**Relatório**

Lista de integrantes:

Lucas Coutinho Lira

RGM: 27592308

João Victor dos Santos Vargens

RGM: 27737161

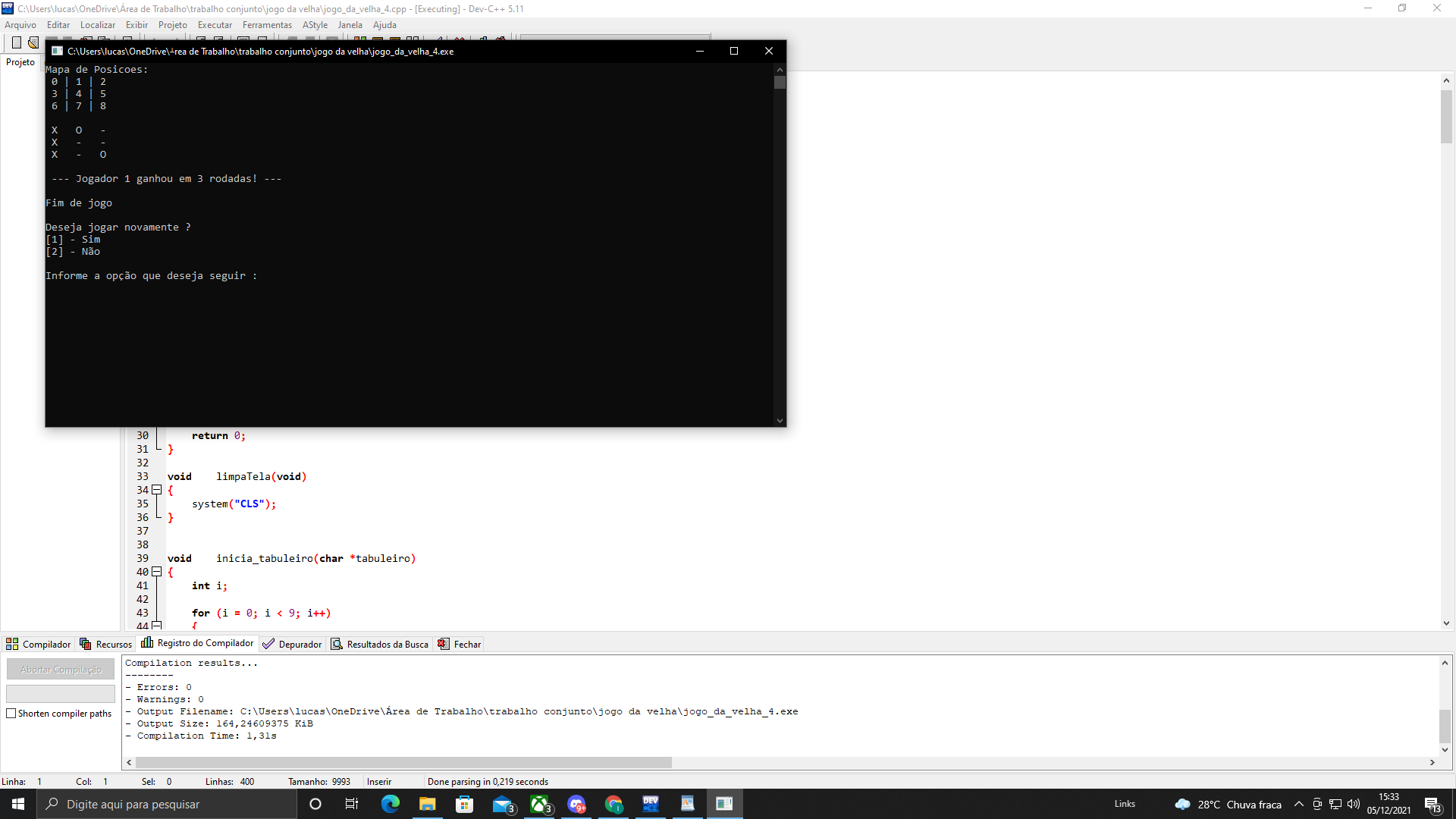
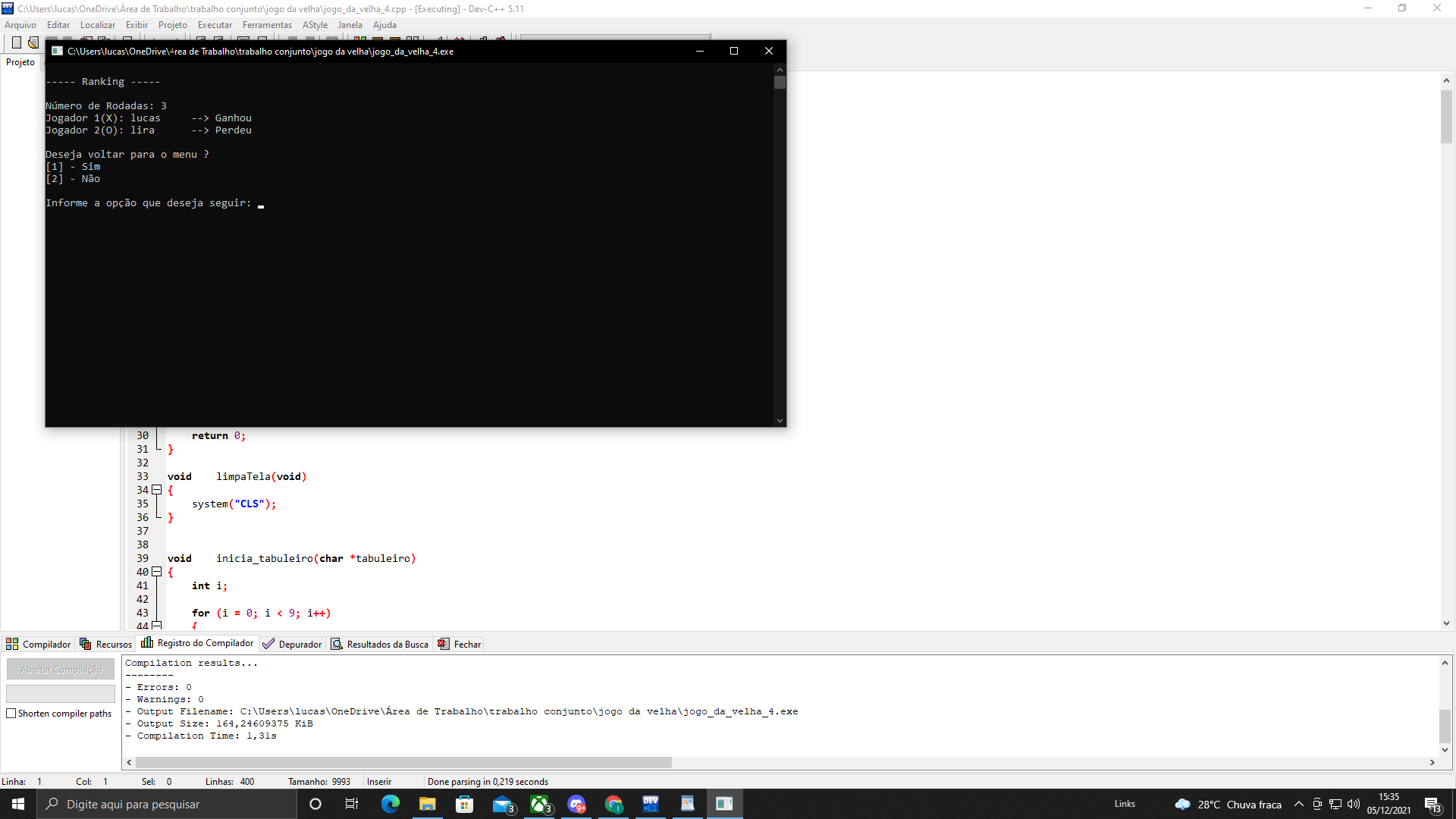
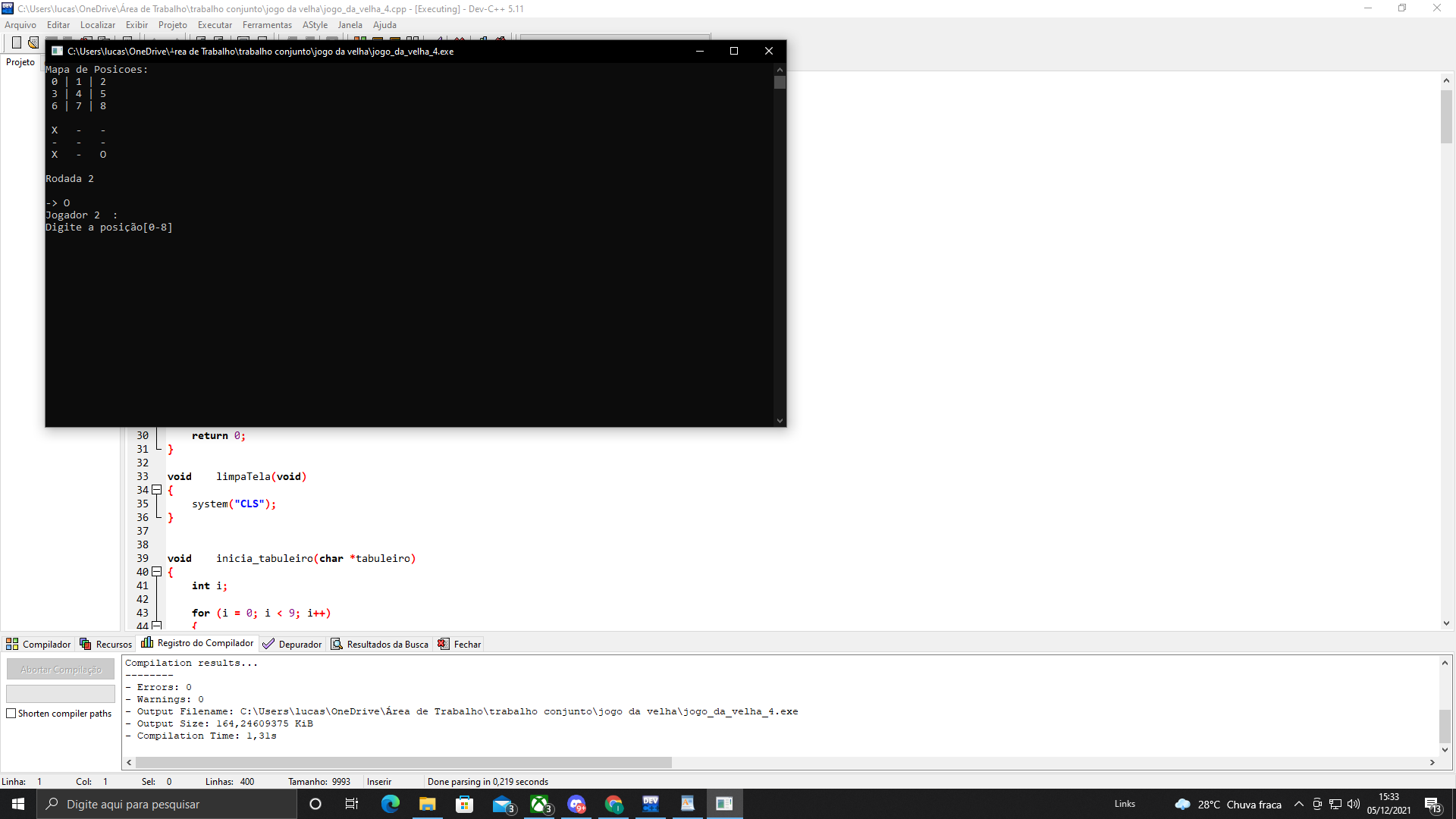
Hiago Maciel Aquino de Macêdo

RGM : 28180607

**Sobre o jogo:** Jogo da velha é um jogo e/ou passatempo popular. É um jogo de regras extremamente simples, que não traz grandes dificuldades para seus jogadores e é facilmente aprendido. A origem é desconhecida, com indicações de que pode ter começado no antigo Egito, onde foram encontrados *tabuleiros* esculpidos na rocha, que teriam mais de 3.500 anos.

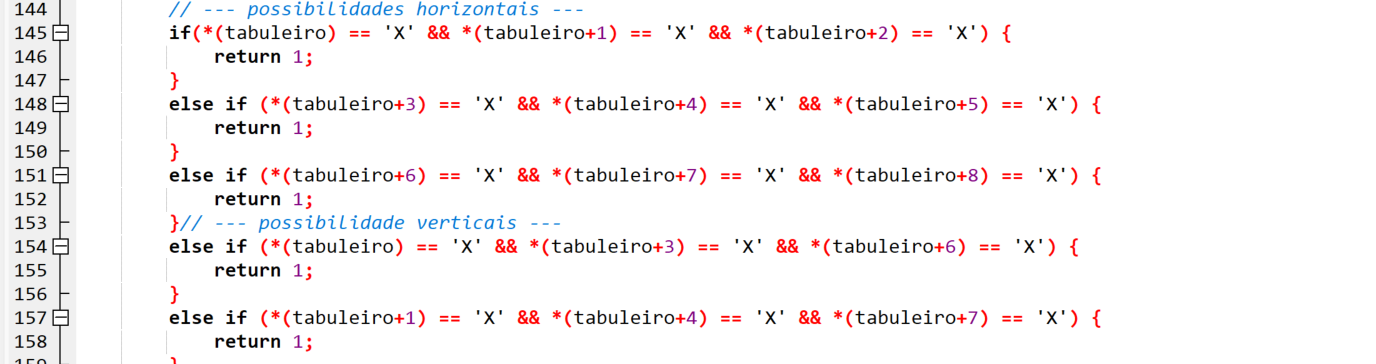
**Regras do jogo:** No modo básico do jogo, participam duas pessoas, que jogam alternadamente, preenchendo cada um dos espaços vazios. Cada participante deve usar um símbolo (X ou O). Vence o jogador que conseguir formar primeiro uma linha com três símbolos iguais, seja ela na horizontal, vertical ou diagonal.

**Resultados:** Para criarmos o tabuleiro criamos duas funções: “Void inicia\_tabulerio e Void exibe\_tabuleiro” e assim deixando o tabuleiro visível durante o jogo, em seguida criamos a função Void jogo, onde tudo que aconteceria no jogo estaria dentro dessa função e assim utilizando varios if´s e else´s para verificações de posições de jogadas no tabuleiro e indicando se a jogada pode ser realizada ou não e também determinando qual jogador ganhou a partida.

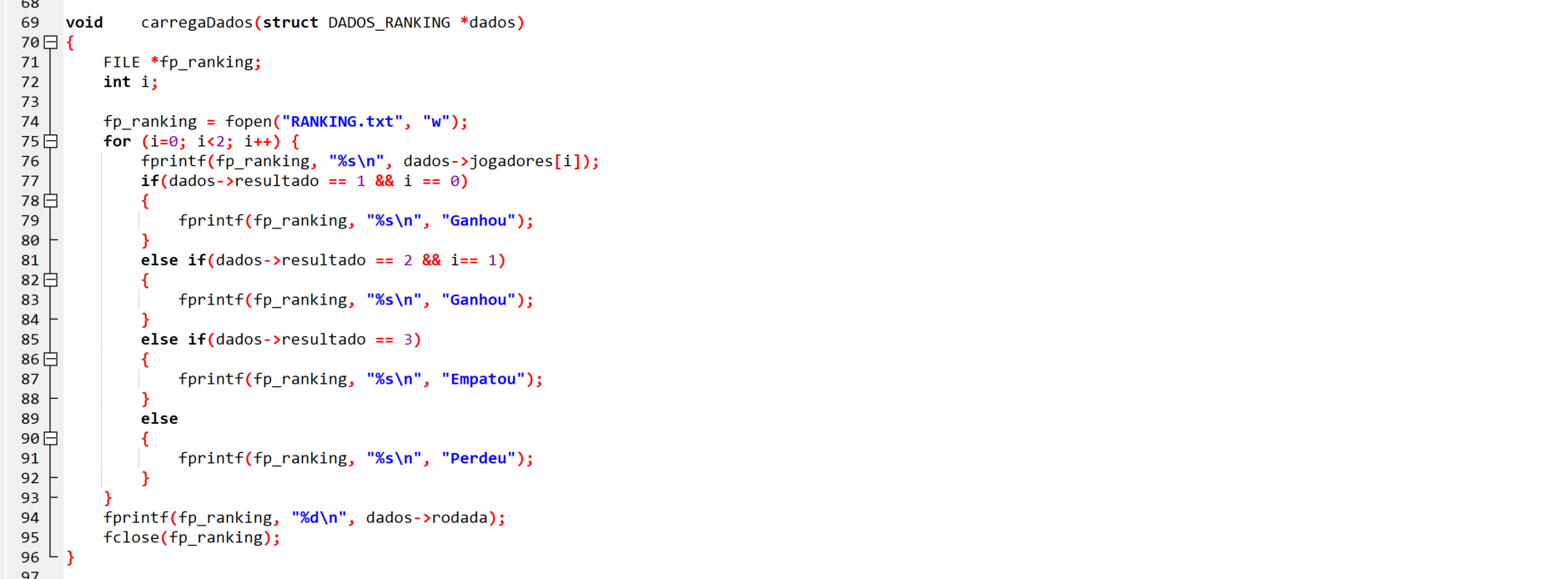


**Dificuldades:**

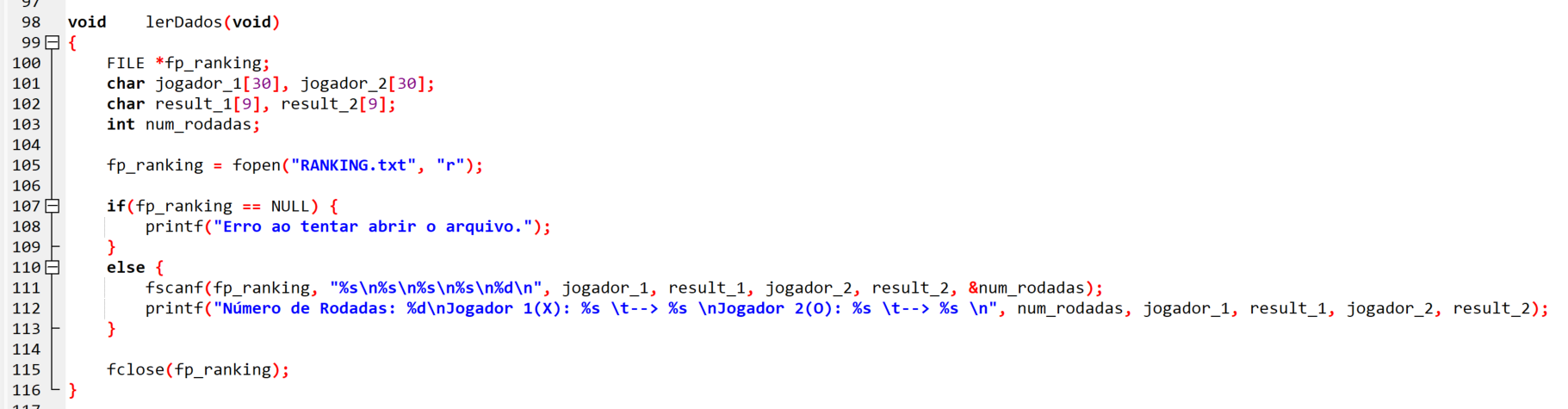
Na função checarJogo seria necessário checar as posições sem ter que mudar completamente o endereço inicial do que estava guardado dentro do ponteiro tabuleiro ou de ter que alterar o valor que estava naquele endereço,

já que tabuleiro+=1 iria mudar definitivamente o endereço e \*tabuleiro+=1 ou \*tabuleiro+1 iria mudar o valor que estava naquele endereço.Para resolver isso foi utilizado parênteses para mudar a prioridade de operações, demonstrado, por exemplo, como \*(tabuleiro+1) no código e assim poder mudar de endereço de uma forma não definitiva e, também, evitar de fazer operações de soma com o valor que está naquele endereço.

Na criação do sistema de ranking, para que guardasse, em um arquivo, os nomes do jogador 1 e jogador 2, o resultado da partida e o número de rodadas, e depois fosse possível ler os dados e apresentar para o usuário. A solução encontrada seria de criar uma função para escrever os dados (função carregaDados) e outra para ler os dados (função lerDados), assim como, formatar a escrita dos dados, no arquivo, com um padrão para que na leitura dos dados pudesse fazer o uso da função fscanf para ler os dados de uma forma formatada, dessa forma, tendo certeza de que leu tudo que é necessário para o ranking. A Formatação utilizada foi a de deixar cada uma das informações em uma linha diferente, na ordem: jogador1, resultado1, jogador2, resultado2 e , por último, o número de rodadas, sendo representado no código, na hora da leitura, como fscanf(fp\_ranking, "%s\n%s\n%s\n%s\n%d\n", jogador\_1, result\_1, jogador\_2, result\_2, &num\_rodadas);



(escrevendo os dados utilizando a função carregaDados)



(lendo os dados utilizando a função lerDados)

Código fonte:

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <string.h>

struct DADOS\_RANKING {

char jogadores[2][30];

int resultado;

int rodada;

};

void limpaTela(void);

void inicia\_tabuleiro(char \*tabuleiro);

void exibe\_tabuleiro(char \*tabuleiro);

void carregaDados(struct DADOS\_RANKING \*dados);

void lerDados(void);

int checarJogo(char \*tabuleiro, char \*reset\_tabuleiro, int turno);

void jogo(struct DADOS\_RANKING ranking);

void menu\_principal(void);

int main(void)

{

setlocale (LC\_ALL, "Portuguese");

menu\_principal();

printf("\nPrograma Encerrado.\n");

return 0;

}

void limpaTela(void)

{

system("CLS");

}

void inicia\_tabuleiro(char \*tabuleiro)

{

int i;

for (i = 0; i < 9; i++)

{

\*tabuleiro = '-';

if (i < 8) {

tabuleiro++;

}

}

}

void exibe\_tabuleiro(char \*tabuleiro)

{

int i;

printf("Mapa de Posicoes:\n 0 | 1 | 2\n 3 | 4 | 5\n 6 | 7 | 8\n\n");

for (i = 0; i < 9; i++)

{

printf(" %c ", \*tabuleiro);

if (i < 8) {

tabuleiro++;

}

if (i == 2 || i == 5) {

printf("\n");

}

}

}

void carregaDados(struct DADOS\_RANKING \*dados)

{

FILE \*fp\_ranking;

int i;

fp\_ranking = fopen("RANKING.txt", "w");

for (i=0; i<2; i++) {

fprintf(fp\_ranking, "%s\n", dados->jogadores[i]);

if(dados->resultado == 1 && i == 0)

{

fprintf(fp\_ranking, "%s\n", "Ganhou");

}

else if(dados->resultado == 2 && i== 1)

{

fprintf(fp\_ranking, "%s\n", "Ganhou");

}

else if(dados->resultado == 3)

{

fprintf(fp\_ranking, "%s\n", "Empatou");

}

else

{

fprintf(fp\_ranking, "%s\n", "Perdeu");

}

}

fprintf(fp\_ranking, "%d\n", dados->rodada);

fclose(fp\_ranking);

}

void lerDados(void)

{

FILE \*fp\_ranking;

char jogador\_1[30], jogador\_2[30];

char result\_1[9], result\_2[9];

int num\_rodadas;

fp\_ranking = fopen("RANKING.txt", "r");

if(fp\_ranking == NULL) {

printf("Erro ao tentar abrir o arquivo.");

}

else {

fscanf(fp\_ranking, "%s\n%s\n%s\n%s\n%d\n", jogador\_1, result\_1, jogador\_2, result\_2, &num\_rodadas);

printf("Número de Rodadas: %d\nJogador 1(X): %s \t--> %s \nJogador 2(O): %s \t--> %s \n", num\_rodadas, jogador\_1, result\_1, jogador\_2, result\_2);

}

fclose(fp\_ranking);

}

int checarJogo(char \*tabuleiro, char \*reset\_tabuleiro, int turno)

{

/\*

Retorna (0) se ninguem tiver ganhado ou empatado

Retorna (1) se o jogador 1 tiver ganhado

Retorna (2) se o jogador 2 tiver ganhado

Retorna (3) se empatou

\*/

int i;

int qtd\_posicoes\_restantes=0;

for (i = 0; i < 9; i++)

{

if (\*tabuleiro == '-') {

qtd\_posicoes\_restantes++;

}

if (i < 8) {

tabuleiro++;

}

}

tabuleiro = reset\_tabuleiro;

// Checando se ganhou ou empatou

if (turno == 1)

{

// --- possibilidades horizontais ---

if(\*(tabuleiro) == 'X' && \*(tabuleiro+1) == 'X' && \*(tabuleiro+2) == 'X') {

return 1;

}

else if (\*(tabuleiro+3) == 'X' && \*(tabuleiro+4) == 'X' && \*(tabuleiro+5) == 'X') {

return 1;

}

else if (\*(tabuleiro+6) == 'X' && \*(tabuleiro+7) == 'X' && \*(tabuleiro+8) == 'X') {

return 1;

}// --- possibilidade verticais ---

else if (\*(tabuleiro) == 'X' && \*(tabuleiro+3) == 'X' && \*(tabuleiro+6) == 'X') {

return 1;

}

else if (\*(tabuleiro+1) == 'X' && \*(tabuleiro+4) == 'X' && \*(tabuleiro+7) == 'X') {

return 1;

}

else if (\*(tabuleiro+2) == 'X' && \*(tabuleiro+5) == 'X' && \*(tabuleiro+8) == 'X') {

return 1;

}// --- possibilidades diagonais ---

else if (\*(tabuleiro) == 'X' && \*(tabuleiro+4) == 'X' && \*(tabuleiro+8) == 'X') {

return 1;

}

else if (\*(tabuleiro+2) == 'X' && \*(tabuleiro+4) == 'X' && \*(tabuleiro+6) == 'X') {

return 1;

}

else if (qtd\_posicoes\_restantes == 0) { // Empatou

return 3;

}

else { // ninguem resultado ou empatou nessa rodada

return 0;

}

}

else

{

// --- possibilidades horizontais ---

if(\*(tabuleiro) == 'O' && \*(tabuleiro+1) == 'O' && \*(tabuleiro+2) == 'O') {

return 2;

}

else if (\*(tabuleiro+3) == 'O' && \*(tabuleiro+4) == 'O' && \*(tabuleiro+5) == 'O') {

return 2;

}

else if (\*(tabuleiro+6) == 'O' && \*(tabuleiro+7) == 'O' && \*(tabuleiro+8) == 'O') {

return 2;

}// --- possibilidade verticais ---

else if (\*(tabuleiro) == 'O' && \*(tabuleiro+3) == 'O' && \*(tabuleiro+6) == 'O') {

return 2;

}

else if (\*(tabuleiro+1) == 'O' && \*(tabuleiro+4) == 'O' && \*(tabuleiro+7) == 'O') {

return 2;

}

else if (\*(tabuleiro+2) == 'O' && \*(tabuleiro+5) == 'O' && \*(tabuleiro+8) == 'O') {

return 2;

}// --- possibilidades diagonais ---

else if (\*(tabuleiro) == 'O' && \*(tabuleiro+4) == 'O' && \*(tabuleiro+8) == 'O') {

return 2;

}

else if (\*(tabuleiro+2) == 'O' && \*(tabuleiro+4) == 'O' && \*(tabuleiro+6) == 'O') {

return 2;

}

else if (qtd\_posicoes\_restantes == 0) { // Empatou

return 3;

}

else { // ninguem resultado ou empatou nessa rodada

return 0;

}

}

}

void jogo(struct DADOS\_RANKING \*ranking) //onde o jogo fica

{

char \*tabuleiro, \*reset\_tabuleiro;

int posicao\_jogada;

int estadodejogo = 1;

int checar\_jogo;

int rodada = 1;

int turno = 0;

int opcao;

tabuleiro = (char\*) malloc(sizeof(char) \* 9);

reset\_tabuleiro = tabuleiro;

ranking->rodada = 1;

inicia\_tabuleiro(tabuleiro);

while(rodada < 10 && estadodejogo == 1 )

{

tabuleiro = reset\_tabuleiro; // Resetando o endereço do tabuleiro

exibe\_tabuleiro(tabuleiro);

printf("\n\nRodada %d\n\n", ranking->rodada);

if (turno == 0) {

printf("-> X");

}

else{

printf("-> O");

}

printf("\n%s :\nDigite a posição[0-8]\n", ranking->jogadores[turno]);

scanf("%d", &posicao\_jogada);

fflush(stdin);

if (posicao\_jogada >= 0 && posicao\_jogada <= 8) {

tabuleiro += posicao\_jogada;

}

else {

limpaTela();

printf("----- Erro! Posição INVÁLIDA, digite uma posição válida. -----\n\n");

continue;

}

if (\*tabuleiro != '-') {

limpaTela();

printf("----- Erro! Posição está OCUPADA, escolha outra posição. -----\n\n");

continue;

}

else if(turno == 0) {

limpaTela();

\*tabuleiro = 'X';

turno++;

}

else {

limpaTela();

\*tabuleiro = 'O';

turno--;

ranking->rodada++;

}

tabuleiro = reset\_tabuleiro; // Resetando o endereço do tabuleiro

checar\_jogo = checarJogo(tabuleiro, reset\_tabuleiro, turno);

if (checar\_jogo == 1) {

ranking->resultado = 1;

exibe\_tabuleiro(tabuleiro);

printf("\n\n --- %s ganhou em %d rodadas! ---\n", ranking->jogadores[0], ranking->rodada);

carregaDados(ranking);

estadodejogo = 0;

}

else if (checar\_jogo == 2) {

ranking->resultado = 2;

exibe\_tabuleiro(tabuleiro);

printf("\n\n --- %s ganhou em %d rodadas! ---\n", ranking->jogadores[1], ranking->rodada);

carregaDados(ranking);

estadodejogo = 0;

}

else if (checar\_jogo == 3) {

ranking->resultado = 3;

exibe\_tabuleiro(tabuleiro);

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

carregaDados(ranking);

estadodejogo = 0;

}

else {

estadodejogo = 1;

}

}

free(tabuleiro);

printf("\nFim de jogo\n");

printf("\nDeseja jogar novamente ?\n[1] - Sim\n[2] - Não\n");

printf("\nInforme a opção que deseja seguir : ");

scanf("%d", &opcao);

if (opcao == 1){

limpaTela();

jogo(ranking);

} else {

limpaTela();

menu\_principal();

}

}

void menu\_principal(void)

{

struct DADOS\_RANKING dadosJogo, \*ranking;

int opcao = 0;

int i;

int len;

ranking = &dadosJogo;

//menu do jogo

do

{

//limpaTela();

printf("Bem vindo ao jogo da velha\nEscolha uma das opções para prosseguir\n");

printf("\n[1] - Iniciar o jogo\n");

printf("[2] - Sobre o jogo\n");

printf("[3] - Créditos\n");

printf("[4] - Ranking\n");

printf("[5] - Sair do jogo\n");

printf("\nInforme a opção que deseja seguir : ");

scanf("%d", &opcao);

fflush(stdin);

switch(opcao){

case 1:

limpaTela();

printf("\nJogo iniciado!\n\n");

for (i = 0; i < 2; i++)

{

printf("Informe o nome do Jogador %d: ", i+1);

fgets(ranking->jogadores[i], 30, stdin);

len = strlen(ranking->jogadores[i]);

if (ranking->jogadores[i][len -1] == '\n') {

ranking->jogadores[i][len -1] = 0; // Removendo o Enter(\n) que o fgets põe

}

}

jogo(ranking);

break;

case 2:

limpaTela();

printf("\nO jogo da velha ou jogo do galo ou três em linha é um jogo e/ou passatempo popular.\n");

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

printf("\nDeseja voltar para o menu ?\n[1] - Sim\n[2] - Não\n");

printf("\nInforme a opção que deseja seguir: ");

scanf("%d", &opcao);

if (opcao == 1){

limpaTela();

menu\_principal();

continue;

}

break;

case 3:

limpaTela();

printf("\nFeito por Lucas Lira, João Victor Vargens, Rayan Antonio e Hiago Maciel\n");

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

printf("\nDeseja voltar para o menu ?\n[1] - Sim\n[2] - Não\n");

printf("\nInforme a opção que deseja seguir: ");

scanf("%d", &opcao);

if (opcao == 1){

limpaTela();

menu\_principal();

}

break;

case 4:

limpaTela();

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

lerDados();

printf("\nDeseja voltar para o menu ?\n[1] - Sim\n[2] - Não\n");

printf("\nInforme a opção que deseja seguir: ");

scanf("%d", &opcao);

if (opcao == 1){

limpaTela();

menu\_principal();

}

break;

case 5:

// Saindo do jogo

break;

default:

limpaTela();

printf(" --- Opção inválida. ---\n");

}

}while(opcao < 1 || opcao > 5);

}