PROJETO DE TÉCNICAS E DESENVOLVIMENTO DE ALGORITMOS

Relatório jogo da velha em C

Alunos:

Bernardo Souza – rgm: 34421637

Jeová Moreira Ferreira Filho - rgm: 38914247

Vinicius Cavalcante - rgm: 38589753

Lucas Amorim pontes - rgm: 38419084

Introdução

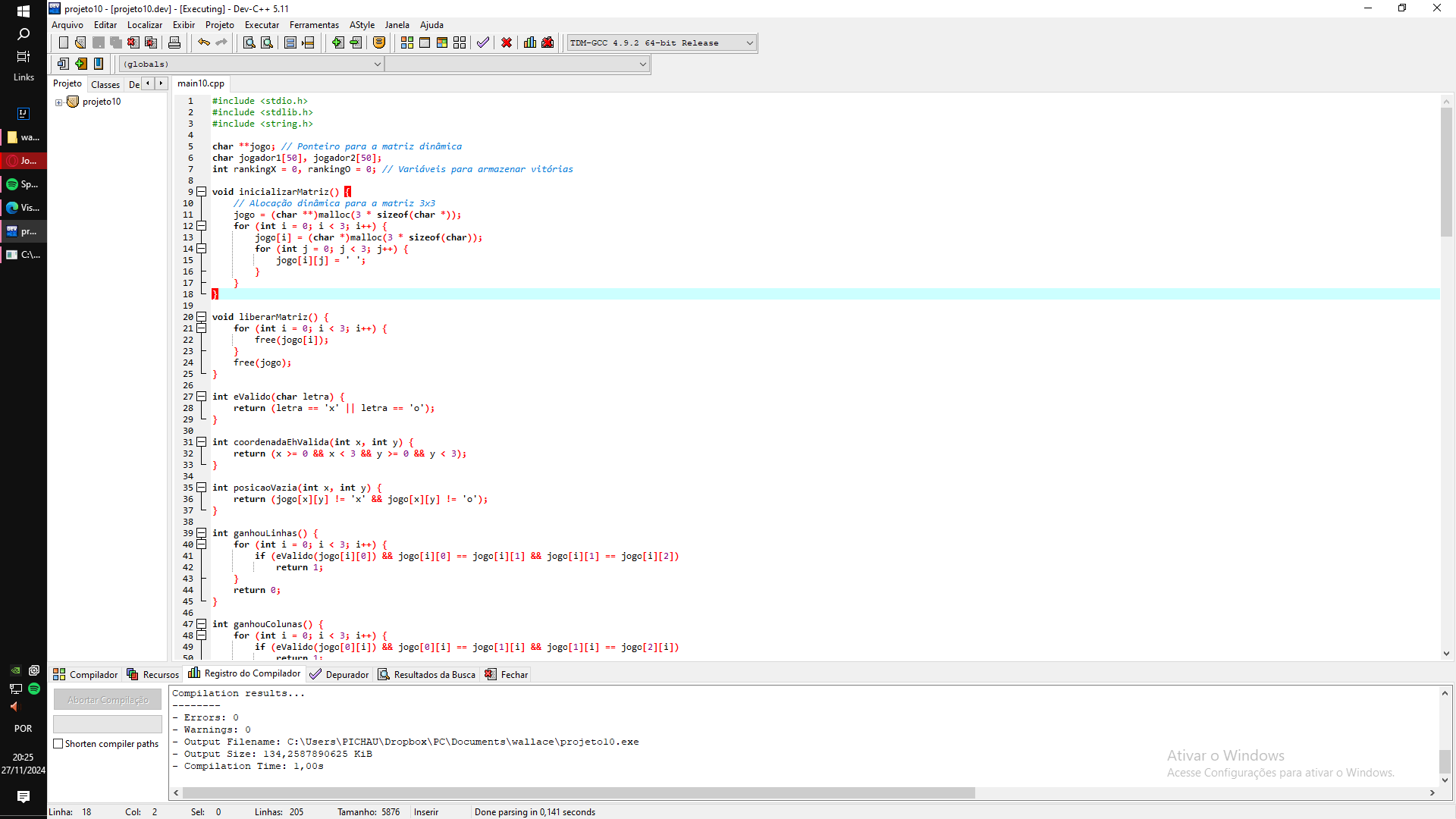
O projeto consiste em desenvolver um jogo da velha em linguagem C. O jogo segue as regras tradicionais, onde dois jogadores alternam jogadas em um tabuleiro 3x3, com o objetivo de alinhar três símbolos idênticos (X ou O) na horizontal, vertical ou diagonal. O jogo implementa alocação dinâmica para gerenciar o tabuleiro, manipulação de arquivos para armazenar rankings, e um menu interativo com opções de jogo, consulta ao ranking, créditos e saída.

Descrição Geral do Jogo

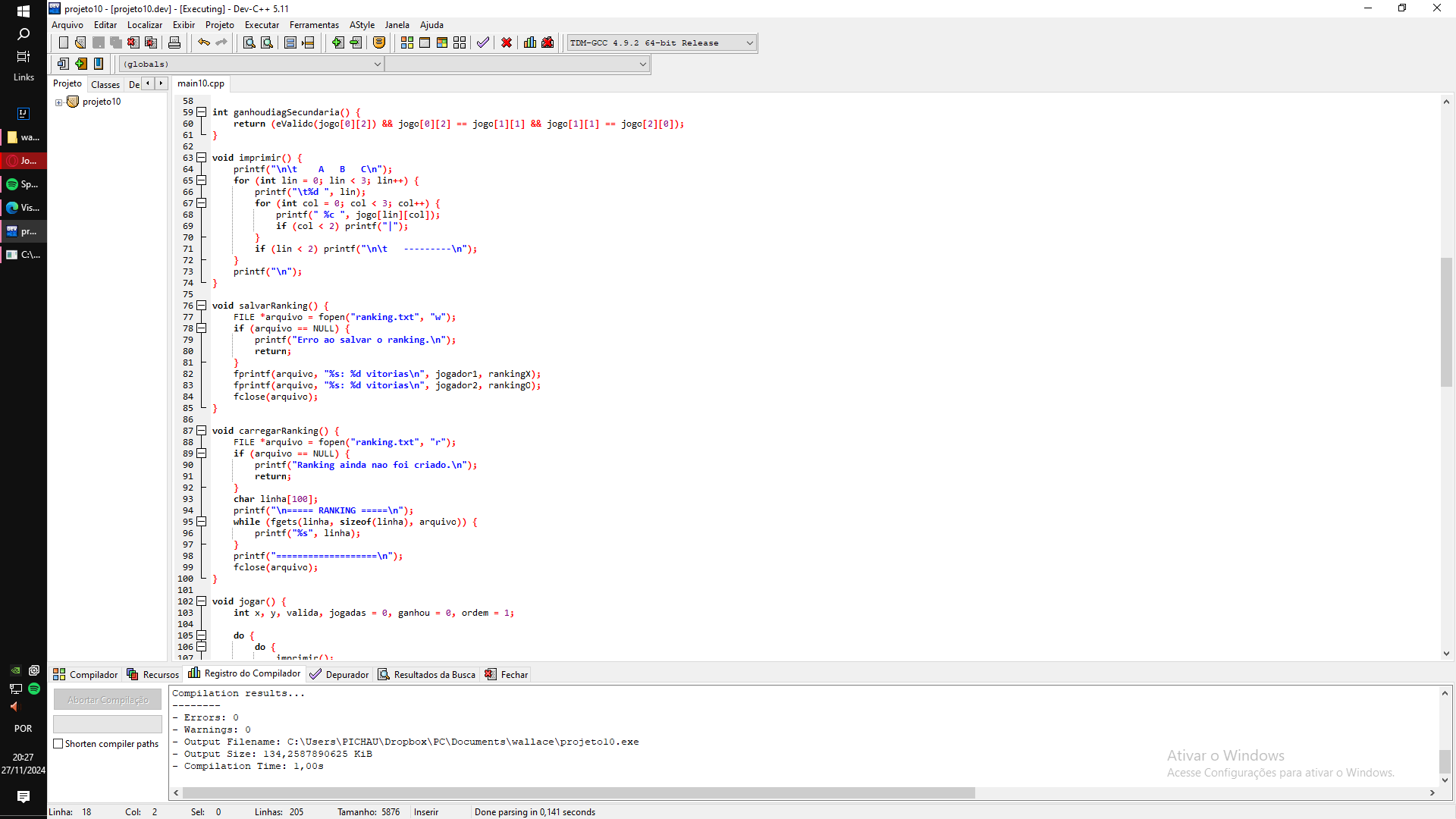
O jogo é estruturado com diversas funções modulares que facilitam o gerenciamento da matriz, a validação de entradas, a detecção de vitórias e a manipulação de arquivos. O código utiliza uma estrutura de menu, permitindo que os usuários escolham entre jogar, visualizar o ranking ou sair do programa.

Exemplo de Código:

Função de inicialização da matriz (tabuleiro):

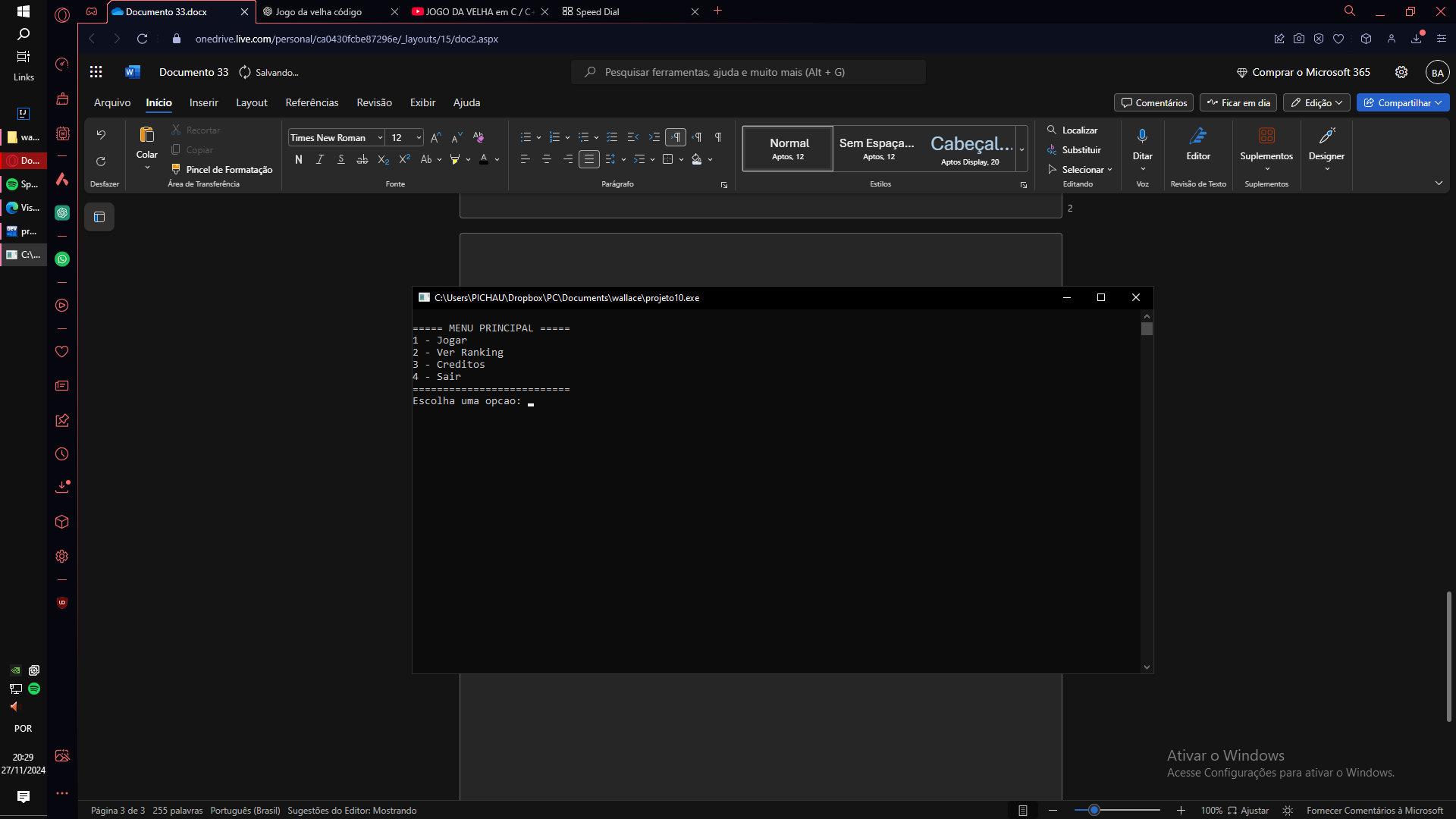


Função de salvar o ranking em um arquivo:



Funcionalidades Implementadas

1. Jogar:Os jogadores alternam suas jogadas até que haja um vencedor ou empate. A matriz 3x3 é preenchida dinamicamente com os símbolos X e O, e o jogo detecta vitórias ou empates após cada jogada.
2. Ver Ranking**:** O programa lê um arquivo chamado ranking.txt para exibir o número de vitórias de cada jogador. Se o arquivo não existir, ele informa que o ranking ainda não foi criado.
3. Créditos:Exibe uma tela com informações sobre os desenvolvedores do jogo.
4. Sair:Encerra o programa de forma segura, liberando a memória alocada.



#### Dificuldades Encontradas e Soluções Implementadas

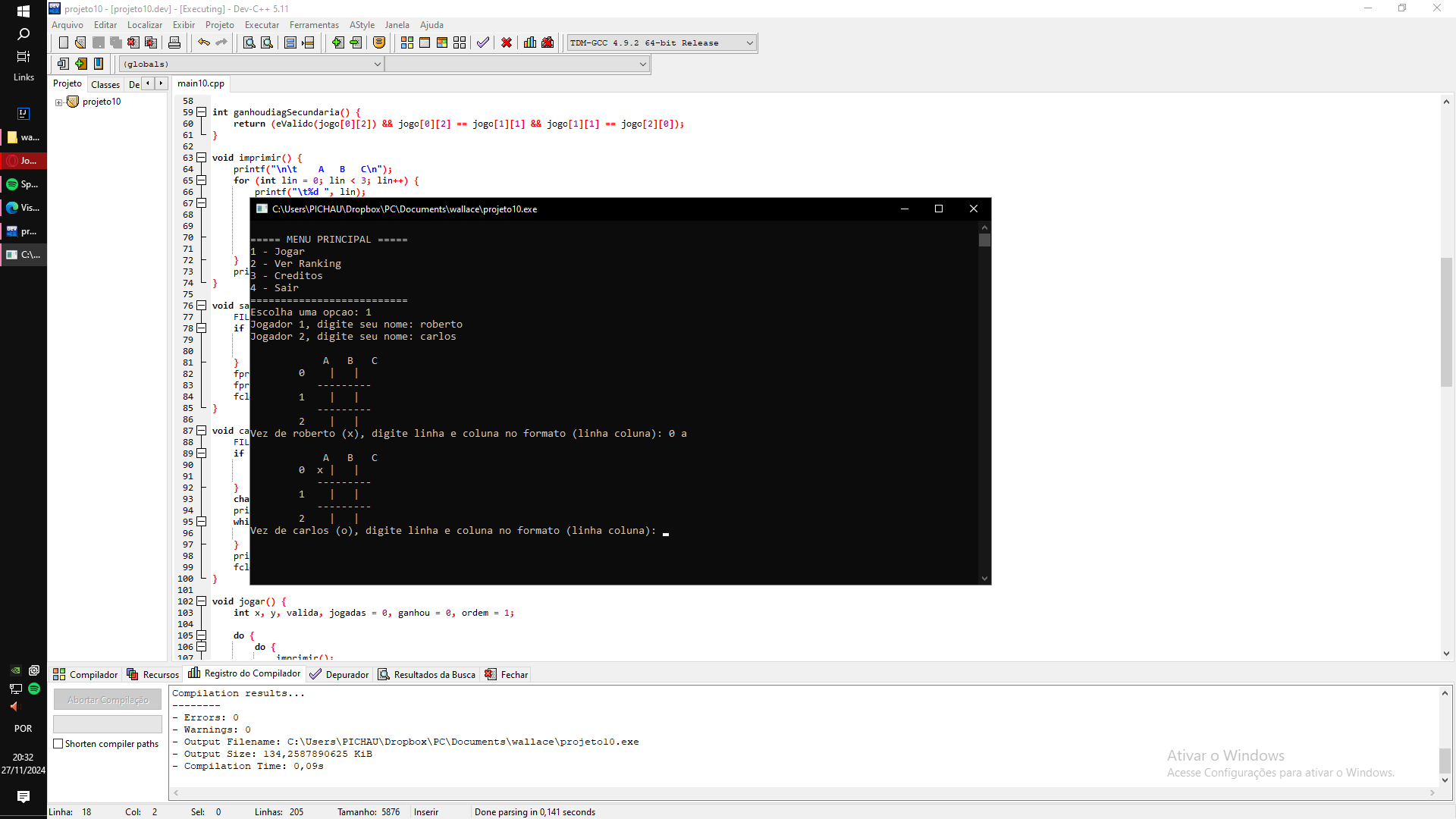
Validação de Entradas: A principal dificuldade foi garantir que as entradas dos jogadores fossem válidas e no formato correto. A solução foi implementar verificações para coordenadas de linhas e colunas e assegurar que a posição escolhida estivesse vazia.

Manipulação de Arquivos: Foi necessário garantir que o ranking fosse salvo e carregado corretamente. A abordagem usada envolveu funções de manipulação de arquivos (fopen, fclose, fprintf e fgets), com tratamento de erros para garantir a robustez do código.

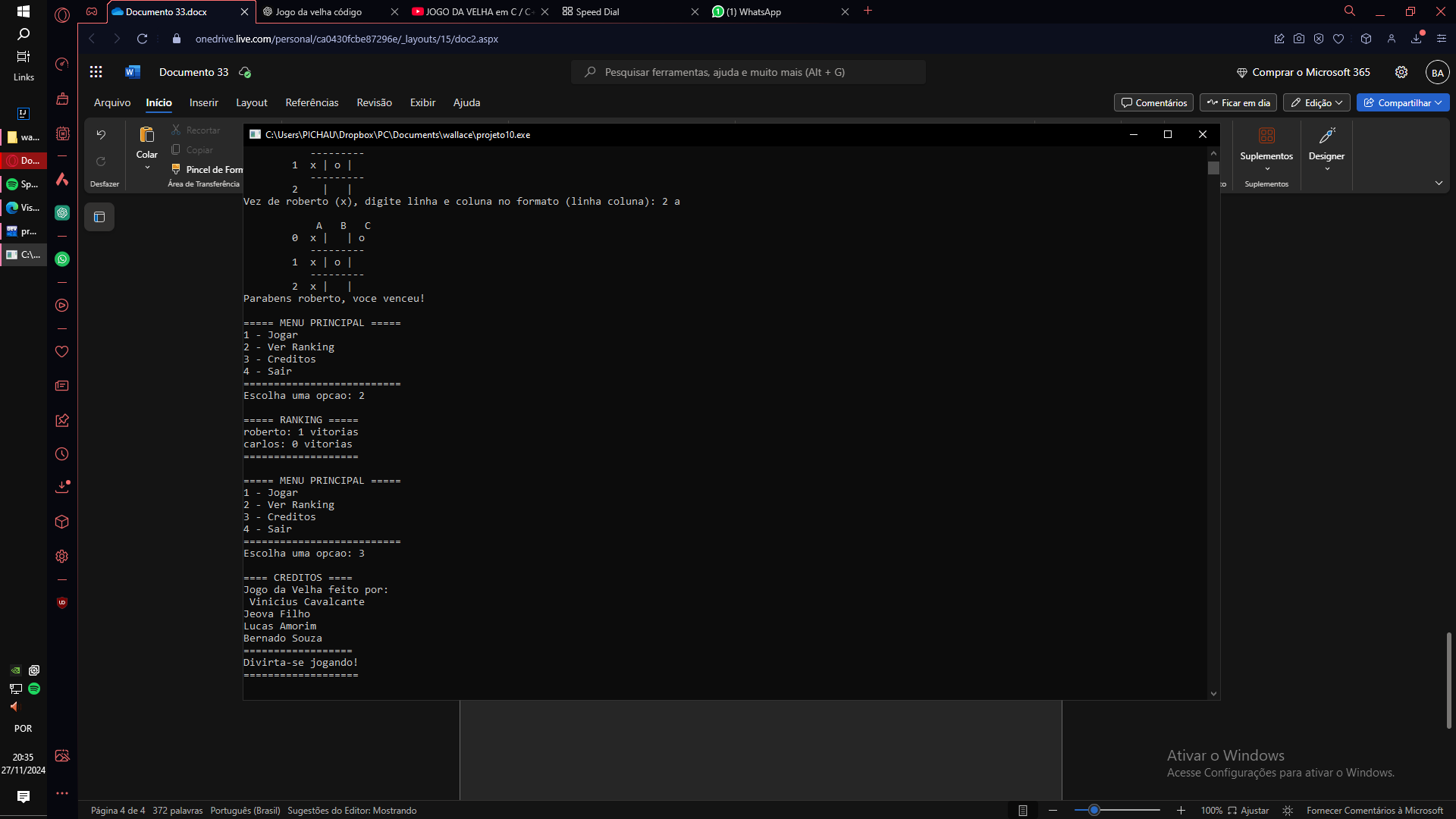
Gerenciamento de Memória Dinâmica: A alocação e liberação de memória para a matriz foram cuidadosamente implementadas para evitar vazamentos. A função free() foi usada para liberar a memória após o uso.

Demonstrativo das Funcionalidades

Tabuleiro durante o jogo:



Tabuleiro com a mensagem de vitória, visualização do ranking e créditos.



Apêndice com o código completo:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char \*\*jogo; // Ponteiro para a matriz dinâmica

char jogador1[50], jogador2[50];

int rankingX = 0, rankingO = 0; // Variáveis para armazenar vitórias

void inicializarMatriz() {

// Alocação dinâmica para a matriz 3x3

jogo = (char \*\*)malloc(3 \* sizeof(char \*));

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

jogo[i] = (char \*)malloc(3 \* sizeof(char));

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

jogo[i][j] = ' ';

}

}

}

void liberarMatriz() {

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

free(jogo[i]);

}

free(jogo);

}

int eValido(char letra) {

return (letra == 'x' || letra == 'o');

}

int coordenadaEhValida(int x, int y) {

return (x >= 0 && x < 3 && y >= 0 && y < 3);

}

int posicaoVazia(int x, int y) {

return (jogo[x][y] != 'x' && jogo[x][y] != 'o');

}

int ganhouLinhas() {

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

if (eValido(jogo[i][0]) && jogo[i][0] == jogo[i][1] && jogo[i][1] == jogo[i][2])

return 1;

}

return 0;

}

int ganhouColunas() {

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

if (eValido(jogo[0][i]) && jogo[0][i] == jogo[1][i] && jogo[1][i] == jogo[2][i])

return 1;

}

return 0;

}

int ganhoudiagPrincipal() {

return (eValido(jogo[0][0]) && jogo[0][0] == jogo[1][1] && jogo[1][1] == jogo[2][2]);

}

int ganhoudiagSecundaria() {

return (eValido(jogo[0][2]) && jogo[0][2] == jogo[1][1] && jogo[1][1] == jogo[2][0]);

}

void imprimir() {

printf("\n\t A B C\n");

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

printf("\t%d ", lin);

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

printf(" %c ", jogo[lin][col]);

if (col < 2) printf("|");

}

if (lin < 2) printf("\n\t ---------\n");

}

printf("\n");

}

void salvarRanking() {

FILE \*arquivo = fopen("ranking.txt", "w");

if (arquivo == NULL) {

printf("Erro ao salvar o ranking.\n");

return;

}

fprintf(arquivo, "%s: %d vitorias\n", jogador1, rankingX);

fprintf(arquivo, "%s: %d vitorias\n", jogador2, rankingO);

fclose(arquivo);

}

void carregarRanking() {

FILE \*arquivo = fopen("ranking.txt", "r");

if (arquivo == NULL) {

printf("Ranking ainda nao foi criado.\n");

return;

}

char linha[100];

printf("\n===== RANKING =====\n");

while (fgets(linha, sizeof(linha), arquivo)) {

printf("%s", linha);

}

printf("===================\n");

fclose(arquivo);

}

void jogar() {

int x, y, valida, jogadas = 0, ganhou = 0, ordem = 1;

do {

do {

imprimir();

printf("Vez de %s (%c), digite linha e coluna no formato (linha coluna): ",

(ordem == 1 ? jogador1 : jogador2),

(ordem == 1 ? 'x' : 'o'));

char linha, coluna;

scanf(" %c %c", &linha, &coluna);

// Verifica se a entrada está no formato correto

if (linha >= '0' && linha <= '2' && ((coluna >= 'A' && coluna <= 'C') || (coluna >= 'a' && coluna <= 'c'))) {

x = linha - '0'; // Converte linha para número

y = (coluna >= 'a') ? coluna - 'a' : coluna - 'A'; // Converte coluna para número

valida = coordenadaEhValida(x, y) && posicaoVazia(x, y);

} else {

valida = 0; // Entrada inválida se não estiver no formato correto

}

if (!valida)

printf("Formato invalido ou posicao ocupada! Use (linha coluna), por exemplo: 0 A.\n");

} while (!valida);

jogo[x][y] = (ordem == 1) ? 'x' : 'o';

jogadas++;

ordem = (ordem == 1) ? 2 : 1;

ganhou = ganhouLinhas() || ganhouColunas() || ganhoudiagPrincipal() || ganhoudiagSecundaria();

} while (!ganhou && jogadas < 9);

imprimir();

if (ganhou) {

if (ordem == 2) {

printf("Parabens %s, voce venceu!\n", jogador1);

rankingX++;

} else {

printf("Parabens %s, voce venceu!\n", jogador2);

rankingO++;

}

salvarRanking();

} else {

printf("Empate! Ninguem venceu.\n");

}

}

void mostrarCreditos() {

printf("\n==== CREDITOS ====\n");

printf("Jogo da Velha feito por:\n Vinicius Cavalcante\nJeova Filho\nLucas Amorim\nBernado Souza\n==================\n");

printf("Divirta-se jogando!\n");

printf("===================\n");

}

void menuPrincipal() {

int escolha;

do {

printf("\n===== MENU PRINCIPAL =====\n");

printf("1 - Jogar\n");

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

printf("3 - Creditos\n");

printf("4 - Sair\n");

printf("==========================\n");

printf("Escolha uma opcao: ");

scanf("%d", &escolha);

getchar(); // Limpa o buffer

switch (escolha) {

case 1:

printf("Jogador 1, digite seu nome: ");

scanf("%s", jogador1);

printf("Jogador 2, digite seu nome: ");

scanf("%s", jogador2);

inicializarMatriz();

jogar();

liberarMatriz(); // Libera a matriz apos o jogo

break;

case 2:

carregarRanking();

break;

case 3:

mostrarCreditos();

break;

case 4:

printf("Saindo do jogo... Ate a proxima!\n");

break;

default:

printf("Opcao invalida. Tente novamente.\n");

}

} while (escolha != 4);

}

int main() {

menuPrincipal();

return 0;

}