Grupo/ RGM:
Maria Heloiza - 38608189
Zafnny Carvalho - 39215253
Leonardo Saraiva - 26736926
Curso(s)/Área(s): Ciência da Computação 2 º Período B
Relatório: Projeto de Técnicas e Desenvolvimento de Algoritmos

Relatório: Jogo da Velha - Implementação em C

1. Introdução: Descrição do Jogo e Regras

O jogo desenvolvido é o **Jogo da Velha** é um jogo de tabuleiro para dois jogadores onde o objetivo é alinhar três símbolos (X ou O) de forma contínua, seja na horizontal, vertical ou diagonal.

- **Regras do Jogo:**
    1. O jogo é jogado em um tabuleiro 3x3, onde cada posição pode estar vazia ou ocupada por um "X" ou um "O".
    2. O jogador "X" sempre começa, e os jogadores alternam suas jogadas.
    3. O vencedor é aquele que conseguir alinhar três símbolos consecutivos (na linha, coluna ou diagonal).
    4. Se o tabuleiro estiver cheio e não houver vencedor, o jogo termina em **empate**.
    5. O jogo oferece a opção de jogar contra o computador, onde o computador faz jogadas aleatórias.

Além disso, o jogo mantém um **ranking** de vitórias, permitindo que o jogador registre suas vitórias e visualize um histórico.

2. Resultados: Descrição Geral do Jogo

- **Estrutura de Dados:**
  - A estrutura Jogador armazena o nome do jogador e o número de vitórias.
  - O tabuleiro é representado por uma matriz 3x3 de caracteres (char), onde os espaços vazios são representados por ' ' (espaço em branco), e os símbolos dos jogadores são representados por 'X' ou 'O'.
  -
- **Funções principais do jogo:**
  - **criarTabuleiro:** Cria o tabuleiro 3x3, alocando memória dinamicamente.
  - **exibirTabuleiro:** Exibe o estado atual do tabuleiro no console.
  - **verificarVencedor:** Verifica se algum jogador conseguiu formar uma linha, coluna ou diagonal com três símbolos consecutivos.
  - **tabuleiroCheio:** Verifica se o tabuleiro está completamente preenchido, indicando empate.
  - **jogarJogador:** Permite ao jogador humano inserir sua jogada (linha e coluna).
  - **jogarComputador:** Faz a jogada do computador, escolhendo uma posição aleatória disponível.
  - **salvarRanking e exibirRanking:** Permite salvar e exibir o ranking de vitórias dos jogadores.

3. Exemplificação de Código Fonte

Abaixo estão os trechos de código principais que implementam a lógica do jogo.

Criação do Tabuleiro:

```
char** criarTabuleiro() {
    char** tabuleiro = (char**)malloc(3 * sizeof(char*));
    for (int i = 0; i < 3; i++)
        tabuleiro[i] = (char*)malloc(3 * sizeof(char));

    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            tabuleiro[i][j] = ' ';  // Posicoes vazias do tabuleiro

    return tabuleiro;
}
```

Exibição do Tabuleiro:

```
void exibirTabuleiro(char** tabuleiro) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf(" %c ", tabuleiro[i][j]);
            if (j < 2) printf("|");
        }
        printf("\n");
        if (i < 2) printf("---+---+---\n");
    }
}
```

Verificação de Vencedor:

```
int verificarVencedor(char** tabuleiro, char jogador) {
    for (int i = 0; i < 3; i++) {
```

```
        if (tabuleiro[i][0] == jogador && tabuleiro[i][1] == jogador && tabuleiro[i][2] ==
jogador) return 1;
        if (tabuleiro[0][i] == jogador && tabuleiro[1][i] == jogador && tabuleiro[2][i] ==
jogador) return 1;
    }
    if (tabuleiro[0][0] == jogador && tabuleiro[1][1] == jogador && tabuleiro[2][2] ==
jogador) return 1;
    if (tabuleiro[0][2] == jogador && tabuleiro[1][1] == jogador && tabuleiro[2][0] ==
jogador) return 1;

    return 0;
}
```

Jogada do Computador (aleatória):

```
void jogarComputador(char** tabuleiro) {
    int linha, coluna;
    linha = rand() % 3;
    coluna = rand() % 3;

    while (tabuleiro[linha][coluna] != ' ') {
        linha = rand() % 3;
        coluna = rand() % 3;
    }

    tabuleiro[linha][coluna] = 'O';  // Coloca a jogada do computador
}
```

Dificuldades Encontradas e Soluções Implementadas

1. **Problema com a verificação de vitória**: Inicialmente, a verificação de vitória estava apenas comparando a linha e a coluna diretamente. No entanto, era necessário garantir que as duas diagonais também fossem verificadas.

- **Solução**: Implementamos uma função que verifica as duas diagonais, além das linhas e colunas.

2. **Controle de jogadas no computador**: O computador, ao jogar, inicialmente não conseguia realizar jogadas inteligentes. Ele apenas escolhia uma posição aleatória.
   - **Solução**: A solução foi deixar o computador escolher posições aleatórias dentro do tabuleiro, mas garantindo que a posição fosse vazia, o que foi feito com um laço while.

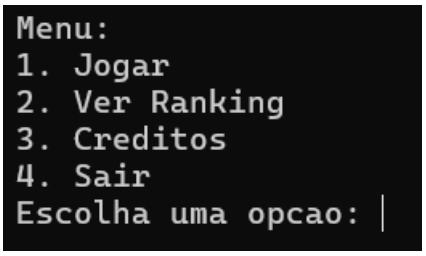
Funcionalidades Implementadas

- **Menu de Opções**: O menu inicial permite ao jogador escolher entre jogar, ver o ranking, visualizar os créditos ou sair do jogo.
- **Jogo com Computador**: Após o jogador escolher a opção "Jogar", o jogo se inicia. O jogador faz sua jogada, depois o computador faz a dele. O jogo continua até alguém vencer ou empatar.
- **Ranking**: O número de vitórias do jogador é armazenado em um arquivo de texto chamado "ranking.txt". A cada vitória, o jogador tem seu nome e número de vitórias registrados nesse arquivo.

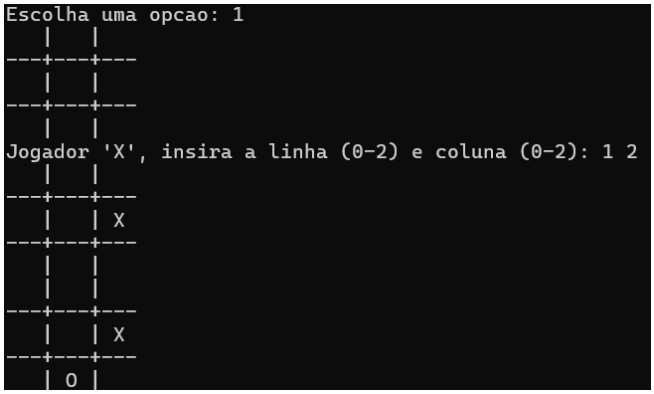
5. Demonstrativo das Funcionalidades Implementadas

Prints da tela do funcionamento do jogo:

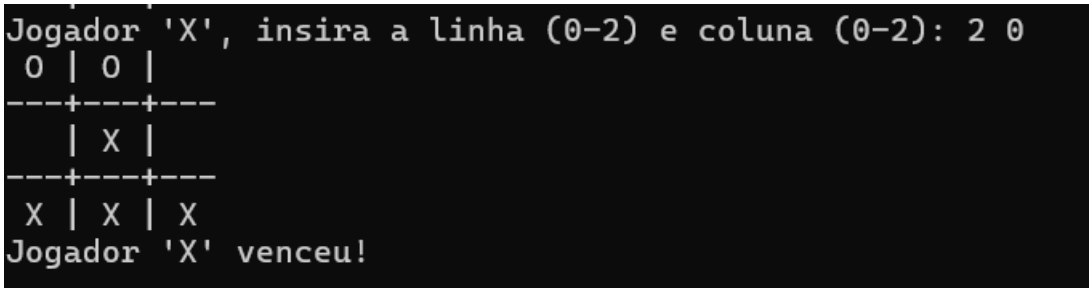
1. **Tela Inicial - Menu Principal:**



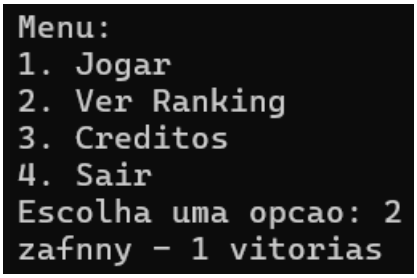
2. **Jogo em andamento:**



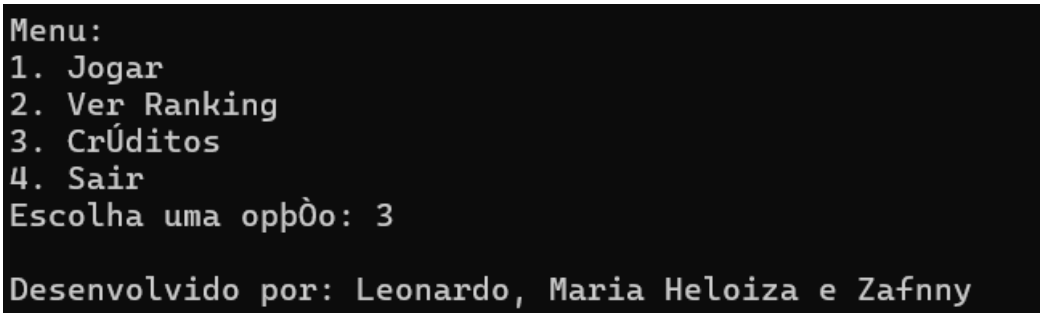
3. **Fim do Jogo - Vencedor:**



4. **Ranking: Opção 2**



5. **Créditos: Opção 3**



6. Apêndice - Código Fonte

Este é o código completo, com as principais funcionalidades de jogo, gerenciamento de ranking, e interação com o jogador e o computador.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

//definir a estrutura para armazenar informacao de jogador
typedef struct {
    char nome[50];  //nome do jogador
    int vitorias;   //numero de vitorias do jogador
} Jogador;

//Funcao para criar o tabuleiro 3x3
char** criarTabuleiro() {
    char** tabuleiro = (char**)malloc(3 * sizeof(char*));  //Aloca memoria para 3 linhas
    for (int i = 0; i < 3; i++)
        tabuleiro[i] = (char*)malloc(3 * sizeof(char)); //Aloca memoria para 3 colunas em
cada linha

    //inicializa todas as posicoes do tabuleiro com espacos em branco (' ')
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            tabuleiro[i][j] = ' ';  //posicoes vazias do tabuleiro

    return tabuleiro;  //retorna o tabuleiro criado
}

//funcao para exibir o tabuleiro
void exibirTabuleiro(char** tabuleiro) {

    //imprime o tabuleiro linha por linha
```

```
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf(" %c ", tabuleiro[i][j]);  //exibe o valor na posicao (i, j)
            if (j < 2) printf("|");  //imprime separador entre as colunas
        }
        printf("\n");
        if (i < 2) printf("---+---+---\n");  //imprime a linha separadora entre as linhas do tabuleiro
    }
    printf("\n");
}

//funcao para verificar se algum jogador venceu
int verificarVencedor(char** tabuleiro, char jogador) {
    //verifica se o jogador fez uma linha - coluna - diagonal completa
    for (int i = 0; i < 3; i++) {
        //verifica linhas
        if (tabuleiro[i][0] == jogador && tabuleiro[i][1] == jogador && tabuleiro[i][2] == jogador) return 1;
        //verifica colunas
        if (tabuleiro[0][i] == jogador && tabuleiro[1][i] == jogador && tabuleiro[2][i] == jogador) return 1;
    }
    //verifica diagonais
    if (tabuleiro[0][0] == jogador && tabuleiro[1][1] == jogador && tabuleiro[2][2] == jogador) return 1;
    if (tabuleiro[0][2] == jogador && tabuleiro[1][1] == jogador && tabuleiro[2][0] == jogador) return 1;

    return 0;  //retorna 0 se nao houve vencedor
```

```
}

//funÃ§Ã£o para verificar se o tabuleiro estÃ¡ cheio (empate)
int tabuleiroCheio(char** tabuleiro) {
    //verifica se todas as posiÃ§Ãµes do tabuleiro estÃ£o preenchidas
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (tabuleiro[i][j] == ' ') return 0;  //se houver espaÃ§o vazio, retorna 0
    return 1;  //retorna 1 se o tabuleiro estiver cheio
}

//funÃ§Ã£o para a jogada do jogador
void jogarJogador(char** tabuleiro, char jogador) {
    int linha, coluna;

    while (1) {
    //solicita ao jogador a posiÃ§Ã£o da linha e coluna onde deseja jogar
    printf("Jogador '%c', insira a linha (0-2) e coluna (0-2): ", jogador);
    scanf("%d %d", &linha, &coluna);
        if (linha >= 0 && linha < 3 && coluna >= 0 && coluna < 3 &&
tabuleiro[linha][coluna] == ' ') {
            tabuleiro[linha][coluna] = jogador; //coloca a jogada do jogador no tabuleiro
            break;
        }else{
            printf("A posição está incorreta ou já utilizada pelo jogador O. Tente
novamente.\n");
        }
    }
}
```

```
//funcao para a jogada do computador (aleatoria)
void jogarComputador(char** tabuleiro) {
    int linha, coluna;
    //gera nÃºmeros aleatÃ³rios para a jogada do computador
    linha = rand() % 3;  //gera uma linha aleatoria entre 0 e 2
    coluna = rand() % 3;  //gera uma coluna aleatoria entre 0 e 2

    //continua tentando ate encontrar uma posicao vazia
    while (tabuleiro[linha][coluna] != ' ') {
        linha = rand() % 3;  //gera uma linha aleatoria entre 0 e 2
        coluna = rand() % 3;  //gera uma coluna aleatoria entre 0 e 2
    }

    tabuleiro[linha][coluna] = 'O';  //coloca a jogada do computador no tabuleiro
}

//funÃ§Ã£o para salvar o ranking do jogador em um arquivo
void salvarRanking(Jogador jogador) {
    FILE* arquivo = fopen("ranking.txt", "a");  //abre o arquivo para adicionar
informaÃ§Ãµes
    if (arquivo) {
        fprintf(arquivo, "%s %d\n", jogador.nome, jogador.vitorias);  //salva o nome e
vitÃ³rias do jogador
        fclose(arquivo);  //fecha o arquivo
    }
}

//funÃ§Ã£o para exibir o ranking
void exibirRanking() {
    FILE* arquivo = fopen("ranking.txt", "r");  //abre o arquivo de ranking para leitura
```

```
    if (arquivo) {
        char nome[50];
        int vitorias;
        //le e exibe as vitÃ³rias de todos os jogadores registrados no arquivo
        while (fscanf(arquivo, "%s %d", nome, &vitorias) != EOF) {
            printf("%s - %d vitorias\n", nome, vitorias);
        }
        fclose(arquivo);  //fecha o arquivo
    } else {
        printf("Nao possui ranking registrado.\n");  //caso nÃ£o haja arquivo de ranking
    }
}

//funcao para exibir o menu de opcoes
void mostrarMenu() {
    printf("\nMenu:\n1. Jogar\n2. Ver Ranking\n3. Creditos\n4. Sair\n");
}

//funcao para exibir os creditos
void exibirCreditos() {
    printf("\nDesenvolvido por: Leonardo, Heloisa e Zafnny\n");
}

//funcao principal
int main() {
    Jogador jogador = {"Jogador", 0};  //cria a estrutura do jogador e inicializa o nome e vitorias
    int opcao;
    printf("Digite seu nome: ");
```

```
    scanf("%s", jogador.nome);  //Le o nome do jogador

    srand(time(NULL));  //inicializa o gerador de nÃºmeros aleatÃ³rios (agora fora do laÃ§o de jogo)

    while (1) {
        mostrarMenu();  //exibe o menu de opcoes
        printf("Escolha uma opcao: ");
        scanf("%d", &opcao);  //Le a opÃ§Ã£o escolhida

        switch (opcao) {
            case 1: {
                char** tabuleiro = criarTabuleiro();  //cria o tabuleiro
                int jogoAtivo = 1;
                char jogadorAtual = 'X';  //inicia o jogo com o jogador 'X'

                while (jogoAtivo) {
                    exibirTabuleiro(tabuleiro);  //exibe o tabuleiro atual
                    if (jogadorAtual == 'X')
                        jogarJogador(tabuleiro, jogadorAtual);  //jogada do jogador
                    else
                        jogarComputador(tabuleiro);  //jogada do computador

                    if (verificarVencedor(tabuleiro, jogadorAtual)) {  //verifica se alguem venceu
                        exibirTabuleiro(tabuleiro);
                        printf("Jogador '%c' venceu!\n", jogadorAtual);
                        if (jogadorAtual == 'X') jogador.vitorias++;  //atualiza as vitorias do jogador
                        jogoAtivo = 0;  //termina o jogo
```

```
            } else if (tabuleiroCheio(tabuleiro)) {  //verifica se o tabuleiro esta cheio
(empate)
                exibirTabuleiro(tabuleiro);
                printf("Empate!\n");
                jogoAtivo = 0;
            }

            jogadorAtual = (jogadorAtual == 'X') ? 'O' : 'X';  //alterna entre os
jogadores
        }

        salvarRanking(jogador);  //salva o ranking do jogador
        break;
      }
      case 2: exibirRanking(); break;  //exibe o ranking
      case 3: exibirCreditos(); break;  //exibe os creditos
      case 4: return 0;  //sai do programa
      default: printf("Opcao invalida tente novamente.\n");  //caso a opcao nao seja
valida
    }
  }
  return 0;
}
```