Nome: Cleverson Pereira da Silva **TIA:** 32198531

Nome: Gustavo Teixeira dos Santos **TIA:** 32197020

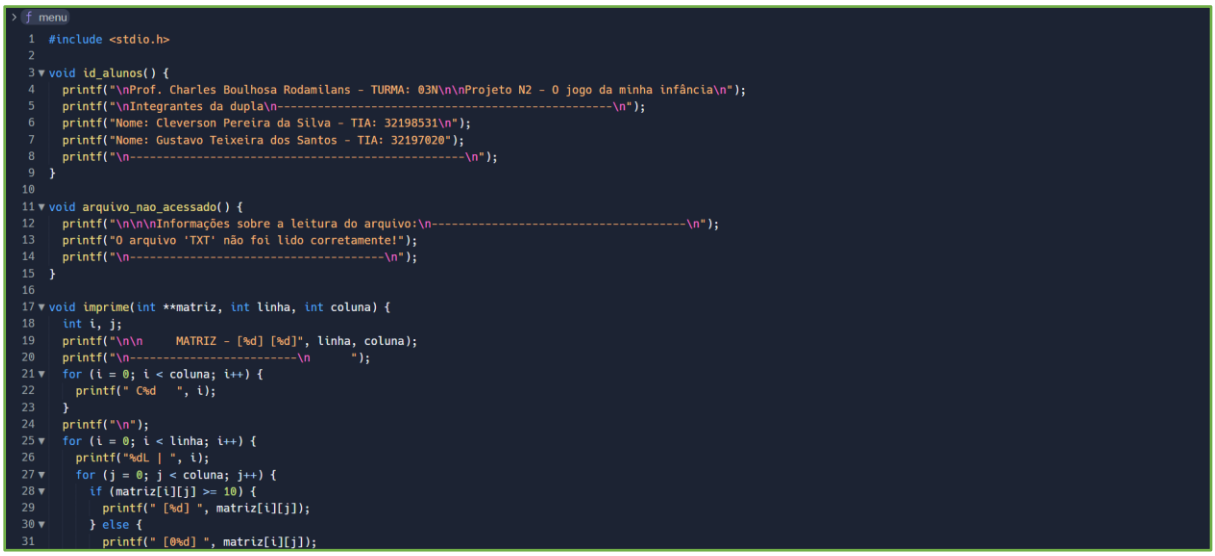
Prof. Charles Boulhosa Rodamilans - TURMA: 03N

Projeto N2 – O jogo da minha infância

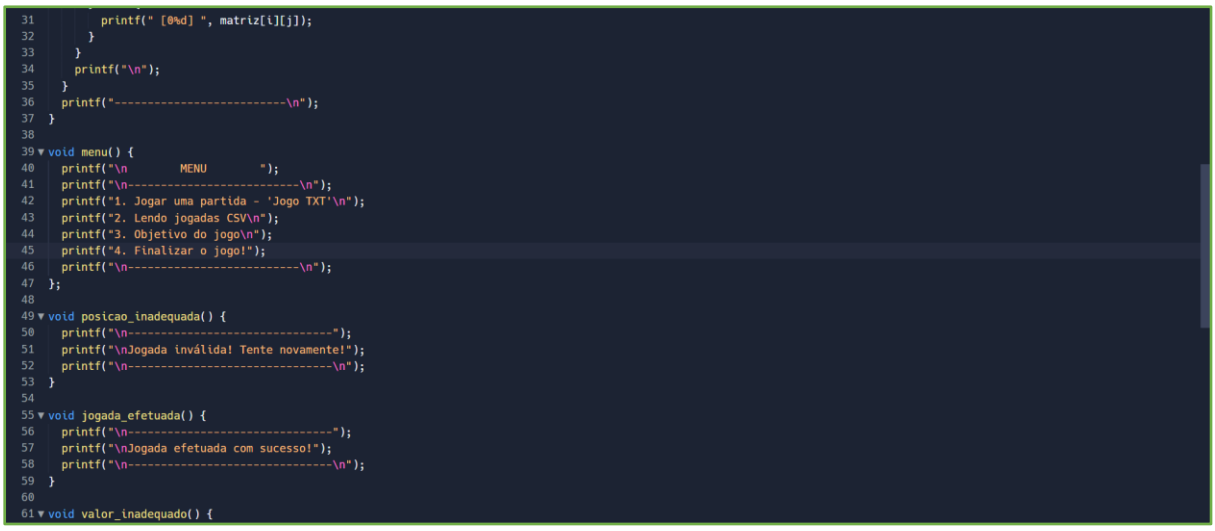
Código em execução - https://www.youtube.com/watch?v=REAOnhxpi48

Relatório

1. **O arquivo “Exibição.h” foi criado no intuito de diminuição dos print’s dentro do main.c**



```c
f menu
#include <stdio.h>

void id_alunos() {
  printf("\nProf. Charles Boulhosa Rodamilans - TURMA: 03N\n\nProjeto N2 - O jogo da minha infância\n");
  printf("\nIntegrantes da dupla\n-----------------------------------------------------\n");
  printf("Nome: Cleverson Pereira da Silva - TIA: 32198531\n");
  printf("Nome: Gustavo Teixeira dos Santos - TIA: 32197020");
  printf("\n-----------------------------------------------------\n");
}

void arquivo_nao_acessado() {
  printf("\n\n\nInformações sobre a leitura do arquivo:\n-------------------------------------\n");
  printf("O arquivo 'TXT' não foi lido corretamente!");
  printf("\n-------------------------------------\n");
}

void imprime(int **matriz, int linha, int coluna) {
  int i, j;
  printf("\n\n     MATRIZ - [%d] [%d]", linha, coluna);
  printf("\n-------------------------\n      ");
  for (i = 0; i < coluna; i++) {
    printf(" C%d   ", i);
  }
  printf("\n");
  for (i = 0; i < linha; i++) {
    printf("%dL | ", i);
    for (j = 0; j < coluna; j++) {
      if (matriz[i][j] >= 10) {
        printf(" [%d] ", matriz[i][j]);
      } else {
        printf(" [0%d] ", matriz[i][j]);
```



```c
        printf(" [0%d] ", matriz[i][j]);
      }
    }
    printf("\n");
  }
  printf("-------------------------\n");
}

void menu() {
  printf("\n        MENU        ");
  printf("\n-------------------------\n");
  printf("1. Jogar uma partida - 'Jogo TXT'\n");
  printf("2. Lendo jogadas CSV\n");
  printf("3. Objetivo do jogo\n");
  printf("4. Finalizar o jogo!");
  printf("\n-------------------------\n");
};

void posicao_inadequada() {
  printf("\n------------------------------");
  printf("\nJogada inválida! Tente novamente!");
  printf("\n------------------------------\n");
}

void jogada_efetuada() {
  printf("\n------------------------------");
  printf("\nJogada efetuada com sucesso!");
  printf("\n------------------------------\n");
}

void valor_inadequado() {
```

```c
void jogada_efetuada() {
  printf("\n--------------------------------");
  printf("\nJogada efetuada com sucesso!");
  printf("\n--------------------------------\n");
}

void valor_inadequado() {
  printf("--------------------------------");
  printf("\nValor inválido! Tente novamente!");
  printf("\n--------------------------------\n");
}

void jogador_ganhou() {
  printf("--------------------------------");
  printf("\nParabéns! Você venceu o jogo!");
  printf("\n--------------------------------\n");
}

void objetivo() {
  printf("\n------------------\nObjetivo\n------------------\n\nOrdenar os "
  "quadrados, da esquerda para a direita e de cima para baixo, isto "
  "é,\nobter a disposição original dos contadores depois de terem sido "
  "aleatoriamente deslocados.\n\nFonte: Wikipedia\n\n");
}

```

2. **O arquivo “metodo_csv” foi criado para contém as funções devidas para realização da opção “2 – Lendo jogadas”.** **LEITURA DO CSV**

metodos_csv.h

```c
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include "exibicao.h"

typedef struct { // Define Struct para armazenar os passos realizados pelo
                 // usuário

  char direcao;
  int valor;

} passos;

// FUNÇÃO PARA CHECAR SE A MATRIZ ESTÁ ORDENADA DE FORMA ITERATIVA
// Essa função trabalha seguindo algumas regras:
// 1 - Se o valor for checado for 0, então ele obrigatoriamente precisa estar na
// última posição da matriz 2 - Se o j (índice da coluna) for 0 e não estiver na
// primeira linha, ele checa o valor atual com a última coluna da linha anterior
// (Sua posição anterior na ordenação) 3 - Se o j (índice da coluna) for maior
// que 0 e não for a última coluna da linha da matriz, checa a posição atual com
// a próxima 4 - Se os laços finalizarem e a função ainda não tiver retornado
// false, então a matriz está ordenada



// FUNÇÃO PARA VERIFICAR SE A MATRIZ ESTÁ ORDENADA DE FORMA RECURSIVA
// Essa função recebe um vetor com todos os valores da matriz para fazer a
// checagem de forma recursiva Essa função segue algumas regras: 1 - Se a
// posição atual do vetor for maior que a anterior, ativa a recursividade
// passando tamanho = tamanho - 1. Caso contrário retorna false 2 - Se a função
// recursiva conseguir chegar no primeiro índice da matriz (Último valor da
```

```c
// checagem), então a função é verdadeira.] OBS: A posição o 0 já é checada
// antes da chamada da função
bool checaOrdenacao(int *vetor, int tamanhoVetor) {

  if (tamanhoVetor == 1) { // Se a variável tamanhoVetor estiver apontando para
                           // a primeira posição do vetor
return true;           // Retorna true
  } else {                 // Caso contrário
    if (vetor[tamanhoVetor - 1] >=
        vetor[tamanhoVetor -
              2]) { // Checa se a posição atual é >= à posição anterior
      return checaOrdenacao(
          vetor,
          tamanhoVetor - 1); // Caso a condição seja verdadeira, ativa a
                             // recursividade checando as posições anteriores
    } else {                 // Caso contrário
      return false;          // Retorna false
    }
  }
}

// FUNÇÃO PARA PRINTAR O PASSO ATUAL DA MATRIZ PARA O USUÁRIO
// Essa função foi criada pois em diversas áreas do software a matriz é exibida
// ao usuário

void printMatriz(int **matriz, int tamanhoMatriz) {

  for (int i = 0; i < tamanhoMatriz; i++) {
    for (int j = 0; j < tamanhoMatriz; j++) {
      if (matriz[i][j] >= 10)        // Se o valor atual da checagem for >= 10
        printf("%d ", matriz[i][j]); // Exibe somente o valor
```

```
      else                          // Caso contrário
        printf( "0%d ", matriz[i][j]); // Exibe um 0 antes do valor para ficar padronizado
    }
    printf("\n");
  }
}

// FUNÇÃO PARA RECEBER OS PASSOS DA MATRIZ E RESOLVE O JOGO
// Essa função procura onde está o 0 e faz as checagens e alterações a partir do
// 'ponto de vista' do mesmo

bool resolveMatriz(passos *passos, int **matriz, int tamanhoPassos, int tamanhoMatriz) {
  int linha, coluna, contador, *aux;
  // Bloco de código que encontra a exata posição do 0;

  linha = -1; // Linha e coluna recebem -1. Para checar se a tabela está no
  // formato certo
  coluna = -1;
  for (int i = 0; i < tamanhoMatriz; i++) {
    for (int j = 0; j < tamanhoMatriz; j++) {
      if (matriz[i][j] == 0) { // Se a posição em específico da matriz for 0
        if (linha == -1 &&
            coluna ==
                -1) { // E ainda não tiver um valor atribuído à linha e coluna
                      // (Ou seja, ainda não foi encontrado nenhum 0)
          // Variáveis linha e coluna recebem a posição exata onde está o 0
          linha = i;
          coluna = j;
        } else {        // Caso contrário
          return false; // Retorna falso
        }
      }
    }
  }

  if (linha == -1 || coluna == -1) { // Se não encontrou um 0 na matriz
    return false;                    // retorna falso
  }

  // Bloco de código dedicado a resolução dos passos inseridos pelo usuário
  for (int i = 0; i < tamanhoPassos; i++) {

    switch (passos[i].direcao) { // Switch case para cada passo inserido

    case 'c': // Caso o passo indique 'para cima'

      if ((linha < tamanhoMatriz - 1) &&
          (matriz[linha + 1][coluna] ==
           passos[i].valor)) { // Se a posição abaixo do 0 for o número do passo
                               // e não estiver na última linha da matriz
        // troca a posição do número com a do 0 e atualiza a posição do mesmo
        matriz[linha + 1][coluna] = 0;
        matriz[linha][coluna] = passos[i].valor;
        linha++;
        printf("\n %dº passo: Colocando o número %d para cima\n", i + 1,
               passos[i].valor);

      } else {        // Caso contrário
        return false; // Retorna false
      }

      break;

    case 'b': // Caso o passo indique 'para baixo'

      if ((linha > 0) && (matriz[linha - 1][coluna] == passos[i].valor)) {
        // Se a posição acima do 0 for o número do passo // e não estiver na primeira posição da matriz
        // Troca a posição do número com a do 0 e atualiza a posição do mesmo
        matriz[linha - 1][coluna] = 0;
        matriz[linha][coluna] = passos[i].valor;
        linha--;
        printf("\n %dº passo: Colocando o número %d para baixo\n", i + 1, passos[i].valor);
      } else {        // Caso contrário
        return false; // Retorna false
      }

      break;
    case 'd': // Caso o passo indique 'à direita'

      if ((coluna > 0) && (matriz[linha][coluna - 1] == passos[i].valor)) {
        // Se a posição à esquerda do 0 for o número do passo // e não estiver na primeira posição da coluna
        // Troca a posição do número com a do 0 e atualiza a posição do mesmo
        matriz[linha][coluna - 1] = 0;
        matriz[linha][coluna] = passos[i].valor;
        coluna--;
        printf("\n %dº passo: Colocando o número %d para direita\n", i + 1, passos[i].valor);
      } else {        // Caso contrário
        return false; // Retorna false
      }

      break;
    case 'e': // Caso o passo indique 'à esquerda'

      if ((coluna < tamanhoMatriz - 1) &&
```

```
155       if ((coluna < tamanhoMatriz - 1) &&
156         (matriz[linha][coluna + 1] == passos[i].valor)) {
157         // Se a posição à direita do 0 for o numero do passo e// não estiver na última posição da coluna
158         // Troca a posição do número com a do 0 e atualiza a posição do mesmo
159         matriz[linha][coluna + 1] = 0;
160         matriz[linha][coluna] = passos[i].valor;
161         coluna++;
162         printf("\n %dº passo: Colocando o número %d para esquerda\n", i + 1, passos[i].valor);
163       } else {        // Caso contrário
164         return false; // Retorna false
165       }
166
167       break;
168     }
169
170     imprime(matriz, tamanhoMatriz, tamanhoMatriz); // Ao final de cada passo, printa o // estado atual da matriz
171   }
172
173   // Inicia os passos para chamar as funções iterativas e recursivas
174   if (matriz[tamanhoMatriz - 1][tamanhoMatriz - 1] == 0) { // Se a última posição da matriz for 0
175     aux = (int*)malloc((tamanhoMatriz * tamanhoMatriz) * sizeof(int)); // Inicia um vetor para receber os valores da // matriz e ser enviado para a função recursiva
176     contador = 0;
177
178     for (int i = 0; i < tamanhoMatriz; i++) {
179       for (int j = 0; j < tamanhoMatriz; j++) {
180         aux[contador] = matriz[i][j]; // Copia o valor em específico da matriz // para a posição em específico do vetor
181         contador++; // Atualiza o contador
182       }
183     }
184
185     if (checaOrdenacao(aux, tamanhoMatriz * tamanhoMatriz - 1) == true) { // Se estiver ordenado
```

```
183     }
184
185     if (checaOrdenacao(aux, tamanhoMatriz * tamanhoMatriz - 1) == true) { // Se estiver ordenado
186       return true;  // Retorna true
187     }else{        // Caso contrário
188       return false; // Retorna false
189     }
190   } else { // Se a última posição da matriz não for 0, nem realiza a checagem e // ja retorna false
191     return false; // Retorna false
192   }
193 }
```

3. O arquivo “métodos.h” é para realização da opção 1, que foi desenvolvido para o usuário jogar! “OBS: Ficar atento ao caminho do arquivo” . LEITURA DO TXT

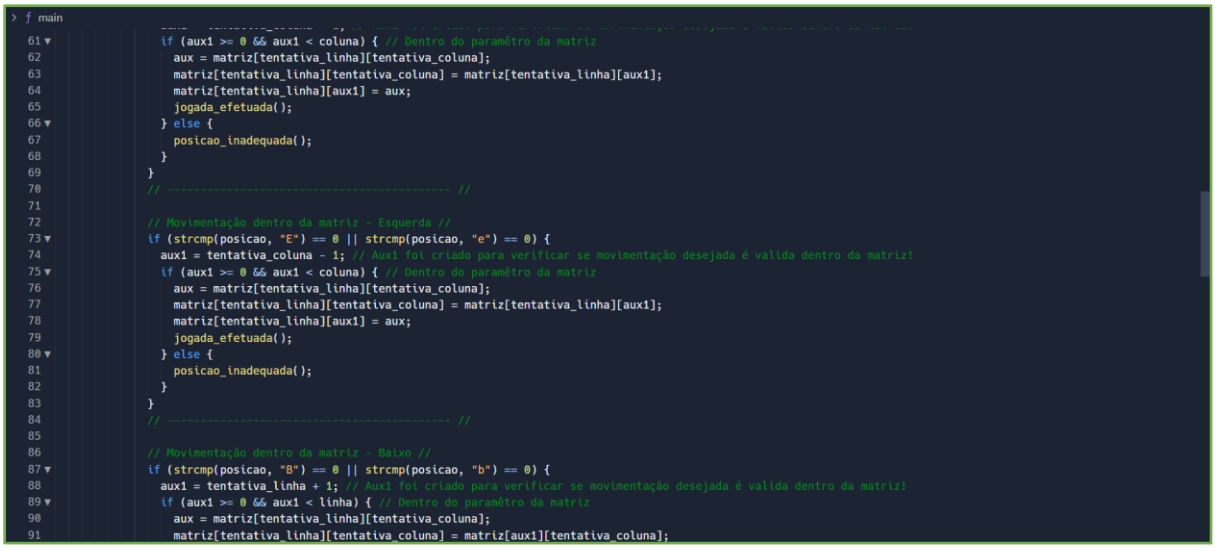
```
› ...
1  #include <stdio.h>
2
3  int leitura_txt(int **matriz, int linha, int coluna) {
4    int i, j, aux_integer;
5    FILE *arquivo;
6    arquivo = fopen("ladrilho_para_jogar.txt", "r");
7
8    if (arquivo != NULL) {
9      for (i = 0; i < linha; i++) {
10       for (j = 0; j < coluna; j++) {
11         fscanf(arquivo, "%d", &aux_integer);
12         matriz[i][j] = aux_integer;
13       }
14     }
15     printf("\nInformações sobre txt:\n-------------------------------------\n");
16     printf("Leitura realizada com sucesso!\n");
17     printf("-------------------------------------\n");
18     return 1;
19   } else {
20     printf("\nInformações sobre txt:\n-------------------------------------\n");
21     printf("Leitura não foi realizada com sucesso!\n");
22     printf("-------------------------------------\n");
23     printf("Verifique se diretório está correto\n\n");
24     return 0;
25   }
26 }
27
28 void gerar_resposta(int **matriz_resposta, int linha, int coluna) {
29   int i, j, contador = 1, ultima_l = linha - 1;
30   for (i = 0; i < linha; i++) {
31     for (j = 0; j < coluna; j++) {
```

```
30   for (i = 0; i < linha; i++) {
31     for (j = 0; j < coluna; j++) {
32       matriz_resposta[i][j] = contador;
33       contador++;
34     }
35   }
36   matriz_resposta[ultima_l][ultima_l] = 0;
37 }
38
39 int ganhou(int **matriz, int **matriz_resposta, int linha, int coluna) {
40   int i, j, contador = 0, ultima_l = linha - 1, ultima_c = coluna - 1;
41   for (i = 0; i < linha; i++) {
42     for (j = 0; j < coluna; j++) {
43       if (matriz[i][j] != matriz_resposta[i][j]) {
44         return 0;
45       }
46     }
47   }
48   return 1;
49 }
```

4. **No arquivo “main.c” foi incluído todos os arquivos auxiliares e chamados na função principal.**

```c
#include "metodos.h"
#include "metodos_csv.h"
#include <stdio.h>
#include <stdlib.h>

int main() {
  id_alunos();
  int **matriz, **matriz_resposta, linha, coluna, i, j, opcao = 0, contador = 0;
  int tentativa_linha, tentativa_coluna, tentativa_valor, aux = 0, verifica_ganhou = 0;
  char posicao[1];  int aux1 = 0;

  while (opcao != 4) {
    menu();
    printf("\nQual opção desejada: ");
    scanf("%d", &opcao);
    if (opcao == 1) {
      // -------------- Os valores de linha e coluna para matriz dinâmica ------------------ //
      printf("\n\nDigite número de linhas da matriz: "); scanf("%d", &linha);
      printf("Digite número de colunas da matriz: "); scanf("%d", &coluna); printf("\n");
      // -------------- Alocação feita para matriz ------------------ //
      matriz = malloc(linha * sizeof(int));
      for (i = 0; i < linha; i++) {
        matriz[i] = malloc(coluna * sizeof(int));
      }
      matriz_resposta = malloc(linha * sizeof(int));
      for (i = 0; i < linha; i++){
        matriz_resposta[i] = malloc(coluna * sizeof(int));
      }
      // ------------------------------------------------------------ //
      if (leitura_txt(matriz, linha, coluna) == 1){
```

```c
        printf("\nJogo começou!\n");
        // Gerar_resposta é uma forma de comparar se o jogador venceu o jogo!
        //-------------------------------------------------------------------//
        gerar_resposta(matriz_resposta, linha, coluna);
        //-------------------------------------------------------------------//
        while (verifica_ganhou != 1){
          //Esta função imprime e molde de visualização do jogo//
          imprime(matriz, linha, coluna);
          //-------------------------------------------------//

          // O jogador irá inserir os valores de linha e coluna
          printf("\nDigite a linha desejada: ");
          scanf("%d", &tentativa_linha);
          printf("\nDigite a coluna desejada: ");
          scanf("%d", &tentativa_coluna);
          printf("\n");
          //-------------------------------------------------//

          // A posição inserida pelo jogador deve está dentro do parametro da matriz
          if (tentativa_linha >= 0 && tentativa_linha <= linha && tentativa_coluna >= 0 && tentativa_coluna <= coluna) {
            // Selecionando a posição que possue a posição vazia "0"
            if (matriz[tentativa_linha][tentativa_coluna] == 0) {
              printf("Posição escolhida: [%d][%d] = %d", tentativa_linha, tentativa_coluna, matriz[tentativa_linha][tentativa_coluna]);
              printf("\n\n----------\nE = Esquerda | \nD = Direita  | \nB = Baixo    |\nC = Cima     |\n----------\n");
              printf("\nQual a movimentação?: ");
              scanf("%s", posicao);
            //-------------------------------------------------//
              // Movimentação dentro da matriz - Direita //
              if (strcmp(posicao, "D") == 0 || strcmp(posicao, "d") == 0){
                aux1 = tentativa_coluna + 1; // Aux1 foi criado para verificar se movimentação desejada é valida dentro da matriz!
                if (aux1 >= 0 && aux1 < coluna) { // Dentro do paramêtro da matriz
```



```c
                if (aux1 >= 0 && aux1 < coluna) { // Dentro do paramêtro da matriz
                  aux = matriz[tentativa_linha][tentativa_coluna];
                  matriz[tentativa_linha][tentativa_coluna] = matriz[tentativa_linha][aux1];
                  matriz[tentativa_linha][aux1] = aux;
                  jogada_efetuada();
                } else {
                  posicao_inadequada();
                }
              }
              // ------------------------------------------- //

              // Movimentação dentro da matriz - Esquerda //
              if (strcmp(posicao, "E") == 0 || strcmp(posicao, "e") == 0) {
                aux1 = tentativa_coluna - 1; // Aux1 foi criado para verificar se movimentação desejada é valida dentro da matriz!
                if (aux1 >= 0 && aux1 < coluna) { // Dentro do paramêtro da matriz
                  aux = matriz[tentativa_linha][tentativa_coluna];
                  matriz[tentativa_linha][tentativa_coluna] = matriz[tentativa_linha][aux1];
                  matriz[tentativa_linha][aux1] = aux;
                  jogada_efetuada();
                } else {
                  posicao_inadequada();
                }
              }
              // ------------------------------------------- //

              // Movimentação dentro da matriz - Baixo //
              if (strcmp(posicao, "B") == 0 || strcmp(posicao, "b") == 0) {
                aux1 = tentativa_linha + 1; // Aux1 foi criado para verificar se movimentação desejada é valida dentro da matriz!
                if (aux1 >= 0 && aux1 < linha) { // Dentro do paramêtro da matriz
                  aux = matriz[tentativa_linha][tentativa_coluna];
                  matriz[tentativa_linha][tentativa_coluna] = matriz[aux1][tentativa_coluna];
```

```
> ƒ main
                matriz[aux1][tentativa_coluna] = aux;
                jogada_efetuada();
              } else {
                posicao_inadequada();
              }
            }
            // ---------------------------------------- //

            // Movimentação dentro da matriz - Cima //
            if (strcmp(posicao, "C") == 0 || strcmp(posicao, "c") == 0) {
              aux1 = tentativa_linha - 1; // Aux1 foi criado para verificar se movimentação desejada é valida dentro da matriz!
              if (aux1 >= 0 && aux1 < linha) { // Dentro do paramêtro da matriz
                aux = matriz[tentativa_linha][tentativa_coluna];
                matriz[tentativa_linha][tentativa_coluna] = matriz[aux1][tentativa_coluna];
                matriz[aux1][tentativa_coluna] = aux;
                jogada_efetuada();
              }else{
                posicao_inadequada();
              }
            }
            // ---------------------------------------- //
          }
        }
        if (ganhou(matriz, matriz_resposta, linha, coluna) == 1) {
          jogador_ganhou();
          verifica_ganhou = 1;
        }
      }
    }
  }
  if (opcao == 2) {
```

```
> ƒ main
  if (opcao == 2) {

    int tamanhoMatriz, tamanhoPassos;
    int **matriz = NULL;   // ponteiro que receberá a matriz de valores
    passos *Passos = NULL; // ponteiro da struct que receberá os passos
    FILE *input = fopen("ladrilhos.csv", "r"); // Inicia a conexão com o arquivo csv

    fscanf(input, "%d", &tamanhoMatriz); // Lê o tamanho da matriz quadrada
    matriz = malloc(tamanhoMatriz * sizeof(int)); // Aloca dinâmicamente as colunas da matriz
    for (int i = 0; i < tamanhoMatriz; i++) {
      matriz[i] = malloc(tamanhoMatriz * sizeof(int)); // Aloca dinâmicamente as linhas da matriz
      for (int j = 0; j < tamanhoMatriz; j++) { // Logo após isso insere os valores
        fscanf(input, "%d", &matriz[i][j]);
      }
    }

    fscanf(input, "%d", &tamanhoPassos); // Lê a quantidade de passos que o usuário vai passar
    Passos = malloc(tamanhoPassos * sizeof(passos)); // Inicia o vetor de structs que receberá os passos
    for (int i = 0; i < tamanhoPassos; i++) {
      fscanf( input, "%d %c", &Passos[i].valor, &Passos[i].direcao); // Lê os passos do usuário e insere na struct
    }

    printf("\nMatriz inicial:\n");
    linha = tamanhoMatriz;
    coluna = tamanhoMatriz;
    imprime(matriz, linha, coluna); // Exibe a posição inicial da matriz
    if (resolveMatriz(Passos, matriz, tamanhoPassos,tamanhoMatriz)) { // Se o software conseguir resolver os passos e // no final a matriz estiver ordenada
      printf("\n\nPROBLEMA RESOLVIDO!!!\n\n");
      // Exibe mensagem indicando que // o usuário conseguiu  // completar o jogo // Caso contrário
    } else {
      printf("\n\nERO0000000!!!");
```

```
      // Exibe mensagem indicando que o usuário // não conseguiu completar o jogo
    }
  }
  if (opcao == 3){
    objetivo();
  }
  if (opcao == 4){
    printf("\n\n-----------------------------------------\n");
    printf("Obrigado por jogar! Até breve");
    printf("\n-----------------------------------------\n\n");
  }
 }
}
```