogo[0][2] == '0' && jogo[1][2] == '0' && jogo[2][2] == '0'){

printf("\nParabéns!!! a vitória é sua jogador 1\n");

ganhou = 1;

}

if(jogo[0][0] == 'X' && jogo[1][0] == 'X' && jogo[2][0] == 'X' ||

jogo[0][1] == 'X' && jogo[1][1] == 'X' && jogo[2][1] == 'X' ||

jogo[0][2] == 'X' && jogo[1][2] == 'X' && jogo[2][2] == 'X'){

printf("\nParabéns!!! a vitória é sua jogador 2\n");

ganhou = 1;

}

if(jogo[0][0] == '0' && jogo[1][1] == '0' && jogo[2][2] == '0'){

printf("\nParabéns!!! a vitória é sua jogador 1\n");

ganhou = 1;

}

if(jogo[0][0] == 'X' && jogo[1][1] == 'X' && jogo[2][2] == 'X'){

printf("\nParabéns!!! a vitória é sua jogador 2\n");

ganhou = 1;

}

if(jogo[0][2] == '0' && jogo[1][1] == '0' && jogo[2][0] == '0'){

printf("\nParabéns!!! a vitória é sua jogador 1\n");

ganhou = 1;

}

if(jogo[0][2] == 'X' && jogo[1][1] == 'X' && jogo[2][0] == 'X'){

printf("\nParabéns!!! a vitória é sua jogador 2\n");

ganhou = 1;

}

} while(ganhou == 0 && jogadas < 9);

printf("\n\n\t 0 1 2\n\n");

for(l = 0; l < 3; l++){

for(c = 0; c < 3; c++){

if(c == 0)

printf("\t");

printf(" %c ", jogo[l][c]);

if(c < 2)

printf("|");

if(c == 2)

printf(" %d", l);

}

if(l < 2)

printf("\n\t-----------");

printf("\n");

}

if(ganhou == 0)

printf("\nDeu Velha! \n");

printf("\nDigite 1 para jogar novamente e 2 para voltar ao MENU: ");

scanf("%d", &opcao);

}while(opcao == 1);

scanf("%d", &sair);

return 0;

}

//Inicia a Visualização do Ranking, fazendo a leitura do arquivo e apresentando ao usuario

int Ranking(){

void ordenarVetor(struct Jogador \*jogadores, int);

int mostrarUsuarioRanking(struct Jogador);

int sair;

struct Jogador jogadores[POSICOES\_PLACAR]; //Array de jogadores do Placar

//variaveis auxiliares

char nome[20];

int pontuacao;

int ranking\_total = 0;

system("clear");

printf(".......... Ranking ..........");

//ponteiro do arquivo

FILE \*arquivo;

//abrindo o arquivo somente para leitura

arquivo = fopen("placar.txt", "r");

//caso tiver um erro na abertura do arquivo ele não executa as operações

if(arquivo == NULL){

printf("\nErro ao abrir o arquivo!\n");

}else{

//enquanto não for fim de arquivo o looping será executado e será impresso o texto

while(fscanf(arquivo, "%s %d", nome, &pontuacao)!=EOF){

strcpy(jogadores[ranking\_total].nome, nome);

jogadores[ranking\_total].pontuacao = pontuacao;

ranking\_total++;

}

//fecha o arquivo

fclose(arquivo);

// testa se existem pessoas no ranking e ordena para apresenta-lo

if(ranking\_total>0){

ordenarVetor(jogadores, ranking\_total);

for(int i=0; i < ranking\_total; i++){

//Fazer a apresentação

mostrarUsuarioRanking(jogadores[i]);

}

}

}

//travar o loop

printf("\n\nSelecione 1 para voltar:");

scanf("%d", &sair);

return 0;

}

//Fazer a ordenação dos ranking

void ordenarVetor(struct Jogador jogadores[POSICOES\_PLACAR], int ranking\_total){

int status;

do{

status = 0;

for(int i=ranking\_total; i>0; i--){

if(jogadores[i].pontuacao> jogadores[i-1].pontuacao){

struct Jogador aux;

aux = jogadores[i];

jogadores[i] = jogadores[i-1];

jogadores[i-1] = aux;

status = 1;

}

}

}while(status);

}

//Adicionar Jogador ao arquivo de ranking

void inserirRanking(struct Jogador jogador){

FILE \*arquivo;

//abrindo o arquivo

arquivo = fopen("placar.txt", "a");

//salvando dados no aquivo

fprintf(arquivo, "\n%s %d", jogador.nome, jogador.pontuacao);

//fechando o arquivo

fclose(arquivo);

}

//Apresenta os dados do usuario

void mostrarUsuario(struct Jogador \*jogador){

printf("\n---------------------------");

printf("\nNome: %s \t", (\*jogador).nome);

printf("Pontuação: %d\n", (\*jogador).pontuacao);

}

//Apresenta os dados do usuario

int mostrarUsuarioRanking(struct Jogador jogador){

for(int i = 0; i < strlen(jogador.nome); i++){

if((jogador.nome[i] < 'a' && jogador.nome[i] > 'z') || (jogador.nome[i] < 'A' && jogador.nome[i] > 'Z') ||

(jogador.pontuacao > 10000) || (jogador.pontuacao < 0)){

return 0;

}

}

printf("\n\n%s \t", jogador.nome);

printf("%d\n", jogador.pontuacao);

printf("-----------------------------");

return 1;

}

//Apresenta o placar

void mostrarPlacar(struct Placar placar){

printf("\n| Você || Velha || Bot |\n");

printf("| %d || %d || %d | \n\n", placar.vitorias, placar.velhas, placar.derrotas);

}

//Recebe o placar e faz o calculo da sua pontuacao

int calculaPontuacao(struct Placar placar){

int pontuacao = 0;

int vitorias = placar.vitorias \* 3;

int derrotas = placar.derrotas \* 2;

int empates = placar.velhas;

pontuacao = ((vitorias + empates)-derrotas);

//Caso a pontuacao seja menor que 0, será retornado o valor 0

return pontuacao > 0 ? pontuacao : 0;

}

int creditos(){

int sair;

printf("\n.......... Créditos ..........\n\n");

printf("Jogo criado e desenvolvido por: \n");

printf("\nNome/RGM - Robert Emanuel /31162509");

printf("\nNome/RGM - Brenno José Vaz C. Lacerda /29407583");

printf("\nNome/RGM - Nicolas Matias Dantas /29984106");

printf("\nNome/RGM - João Vitor F. de Souza /31719660");

printf("\n\nSelecione 1 para voltar: ");

scanf("%d", &sair);

return 0;

}