IPS Instituto Politécnico de Setúbal

Mastermind

Desenvolvido por:

- Edgar Santos, Nº 160210035
- Miguel Rodrigues, Nº 160210012

Professora: Graça Fraga

Curso: CTeSP TPSI

Unidade curricular: Algoritmo e Fundamentos de Programação

Ano Letivo: 2016/2017

IPS Instituto Politécnico de Setúbal

Índice

IPS Instituto Politécnico de Setúbal

Introdução

Este projeto foi realizado no âmbito da disciplina de Algoritmo e Fundamentos de Programação, onde foi proposto a realização de um projeto entre três possibilidades, sendo que escolhemos o jogo Mastermind.

Neste relatório iremos apresentar o nosso projeto, em que consiste e como se usa a aplicação, tal como alguns detalhes técnicos, tais como as estruturas de dados utilizadas, as limitações da aplicação, que tentámos que fossem mínimas e mostraremos também alguns algoritmos de maior relevância.

IPS Instituto Politécnico de Setúbal

Manual Técnico

Apresentação do projeto

Este projeto foi desenvolvido no âmbito da Unidade Curricular Algoritmo e Fundamentos de Programação, e tem como objetivo demonstrar os conhecimentos adquiridos ao longo do semestre na linguagem de programação C.

A aplicação desenvolvida foi o jogo Mastermind, que tem como objetivo descobrir uma combinação de cores e as suas posições numa linha de 4 pinos, gerada pelo computador.

Estruturas de dados

Foram criadas duas estruturas de dados neste projeto.

As estruturas de dados criadas foram definidas como tipos, nomeadamente:

- **ScreenResolution**
  - **Objetivo:** Guardar a resolução do ecrã definida pelo sistema.
  - **Estrutura:**
    - Variavel width, do tipo DWORD;
    - Variavel height, do tipo DWORD.

- **PlayerScore**
  - **Objetivo:** Guardar a data, a hora, o nome e a pontuação do jogador.
  - **Estrutura:**
    - Vetor name, do tipo char, de dimensão definida pela constante MAX_PLAYER_NAME;
    - Variavel score, do tipo int;
    - Estrutura time, do tipo struct tm, que faz parte da biblioteca time.h.

Limitações da aplicação

Tendo em consideração os nossos objetivos neste projeto consideramos que a nossa aplicação possui as seguintes limitações:

- O código apenas pode ser compilado no sistema operativo Windows, devido a limitações de algumas bibliotecas usadas;
- A aplicação pode ser compilada no Windows 10, no entanto alguns textos serão apresentados de forma incompleta ou incorreta, esta situação deve-se as alterações na API do Windows que foram aplicadas nas compilações mais recentes do Windows 10;
- A falta da possibilidade de alterar algumas configurações durante a execução do jogo, pois os sistemas do jogo foram desenhados para funcionar de forma dinâmica, ou seja, ao alterar determinadas configurações no ficheiro settings.h, tais como, o numero de peças da solução, o numero de tentativas, as cores usadas nas peças, entre outras, o resultado será visível no jogo, não sendo necessário realizar modificações no código.
- O compilador usado, **MinGW**, apresenta varias limitações devido à falta de atualizações do mesmo, algumas funcionalidades mais recentes não estão disponíveis neste compilador, um exemplo de uma restrição foi o uso das funções que permitem alterar a fonte usada pela consola, fornecidas pela API do Windows, que foram adicionadas apenas a partir do Windows Vista.

Opções de implementação usadas

Foi usada a biblioteca windows.h por possibilitar o uso de determinadas funcionalidades de forma mais acessível no sistema operativo Windows;

Foi usada a biblioteca locale.h por permitir o uso de carateres da língua portuguesa;

A aplicação foi desenhada fazendo uso de ficheiros de cabeçalho (.h), separando assim as funcionalidades de maior relevância, permitindo uma melhor e mais acessível analise do código;

Foram usados métodos e funções na construção da aplicação, sendo que cada método ou função de maior relevância é constituído por um conjunto de outros métodos e funções, permitindo assim remover as funcionalidades mais genéricas dos métodos/funções principais;

Os métodos e funções comuns aos vários ficheiros da aplicação foram separados dos demais, estando presentes na biblioteca util.h, permitindo assim uma maior organização do código;

O menu apenas é desenhado no ecrã quando necessário, ou seja, caso a opção introduzida pelo utilizador seja inválida, o ecrã não sofre qualquer alteração no conteúdo, poupando assim tempo de processamento e melhorando a aparência gráfica;

O vetor do tabuleiro é gerado apenas uma vez durante a execução da aplicação, no caso é gerado na primeira vez que o jogador começar uma jogada desde que abriu a aplicação;

Os únicos casos onde foi usada a estrutura de seleção switch em vez de if foi no algoritmo do menu, por haver um elevado numero de opções, e no algoritmo do jogo para realizar uma ação dependendo da tecla direcional pressionada;

Nos casos onde é pretendido que o utilizador pressione uma tecla em especifico foi usada a função getch(); da biblioteca conio.h para obter a tecla pressionada sem que esta seja escrita no ecrã;

Os textos escritos com um conjunto de carateres ASCII foram guardados numa constante a nível da função onde o respetivo texto é apresentado, para facilitar a leitura e alteração dos mesmos;

Os textos corridos, tais como a Introdução e as Instruções do jogo, foram escritos usando apenas uma vez a função printf();, fazendo uso do carater \ que permite continuar o texto na linha a seguinte e do carater \r que move o cursor para o inicio da linha, desta forma foi possível manter todos os textos longos, organizados e de fácil interpretação e edição no código;

As cores das peças usadas no jogo foram definidas no vetor constante PIECE_COLORS do tipo WORD, facilitando o ciclo pelas peças e a geração da chave secreta;

Foi usada a função Sleep(); para realizar uma animação enquanto a chave é gerada, o uso desta função tem algumas desvantagens, no entanto era a opção de mais fácil implementação, sendo que a desvantagem com maior impacto no nosso projeto, no caso, as teclas pressionadas pelo utilizador durante o período definido na função eram guardadas em fila de espera e executadas quando o programa retornasse a execução, foi corrigida descartando toda e qualquer entrada realizada pelo utilizador durante esse período;

O ranking dos jogadores foi guardado num ficheiro de texto, devido a não ser necessário efetuar a gestão nem analise desses dados, pois sempre que o ficheiro é aberto é carregado na sua totalidade para um vetor da estrutura do tipo PlayerScore;

O ficheiro de ranking é criado ou reescrito, caso já exista, sempre que for adicionado um novo jogador ao ranking, não foi usada um sistema mais complexo pois, tratando-se de um TOP de jogadores, o número de registos é reduzido, sendo neste caso apenas 10.

Algoritmos de maior relevância

Os algoritmos serão apresentados em pseudocódigo.

- Algoritmo do menu do jogo (displayMenu();):

```
FUNÇÃO displayMenu ( hStdout, hStdin: HANDLE, screenRes: ScreenResolution ) : void
INICIO
VAR menuOption: char, drawMenu: int
drawMenu <- 1
ENQUANTO ( 1 )
REPETIR
        SE drawMenu != 0 ENTÃO
                clearScreen ( hStdout )
setWindowSize ( hStdout, screenRes.width, screenRes.height )
SetConsoleTextAttribute ( hStdout, FOREGROUND_CYAN_INTENSE )
ESCREVA ( “
======================
| Mastermind - Menu    |
======================
“ )
SetConsoleTextAttribute ( hStdout, FOREGROUND_GREEN_INTENSE )
ESCREVA ( “1 – Jogar” )
SetConsoleTextAttribute ( hStdout, FOREGROUND_WHITE )
ESCREVA ( “
2 – Ranking
        3 – Instruções
        4 – Som: “ ( SE gameSound == 0 ENTÃO “Ligado” SENÃO “Desligado” ) “
        5 – Sobre

        0 – Sair
        “ )
SetConsoleTextAttribute ( hStdout, FOREGROUND_RED_INTENSE )
ESCREVA ( “Escolha uma opção!” )
```

```
SetConsoleCursorPosition ( hStdout,  ( COORD ) { 1, 14 } )
setWindowSize ( hStdout, 23, 16 )
        REPETIR
                menuOption <- getch ( )
        ENQUANTO ( menuOption == 13 )
        drawMenu <- 1
        ESCOLHA ( menuOption )
                CASO ( '1' ):
                        prepareGame ( hStdout, screenRes )
                        PARE
                CASO ( '2' ):
                        displayRanking ( hStdout, screenRes )
                        PARE
                CASO ( '3' ):
                        displayInstructions ( hStdout, screenRes )
                        PARE
                CASO ( '4' ):
                        SE gameSound != 0 ENTÃO
                                gameSound <- 0
                        SENÃO
                                gameSound <- 1
                                playBeep ( gameSound, MB_ICONINFORMATION )
                        PARE
                CASO ( '5' ):
                        displayWelcome ( hStdout, screenRes )
                        PARA
                CASO ( '0' ):
                        RETORNE
                PADRÃO:
                        drawMenu <- 0
FIMFUNÇÃO
```

- Algoritmo de criação do ranking (checkTopScore(); e addTopScore();):

```
FUNÇÃO checkTopScore ( hStdout: HANDLE, screenRes: ScreenResolution, attempts: int ) : void
INICIO
VAR *ranking: FILE, player [ RANK_TOP_SIZE ]: PlayerScore, players, i: int, name [ MAX_PLAYER_NAME ]: char
players <- 1
i <- 0
ranking <- fopen ( FILE_RANKING_PATH, “r” )
SE ranking == 0 ENTÃO
requestPlayerName ( hStdout, screenRes, name, MAX_PLAYER_NAME - 1 )
SE name [ 0 ] != ‘\0’ ENTÃO
addTopScore ( hStdout, player, players, 0, attempts, name )
                RETORNE
ENQUANTO ( fscanf ( ranking, "%d-%d-%d %d:%d:%d %d %" MAX_PLAYER_NAME_STRING "[^\n]", &player [ players ].time.tm_mday, &player [ players ].time.tm_mon, &player [ players ].time.tm_year, &player [ players ].time.tm_hour, &player [ players ].time.tm_min, &player [ players ].time.tm_sec, &player [ players ].score, player [ players ].name ) != EOF && players < RANK_TOP_SIZE )
REPETIR
        players <- players + 1
fclose ( ranking )
SE players == 0 ENTÃO
        requestPlayerName ( hStdout, screenRes, name, MAX_PLAYER_NAME - 1 )
        SE name [ 0 ] != ‘\0’ ENTÃO
                addTopScore ( hStdout, player, players, 0, attempts, name )
        RETORNE
PARA ( i <- 0, I < players, i <- i + 1 ) FAÇA
        SE attempts < player [ i ].score ENTÃO
                requestPlayerName ( hStdout, screenRes, name, MAX_PLAYER_NAME - 1 )
                SE name [ 0 ] != ‘\0’ ENTÃO
                        addTopScore ( hStdout, player, players, i, attempts, name )
                PARE
        SENÃO SE i == player – 1 && I < RANK_TOP_SIZE – 1 ENTÃO
                requestPlayerName ( hStdout, screenRes, name, MAX_PLAYER_NAME - 1 )
                SE name [ 0 ] != ‘\0’ ENTÃO
                        addTopScore ( hStdout, player, players, i + 1, attempts, name )
                PARE
```

```
FIMPARA

FIMFUNÇÃO
```

```
FUNÇÃO addTopScore ( hStdout: HANDLE, player [ ]: PlayerScore, players, index, attempts: int, *name: const char ) : void

INICIO

VAR *rankinf: FILE, rawtime: time_t, i: int

SE players + 1 <= RANK_TOP_SIZE && players != 0 ENTÃO

        players <- players + 1

PARA ( i = players – 1, I > index, i <- i - 1 ) FAÇA

        Player [ i ] <- player [ i – 1 ]

FIMPARA

time ( &rawtime )

player [ index ].time <- *localtime ( &rawtime )

player [ index ].time.tm_mon <- player [ index ].time.tm_mon + 1

player [ index ].time.tm_year <- player [ index ].time.tm_year + 1900

player [ index ].score <- attempts

strcpy ( player [ index ].name, name )

ranking = fopen ( FILE_RANKING_PATH, "w" )

SE ranking == 0 ENTÃO

        clearScreen ( hStdout )

        SetConsoleTextAttribute ( hStdout, FOREGROUND_RED_INTENSE )

        printfXY ( hStdout, 1, 1, "Erro: A gravação da pontuação falhou!" )

        SetConsoleTextAttribute ( hStdout, FOREGROUND_RED_INTENSE )

        printfXY ( hStdout, 1, 3, "Prima qualquer tecla para continuar!" )

        SetConsoleCursorPosition ( hStdout, ( COORD ) { 1, 3 } )

        getch ( )

        RETORNE

SE players == 0 ENTÃO

        players <- players + 1

PARA ( i = 0, i < players, I <- i +1 ) FAÇA

        fprintf ( ranking, "%02d-%02d-%d %02d:%02d:%02d %d %s\n", player [ i ].time.tm_mday, player [ i ].time.tm_mon, player [ i ].time.tm_year, player [ i ].time.tm_hour, player [ i ].time.tm_min, player [ i ].time.tm_sec, player [ i ].score, player [ i ].name )

FIMPARA
```

```
fclose ( ranking )

FIMFUNÇÃO
```

- Algoritmo principal do jogo (startGame();):

```
FUNÇÃO startGame ( hStdout: HANDLE, screenRes: ScreenResolution, csbi: CONSOLE_SCREEN_BUFFER_INFO ) : void

INICIO

VAR i, x, keyInput, attemptPieces [ GAME_PIECES ] <- { 0 }, gameAttempt, cursorPiece, confirmAttempt, blackPiece, whitePiece, checkSecret [ GAME_PIECES ], checkAttempt [ GAME_PIECES ]: int

gameAttempt <- 1

ENQUANTO ( gameAttempt <= GAME_ATTEMPTS )

REPETIR

playBeep ( gameSound, MB_ICONWARNING )

SetConsoleTextAttribute ( hStdout, BOARD_COLOR | FOREGROUND_WHITE )

clearScreenBufferAt ( hStdout, BOARD_SIZE_Y + 4 )

printfXY ( hStdout, 1, BOARD_SIZE_Y + 4, "%d tent. restante", GAME_ATTEMPTS - gameAttempt + 1 );

SE GAME_ATTEMPTS – gameAttempt + 1 != 1 ENTÃO

        ESCREVE “s”

PARA ( i <- 0, i < GAME_PIECES, i <- i + 1 ) FAÇA

SetConsoleTextAttribute ( hStdout, BOARD_COLOR | PIECE_COLORS [ attemptPieces [ i ] ] )

SetConsoleCursorPosition ( hStdout, ( COORD ) { 1 + ( i + 1 ) * 2, 1 + gameAttempt } )

printSpecialChar ( 254 );

FIMPARA

blackPiece <- 0

 whitePiece <- 0

confirmAttempt <- 0

cursorPiece <- 0

PARA ( i <- 0, i < GAME_ATTEMPTS, i <- i + 1 ) FAÇA

checkSecret [ i ] <- 1

checkAttempt [ i ] <- 1

FIMPARA

SetConsoleCursorPosition ( hStdout, ( COORD ) { 1 + ( cursorPiece + 1 ) * 2, 1 + gameAttempt } )

REPETIR

        keyInput <- getch ( )
```

```
        SE keyInput == 0 || keyInput == 224 ENTÃO
                keyInput <- getch ( )
        SENÃO
                SE keyInput == KEY_SPACE ENTÃO
                        confirmAttempt <- 1
                CONTINUE
        ESCOLHA ( keyInput )
                CASO ( KEY_ARROW_UP ):
                        SE attemptPieces [ cursorPiece ] == GAME_COLORS – 1 ENTÃO
                                attemptPieces [ cursorPiece ] <- 0
                        SENÃO
                                attemptPieces [ cursorPiece ] <- attemptPieces [ cursorPiece ] + 1
SetConsoleTextAttribute ( hStdout, BOARD_COLOR | PIECE_COLORS [ attemptPieces [ cursorPiece ] ] )
printSpecialChar ( 254 )
SetConsoleCursorPosition ( hStdout, ( COORD ) { 1 + ( cursorPiece + 1 ) * 2, 1 + gameAttempt } )
                        PARE
                CASO ( KEY_ARROW_DOWN ):
                        SE attemptPieces [ cursorPiece ] == 0 ENTÃO
                                attemptPieces [ cursorPiece ] <- GAME_COLORS – 1
SENÃO
                                attemptPieces [ cursorPiece ] <- attemptPieces [ cursorPiece ] - 1
SetConsoleTextAttribute ( hStdout, BOARD_COLOR | PIECE_COLORS [ attemptPieces [ cursorPiece ] ] )
printSpecialChar ( 254 )
SetConsoleCursorPosition ( hStdout, ( COORD ) { 1 + ( cursorPiece + 1 ) * 2, 1 + gameAttempt } )
                        PARE
                CASO ( KEY_ARROW_LEFT ):
                        SE cursorPiece == 0 ENTÃO
                                cursorPiece <- GAME_PIECES – 1
                        SENÃO
                                cursorPiece <- cursorPiece – 1
                        SetConsoleCursorPosition ( hStdout, ( COORD ) { 1 + ( cursorPiece + 1 ) * 2, 1 +
gameAttempt } )
                        PARE
                CASO ( KEY_ARROW_RIGHT ):
                        SE cursorPiece == GAME_PIECES – 1 ENTÃO
```

```
                                        cursorPiece <- 0
                                SENÃO
                                        cursorPiece <- cursorPiece + 1
                                SetConsoleCursorPosition ( hStdout, ( COORD ) { 1 + ( cursorPiece + 1 ) * 2, 1 +
gameAttempt } )
                                PARE
ENQUANTO ( confirmAttempt == 0 )
PARA ( i <- 0, i < GAME_PIECES, i <- i + 1 ) FAÇA
        SE gameSecretKey [ i ] == attemptPieces [ i ] ENTÃO
SetConsoleCursorPosition ( hStdout, ( COORD ) { GAME_PIECES * 2 + 5 + blackPiece, 1 + gameAttempt } )
SetConsoleTextAttribute ( hStdout, BOARD_COLOR )
printSpecialChar ( 254 )
blackPiece <- blackPiece + 1
checkSecret [ i ] <- 0
checkAttempt [ i ] <- 0
FIMPARA
PARA ( i <- 0, i < GAME_PIECES, i <- i + 1 ) FAÇA
        PARA ( x <- 0, x < GAME_PIECES, x <- x + 1 ) FAÇA
                SE gameSecretKey [ i ] == attemptPieces [ x ] && checkSecret [ x ] && checkAttempt [ i ]
&& i != x ENTÃO
SetConsoleCursorPosition ( hStdout, ( COORD ) { GAME_PIECES * 2 + 5 + blackPiece + whitePiece, 1 +
gameAttempt } )
SetConsoleTextAttribute ( hStdout, BOARD_COLOR | FOREGROUND_WHITE )
printSpecialChar ( 254 )
whitePiece <- whitePiece + 1
checkSecret [ x ] <- 0
checkAttempt [ i ] <- 0
FIMPARA
FIMPARA
SE blackPiece == GAME_PIECES ENTÃO
playBeep ( gameSound, MB_OK )
displayWinScreen ( hStdout, screenRes, gameAttempt )
checkTopScore ( hStdout, screenRes, gameAttempt )
RETORNE
        gameAttempt <- gameAttempt + 1
playBeep ( gameSound, MB_ICONERROR )
```

```
PARA ( i = 0, i < GAME_PIECES, I <- i + 1 ) FAÇA
SetConsoleCursorPosition ( hStdout, ( COORD ) { 1 + ( i + 1 ) * 2, BOARD_SIZE_Y + 1 } )
SetConsoleTextAttribute ( hStdout, BOARD_COLOR | PIECE_COLORS [ gameSecretKey [ i ] ] )
printSpecialChar ( 254 );
FIMPARA
setWindowSize ( hStdout, screenRes.width, screenRes.height )
SetConsoleTextAttribute ( hStdout, BOARD_COLOR | FOREGROUND_WHITE )
clearScreenBufferAt ( hStdout, BOARD_SIZE_Y + 4 )
printfXY ( hStdout, 1, BOARD_SIZE_Y + 4, "%d tent. restantes", GAME_ATTEMPTS - gameAttempt + 1 )
printfXY ( hStdout, 1, BOARD_SIZE_Y + 6, "Pressione qualquer
        tecla para continuar!" )
SetConsoleCursorPosition ( hStdout, ( COORD ) { 1, BOARD_SIZE_Y + 6 } )
setWindowSize ( hStdout, csbi.dwSize.X + 3, csbi.dwSize.Y + 5 )
getch ( )
displayLossScreen ( hStdout, screenRes )
FIMFUNÇÃO
```

Código fonte

O código fonte da aplicação esta disponível no ficheiro compactado fornecido na entrega deste relatório.

As funções de maior relevância ou mais complexas foram comentadas, explicando o seu objetivo e em alguns casos o seu funcionamento, em determinadas situações apenas a primeira ocorrência da função foi comentada.

Alguns comentários encontram-se em Inglês devido ao trecho de código em questão não ter sido desenvolvido por nós, mesmo que tenhamos efetuado ligeiras alterações no mesmo.

Manual de Utilizador

Início

Ao abrir o ficheiro mastermind.exe irá aparecer uma janela, como pode ver na imagem abaixo, com algumas informações em relação ao projeto, para continuar para o menu principal do jogo basta a qualquer momento pressionar qualquer tecla.

Esta tela pode ser acedida futuramente através da opção do número 5 do menu.

Figura 1 – Tela de Boas-Vinda ou Sobre

Menu

No menu são apresentadas varias opções. Para escolher uma opção basta pressionar o número correspondente a opção desejada.

É possível ligar ou desligar o som no jogo escolhendo a opção 4.

Para terminar a execução da aplicação basta escolher a opção 0.

Figura 2 – Tela do Menu

Opção 1 - Jogo

Ao começar, o jogo irá preparar uma chave secreta, este processo poderá levar alguns segundos, após a criação da chave secreta é pedido que pressione qualquer tecla para começar o jogo.

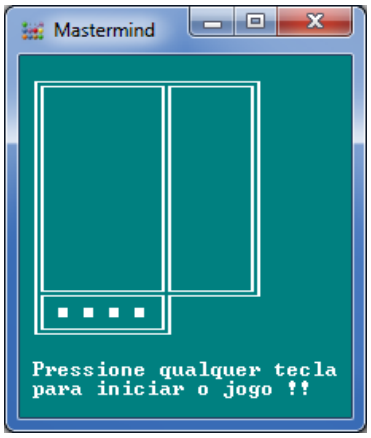

Figura 3 – Tela do Jogo 1

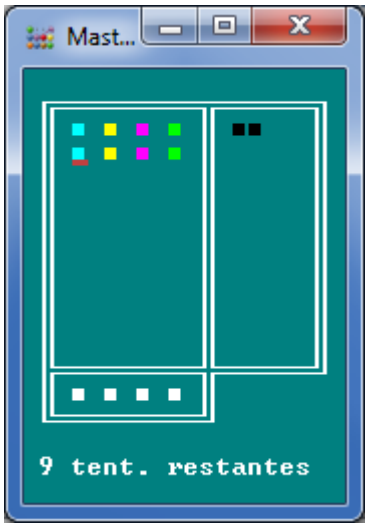

Figura 4 – Tela do Jogo 2

Para jogar usam-se as setas direcionais e o espaço para confirmar a jogada.

As setas para a esquerda e para a direita servem para mudar a seleção do bloco, é possível verificar o bloco selecionado olhando para o cursor na tela.

As setas para cima e para baixo servem para mudar a cor do bloco selecionado.

Após confirmar a jogada serão apresentados, no lado direito da tela, blocos pretos caso acerte em alguma cor na posição correta, ou blocos brancos caso a cor escolhida exista, mas esteja na posição errada. É possível verificar o numero de jogadas restantes na parte inferior da tela.

Caso consiga descobrir a combinação secreta antes de terminar o número de tentativas disponíveis será apresentada uma tela de vitória, com a indicação da quantidade de tentativas usadas.

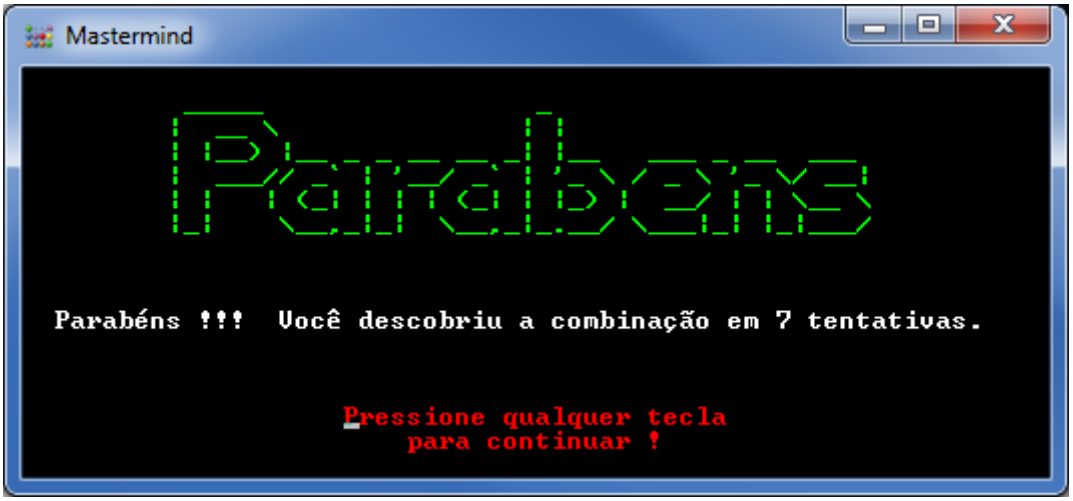

Figura 5 – Tela de Vitoria

No caso da sua pontuação estar entre os 10 melhores jogos, será apresentada uma tela onde poderá introduzir o seu nome para entrar para o ranking.

A pontuação é calculada pelo número de tentativas usadas para descobrir a combinação, ou seja, quanto menor o número de tentativas melhor a pontuação.

Caso não pretenda que o seu jogo seja adicionado ao ranking basta pressionar a tecla ENTER sem introduzir qualquer texto.

Nota: O tamanho máximo permitido para o nome é de 20 carateres.

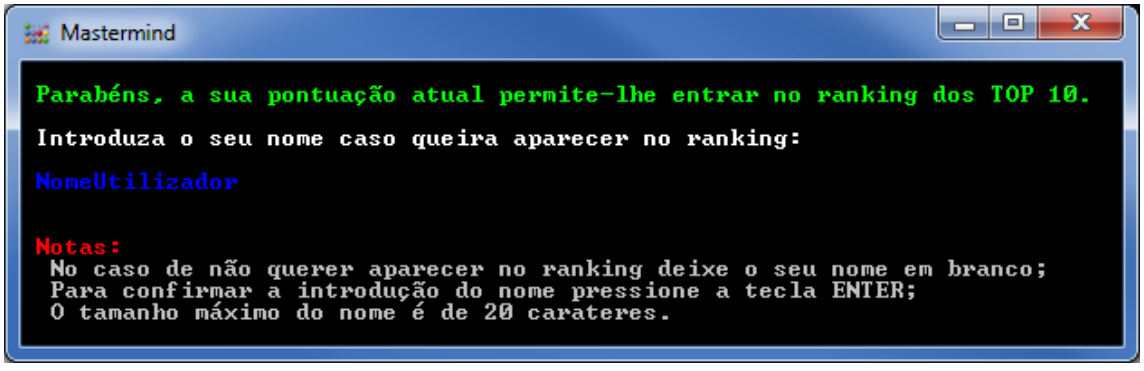

Figura 6 - Tela de Entrada no Ranking

Caso esgote o número de tentativas disponíveis e não consiga descobrir a combinação correta será informado que tem 0 tentativas restantes e na caixa inferior do tabuleiro é apresentada a solução pretendida.

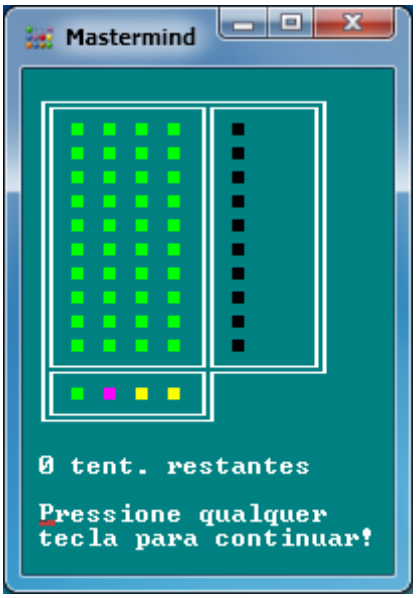

Figura 7 - Tele do Jogo 3

Ao pressionar qualquer tecla, como informado na tela, irá aparecer uma tela a informar que perdeu por ter esgotado o número de tentativas disponíveis.

Poderá voltar ao menu do jogo ao pressionar qualquer tecla.

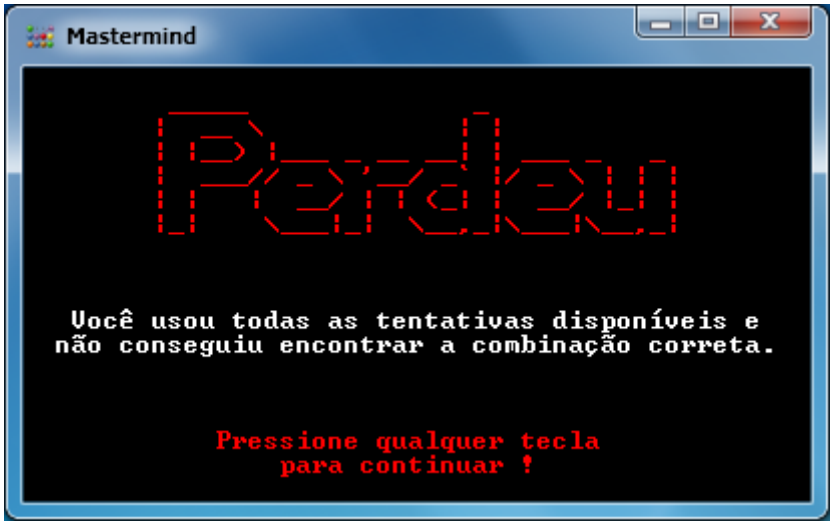

Figura 8 - Tela de Perdedor

Opção 2 - Ranking

Será apresentada a tela de ranking, onde irá mostrar as 10 melhores jogadas guardadas, informando a data e a hora a que o jogo foi realizado.

Caso não existam ainda jogadas guardadas será apresentada uma mensagem informativa com essa indicação.

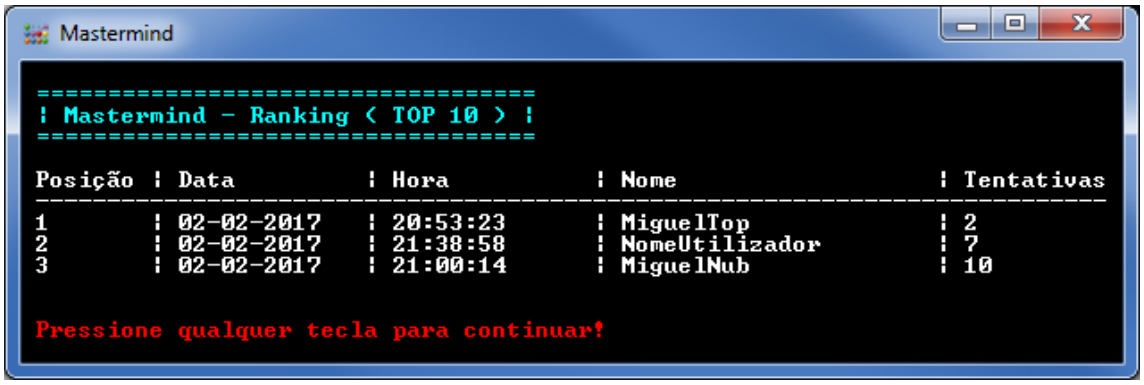

Figura 9 - Tela de Ranking

Opção 3 - Instruções

Será apresentada uma tela com as instruções principais do jogo, fornecendo assim uma rápida introdução da jogabilidade ao jogador.

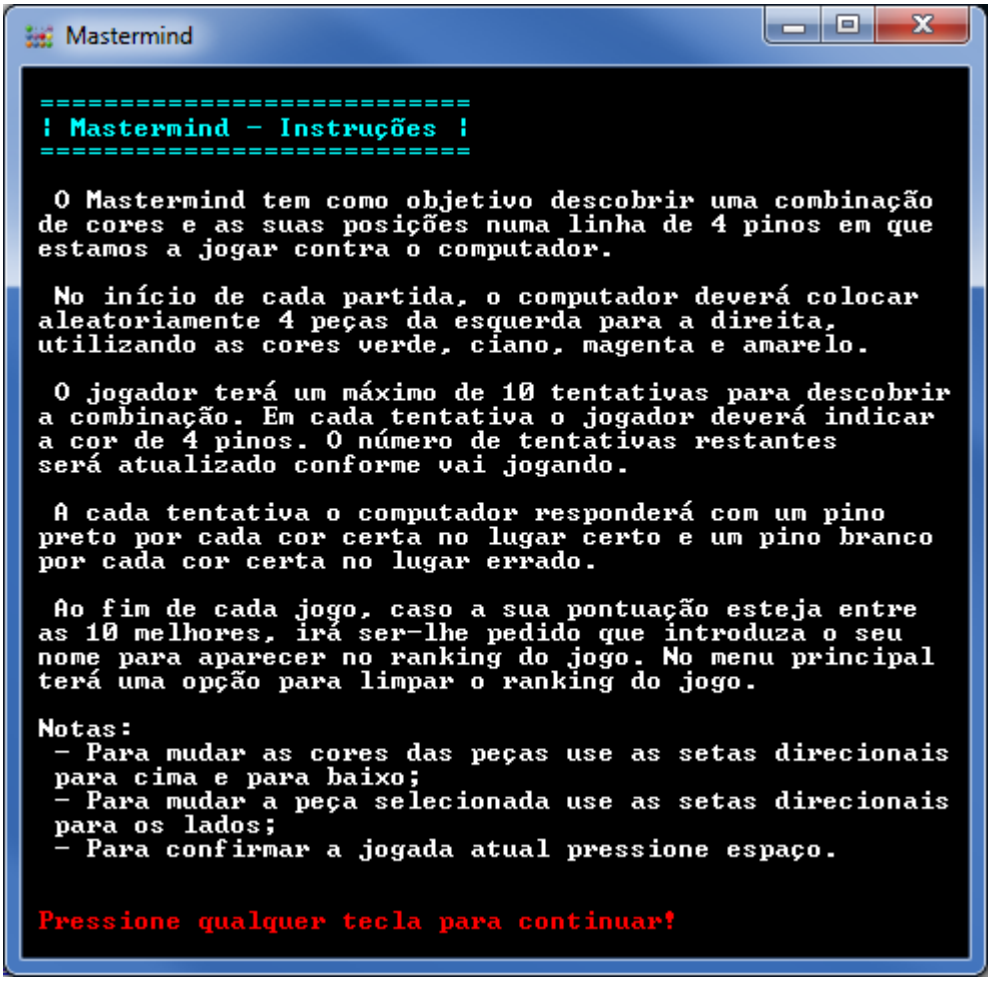

Figura 10 - Tela de Instruções