UNIPÊ- CENTRO UNIVERSITÁRIO DE JOÃO PESSOA

CURSO: CIÊNCIA DA COMPUTAÇÃO

TURMA: P2-B

Relatório de Projeto da Disciplina Técnicas e Desenvolvimento de Algoritmo

Jogo da Velha em C

Alunos:

Diego da Silva Feitosa (RGM 37625446)

Alex Dantas de Oliveira e Silva (RGM 38305585)

José Kleyton Francelino da silva Filho (RGM 38577305)

Handrey Kaleu Matias Souza de Carvalho (RGM 38196883)

João Pessoa- PB 2024

Introdução

O Jogo da Velha, é um jogo de estratégia simples, popular em todo o mundo. A versão apresentada neste relatório foi implementada em linguagem C, utilizando a biblioteca padrão para interação com o usuário, manipulação de arquivos e alocação dinâmica de memória. O objetivo principal do jogo é alinhar três símbolos do mesmo jogador (geralmente 'X' ou 'O') em uma linha reta, seja horizontal, vertical ou diagonal, em uma grade de 3x3. O jogo pode terminar de três maneiras: um dos jogadores vence, o jogo empata (velha) ou o jogo é encerrado pelo usuário.

Regras do Jogo

1. **Início do Jogo**: O jogo começa com dois jogadores. Cada jogador escolhe um símbolo: um usa 'X' e o outro usa 'O'. O jogador que utiliza o 'X' sempre começa.
2. **Movimentos**: Os jogadores se alternam para colocar seus símbolos em espaços vazios na grade. Eles devem indicar a linha e a coluna desejadas (ambas variando de 0 a 2).
3. **Condição de Vitória**: O primeiro jogador a alinhar três de seus símbolos, seja em uma linha, coluna ou diagonal, vence o jogo.
4. **Condição de Empate (Velha)**: Caso todos os espaços da grade sejam preenchidos e nenhum jogador tenha alinhado três símbolos consecutivos, o jogo termina em empate.

Resultados

A programação do jogo foi feita na linguagem C, por meio do Dev C + +.

Em primeiro lugar, pensamos nos requisitos principais do jogo, para assim começarmos o protótipo do código.

Estrutura de Dados

A principal estrutura de dados usada no código é a **struct Jogador**, que armazena as informações de cada jogador:

- **nome[50]**: Um array de caracteres que armazena o nome do jogador (limite de 49 caracteres + o caractere nulo de terminação).
- **vitórias**: Um inteiro que armazena o número de vitórias do jogador.

```c
struct Jogador {
    char nome[50];
    int vitorias;
};
```

Funções

1. **Função lerNumero:**
    - Esta função tem como objetivo garantir que a entrada do usuário seja um número inteiro válido. Caso a entrada não seja um número, o código exibe uma mensagem de erro e limpa o buffer de entrada.
    - Essa função é essencial para garantir que o jogo não quebre devido a entradas inválidas de números.

```c
// Função para ler um número inteiro e garantir que a entrada seja válida
int lerNumero() {
    int numero;
    while (scanf("%d", &numero) != 1) {
        printf("\nOpção Inválida! Digite um número: \n ");
        while (getchar() != '\n');
    }
    return numero;
}
```

2. **Função exibirTabuleiro:**
    - Exibe o estado atual do tabuleiro. Cada célula do tabuleiro é exibida, separada por barras verticais e com as linhas separadas por hífens.

```c
// Função para exibir o tabuleiro do jogo
void exibirTabuleiro(char **jogo) {
    for (int l = 0; l < 3; l++) {
        for (int c = 0; c < 3; c++) {
            if (c == 0) printf("\t");
            printf(" %c ", jogo[l][c]);
            if (c < 2) printf("|");
        }
        if (l < 2) printf("\n\t-----------\n");
        else printf("\n");
    }
}
```

    - Essa função é chamada a cada rodada para que os jogadores possam visualizar o andamento do jogo.

3. **Função verificarVencedor:**

- Verifica se algum jogador venceu. O código analisa se há uma linha, coluna ou diagonal com três símbolos iguais (diferente de espaço vazio).

4. **Função jogoVelha:**
   - Verifica se o jogo terminou em empate. Se todas as células do tabuleiro estão preenchidas e não houve vencedor, o jogo é dado como "velha".
   -

5. **Função salvarRanking:**
   - Salva o nome dos jogadores e o número de vitórias de ambos em um arquivo de texto ("ranking.txt"). Cada vez que o jogo termina, o ranking é atualizado. -

```c
// Função para salvar o ranking em arquivo
void salvarRanking(struct Jogador jogadores[2]) {
    FILE *arquivo = fopen("ranking.txt", "a");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo de ranking!\n");
        return;
    }

    fprintf(arquivo, "%s: %d vitórias\n", jogadores[0].nome, jogadores[0].vitorias);
    fprintf(arquivo, "%s: %d vitórias\n", jogadores[1].nome, jogadores[1].vitorias);
    fclose(arquivo);
}
```

6. **Função mostrarRanking:**
   - Lê e exibe o conteúdo do arquivo "ranking.txt", mostrando as vitórias acumuladas pelos jogadores.

7. **Função exibirCreditos:**
   - Exibe os créditos de desenvolvimento do jogo, incluindo o nome dos desenvolvedores e a turma da universidade responsável pelo código.

Função Principal main

1. **Declaração de Variáveis:**
   - São declaradas variáveis para controle do jogo como: jogador, ganhou, jogadas, numero (para escolher a opção do menu), e jogo (o tabuleiro).

- A variável loop controla o menu principal do jogo.

```c
int main() {
    setlocale(LC_ALL, "portuguese");

    struct Jogador jogadores[2];
    int numero, jogador, linha, coluna, ganhou, jogadas;
    char **jogo;
    int loop = 1;
```

2. **Alocação Dinâmica de Memória**:
    - O tabuleiro é alocado dinamicamente usando malloc, criando uma matriz de 3x3 de caracteres.

```c
// Aloca memória dinamicamente para a matriz jogo
jogo = (char **)malloc(3 * sizeof(char *));
for (int i = 0; i < 3; i++) {
    jogo[i] = (char *)malloc(3 * sizeof(char));
}
```

3. **Loop do Menu**:
    - O programa apresenta um menu principal com as opções: Jogar, Ver Ranking, Créditos e Sair. O menu continua sendo exibido até que o usuário opte por sair.
4. **Opção 1 (Jogar)**:
    - Solicita os nomes dos jogadores e inicializa o tabuleiro com espaços vazios.
    - Dentro de um loop do-while, o jogo é executado, com alternância entre os jogadores para fazer suas jogadas. Após cada jogada, é verificado se houve vitória ou empate.
    - O ranking é atualizado ao final do jogo e salvo no arquivo.
5. **Opção 2 (Ver Ranking)**: ○ Exibe o ranking de vitórias dos jogadores, lido do arquivo "ranking.txt".
6. **Opção 3 (Créditos)**: ○ Exibe os créditos de desenvolvimento do jogo.
7. **Opção 4 (Sair)**: ○ Encerra o jogo e sai do loop principal.

Controle de Jogo

- O jogo alterna entre os dois jogadores, marcando 'O' ou 'X' no tabuleiro, conforme a escolha de cada um.
- O código verifica as condições de vitória após cada jogada (linhas, colunas e diagonais).
- Em caso de empate (quando todas as células são preenchidas e não há vencedor), o jogo informa que houve um empate.
- Após cada jogo, o usuário tem a opção de retornar ao menu ou sair.

Conclusão

O código implementa de maneira funcional o Jogo da Velha com algumas melhorias, como a exibição de um ranking de vitórias e o controle de entradas inválidas. A utilização de alocação dinâmica para o tabuleiro torna o código mais flexível. Além disso, o código é modular, com funções bem definidas para tarefas específicas, como a verificação de vitória e a exibição do tabuleiro. O uso de arquivos para salvar o ranking permite que os dados persistam entre as sessões de jogo. Em resumo, o código apresenta uma solução simples e eficaz para o Jogo da Velha com recursos adicionais de ranking e controle de erros.

Apêndice

Código Fonte:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>

struct Jogador {
   char nome[50];
   int vitorias;
};

// Função para ler um número inteiro e garantir que a entrada seja
válida int lerNumero() { int numero; while (scanf("%d", &numero) != 1) {
printf("\nOpção Inválida! Digite um número: \n "); while (getchar() != '\n');
   }
   return numero;
}
// Função para exibir o tabuleiro do
jogo void exibirTabuleiro(char **jogo) {
```

```
for (int l = 0; l < 3; l++) { for (int c = 0; c
< 3; c++) {
            if (c == 0) printf("\t");
            printf(" %c ", jogo[l][c]);
            if (c < 2) printf("|");
        } if (l < 2) printf("\n\t-----------
        \n"); else printf("\n");
    }
}

// Função para verificar se há um vencedor
int verificarVencedor(char **jogo) {
    // Verifica se algum jogador venceu em linhas, colunas ou diagonais for (int i
    = 0; i < 3; i++) { if ((jogo[i][0] == jogo[i][1] && jogo[i][1] == jogo[i][2] &&
    jogo[i][0] != ' ') || (jogo[0][i] == jogo[1][i] && jogo[1][i] == jogo[2][i] &&
    jogo[0][i] != ' ')) { return 1;
        }
    }

    // Verifica diagonais if ((jogo[0][0] == jogo[1][1] && jogo[1][1] == jogo[2][2]
    && jogo[0][0] != ' ') || (jogo[0][2] == jogo[1][1] && jogo[1][1] == jogo[2][0] &&
    jogo[0][2] != ' ')) { return 1;
    }

    return 0;
}
// Função para verificar se o jogo deu velha int jogoVelha(char **jogo) { for (int i =
0; i < 3; i++) { for (int j = 0; j < 3; j++) { if (jogo[i][j] == ' ') return 0; // Se ainda
houver espaço, o jogo não terminou
        }
    }
```

```
    return 1; // Se não houver espaço, é velha
}

// Função para salvar o ranking em arquivo void
salvarRanking(struct Jogador jogadores[2]) { FILE
*arquivo = fopen("ranking.txt", "a"); if (arquivo ==
NULL) { printf("Erro ao abrir o arquivo de
ranking!\n"); return;
    }

    fprintf(arquivo, "%s: %d vitórias\n", jogadores[0].nome, jogadores[0].vitorias);
    fprintf(arquivo, "%s: %d vitórias\n", jogadores[1].nome, jogadores[1].vitorias);
    fclose(arquivo);
}

// Função para mostrar o ranking
void mostrarRanking() {
    FILE *arquivo = fopen("ranking.txt", "r"); if
    (arquivo == NULL) { printf("Erro ao abrir o
    arquivo de ranking!\n"); return;
    }

    char linha[100];
    printf("\nRanking:\n"); while (fgets(linha,
    sizeof(linha), arquivo)) { printf("%s", linha);
    }
    fclose(arquivo);
}
```

```
// Função para exibir os créditos
void exibirCreditos() {
printf("\n\t\t\t\tCREDITOS\n\n");
   printf("\n\tDesenvolvido por Diego Feitosa, Alex Dantas, José Kleyton e Handrey Kaleu.\n"); printf("\n\tAlunos do UNIPÊ - Ciência da Computação\n");
   printf("\n\tTurma: Técnicas e Desenvolvimento de Algoritmos - 2B 2024.2\n\n");
}

int main() { setlocale(LC_ALL,
   "portuguese");

   struct Jogador jogadores[2]; int numero, jogador,
   linha, coluna, ganhou, jogadas; char **jogo; int loop
   = 1;

   // Aloca memória dinamicamente para a matriz jogo
   jogo = (char **)malloc(3 * sizeof(char *)); for (int i =
   0; i < 3; i++) { jogo[i] = (char *)malloc(3 *
   sizeof(char));
   }

   while (loop == 1) {
      system("cls");
      printf("\n\tJOGO DA VELHA\n\n");
      printf("1 - JOGAR\n"); printf("2 -
      VER RANKING\n"); printf("3 -
      CRÉDITOS\n"); printf("4 -
      SAIR\n\n"); printf("Digite a opção
      desejada: "); numero =
      lerNumero();
```

```
switch (numero) { case 1: { printf("\nDigite o nome do jogador
1 (apenas 1 nome): "); scanf("%s", jogadores[0].nome);

printf("\nDigite o nome do jogador 2 (apenas 1 nome): ");
scanf("%s", jogadores[1].nome);

jogador = 1; ganhou = 0;
jogadas = 0; // Inicializa a
matriz for (int l = 0; l < 3;
l++) { for (int c = 0; c < 3;
c++) {
    jogo[l][c] = ' ';
  }
}

do {
  exibirTabuleiro(jogo);

  if (jogador == 1) {
    printf("\nJOGADOR 1 (%s) = o\nJOGADOR 2 (%s) = x\n", jogadores[0].nome,
jogadores[1].nome);
    printf("\nJOGADOR 1: Digite a linha e a coluna (0 a 2): ");
  } else {
    printf("\nJOGADOR 1 (%s) = o\nJOGADOR 2 (%s) = x\n", jogadores[0].nome,
jogadores[1].nome);
    printf("\nJOGADOR 2: Digite a linha e a coluna (0 a 2): ");
  }

  // Loop para garantir que a jogada seja válida
  do {
    if (scanf("%d %d", &linha, &coluna) != 2 || linha < 0 || linha > 2 || coluna
< 0 || coluna > 2 || jogo[linha][coluna] != ' ') {
```

```
                printf("\nEntrada inválida! Certifique-se de que a linha e a coluna estão
entre 0 e 2, e a célula está vazia.\n");
                while (getchar() != '\n');  // Limpar o buffer
            } else { break;        //
                Jogada válida
            }
        } while (1);

        // Realiza a jogada if
        (jogador == 1) {
        jogo[linha][coluna] = 'o';
        jogador = 2;
        } else { jogo[linha][coluna]
            = 'x'; jogador = 1;
        }
        jogadas++;

        // Verifica se alguém
        venceu if
        (verificarVencedor(jogo)) {
        exibirTabuleiro(jogo); if
        (jogador == 2) {
                printf("\nO jogador 1 (%s) venceu!\n", jogadores[0].nome);
                jogadores[0].vitorias++;
            } else { printf("\nO jogador 2 (%s) venceu!\n",
                jogadores[1].nome); jogadores[1].vitorias++;
            }
            ganhou = 1;
        }
```

```
            // Verifica se deu velha if
            (jogoVelha(jogo)) { exibirTabuleiro(jogo);
            printf("\nO jogo terminou em velha!\n");
            ganhou = 1;
            }

        } while (!ganhou);

        salvarRanking(jogadores);
        break;
    }
    case 2:
    mostrarRanking();
    break; case 3:
        exibirCreditos()
    ; break; case 4:
        printf("\nSaindo do
    jogo...\n"); loop = 0; break;
    default:
        printf("\nOpção Inválida!\n");
        break;
  }

  if (loop != 0) { printf("\nDigite '1' para voltar ao menu, ou '0' para sair
    do jogo.\n"); loop = lerNumero();
  }
}

// Libera a memória alocada para o tabuleiro
for (int i = 0; i < 3; i++) {
  free(jogo[i]);
```

```
    }
    free(jogo);

    return 0;
}
```