**Relatório de Desenvolvimento - Jogo da Velha em C**

**Elielton Dias Ramos de Souza - 36498351**

**Gabriel Xavier Soares - 35535288**

**Lucas Pontes Viana de Castro - 36648892**

**João Victor Araújo de Andrade - 35637021**

**João Vitor Virginio Nunes - 35809221**

**1. Introdução:**

O objetivo deste projeto foi criar um jogo da velha em linguagem C, o código foi organizado de maneira a facilitar a compreensão e manutenção, envolvendo conceitos como estruturas de controle, manipulação de matrizes, strings, modularização com a utilização de cabeçalho.

**2. Estrutura do Projeto:**

**jogo\_da\_velha.h:** Este arquivo contém as definições de constantes, declaração de variáveis globais, e protótipos das funções que serão implementadas nos arquivos de código-fonte.

**jogo\_da\_velha.c:** Contém as implementações das funções declaradas em "jogo\_da\_velha.h". Aqui, as regras do jogo, manipulação do tabuleiro, e interações com o usuário foram implementadas.

**main.c:** Este é o arquivo principal onde a lógica do jogo é coordenada. Ele inclui "jogo\_da\_velha.h" e utiliza as funções implementadas para controlar o fluxo do jogo.

**3. Estrutura do Código:**

**Constantes:** As constantes `X`, `O`, e `VAZIO` são utilizadas para representar os jogadores e o estado do tabuleiro, proporcionando clareza ao código.

**Variáveis Globais:** `jogador1\_nome`, `jogador2\_nome`, e `tabuleiro` são declaradas como variáveis globais no arquivo de cabeçalho para que possam ser compartilhadas entre diferentes partes do programa.

**Funções:**

**exibir\_tabuleiro():** Responsável por mostrar o estado atual do tabuleiro de forma clara e legível.

**inicializar\_tabuleiro():** Inicializa o tabuleiro com espaços vazios, preparando-o para um novo jogo.

**realizar\_jogada():** Recebe as coordenadas da jogada do jogador, verifica a validade e a realiza, alternando entre X e O.

**verificar\_vencedor():** Analisa o tabuleiro para determinar se há um vencedor ou não.

**tabuleiro\_cheio():** Verifica se o tabuleiro está cheio, indicando um empate.

**limpar\_tela():** Limpa a tela do console, proporcionando uma melhor experiência de jogo.

**4. Boas Práticas:**

**Modularização:** O código foi modularizado, separando as responsabilidades em funções específicas para facilitar a leitura, manutenção e reutilização.

**Uso de Constantes:** O uso de constantes, como `TAMANHO\_NOME` e `TAMANHO\_TABULEIRO`, torna o código mais legível e adaptável a possíveis alterações futuras.

**Tratamento de Erros:** O código inclui tratamentos para entradas inválidas, garantindo uma interação mais amigável com o usuário.

**Limpeza de Buffer:** A função `limpar\_tela()` é utilizada para manter uma interface mais limpa entre as jogadas.

**5. Considerações Finais:**

O código foi desenvolvido de forma a atender aos requisitos do projeto de Técnicas e desenvolvimento de Algoritmo, implementando um jogo funcional e interativo com a utilização da linguagem C.

Este relatório resume o desenvolvimento do projeto, destacando aspectos importantes da estrutura e boas práticas adotadas.

**6. Apêndice – Codigo fonte da aplicação**

**Arquivo principal (main.c)**

#include <stdio.h>

#include "jogo\_da\_velha.h"

int main() {

int jogar\_novamente = 1; // Variável para controlar se quer jogar novamente

limpar\_tela(); // Limpa a tela do console

while (jogar\_novamente) {

printf("\tBEM-VINDO AO JOGO DA VELHA\n\n");

printf("Digite o nome do Jogador 1: ");

scanf("%s", jogador1\_nome);

printf("Digite o nome do Jogador 2: ");

scanf("%s", jogador2\_nome);

int jogador\_atual = X; // Define o jogador inicial como X

int vencedor = 0; // Variável para armazenar se houve um vencedor

// Loop principal do jogo

while (!vencedor && !tabuleiro\_cheio()) {

exibir\_tabuleiro(); // Mostra o estado atual do tabuleiro

vencedor = realizar\_jogada(jogador\_atual); // Realiza a jogada e verifica se há um vencedor

if (!vencedor) {

jogador\_atual = (jogador\_atual == X) ? O : X; // Alterna entre os jogadores X e O

}

}

exibir\_tabuleiro(); // Exibe o tabuleiro final

if (vencedor) {

printf("Parabens, %s! Voce venceu!\n", (vencedor == X) ? jogador1\_nome : jogador2\_nome);

} else {

printf("O jogo empatou. VELHA!\n");

}

// Pergunta aos jogadores se querem jogar novamente

printf("Deseja jogar novamente? (1 para sim, 0 para nao): ");

scanf("%d", &jogar\_novamente);

limpar\_tela(); // Limpa a tela para o próximo jogo

// Reinicia o estado do jogo se estiver jogando novamente

if (jogar\_novamente) {

inicializar\_tabuleiro(); // Inicializa o tabuleiro para um novo jogo

}

}

printf("Obrigado por jogar! Adeus!\n");

return 0;

}

**Arquivo contendo funções (jogo\_da\_velha.c)**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "jogo\_da\_velha.h"

// Definição do tamanho do nome dos jogadores

char jogador1\_nome[TAMANHO\_NOME];

char jogador2\_nome[TAMANHO\_NOME];

// Representação do tabuleiro do jogo

int tabuleiro[3][3];

// Implementações das funções do jogo

// Função para exibir o estado atual do tabuleiro

void exibir\_tabuleiro() {

limpar\_tela();

printf(" 1 2 3\n");

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

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

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

if (tabuleiro[i][j] == X) {

printf("X|");

} else if (tabuleiro[i][j] == O) {

printf("O|");

} else {

printf("\_|");

}

}

printf("\n");

}

printf("\n");

}

// Função para inicializar o tabuleiro com espaços vazios

void inicializar\_tabuleiro() {

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

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

tabuleiro[i][j] = VAZIO;

}

}

}

// Função para realizar uma jogada de um jogador específico

int realizar\_jogada(int jogador) {

int linha, coluna;

printf("Vez do Jogador %c.\n", (jogador == X) ? 'X' : 'O');

// Loop para garantir uma entrada válida

while (1) {

printf("Digite a linha e a coluna (por exemplo, 1 2): ");

// Verificar se a leitura foi bem-sucedida

if (scanf("%d %d", &linha, &coluna) == 2) {

linha--;

coluna--;

// Verificar se a posição está dentro dos limites e vazia

if (linha >= 0 && linha < 3 && coluna >= 0 && coluna < 3 && tabuleiro[linha][coluna] == VAZIO) {

break; // Jogada válida, sair do loop

} else {

printf("Posicao invalida ou ja ocupada. Tente novamente.\n");

}

} else {

printf("Entrada invalida. Digite dois numeros separados por espaco.\n");

// Limpar o buffer do teclado para evitar problemas com entradas inválidas

while (getchar() != '\n');

}

}

tabuleiro[linha][coluna] = jogador;

return verificar\_vencedor();

}

// Função para verificar se há um vencedor

int verificar\_vencedor() {

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

// Verificar linhas e colunas

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

return tabuleiro[i][0];

}

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

return tabuleiro[0][i];

}

}

// Verificar diagonais

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

return tabuleiro[0][0];

}

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

return tabuleiro[0][2];

}

return 0; // Nenhum vencedor ainda

}

// Função para verificar se o tabuleiro está cheio (empate)

int tabuleiro\_cheio() {

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

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

if (tabuleiro[i][j] == VAZIO) {

return 0; // Ainda há espaços vazios

}

}

}

return 1; // Tabuleiro está cheio

}

// Função para limpar a tela (dependente do sistema operacional)

void limpar\_tela() {

system("cls || clear");

}

**Arquivo de cabeçalho(jogo\_da\_velha.h)**

#ifndef JOGO\_DA\_VELHA\_H

#define JOGO\_DA\_VELHA\_H

// Definições de constantes para representar os jogadores e o estado do tabuleiro

#define X 1

#define O 2

#define VAZIO 0

#define TAMANHO\_NOME 50 // Tamanho máximo dos nomes dos jogadores

// Declaração das variáveis globais que armazenam os nomes dos jogadores e o estado do tabuleiro

extern char jogador1\_nome[TAMANHO\_NOME];

extern char jogador2\_nome[TAMANHO\_NOME];

extern int tabuleiro[3][3];

// Declaração das funções que serão implementadas no código-fonte (arquivo .c) correspondente

void exibir\_tabuleiro();

void inicializar\_tabuleiro();

int realizar\_jogada(int jogador);

int verificar\_vencedor();

int tabuleiro\_cheio();

void limpar\_tela();

#endif