Jogo de Xadrez

Gerado por Doxygen 1.8.13

Sumário

Capítulo 1

jogo_de_xadrez

Um jogo de xadrez escrito em C.

Instruções:

Primeiramente, abre-se a pasta src, que está dentro do diretório jogo_de_xadrez, no terminal. Em seguida, compila-se o programa escrevendo make. Para iniciar a execução do programa, digita-se no terminal, dentro da pasta src, a seguinte instrução: ./main.

Em seguida, Aparecerá uma interface com 4 opções, uma para iniciar um jogo com um tabuleiro na posição de peças padrão do xadrez, a segunda que permite recuperar um jogo salvo por meio de um arquivo contendo um tabuleiro salvo, a terceira que permite criar um tabuleiro do zero e a última que permite sair do jogo.

Ao selecionar a primeira opção e a segunda, abrirá outra janela de opções que dispõem de 3 opções, a primeira em que se joga jogadorXjogador, a segunda que é jogadorxComputador e a opção de sair do jogo.

Ao selecionar a terceira opção, primeiro abre-se uma interface para que se crie a sua própria disposição em um tabuleiro de xadrez, usando a linha de comando própria da interface para adicionar as peças, começando pelos reis, e clicando na posição desejada. Ao finalizar esse processo, aparecem as 2 opções de modo de jogo.

Como jogar:

. Modo Jogador x Jogador:

Aparece a interface com o tabuleiro selecionado, começando pelas peças brancas. O jogador deve escrever na linha de comando usando o padrão pré estabelecido para determinar a peça em sua posição inicial e a posição final do movimento. Em seguida, é a vez das peça pretas jogarem, seguindo o mesmo procedimento. O jogo termina ao se chegar em uma condição de checkmate ou empate.

. Modo Jogador x Computador:

Aparece a interface com o tabuleiro selecionado, começando pelas peças brancas. O jogador recebe escrito naa tela as melhores jogadas possíveis para o momento, podendo escolhê-las ou não. Em seguida, a maquina joga baseada em sua IA para tentar ganhar do jogador. O jogo termina quando se chega em uma posição de checkmate ou empate.

Capítulo 2

Índice das Estruturas de Dados

2.1 Estruturas de Dados

Aqui estão as estruturas de dados, uniões e suas respectivas descrições:

board . . . . . **??**
list_past_move . . . . . **??**
ListNode
Estrutura do nó da lista que armazena as jogadas passadas . . . . . **??**
ListOfMoves
Estrutura representará todas as possíveis jogadas de um tabuleiro . . . . . **??**
ListPastMoves
Estrutura da lista que armazena as jogadas passadas . . . . . **??**
Move
Estrutura que irá caracterizar a jogada . . . . . **??**
node_list . . . . . **??**
NodeList . . . . . **??**
NodeTree
Estrutura de um nó da árvore . . . . . **??**
TBoard
Estrutura do tabuleiro . . . . . **??**
Tree
Estrutura da árvore . . . . . **??**

Capítulo 3

Índice dos Arquivos

3.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

arv_decisoes.c . . . . . **??**
arv_decisoes.h . . . . . **??**
ia.c . . . . . **??**
ia.h . . . . . **??**
in-out.c . . . . . **??**
in-out.h . . . . . **??**
interface.c . . . . . **??**
interface.h . . . . . **??**
list_of_moves.c . . . . . **??**
list_of_moves.h . . . . . **??**
logica.c . . . . . **??**
logica.h . . . . . **??**
main.c . . . . . **??**
tabuleiro.c . . . . . **??**
tabuleiro.h . . . . . **??**
TEST_arv_decisoes.c . . . . . **??**
TEST_ia.c . . . . . **??**
TEST_in-out.c . . . . . **??**
TEST_list.c . . . . . **??**
TEST_logica.c . . . . . **??**
TEST_tabuleiro.c . . . . . **??**

Capítulo 4

Estruturas

4.1 Referência da Estrutura board

`#include <tabuleiro.h>`

Campos de Dados

- char **Board** [8][8]
- int **Weight**
- int **WhiteCheck**
- int **BlackCheck**

4.1.1 Campos

4.1.1.1 BlackCheck

`int BlackCheck`

Inteiro que armazena a condição de xeque do rei preto.

4.1.1.2 Board

`char Board[8][8]`

Matriz de caracteres que representa o tabuleiro.

4.1.1.3 Weight

`int Weight`

Inteiro que armazena o peso ponderado do tabuleiro.

4.1.1.4 WhiteCheck

```
int WhiteCheck
```

Inteiro que armazena a condição de xeque do rei branco.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **tabuleiro.h**

4.2 Referência da Estrutura list_past_move

```
#include <in-out.h>
```

Campos de Dados

- **ListNode** * **head**
- **ListNode** * **last**

4.2.1 Campos

4.2.1.1 head

```
ListNode* head
```

Ponteiro para a cabeça da lista.

4.2.1.2 last

```
ListNode* last
```

Ponteiro para o último elemento da lista.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **in-out.h**

4.3 Referência da Estrutura ListNode

Estrutura do nó da lista que armazena as jogadas passadas.

```
#include <in-out.h>
```

4.3.1 Descrição Detalhada

Estrutura do nó da lista que armazena as jogadas passadas.

Essa estrutura armazena os elemnetos ncessários para recuperar uma jogada feita anteriormente.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **in-out.h**

4.4 Referência da Estrutura ListOfMoves

Estrutura representará todas as possíveis jogadas de um tabuleiro.

`#include <list_of_moves.h>`

Campos de Dados

- int **howmany**
- **NodeList** * **first**
- **NodeList** * **current**
- **NodeList** * **last**

4.4.1 Descrição Detalhada

Estrutura representará todas as possíveis jogadas de um tabuleiro.

Lista do tipo **Move** (pag. **??**) que é importante para a análise das jogadas.

4.4.2 Campos

4.4.2.1 current

```
NodeList* current
```

4.4.2.2 first

```
NodeList* first
```

4.4.2.3 howmany

```
int howmany
```

Inteiro que guarda quantas jogadas são possíveis de serem feitas.

4.4.2.4 last

```
NodeList* last
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **list_of_moves.h**

4.5 Referência da Estrutura ListPastMoves

Estrutura da lista que armazena as jogadas passadas.

```
#include <in-out.h>
```

4.5.1 Descrição Detalhada

Estrutura da lista que armazena as jogadas passadas.

Essa estrutura serva para recuperação de jogadas passadas e salvamentos de jogos em PGN.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **in-out.h**

4.6 Referência da Estrutura Move

Estrutura que irá caracterizar a jogada.

```
#include <list_of_moves.h>
```

Campos de Dados

- int **origin** [2]
- int **destiny** [2]

4.6.1 Descrição Detalhada

Estrutura que irá caracterizar a jogada.

Estrutura que armazena uma maneira de interpretar uma jogada.

4.6.2 Campos

4.6.2.1 destiny

```
int destiny[2]
```

Array que armazena as coordenadas de destino do movimento.

4.6.2.2 origin

```
int origin[2]
```

Array que armazena as coordenadas de origem do movimento.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **list_of_moves.h**

4.7 Referência da Estrutura node_list

```
#include <in-out.h>
```

Campos de Dados

- char **move** [7]
- struct **node_list** * **next**

4.7.1 Campos

4.7.1.1 move

```
char move[7]
```

String com a jogada feita na notação de xadrez.

4.7.1.2 next

```
struct node_list* next
```

Ponteiro para o próximo elemento da lista.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **in-out.h**

4.8 Referência da Estrutura NodeList

`#include <list_of_moves.h>`

Campos de Dados

- **Move play**
- struct **NodeList** * **next**

4.8.1 Campos

4.8.1.1 next

`struct NodeList* next`

4.8.1.2 play

`Move play`

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **list_of_moves.h**

4.9 Referência da Estrutura NodeTree

Estrutura de um nó da árvore.

`#include <arv_decisoes.h>`

Campos de Dados

- **TBoard** * **board**
- **Move** * **play**
- int **n_child**
- **NodeTree** ** **child**

4.9.1 Descrição Detalhada

Estrutura de um nó da árvore.

Estrutura que armazena os componentes básicos de um nó da árvore de decisões

4.9.2 Campos

4.9.2.1 board

```
TBoard* board
```

Ponteiro para uma variável do tipo Tboard que representa a organização do tabuleiro depois de alguma jogada

4.9.2.2 child

```
NodeTree** child
```

Vetor de ponteiros para os filhos do nó

4.9.2.3 n_child

```
int n_child
```

Inteiro que armazena o número de filhos que o nó da árvore tem

4.9.2.4 play

```
Move* play
```

Ponteiro para uma variável do tipo **Move** (pag. **??**) que representa a jogada que originou a configuração do tabuleiro

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **arv_decisoes.h**

4.10 Referência da Estrutura TBoard

Estrutura do tabuleiro.

```
#include <tabuleiro.h>
```

4.10.1 Descrição Detalhada

Estrutura do tabuleiro.

Estrutura que armazena os elementos necessário para definir um tabuleiro.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **tabuleiro.h**

4.11 Referência da Estrutura Tree

Estrutura da árvore.

`#include <arv_decisoes.h>`

Campos de Dados

- **NodeTree** * **root**

4.11.1 Descrição Detalhada

Estrutura da árvore.

Estrutura que armazena uma referência para a raíz da árvore de decisões

4.11.2 Campos

4.11.2.1 root

NodeTree* root

Ponteiro para uma variável do tipo ponteiro que representa a raiz da árvore

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- **arv_decisoes.h**

Capítulo 5

Arquivos

5.1 Referência do Arquivo arv_decisoes.c

```
#include "../include/arv_decisoes.h"
#include <stdio.h>
#include <stdlib.h>
```

Funções

- **Tree** ∗ **AlocateTree** (void)

  Aloca espaço em memória para uma árvore.
- **NodeTree** ∗ **AlocateNodeTree** (int n_child, **TBoard** ∗ **board**, **Move** ∗play)
- int **AddChildNode** (**NodeTree** ∗father, **NodeTree** ∗child, int position)

  Insere um nó como filho de outro nó
- **NodeTree** ∗ **FreeTreeNodes** (**NodeTree** ∗node)

5.1.1 Funções

5.1.1.1 AddChildNode()

```
int AddChildNode (
              NodeTree * father,
              NodeTree * child,
             int position )
```

Insere um nó como filho de outro nó

Parâmetros

father	Ponteiro para o tipo **NodeTree** (pag. **??**) que armazena o nó pai
child	Ponteiro para o tipo **NodeTree** (pag. **??**) que armazena o novo filho de "father"
posititon	Inteiro que armazena qual é o número do novo filho no vetor de filhos

Retorna

Um inteiro indicando 0 será a inserção foi um fracasso ou 1 se foi um sucesso

5.1.1.2 AlocateNodeTree()

```
NodeTree* AlocateNodeTree (
            int n_child,
             TBoard * board,
             Move * play )
```

5.1.1.3 AlocateTree()

```
Tree* AlocateTree (
            void  )
```

Aloca espaço em memória para uma árvore.

Retorna

Uma árvore inicializada e diferente de nulo

5.1.1.4 FreeTreeNodes()

```
NodeTree* FreeTreeNodes (
             NodeTree * node )
```

5.2 Referência do Arquivo arv_decisoes.h

```
#include "../include/tabuleiro.h"
#include "../include/logica.h"
```

Estruturas de Dados

- struct **NodeTree**

  Estrutura de um nó da árvore.
- struct **Tree**

  Estrutura da árvore.

Definições de Tipos

- typedef struct **NodeTree** **NodeTree**
- typedef struct **Tree** **Tree**

Funções

- **Tree** ∗ **AlocateTree** (void)

  Aloca espaço em memória para uma árvore.
- **NodeTree** ∗ **AlocateNodeTree** (int n_child, **TBoard** ∗ **board**, **Move** ∗play)
- int **AddChildNode** (**NodeTree** ∗father, **NodeTree** ∗child, int position)

  Insere um nó como filho de outro nó
- **NodeTree** ∗ **FreeTreeNodes** (**NodeTree** ∗node)

5.2.1 Definições dos tipos

5.2.1.1 NodeTree

```
typedef struct  NodeTree  NodeTree
```

5.2.1.2 Tree

```
typedef struct  Tree  Tree
```

5.2.2 Funções

5.2.2.1 AddChildNode()

```
int AddChildNode (
              NodeTree * father,
              NodeTree * child,
             int position )
```

Insere um nó como filho de outro nó

Parâmetros

father	Ponteiro para o tipo **NodeTree** (pag. **??**) que armazena o nó pai
child	Ponteiro para o tipo **NodeTree** (pag. **??**) que armazena o novo filho de "father"
posititon	Inteiro que armazena qual é o número do novo filho no vetor de filhos

Retorna

Um inteiro indicando 0 será a inserção foi um fracasso ou 1 se foi um sucesso

5.2.2.2 AlocateNodeTree()

```
NodeTree* AlocateNodeTree (
            int n_child,
             TBoard * board,
             Move * play )
```

5.2.2.3 AlocateTree()

```
Tree * AlocateTree (
            void  )
```

Aloca espaço em memória para uma árvore.

Retorna

Uma árvore inicializada e diferente de nulo

5.2.2.4 FreeTreeNodes()

```
NodeTree* FreeTreeNodes (
             NodeTree * node )
```

5.3 Referência do Arquivo ia.c

```
#include "../include/ia.h"
```

Funções

- **Tree** * **CreateMovesTree** (**TBoard** * **board**, int turn)
- int **SortTree** (**Tree** *tree, int turn)

  Ordena a árvore para achar a melhor jogada.
- **ListOfMoves** * **Best_Plays** (**Tree** *tree, int n_child)

  Extrai as melhores jogadas.

5.3.1 Funções

5.3.1.1 Best_Plays()

```
ListOfMoves* Best_Plays (
             Tree * tree,
            int n_child )
```

Extrai as melhores jogadas.

Parâmetros

tree	Ponteiro para a árvore que se deseja extrair as jogadas
turn	Inteiro contendo o número de filhos da raiz da árvore

Retorna

Um ponteiro para uma lista de movimentos

5.3.1.2 CreateMovesTree()

```
Tree* CreateMovesTree (
              TBoard * board,
             int turn )
```

5.3.1.3 SortTree()

```
int SortTree (
              Tree * tree,
             int turn )
```

Ordena a árvore para achar a melhor jogada.

Parâmetros

board	Ponteiro para o tabuleiro o qual se deseja encontrar as melhores possíveis jogadas
turn	Inteiro contendo a informação de quem é jogada(Black = 0, White = 1)

Retorna

Inteiro indicando sucesso(0) ou fracasso(1) da operação

5.4 Referência do Arquivo ia.h

```
#include <stdio.h>
#include <stdlib.h>
#include "../include/logica.h"
#include "../include/arv_decisoes.h"
```

Funções

- **Tree** * **CreateMovesTree** (**TBoard** * **board**, int turn)

- int **SortTree** (**Tree** ∗tree, int turn)

  Ordena a árvore para achar a melhor jogada.
- **ListOfMoves** ∗ **Best_Plays** (**Tree** ∗tree, int n_child)

  Extrai as melhores jogadas.

Variáveis

- const int **BLACKS_TURN** = 0
- const int **WHITES_TURN** = 1
- const int **CHECK_WEIGHT** = 50

5.4.1 Funções

5.4.1.1 Best_Plays()

```
ListOfMoves * Best_Plays (
              Tree * tree,
             int n_child )
```

Extrai as melhores jogadas.

Parâmetros

tree	Ponteiro para a árvore que se deseja extrair as jogadas
turn	Inteiro contendo o número de filhos da raiz da árvore

Retorna

Um ponteiro para uma lista de movimentos

5.4.1.2 CreateMovesTree()

```
Tree* CreateMovesTree (
              TBoard * board,
             int turn )
```

5.4.1.3 SortTree()

```
int SortTree (
              Tree * tree,
             int turn )
```

Ordena a árvore para achar a melhor jogada.

Parâmetros

board	Ponteiro para o tabuleiro o qual se deseja encontrar as melhores possíveis jogadas
turn	Inteiro contendo a informação de quem é jogada(Black = 0, White = 1)

Retorna

Inteiro indicando sucesso(0) ou fracasso(1) da operação

5.4.2 Variáveis

5.4.2.1 BLACKS_TURN

```
const int BLACKS_TURN = 0
```

5.4.2.2 CHECK_WEIGHT

```
const int CHECK_WEIGHT = 50
```

5.4.2.3 WHITES_TURN

```
const int WHITES_TURN = 1
```

5.5 Referência do Arquivo in-out.c

```
#include "../include/in-out.h"
```

Funções

- **ListPastMoves** ∗ **StartListPM** (void)
  Aloca o espaço para a lista de movimentos passados e sua cabeça.
- int **AddListPM** (**ListPastMoves** ∗list, char move[ ])
  Adiciona um movimento na lista de movimentos passados.
- int **RemoveLastListPM** (**ListPastMoves** ∗list)
  Remove o último elemento da lista.
- int **FreeListPM** (**ListPastMoves** ∗list)
  Libera a lista, desalocando seus elementos.
- int **SaveBoardFile** (**TBoard** ∗ **board**, char ∗file_name)
- int **RecoverBoardFromFile** (**TBoard** ∗ **board**, char ∗file_name)
- int **SavePGNFile** (**ListPastMoves** ∗listmoves, char ∗file_name)
  Salva o tabuleiro em um arquivo.
- int **RecoverMoveListFromFile** (**ListPastMoves** ∗listmoves, char ∗file_name)
  Recupera a lista de movimentos armazenada em um arquivo.

5.5.1 Funções

5.5.1.1 AddListPM()

```
int AddListPM (
              ListPastMoves * list,
             char move[] )
```

Adiciona um movimento na lista de movimentos passados.

Parâmetros

list	Ponteiro para a lista.
move	String com o movimeto na notação de xadrez.

Retorna

Por parâmetro, retorna a lista com o novo elemento e um inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.5.1.2 FreeListPM()

```
int FreeListPM (
              ListPastMoves * list )
```

Libera a lista, desalocando seus elementos.

Parâmetros

list	Ponteiro para a lista.

Retorna

Inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.5.1.3 RecoverBoardFromFile()

```
int RecoverBoardFromFile (
              TBoard * board,
             char * file_name )
```

5.5.1.4 RecoverMoveListFromFile()

```
int RecoverMoveListFromFile (
              ListPastMoves * listmoves,
             char * file_name )
```

Recupera a lista de movimentos armazenada em um arquivo.

Parâmetros

listmoves	Lista de movimentos que será recuperada
file_name	Nome do arquivo contendo o tabuleiro salvo

Retorna

Inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.5.1.5 RemoveLastListPM()

```
int RemoveLastListPM (
              ListPastMoves * list )
```

Remove o último elemento da lista.

Parâmetros

list	Ponteiro para a lista.

Retorna

Por parâmetro, retorna a lista sem o último elemento e um inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.5.1.6 SaveBoardFile()

```
int SaveBoardFile (
              TBoard * board,
             char * file_name )
```

5.5.1.7 SavePGNFile()

```
int SavePGNFile (
              ListPastMoves * listmoves,
             char * file_name )
```

Salva o tabuleiro em um arquivo.

Parâmetros

listmoves	Ponteiro para uma lista de movimentos
file_name	Nome do arquivo

Retorna

Inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.5.1.8 StartListPM()

```
ListPastMoves* StartListPM (
            void  )
```

Aloca o espaço para a lista de movimentos passados e sua cabeça.

Retorna

Lista com espaço e sua cabeca alocados e o elemento seguinte da cabeça como nulo.

5.6 Referência do Arquivo in-out.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../include/tabuleiro.h"
```

Estruturas de Dados

- struct **node_list**
- struct **list_past_move**

Definições de Tipos

- typedef struct **node_list** **ListNode**
- typedef struct **list_past_move** **ListPastMoves**

Funções

- **ListPastMoves** ∗ **StartListPM** (void)

  Aloca o espaço para a lista de movimentos passados e sua cabeça.
- int **AddListPM** (**ListPastMoves** ∗list, char move[ ])

  Adiciona um movimento na lista de movimentos passados.
- int **RemoveLastListPM** (**ListPastMoves** ∗list)

  Remove o último elemento da lista.
- int **FreeListPM** (**ListPastMoves** ∗list)

  Libera a lista, desalocando seus elementos.
- int **SaveBoardFile** (**TBoard** ∗ **board**, char ∗file_name)
- int **RecoverBoardFromFile** (**TBoard** ∗ **board**, char ∗file_name)
- int **SavePGNFile** (**ListPastMoves** ∗listmoves, char ∗file_name)

  Salva o tabuleiro em um arquivo.
- int **RecoverMoveListFromFile** (**ListPastMoves** ∗listmoves, char ∗file_name)

  Recupera a lista de movimentos armazenada em um arquivo.

5.6.1 Definições dos tipos

5.6.1.1 ListNode

```
typedef struct  node_list  ListNode
```

5.6.1.2 ListPastMoves

```
typedef struct  list_past_move  ListPastMoves
```

5.6.2 Funções

5.6.2.1 AddListPM()

```
int AddListPM (
              ListPastMoves * list,
             char move[] )
```

Adiciona um movimento na lista de movimentos passados.

Parâmetros

list	Ponteiro para a lista.
move	String com o movimeto na notação de xadrez.

Retorna

Por parâmetro, retorna a lista com o novo elemento e um inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.6.2.2 FreeListPM()

```
int FreeListPM (
              ListPastMoves * list )
```

Libera a lista, desalocando seus elementos.

Parâmetros

list	Ponteiro para a lista.

Retorna

Inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.6.2.3 RecoverBoardFromFile()

```
int RecoverBoardFromFile (
              TBoard * board,
             char * file_name )
```

5.6.2.4 RecoverMoveListFromFile()

```
int RecoverMoveListFromFile (
              ListPastMoves * listmoves,
             char * file_name )
```

Recupera a lista de movimentos armazenada em um arquivo.

Parâmetros

listmoves	Lista de movimentos que será recuperada
file_name	Nome do arquivo contendo o tabuleiro salvo

Retorna

Inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.6.2.5 RemoveLastListPM()

```
int RemoveLastListPM (
              ListPastMoves * list )
```

Remove o último elemento da lista.

Parâmetros

list	Ponteiro para a lista.

Retorna

Por parâmetro, retorna a lista sem o último elemento e um inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.6.2.6 SaveBoardFile()

```
int SaveBoardFile (
              TBoard * board,
             char * file_name )
```

5.6.2.7 SavePGNFile()

```
int SavePGNFile (
              ListPastMoves * listmoves,
             char * file_name )
```

Salva o tabuleiro em um arquivo.

Parâmetros

listmoves	Ponteiro para uma lista de movimentos
file_name	Nome do arquivo

Retorna

Inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.6.2.8 StartListPM()

```
ListPastMoves * StartListPM (
             void  )
```

Aloca o espaço para a lista de movimentos passados e sua cabeça.

Retorna

Lista com espaço e sua cabeca alocados e o elemento seguinte da cabeça como nulo.

5.7 Referência do Arquivo interface.c

```
#include "../include/interface.h"
```

Funções

- WINDOW * **MakeBoardWin** (void)
  Cria janela do tabuleiro.
- WINDOW * **MakeYaxisWin** (void)
  Cria janela do eixo Y.
- WINDOW * **MakeXaxisWin** (void)
  Cria janela do eixo X.
- WINDOW * **MakeKeyWin** (void)
  Cria a janela do menu de atalhos.
- WINDOW * **MakeMsgWin** (void)
  Cria janela de mensagens.
- WINDOW * **MakeHelpWin** (void)
  Cria a janela de ajuda.
- void **TranslateCoord** (int yscreen, int xscreen, int *yboard, int *xboard)
  Traduz as coordenadas da tela para as do tabuleiro.
- int **verify_turn** (**TBoard** * **board**, **Move** *movement, int turn)
  Verifica se o jogador vai mexer a peça correta.
- int **change_turn** (int turn)
  Muda a vez da jogada.
- void **InitBoard** (WINDOW *boardwin, **TBoard** * **board**)
  Inicializa graficamente um tabuleiro na base inicial.
- void **DrawBoard** (WINDOW *boardwin)
  Desenha um tabuleiro vazio no terminal.
- void **DrawAxis** (WINDOW *yaxis, WINDOW *xaxis)
  Desenha os eixos coordenados que servem para indicar ajudar a identificar a posição de uma peça.
- void **GiveHint** (WINDOW *helpwin, **TBoard** * **board**, int turn)
  Mostra as dicas na tela do jogador.
- void **HelpWinPVP** (WINDOW *helpwin)
  Mostra legenda das peças na tela de ajuda no PVP.
- void **HelpWinPVE** (WINDOW *helpwin)
  Inicializa a janela de ajuda do PVE com o básico.
- void **HelpWinNewBoard** (WINDOW *helpwin)
  Inicializa a janela de ajuda para o menu de criação.
- **TBoard** * **CreateNewBoard** (void)
  Função de criar um novo tabuleiro.
- int **CreateMenu** (WINDOW *menuwin)
  Desenha o menu de escolha do tipo de jogo e faz o usuário escolher entre uma das opções disponíveis.
- **TBoard** * **MenuGetBoard** (void)
  Abre o menu de escolha do tabuleiro.

- void **write_keys_help** (WINDOW ∗keywin, int wintype)
- void **clear_keywin** (WINDOW ∗keywin)

  Limpa a área de comandos da janela de ajuda com atalhos.
- void **init_msg_win** (WINDOW ∗messages)

  Inicializa a janela de mensagens.
- void **clear_message** (WINDOW ∗messages)

  Limpa janela de mensagens.
- void **print_message** (WINDOW ∗messages, int msg)

  Imprime mensagens na janela de mensagens.
- int **reverse_color_in_board** (WINDOW ∗boardwin, **TBoard** ∗ **board**, int line, int column)

  Destaca na interface a posição dada no tabuleiro.
- **Move** ∗ **GetMovement** (WINDOW ∗keywin, char chess_move[ ])

  Obtém do usuário um movimento através da notação.
- void **print_winner** (WINDOW ∗helpwin, int who)

  Mostra na tela quem é o jogador vencedor da partida.
- void **print_turn** (WINDOW ∗helpwin, int turn)

  Mostra de quem é a vez de jogar na partida.
- int **wanna_save** (WINDOW ∗messages)

  Pergunta ao usuário se ele quer salvar o jogo.
- int **verify_evolve_pawn** (WINDOW ∗messages, **TBoard** ∗ **board**)

  Verifica se existe algum peão para ser promovido.
- int **UI_MOVE_PIECE** (WINDOW ∗boardwin, WINDOW ∗messages, **TBoard** ∗ **board**, int turn, **Move** ∗movement)

  ***Move** (*pag. **??**) *a peça e faz todas as verificações pela própria interface.*
- int **UI_MOUSE_MOVE** (WINDOW ∗boardwin, WINDOW ∗messages, **TBoard** ∗ **board**, int turn, MEVENT event, **ListPastMoves** ∗pastmoves)

  ***Move** (*pag. **??**) *a peça através do mouse do usuário.*
- void **play_pvp** (WINDOW ∗boardwin, WINDOW ∗keywin, WINDOW ∗messages, **TBoard** ∗ **board**)

  Função responsável por todo o modo de jogo de humano vs humano.
- void **play_pve** (WINDOW ∗boardwin, WINDOW ∗keywin, WINDOW ∗messages, **TBoard** ∗ **board**)

  Faz todo o processo do modo de jogo Jogador vs Computador.

5.7.1 Funções

5.7.1.1 change_turn()

```
int change_turn (
            int turn )
```

Muda a vez da jogada.

Ela vai mudar o turno da jogada, por exemplo se foi as pretas que jogaram por último ela muda o turno para as brancas

Parâmetros

turn	Indica de quem era a última jogada

Retorna

new_turn Indica de quem é a nova vez de jogar

5.7.1.2 clear_keywin()

```
void clear_keywin (
            WINDOW * keywin )
```

Limpa a área de comandos da janela de ajuda com atalhos.

A área de comando desta janela é onde o usuário irá digitar o movimento desejado, que é logo abaixo das informações de atalhos

Parâmetros

keywin	Janela de ajuda com atalhos

5.7.1.3 clear_message()

```
void clear_message (
            WINDOW * messages )
```

Limpa janela de mensagens.

Ela irá limpar a janela de mensagens mas não limpará o efeito box

Parâmetros

messages	Janela de mensagens

5.7.1.4 CreateMenu()

```
int CreateMenu (
            WINDOW * menuwin )
```

Desenha o menu de escolha do tipo de jogo e faz o usuário escolher entre uma das opções disponíveis.

Parâmetros

menuwin	Janela onde será mostrado o menu

Retorna

1 Se o usuário escolher "Jogador VS Jogador" 2 Se ele escolher "Jogador VS Máquina" 3 Se escolher Sair

5.7.1.5 CreateNewBoard()

```
TBoard* CreateNewBoard (
            void  )
```

Função de criar um novo tabuleiro.

Esta função irá abrir um novo ambiente de interação com o usuário onde ele poderá criar um tabuleiro de sua preferência.

Retorna

board Tabuleiro criado pelo usuário

5.7.1.6 DrawAxis()

```
void DrawAxis (
            WINDOW * yaxis,
            WINDOW * xaxis )
```

Desenha os eixos coordenados que servem para indicar ajudar a identificar a posição de uma peça.

Parâmetros

yaxis	Eixo das coordenadas dadas por números
xaxis	Eixo das coordenadas dadas pelas letras

Retorna

Sem retorno

5.7.1.7 DrawBoard()

```
void DrawBoard (
            WINDOW * boardwin )
```

Desenha um tabuleiro vazio no terminal.

Parâmetros

boardwin	Janela onde será colocado o tabuleiro

Retorna

Não há retorno, ela apenas desenha o quadro na janela dada

5.7.1.8 GetMovement()

```
Move* GetMovement (
            WINDOW * keywin,
            char chess_move[] )
```

Obtém do usuário um movimento através da notação.

Parâmetros

keywin	Janela onde está os atalhos
chess_move	String onde será colocada a jodada do jogador

Retorna

movement Movimento traduzido do jogador para o tipo Move*

5.7.1.9 GiveHint()

```
void GiveHint (
            WINDOW * helpwin,
             TBoard * board,
            int turn )
```

Mostra as dicas na tela do jogador.

Parâmetros

helpwin	Janela de ajuda para o usuário
board	Tabuleiro que será considerado para as dicas
turn	Indica quem é o dono do turno e que vai receber as dicas

5.7.1.10 HelpWinNewBoard()

```
void HelpWinNewBoard (
            WINDOW * helpwin )
```

Inicializa a janela de ajuda para o menu de criação.

Observe que esta função é chamada apenas para o menu de criação de uma nova imagem. Esta função irá colocar na janela de ajuda alguns resumos sobre as peças que o usuário poderá colocar e a explicação de como colocar uma nova peça. Não possui retorno

Parâmetros

helpwin	Janela de ajuda

5.7.1.11 HelpWinPVE()

```
void HelpWinPVE (
            WINDOW * helpwin )
```

Inicializa a janela de ajuda do PVE com o básico.

Parâmetros

helpwin	Janela de ajuda

5.7.1.12 HelpWinPVP()

```
void HelpWinPVP (
            WINDOW * helpwin )
```

Mostra legenda das peças na tela de ajuda no PVP.

Parâmetros

helpwin	Janela de ajuda onde seram colocadas as legendas

5.7.1.13 init_msg_win()

```
void init_msg_win (
            WINDOW * messages )
```

Inicializa a janela de mensagens.

Parâmetros

messages	Janela de mensagens já alocada

5.7.1.14 InitBoard()

```
void InitBoard (
             WINDOW * boardwin,
              TBoard * board )
```

Inicializa graficamente um tabuleiro na base inicial.

Parâmetros

boardwin	Janela onde está o tabuleiro
board	Tabuleiro que será incializado na parte gráfica

Retorna

Não tem retorno

5.7.1.15 MakeBoardWin()

```
WINDOW* MakeBoardWin (
             void  )
```

Cria janela do tabuleiro.

Retorna

boardwin Ponteiro para a janela do tabuleiro

5.7.1.16 MakeHelpWin()

```
WINDOW* MakeHelpWin (
             void  )
```

Cria a janela de ajuda.

Retorna

helpwin Janela da ajuda

5.7.1.17 MakeKeyWin()

```
WINDOW* MakeKeyWin (
            void  )
```

Cria a janela do menu de atalhos.

Este menu de atalhos é o que mostra as opções de atalhos no teclado para o usuário, como a tecla 'j' no menu do PVP que é usada para fazer jogadas usando a notação

Retorna

keywin Janela do menu de atalhos

5.7.1.18 MakeMsgWin()

```
WINDOW* MakeMsgWin (
            void  )
```

Cria janela de mensagens.

Retorna

messages Janela de mensagens

5.7.1.19 MakeXaxisWin()

```
WINDOW* MakeXaxisWin (
            void  )
```

Cria janela do eixo X.

Retorna

xaxis Ponteiro para a janela do eixo X

5.7.1.20 MakeYaxisWin()

```
WINDOW* MakeYaxisWin (
            void  )
```

Cria janela do eixo Y.

Retorna

yaxis Ponteiro para a janela do eixo Y

5.7.1.21 MenuGetBoard()

```
TBoard* MenuGetBoard (
            void  )
```

Abre o menu de escolha do tabuleiro.

Esta função irá fazer outras chamadas internas de funções que poderão inicilizar um novo ambiente de interação com o usuário

Retorna

board Tabuleiro que poderá ter sido carregado de um arquivo, criado pelo usuário ou inicializado de forma padrão

5.7.1.22 play_pve()

```
void play_pve (
            WINDOW * boardwin,
            WINDOW * keywin,
            WINDOW * messages,
             TBoard * board )
```

Faz todo o processo do modo de jogo Jogador vs Computador.

Parâmetros

boardwin	Janela do tabuleiro
keywin	Janela onde estão dicas de atalho e região de interação com o teclado dele
Janela	de mensagens
Tabuleiro	que será utilizado

5.7.1.23 play_pvp()

```
void play_pvp (
            WINDOW * boardwin,
            WINDOW * keywin,
            WINDOW * messages,
             TBoard * board )
```

Função responsável por todo o modo de jogo de humano vs humano.

Parâmetros

boardwin	Janela onde está o tabuleiro gráfico do programa
keywin	Janela das teclas e atalhos disponíveis
messages	Janela onde serão impressas as mensagens
board	Tabueleiro guaradado na memória de forma não gráfica

5.7.1.24 print_message()

```
void print_message (
            WINDOW * messages,
            int msg )
```

Imprime mensagens na janela de mensagens.

Parâmetros

messages	Janela de mensagens
msg	Inteiro que indica qual mensagem deverá aparecer na janela

5.7.1.25 print_turn()

```
void print_turn (
            WINDOW * helpwin,
            int turn )
```

Mostra de quem é a vez de jogar na partida.

Parâmetros

helpwin	Janela de ajuda
turn	Indica de quem é o turno (vez de jogar)

5.7.1.26 print_winner()

```
void print_winner (
            WINDOW * helpwin,
            int who )
```

Mostra na tela quem é o jogador vencedor da partida.

Parâmetros

helpwin	Janela de ajuda
who	Indica quem foi o vencedor do jogo

5.7.1.27 reverse_color_in_board()

```
int reverse_color_in_board (
            WINDOW * boardwin,
             TBoard * board,
            int line,
            int column )
```

Destaca na interface a posição dada no tabuleiro.

Parâmetros

boardwin	Janela do tabuleiro
board	Tabuleiro referência
line	Posição Y a ser destacada (referente a board e não a tela)
column	Posição X a ser destacada

Retorna

true Se havia uma peça na posição dada
false Se não tiha uma peça na posição dada

5.7.1.28 TranslateCoord()

```
void TranslateCoord (
            int yscreen,
            int xscreen,
            int * yboard,
            int * xboard )
```

Traduz as coordenadas da tela para as do tabuleiro.

Esta função é chamada toda vez que o usuário clica na tela e traduz as coordenadas da posição que o usuário digitou para as coordenadas do tabuleiro

Parâmetros

yscreen	Posição Y onde o usuário digitou
xscreen	Posição X onde o usuário digitou
yboard	Posição Y traduzida
xboard	Posição X traduzida

5.7.1.29 UI_MOUSE_MOVE()

```
int UI_MOUSE_MOVE (
            WINDOW * boardwin,
```

```
            WINDOW * messages,
             TBoard * board,
            int turn,
            MEVENT event,
             ListPastMoves * pastmoves )
```

Move (pag. **??**) a peça através do mouse do usuário.

Parâmetros

boadwin	Janela do tabuleiro
messages	Janela de mensagens
board	Tabuleiro a ser modificado
turn	Indica de quem é a vez de jogar agora
event	Evento de mouse
pastmoves	Lista de movimentos passados para o arquivo PGN

Retorna

turn Indicando de quem é a vez de jogar agora

5.7.1.30 UI_MOVE_PIECE()

```
int UI_MOVE_PIECE (
            WINDOW * boardwin,
            WINDOW * messages,
             TBoard * board,
            int turn,
             Move * movement )
```

Move (pag. **??**) a peça e faz todas as verificações pela própria interface.

Parâmetros

boardwin	Janela do tabuleiro
messages	Janela de mensagens
board	Tabuleiro a ser modificado
turn	Indica de quem é a vez de jogar
movement	Movimento a ser feito

Retorna

turn Indicando de quem é a vez de jogar

5.7.1.31 verify_evolve_pawn()

```
int verify_evolve_pawn (
            WINDOW * messages,
             TBoard * board )
```

Verifica se existe algum peão para ser promovido.

Ela também irá promover o peão de acordo com a escolha do jogador

Parâmetros

messages	Janela de mensagens
board	Tabuleiro atual do jogo

Retorna

true Se tinha um peão para promover
false Se não tinha um peão para promover

5.7.1.32 verify_turn()

```
int verify_turn (
             TBoard * board,
             Move * movement,
            int turn )
```

Verifica se o jogador vai mexer a peça correta.

Ela verifica se realmente é o turno da cor da peça que o usuário está querendo movimentar

Parâmetros

board	Tabuleiro atual do jogo
movement	Movimento que o jogador quer fazer
turn	De quem é o turno atual do jogo

Retorna

true Se é o turno da cor da peça a ser mexida
false Se não é o turno da cor da peça a ser mexida

5.7.1.33 wanna_save()

```
int wanna_save (
            WINDOW * messages )
```

Pergunta ao usuário se ele quer salvar o jogo.

Parâmetros

messages	Janela de mensagens

Retorna

true Se o usuário quiser salvar o jogo
false Se o usuário não quiser salvar

5.7.1.34 write_keys_help()

```
void write_keys_help (
            WINDOW * keywin,
            int wintype )
```

5.8 Referência do Arquivo interface.h

```
#include <ncurses.h>
#include <stdlib.h>
#include <string.h>
#include "tabuleiro.h"
#include "logica.h"
#include "in-out.h"
#include "arv_decisoes.h"
#include "ia.h"
#include "list_of_moves.h"
```

Funções

- WINDOW * **MakeBoardWin** (void)
  Cria janela do tabuleiro.
- WINDOW * **MakeYaxisWin** (void)
  Cria janela do eixo Y.
- WINDOW * **MakeXaxisWin** (void)
  Cria janela do eixo X.
- WINDOW * **MakeKeyWin** (void)
  Cria a janela do menu de atalhos.
- WINDOW * **MakeMsgWin** (void)
  Cria janela de mensagens.
- WINDOW * **MakeHelpWin** (void)
  Cria a janela de ajuda.
- void **TranslateCoord** (int yscreen, int xscreen, int *yboard, int *xboard)
  Traduz as coordenadas da tela para as do tabuleiro.
- void **InitBoard** (WINDOW *boardwin, **TBoard** * **board**)
  Inicializa graficamente um tabuleiro na base inicial.

- void **DrawBoard** (WINDOW *boardwin)

  Desenha um tabuleiro vazio no terminal.
- void **DrawAxis** (WINDOW *yaxis, WINDOW *xaxis)

  Desenha os eixos coordenados que servem para indicar ajudar a identificar a posição de uma peça.
- int **CreateMenu** (WINDOW *menuwin)

  Desenha o menu de escolha do tipo de jogo e faz o usuário escolher entre uma das opções disponíveis.
- void **write_keys_help** (WINDOW *keywin, int wintype)
- void **play_pvp** (WINDOW *boardwin, WINDOW *keywin, WINDOW *messages, **TBoard** * **board**)

  Função responsável por todo o modo de jogo de humano vs humano.
- void **play_pve** (WINDOW *boardwin, WINDOW *keywin, WINDOW *messages, **TBoard** * **board**)

  Faz todo o processo do modo de jogo Jogador vs Computador.
- void **clear_keywin** (WINDOW *keywin)

  Limpa a área de comandos da janela de ajuda com atalhos.
- void **init_msg_win** (WINDOW *messages)

  Inicializa a janela de mensagens.
- void **clear_message** (WINDOW *messages)

  Limpa janela de mensagens.
- void **print_message** (WINDOW *messages, int msg)

  Imprime mensagens na janela de mensagens.
- **TBoard** * **MenuGetBoard** (void)

  Abre o menu de escolha do tabuleiro.
- **TBoard** * **CreateNewBoard** (void)

  Função de criar um novo tabuleiro.
- void **HelpWinNewBoard** (WINDOW *helpwin)

  Inicializa a janela de ajuda para o menu de criação.
- int **change_turn** (int turn)

  Muda a vez da jogada.
- int **verify_turn** (**TBoard** * **board**, **Move** *movement, int turn)

  Verifica se o jogador vai mexer a peça correta.
- void **GiveHint** (WINDOW *helpwin, **TBoard** * **board**, int turn)

  Mostra as dicas na tela do jogador.
- void **print_turn** (WINDOW *helpwin, int turn)

  Mostra de quem é a vez de jogar na partida.
- int **wanna_save** (WINDOW *messages)

  Pergunta ao usuário se ele quer salvar o jogo.
- void **print_winner** (WINDOW *helpwin, int who)

  Mostra na tela quem é o jogador vencedor da partida.
- int **reverse_color_in_board** (WINDOW *boardwin, **TBoard** * **board**, int line, int column)

  Destaca na interface a posição dada no tabuleiro.
- int **verify_evolve_pawn** (WINDOW *messages, **TBoard** * **board**)

  Verifica se existe algum peão para ser promovido.
- int **UI_MOVE_PIECE** (WINDOW *boardwin, WINDOW *messages, **TBoard** * **board**, int turn, **Move** *movement)

  ***Move** (*pag. **??**) a peça e faz todas as verificações pela própria interface.*
- int **UI_MOUSE_MOVE** (WINDOW *boardwin, WINDOW *messages, **TBoard** * **board**, int turn, MEVENT event, **ListPastMoves** *pastmoves)

  ***Move** (*pag. **??**) a peça através do mouse do usuário.*
- **Move** * **GetMovement** (WINDOW *keywin, char chess_move[ ])

  Obtém do usuário um movimento através da notação.
- void **HelpWinPVE** (WINDOW *helpwin)

  Inicializa a janela de ajuda do PVE com o básico.
- void **HelpWinPVP** (WINDOW *helpwin)

  Mostra legenda das peças na tela de ajuda no PVP.

Variáveis

- const int **XLIMIT** = 8
- const int **YLIMIT** = 8
- const int **BOARDY** = 2
- const int **BOARDX** = 3
- const int **YOFFSET** = 2
- const int **XOFFSET** = 4
- const int **INVALID_SINTAX** = 0
- const int **ARE_YOU_SURE** = 1
- const int **CONTINUE_GAME** = 2
- const int **INVALID_MOVE** = 3
- const int **WHITE_MOVE** = 4
- const int **BLACK_MOVE** = 5
- const int **NOTBLACKSMOVE** = 6
- const int **NOTWHITESMOVE** = 7
- const int **CONTINUE** = 8
- const int **INSERT_PIECE** = 9
- const int **CLICK** = 10
- const int **INVALID_PIECE** = 11
- const int **INVALID_BOARD** = 12
- const int **USE_MOUSE** = 13
- const int **NOTPIECE** = 14
- const int **CLICK_DESTINY** = 15
- const int **OUT_RANGE** = 16
- const int **SAVED_GAME** = 17
- const int **W_CHECK** = 18
- const int **B_CHECK** = 19
- const int **WHITE_WON** = 20
- const int **BLACK_WON** = 21
- const int **WANNA_SAVE** = 22
- const int **GIVE_A_PIECE** = 23
- const int **YOURCOLOR** = 24
- const int **PVP** = 1
- const int **PVE** = 2
- const int **EXITGAME** = 3
- const int **STD_BOARD** = 1
- const int **SAVED_BOARD** = 2
- const int **NEW_BOARD** = 3
- const int **EXIT_GAME** = 4
- const int **GAMING** = 0
- const int **CREATING** = 1

5.8.1 Funções

5.8.1.1 change_turn()

```
int change_turn (
            int turn )
```

Muda a vez da jogada.

Ela vai mudar o turno da jogada, por exemplo se foi as pretas que jogaram por último ela muda o turno para as brancas

Parâmetros

turn	Indica de quem era a última jogada

Retorna

new_turn Indica de quem é a nova vez de jogar

5.8.1.2 clear_keywin()

```
void clear_keywin (
            WINDOW * keywin )
```

Limpa a área de comandos da janela de ajuda com atalhos.

A área de comando desta janela é onde o usuário irá digitar o movimento desejado, que é logo abaixo das informações de atalhos

Parâmetros

keywin	Janela de ajuda com atalhos

5.8.1.3 clear_message()

```
void clear_message (
            WINDOW * messages )
```

Limpa janela de mensagens.

Ela irá limpar a janela de mensagens mas não limpará o efeito box

Parâmetros

messages	Janela de mensagens

5.8.1.4 CreateMenu()

```
int CreateMenu (
            WINDOW * menuwin )
```

Desenha o menu de escolha do tipo de jogo e faz o usuário escolher entre uma das opções disponíveis.

Parâmetros

menuwin	Janela onde será mostrado o menu

Retorna

1 Se o usuário escolher "Jogador VS Jogador" 2 Se ele escolher "Jogador VS Máquina" 3 Se escolher Sair

5.8.1.5 CreateNewBoard()

```
TBoard * CreateNewBoard (
            void  )
```

Função de criar um novo tabuleiro.

Esta função irá abrir um novo ambiente de interação com o usuário onde ele poderá criar um tabuleiro de sua preferência.

Retorna

board Tabuleiro criado pelo usuário

5.8.1.6 DrawAxis()

```
void DrawAxis (
            WINDOW * yaxis,
            WINDOW * xaxis )
```

Desenha os eixos coordenados que servem para indicar ajudar a identificar a posição de uma peça.

Parâmetros

yaxis	Eixo das coordenadas dadas por números
xaxis	Eixo das coordenadas dadas pelas letras

Retorna

Sem retorno

5.8.1.7 DrawBoard()

```
void DrawBoard (
            WINDOW * boardwin )
```

Desenha um tabuleiro vazio no terminal.

Parâmetros

boardwin	Janela onde será colocado o tabuleiro

Retorna

Não há retorno, ela apenas desenha o quadro na janela dada

5.8.1.8 GetMovement()

```
Move * GetMovement (
            WINDOW * keywin,
            char chess_move[] )
```

Obtém do usuário um movimento através da notação.

Parâmetros

keywin	Janela onde está os atalhos
chess_move	String onde será colocada a jodada do jogador

Retorna

movement Movimento traduzido do jogador para o tipo Move∗

5.8.1.9 GiveHint()

```
void GiveHint (
            WINDOW * helpwin,
             TBoard * board,
            int turn )
```

Mostra as dicas na tela do jogador.

Parâmetros

helpwin	Janela de ajuda para o usuário
board	Tabuleiro que será considerado para as dicas
turn	Indica quem é o dono do turno e que vai receber as dicas

5.8.1.10 HelpWinNewBoard()

```
void HelpWinNewBoard (
            WINDOW * helpwin )
```

Inicializa a janela de ajuda para o menu de criação.

Observe que esta função é chamada apenas para o menu de criação de uma nova imagem. Esta função irá colocar na janela de ajuda alguns resumos sobre as peças que o usuário poderá colocar e a explicação de como colocar uma nova peça. Não possui retorno

Parâmetros

helpwin	Janela de ajuda

5.8.1.11 HelpWinPVE()

```
void HelpWinPVE (
            WINDOW * helpwin )
```

Inicializa a janela de ajuda do PVE com o básico.

Parâmetros

helpwin	Janela de ajuda

5.8.1.12 HelpWinPVP()

```
void HelpWinPVP (
            WINDOW * helpwin )
```

Mostra legenda das peças na tela de ajuda no PVP.

Parâmetros

helpwin	Janela de ajuda onde seram colocadas as legendas

5.8.1.13 init_msg_win()

```
void init_msg_win (
            WINDOW * messages )
```

Inicializa a janela de mensagens.

Parâmetros

messages	Janela de mensagens já alocada

5.8.1.14 InitBoard()

```
void InitBoard (
             WINDOW * boardwin,
              TBoard * board )
```

Inicializa graficamente um tabuleiro na base inicial.

Parâmetros

boardwin	Janela onde está o tabuleiro
board	Tabuleiro que será incializado na parte gráfica

Retorna

Não tem retorno

5.8.1.15 MakeBoardWin()

```
WINDOW * MakeBoardWin (
             void  )
```

Cria janela do tabuleiro.

Retorna

boardwin Ponteiro para a janela do tabuleiro

5.8.1.16 MakeHelpWin()

```
WINDOW * MakeHelpWin (
             void  )
```

Cria a janela de ajuda.

Retorna

helpwin Janela da ajuda

5.8.1.17 MakeKeyWin()

```
WINDOW * MakeKeyWin (
            void  )
```

Cria a janela do menu de atalhos.

Este menu de atalhos é o que mostra as opções de atalhos no teclado para o usuário, como a tecla 'j' no menu do PVP que é usada para fazer jogadas usando a notação

Retorna

keywin Janela do menu de atalhos

5.8.1.18 MakeMsgWin()

```
WINDOW * MakeMsgWin (
            void  )
```

Cria janela de mensagens.

Retorna

messages Janela de mensagens

5.8.1.19 MakeXaxisWin()

```
WINDOW * MakeXaxisWin (
            void  )
```

Cria janela do eixo X.

Retorna

xaxis Ponteiro para a janela do eixo X

5.8.1.20 MakeYaxisWin()

```
WINDOW * MakeYaxisWin (
            void  )
```

Cria janela do eixo Y.

Retorna

yaxis Ponteiro para a janela do eixo Y

5.8.1.21 MenuGetBoard()

```
TBoard * MenuGetBoard (
            void  )
```

Abre o menu de escolha do tabuleiro.

Esta função irá fazer outras chamadas internas de funções que poderão inicilizar um novo ambiente de interação com o usuário

Retorna

board Tabuleiro que poderá ter sido carregado de um arquivo, criado pelo usuário ou inicializado de forma padrão

5.8.1.22 play_pve()

```
void play_pve (
            WINDOW * boardwin,
            WINDOW * keywin,
            WINDOW * messages,
             TBoard * board )
```

Faz todo o processo do modo de jogo Jogador vs Computador.

Parâmetros

boardwin	Janela do tabuleiro
keywin	Janela onde estão dicas de atalho e região de interação com o teclado dele
Janela	de mensagens
Tabuleiro	que será utilizado

5.8.1.23 play_pvp()

```
void play_pvp (
            WINDOW * boardwin,
            WINDOW * keywin,
            WINDOW * messages,
             TBoard * board )
```

Função responsável por todo o modo de jogo de humano vs humano.

Parâmetros

boardwin	Janela onde está o tabuleiro gráfico do programa
keywin	Janela das teclas e atalhos disponíveis
messages	Janela onde serão impressas as mensagens
board	Tabueleiro guaradado na memória de forma não gráfica

5.8.1.24 print_message()

```
void print_message (
            WINDOW * messages,
            int msg )
```

Imprime mensagens na janela de mensagens.

Parâmetros

messages	Janela de mensagens
msg	Inteiro que indica qual mensagem deverá aparecer na janela

5.8.1.25 print_turn()

```
void print_turn (
            WINDOW * helpwin,
            int turn )
```

Mostra de quem é a vez de jogar na partida.

Parâmetros

helpwin	Janela de ajuda
turn	Indica de quem é o turno (vez de jogar)

5.8.1.26 print_winner()

```
void print_winner (
            WINDOW * helpwin,
            int who )
```

Mostra na tela quem é o jogador vencedor da partida.

Parâmetros

helpwin	Janela de ajuda
who	Indica quem foi o vencedor do jogo

5.8.1.27 reverse_color_in_board()

```
int reverse_color_in_board (
             WINDOW * boardwin,
              TBoard * board,
             int line,
             int column )
```

Destaca na interface a posição dada no tabuleiro.

Parâmetros

boardwin	Janela do tabuleiro
board	Tabuleiro referência
line	Posição Y a ser destacada (referente a board e não a tela)
column	Posição X a ser destacada

Retorna

true Se havia uma peça na posição dada
false Se não tiha uma peça na posição dada

5.8.1.28 TranslateCoord()

```
void TranslateCoord (
             int yscreen,
             int xscreen,
             int * yboard,
             int * xboard )
```

Traduz as coordenadas da tela para as do tabuleiro.

Esta função é chamada toda vez que o usuário clica na tela e traduz as coordenadas da posição que o usuário digitou para as coordenadas do tabuleiro

Parâmetros

yscreen	Posição Y onde o usuário digitou
xscreen	Posição X onde o usuário digitou
yboard	Posição Y traduzida
xboard	Posição X traduzida

5.8.1.29 UI_MOUSE_MOVE()

```
int UI_MOUSE_MOVE (
             WINDOW * boardwin,
```

```
              WINDOW * messages,
               TBoard * board,
              int turn,
              MEVENT event,
               ListPastMoves * pastmoves )
```

Move (pag. **??**) a peça através do mouse do usuário.

Parâmetros

boadwin	Janela do tabuleiro
messages	Janela de mensagens
board	Tabuleiro a ser modificado
turn	Indica de quem é a vez de jogar agora
event	Evento de mouse
pastmoves	Lista de movimentos passados para o arquivo PGN

Retorna

turn Indicando de quem é a vez de jogar agora

5.8.1.30 UI_MOVE_PIECE()

```
int UI_MOVE_PIECE (
              WINDOW * boardwin,
              WINDOW * messages,
               TBoard * board,
              int turn,
               Move * movement )
```

Move (pag. **??**) a peça e faz todas as verificações pela própria interface.

Parâmetros

boardwin	Janela do tabuleiro
messages	Janela de mensagens
board	Tabuleiro a ser modificado
turn	Indica de quem é a vez de jogar
movement	Movimento a ser feito

Retorna

turn Indicando de quem é a vez de jogar

5.8.1.31 verify_evolve_pawn()

```
int verify_evolve_pawn (
             WINDOW * messages,
              TBoard * board )
```

Verifica se existe algum peão para ser promovido.

Ela também irá promover o peão de acordo com a escolha do jogador

Parâmetros

messages	Janela de mensagens
board	Tabuleiro atual do jogo

Retorna

true Se tinha um peão para promover
false Se não tinha um peão para promover

5.8.1.32 verify_turn()

```
int verify_turn (
              TBoard * board,
              Move * movement,
             int turn )
```

Verifica se o jogador vai mexer a peça correta.

Ela verifica se realmente é o turno da cor da peça que o usuário está querendo movimentar

Parâmetros

board	Tabuleiro atual do jogo
movement	Movimento que o jogador quer fazer
turn	De quem é o turno atual do jogo

Retorna

true Se é o turno da cor da peça a ser mexida
false Se não é o turno da cor da peça a ser mexida

5.8.1.33 wanna_save()

```
int wanna_save (
             WINDOW * messages )
```

Pergunta ao usuário se ele quer salvar o jogo.

Parâmetros

messages	Janela de mensagens

Retorna

true Se o usuário quiser salvar o jogo
false Se o usuário não quiser salvar

5.8.1.34 write_keys_help()

```
void write_keys_help (
            WINDOW * keywin,
            int wintype )
```

5.8.2 Variáveis

5.8.2.1 ARE_YOU_SURE

```
const int ARE_YOU_SURE = 1
```

Perguntar se usuário tem certeza

5.8.2.2 B_CHECK

```
const int B_CHECK = 19
```

Indica xeque no rei preto

5.8.2.3 BLACK_MOVE

```
const int BLACK_MOVE = 5
```

Avisar que a vez das pretas

5.8.2.4 BLACK_WON

```
const int BLACK_WON = 21
```

Indica que as pretas venceram

5.8.2.5 BOARDX

```
const int BOARDX = 3
```

Indica onde começa o tabuleiro pelo eixo X

5.8.2.6 BOARDY

```
const int BOARDY = 2
```

Indica onde começa o tabuleiro pelo eixo Y

5.8.2.7 CLICK

```
const int CLICK = 10
```

Avisar para clicar na posição desejada

5.8.2.8 CLICK_DESTINY

```
const int CLICK_DESTINY = 15
```

Pedir para clicar no destino

5.8.2.9 CONTINUE

```
const int CONTINUE = 8
```

5.8.2.10 CONTINUE_GAME

```
const int CONTINUE_GAME = 2
```

Avisar que o jogo vai continuar

5.8.2.11 CREATING

```
const int CREATING = 1
```

Indica menu para mode de criação

5.8.2.12 EXIT_GAME

```
const int EXIT_GAME = 4
```

Opção de sair do jogo

5.8.2.13 EXITGAME

```
const int EXITGAME = 3
```

Sair do jogo

5.8.2.14 GAMING

```
const int GAMING = 0
```

Indica menu para modo de jogo

5.8.2.15 GIVE_A_PIECE

```
const int GIVE_A_PIECE = 23
```

Pedir para o usuário digitar uma peça

5.8.2.16 INSERT_PIECE

```
const int INSERT_PIECE = 9
```

5.8.2.17 INVALID_BOARD

```
const int INVALID_BOARD = 12
```

Avisar de tabuleiro inválido

5.8.2.18 INVALID_MOVE

```
const int INVALID_MOVE = 3
```

Avisar sobre movimento inválido

5.8.2.19 INVALID_PIECE

```
const int INVALID_PIECE = 11
```

Avisar de peça inválida

5.8.2.20 INVALID_SINTAX

```
const int INVALID_SINTAX = 0
```

Avisa sobre sintxe inválida

5.8.2.21 NEW_BOARD

```
const int NEW_BOARD = 3
```

Opção de criar novo tabuleiro

5.8.2.22 NOTBLACKSMOVE

```
const int NOTBLACKSMOVE = 6
```

Avisar que não é a vez das pretas

5.8.2.23 NOTPIECE

```
const int NOTPIECE = 14
```

Avisar que não há uma peça naquela posição

5.8.2.24 NOTWHITESMOVE

```
const int NOTWHITESMOVE = 7
```

Avisar que não é a vez das brancas

5.8.2.25 OUT_RANGE

```
const int OUT_RANGE = 16
```

Avisar que o destino está fora do tabuleiro

5.8.2.26 PVE

```
const int PVE = 2
```

Player vs Environment

5.8.2.27 PVP

```
const int PVP = 1
```

Player vs Player

5.8.2.28 SAVED_BOARD

```
const int SAVED_BOARD = 2
```

Opção de carregar tabuleiro salvo

5.8.2.29 SAVED_GAME

```
const int SAVED_GAME = 17
```

Avisa que o jogo foi salvo

5.8.2.30 STD_BOARD

```
const int STD_BOARD = 1
```

Opção de tabuleiro padrão

5.8.2.31 USE_MOUSE

```
const int USE_MOUSE = 13
```

Avisar para usar o mouse

5.8.2.32 W_CHECK

```
const int W_CHECK = 18
```

Indica xeque no rei branco

5.8.2.33 WANNA_SAVE

```
const int WANNA_SAVE = 22
```

Perguntar se o usuário quer salvar o jogo

5.8.2.34 WHITE_MOVE

```
const int WHITE_MOVE = 4
```

Avisar que a vez das brancas

5.8.2.35 WHITE_WON

```
const int WHITE_WON = 20
```

Indica que as brancas veceram

5.8.2.36 XLIMIT

```
const int XLIMIT = 8
```

Indica o tamanho do eixo Y do tabuleiro

5.8.2.37 XOFFSET

```
const int XOFFSET = 4
```

Mesma coisa do YOFFSET só que para o eixo X

5.8.2.38 YLIMIT

```
const int YLIMIT = 8
```

Indica o tamanho do eixo X do tabuleiro

5.8.2.39 YOFFSET

```
const int YOFFSET = 2
```

Número de caracteres entre duas posições no tabuleiro gráfico pelo eixo Y

5.8.2.40 YOURCOLOR

```
const int YOURCOLOR = 24
```

Perguntar para o usuário qual cor ele prefere

5.9 Referência do Arquivo list_of_moves.c

```
#include "../include/list_of_moves.h"
#include <stdio.h>
#include <stdlib.h>
```

Funções

- **ListOfMoves** ∗ **CreateListOfMoves** (void)
  Cria uma lista de movimentos.
- int **InsertMove** (**ListOfMoves** ∗list, int originx, int originy, int destinyx, int destinyy)
  Insere um movimento na lista.
- int **DeleteListOfMoves** (**ListOfMoves** ∗list)
  Libera memória utilizada por uma lista.
- int **SearchListOfMoves** (**ListOfMoves** ∗list, int originx, int originy, int destinyx, int destinyy)
  Busca um movimento na lista de movimentos.

5.9.1 Funções

5.9.1.1 CreateListOfMoves()

```
ListOfMoves* CreateListOfMoves (
            void  )
```

Cria uma lista de movimentos.

Parâmetros

void.	

Retorna

Retorna uma lista de movimentos vazia.

5.9.1.2 DeleteListOfMoves()

```
int DeleteListOfMoves (
              ListOfMoves * list )
```

Libera memória utilizada por uma lista.

Parâmetros

list	uma lista de movimentos.

Retorna

Retorna um inteiro indicando a falha ou sucesso da operação.

5.9.1.3 InsertMove()

```
int InsertMove (
              ListOfMoves * list,
             int originx,
             int originy,
             int destinyx,
             int destinyy )
```

Insere um movimento na lista.

Parâmetros

list	Ponteiro para uma lista de movimentos.
originx	Inteiro com a coordenada x de origem.
originy	Inteiro com a coordenada y de origem.
destinyx	Inteiro com a coordenada x de destino.
destinyy	Inteiro com a coordenada y de destino.

Retorna

Retorna um inteiro indicando a falha ou sucesso da inserção.

5.9.1.4 SearchListOfMoves()

```
int SearchListOfMoves (
             ListOfMoves * list,
            int originx,
            int originy,
            int destinyx,
            int destinyy )
```

Busca um movimento na lista de movimentos.

Parâmetros

list	uma lista de movimentos.
originx	Inteiro com a coordenada x de origem.
originy	Inteiro com a coordenada y de origem.
destinyx	Inteiro com a coordenada x de destino.
destinyy	Inteiro com a coordenada y de destino.

Retorna

Retorna um inteiro indicando a falha ou sucesso da operação.

5.10 Referência do Arquivo list_of_moves.h

Estruturas de Dados

- struct **Move**

  Estrutura que irá caracterizar a jogada.
- struct **NodeList**
- struct **ListOfMoves**

  Estrutura representará todas as possíveis jogadas de um tabuleiro.

Definições de Tipos

- typedef struct **Move** **Move**
- typedef struct **NodeList** **NodeList**
- typedef struct **ListOfMoves** **ListOfMoves**

Funções

- **ListOfMoves** ∗ **CreateListOfMoves** (void)

  Cria uma lista de movimentos.
- int **InsertMove** (**ListOfMoves** ∗list, int originx, int originy, int destinyx, int destinyy)

  Insere um movimento na lista.
- int **DeleteListOfMoves** (**ListOfMoves** ∗list)

  Libera memória utilizada por uma lista.
- int **SearchListOfMoves** (**ListOfMoves** ∗list, int originx, int originy, int destinyx, int destinyy)

  Busca um movimento na lista de movimentos.

5.10.1 Definições dos tipos

5.10.1.1 ListOfMoves

```
typedef struct  ListOfMoves  ListOfMoves
```

5.10.1.2 Move

```
typedef struct  Move  Move
```

5.10.1.3 NodeList

```
typedef struct  NodeList  NodeList
```

5.10.2 Funções

5.10.2.1 CreateListOfMoves()

```
ListOfMoves * CreateListOfMoves (
            void  )
```

Cria uma lista de movimentos.

Parâmetros

void.	

Retorna

Retorna uma lista de movimentos vazia.

5.10.2.2 DeleteListOfMoves()

```
int DeleteListOfMoves (
              ListOfMoves * list )
```

Libera memória utilizada por uma lista.

Parâmetros

list	uma lista de movimentos.

Retorna

Retorna um inteiro indicando a falha ou sucesso da operação.

5.10.2.3 InsertMove()

```
int InsertMove (
              ListOfMoves * list,
             int originx,
             int originy,
             int destinyx,
             int destinyy )
```

Insere um movimento na lista.

Parâmetros

list	Ponteiro para uma lista de movimentos.
originx	Inteiro com a coordenada x de origem.
originy	Inteiro com a coordenada y de origem.
destinyx	Inteiro com a coordenada x de destino.
destinyy	Inteiro com a coordenada y de destino.

Retorna

Retorna um inteiro indicando a falha ou sucesso da inserção.

5.10.2.4 SearchListOfMoves()

```
int SearchListOfMoves (
              ListOfMoves * list,
             int originx,
             int originy,
             int destinyx,
             int destinyy )
```

Busca um movimento na lista de movimentos.

Parâmetros

list	uma lista de movimentos.
originx	Inteiro com a coordenada x de origem.
originy	Inteiro com a coordenada y de origem.
destinyx	Inteiro com a coordenada x de destino.
destinyy	Inteiro com a coordenada y de destino.

Retorna

Retorna um inteiro indicando a falha ou sucesso da operação.

5.11 Referência do Arquivo logica.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../include/logica.h"
```

Funções

- int **verify_syntax_move** (char chess_move[ ])
- **Move** * **algebraic_translate** (char chess_move[ ])

  Traduz uma notação algébrica de um movimento de xadrez para um movimento normal na matriz do tabuleiro.
- void **Move2Algebraic** (**Move** *movement, char chess_move[ ])
- **ListOfMoves** * **WhitePawnMovements** (**TBoard** * **board**, **ListOfMoves** *AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um peão branco, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** * **BlackPawnMovements** (**TBoard** * **board**, **ListOfMoves** *AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um peão preto, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** * **HorseMovements** (**TBoard** * **board**, **ListOfMoves** *AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um Cavalo, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** * **TowerMovements** (**TBoard** * **board**, **ListOfMoves** *AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de uma torre, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** * **BishopMovements** (**TBoard** * **board**, **ListOfMoves** *AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um bispo, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** * **QueenMovements** (**TBoard** * **board**, **ListOfMoves** *AllMoves, int originx, int originy)

Adiciona os movimentos possíveis de uma rainha, dada uma configuração de tabuleiro, em uma lista de movimentos.

- **ListOfMoves** * **KingMovements** (**TBoard** * **board**, **ListOfMoves** *AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um rei, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** * **AnalyzePossibleMovementsWhite** (**TBoard** * **board**)

  Armazena movimentos possíveis para cada peça branca.
- **ListOfMoves** * **AnalyzePossibleMovementsBlack** (**TBoard** * **board**)

  Armazena movimentos possíveis para cada peça preta.
- int **VerifyValidMovement** (**TBoard** * **board**, int originx, int originy, int destinyx, int destinyy)

  Verifica com base nas coordenadas de origem e destino do movimento, se ele é válido.
- **TBoard** * **VerifyCheck** (**TBoard** * **board**, int color)

  Verifica com base em um tabuleiro e uma cor se o rei dessa cor está em xeque.
- **ListOfMoves** * **VerifyCheckMate** (**TBoard** * **board**, int color)

5.11.1 Funções

5.11.1.1 algebraic_translate()

```
Move* algebraic_translate (
            char chess_move[] )
```

Traduz uma notação algébrica de um movimento de xadrez para um movimento normal na matriz do tabuleiro.

Parâmetros

chess_move	String com a notação algébrica do movimento

Retorna

Um elemento de movimento com a origem e o destino do movimento

5.11.1.2 AnalyzePossibleMovementsBlack()

```
ListOfMoves* AnalyzePossibleMovementsBlack (
             TBoard * board )
```

Armazena movimentos possíveis para cada peça preta.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Retorna uma lista com os movimentos possíveis de acordo com as peças pretas.

5.11.1.3 AnalyzePossibleMovementsWhite()

```
ListOfMoves* AnalyzePossibleMovementsWhite (
             TBoard * board )
```

Armazena movimentos possíveis para cada peça branca.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Retorna uma lista com os movimentos possíveis de acordo com as peças brancas.

5.11.1.4 BishopMovements()

```
ListOfMoves* BishopMovements (
             TBoard * board,
             ListOfMoves * AllMoves,
            int originx,
            int originy )
```

Adiciona os movimentos possíveis de um bispo, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.11.1.5 BlackPawnMovements()

```
ListOfMoves* BlackPawnMovements (
             TBoard * board,
```

```
 ListOfMoves * AllMoves,
int originx,
int originy )
```

Adiciona os movimentos possíveis de um peão preto, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.11.1.6 HorseMovements()

```
ListOfMoves* HorseMovements (
             TBoard * board,
             ListOfMoves * AllMoves,
            int originx,
            int originy )
```

Adiciona os movimentos possíveis de um Cavalo, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.11.1.7 KingMovements()

```
ListOfMoves* KingMovements (
             TBoard * board,
             ListOfMoves * AllMoves,
            int originx,
            int originy )
```

Adiciona os movimentos possíveis de um rei, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.11.1.8 Move2Algebraic()

```
void Move2Algebraic (
              Move * movement,
             char chess_move[] )
```

5.11.1.9 QueenMovements()

```
ListOfMoves* QueenMovements (
              TBoard * board,
              ListOfMoves * AllMoves,
             int originx,
             int originy )
```

Adiciona os movimentos possíveis de uma rainha, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.11.1.10 TowerMovements()

```
ListOfMoves* TowerMovements (
              TBoard * board,
```

```
 ListOfMoves * AllMoves,
int originx,
int originy )
```

Adiciona os movimentos possíveis de uma torre, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.11.1.11 verify_syntax_move()

```
int verify_syntax_move (
            char chess_move[] )
```

5.11.1.12 VerifyCheck()

```
TBoard* VerifyCheck (
             TBoard * board,
            int color )
```

Verifica com base em um tabuleiro e uma cor se o rei dessa cor está em xeque.

Verifica com base em um tabuleiro e uma cor se o rei dessa cor está em xeque mate.

Parâmetros

board	Ponteiro para um tabuleiro.
color	Inteiro indicando a cor do rei.

Retorna

Retorna um ponteiro para um tabuleiro com o campo check da cor analisada, atualizado.

Parâmetros

board	Ponteiro para um tabuleiro.
color	Inteiro indicando a cor do rei.

Retorna

Retorna um ponteiro para uma lista de movimentos contendo os movimentos possíveis para sair do xeque ou NULL em caso de xeque mate.

5.11.1.13 VerifyCheckMate()

```
ListOfMoves* VerifyCheckMate (
             TBoard * board,
            int color )
```

5.11.1.14 VerifyValidMovement()

```
int VerifyValidMovement (
             TBoard * board,
            int originx,
            int originy,
            int destinyx,
            int destinyy )
```

Verifica com base nas coordenadas de origem e destino do movimento, se ele é válido.

Parâmetros

board	Ponteiro para um tabuleiro.
originx	Inteiro representando a coordenada x da posição (x,y) a ser verificada.
originy	Inteiro representando a coordenada y da posição (x,y) a ser verificada.
destinyx	Inteiro representando a coordenada x da posição de destino (x,y) a ser verificada.
destinyy	Inteiro representando a coordenada y da posição de destubi (x,y) a ser verificada.

Retorna

Retorna um inteiro 0 ou 1 indicando movimento inválido ou válido respectivamente.

5.11.1.15 WhitePawnMovements()

```
ListOfMoves* WhitePawnMovements (
             TBoard * board,
             ListOfMoves * AllMoves,
            int originx,
            int originy )
```

Adiciona os movimentos possíveis de um peão branco, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.12 Referência do Arquivo logica.h

```
#include "../include/list_of_moves.h"
#include "../include/tabuleiro.h"
```

Funções

- **ListOfMoves** ∗ **AnalyzePossibleMovementsBlack** (**TBoard** ∗ **board**)

  Armazena movimentos possíveis para cada peça preta.
- **ListOfMoves** ∗ **AnalyzePossibleMovementsWhite** (**TBoard** ∗ **board**)

  Armazena movimentos possíveis para cada peça branca.
- int **verify_syntax_move** (char chess_move[ ])
- **Move** ∗ **algebraic_translate** (char chess_move[ ])

  Traduz uma notação algébrica de um movimento de xadrez para um movimento normal na matriz do tabuleiro.
- **ListOfMoves** ∗ **BlackPawnMovements** (**TBoard** ∗ **board**, **ListOfMoves** ∗AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um peão preto, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** ∗ **WhitePawnMovements** (**TBoard** ∗ **board**, **ListOfMoves** ∗AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um peão branco, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** ∗ **TowerMovements** (**TBoard** ∗ **board**, **ListOfMoves** ∗AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de uma torre, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** ∗ **BishopMovements** (**TBoard** ∗ **board**, **ListOfMoves** ∗AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um bispo, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** ∗ **QueenMovements** (**TBoard** ∗ **board**, **ListOfMoves** ∗AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de uma rainha, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** ∗ **KingMovements** (**TBoard** ∗ **board**, **ListOfMoves** ∗AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um rei, dada uma configuração de tabuleiro, em uma lista de movimentos.
- **ListOfMoves** ∗ **HorseMovements** (**TBoard** ∗ **board**, **ListOfMoves** ∗AllMoves, int originx, int originy)

  Adiciona os movimentos possíveis de um Cavalo, dada uma configuração de tabuleiro, em uma lista de movimentos.
- int **VerifyValidMovement** (**TBoard** ∗ **board**, int originx, int originy, int destinyx, int destinyy)

  Verifica com base nas coordenadas de origem e destino do movimento, se ele é válido.
- **TBoard** ∗ **VerifyCheck** (**TBoard** ∗ **board**, int color)

  Verifica com base em um tabuleiro e uma cor se o rei dessa cor está em xeque.
- **ListOfMoves** ∗ **VerifyCheckMate** (**TBoard** ∗ **board**, int color)
- void **Move2Algebraic** (**Move** ∗movement, char chess_move[ ])

5.12.1 Funções

5.12.1.1 algebraic_translate()

```
Move * algebraic_translate (
            char chess_move[] )
```

Traduz uma notação algébrica de um movimento de xadrez para um movimento normal na matriz do tabuleiro.

Parâmetros

chess_move	String com a notação algébrica do movimento

Retorna

Um elemento de movimento com a origem e o destino do movimento

5.12.1.2 AnalyzePossibleMovementsBlack()

```
ListOfMoves AnalyzePossibleMovementsBlack (
             TBoard * board )
```

Armazena movimentos possíveis para cada peça preta.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Retorna uma lista com os movimentos possíveis de acordo com as peças pretas.

5.12.1.3 AnalyzePossibleMovementsWhite()

```
ListOfMoves AnalyzePossibleMovementsWhite (
             TBoard * board )
```

Armazena movimentos possíveis para cada peça branca.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Retorna uma lista com os movimentos possíveis de acordo com as peças brancas.

5.12.1.4 BishopMovements()

```
ListOfMoves * BishopMovements (
              TBoard * board,
              ListOfMoves * AllMoves,
             int originx,
             int originy )
```

Adiciona os movimentos possíveis de um bispo, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.12.1.5 BlackPawnMovements()

```
ListOfMoves * BlackPawnMovements (
              TBoard * board,
              ListOfMoves * AllMoves,
             int originx,
             int originy )
```

Adiciona os movimentos possíveis de um peão preto, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.12.1.6 HorseMovements()

```
ListOfMoves * HorseMovements (
              TBoard * board,
              ListOfMoves * AllMoves,
             int originx,
             int originy )
```

Adiciona os movimentos possíveis de um Cavalo, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.12.1.7 KingMovements()

```
ListOfMoves * KingMovements (
              TBoard * board,
              ListOfMoves * AllMoves,
             int originx,
             int originy )
```

Adiciona os movimentos possíveis de um rei, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.12.1.8 Move2Algebraic()

```
void Move2Algebraic (
              Move * movement,
             char chess_move[] )
```

5.12.1.9 QueenMovements()

```
ListOfMoves * QueenMovements (
              TBoard * board,
              ListOfMoves * AllMoves,
             int originx,
             int originy )
```

Adiciona os movimentos possíveis de uma rainha, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.12.1.10 TowerMovements()

```
ListOfMoves * TowerMovements (
              TBoard * board,
              ListOfMoves * AllMoves,
             int originx,
             int originy )
```

Adiciona os movimentos possíveis de uma torre, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.12.1.11 verify_syntax_move()

```
int verify_syntax_move (
            char chess_move[] )
```

5.12.1.12 VerifyCheck()

```
TBoard * VerifyCheck (
             TBoard * board,
            int color )
```

Verifica com base em um tabuleiro e uma cor se o rei dessa cor está em xeque.

Verifica com base em um tabuleiro e uma cor se o rei dessa cor está em xeque mate.

Parâmetros

board	Ponteiro para um tabuleiro.
color	Inteiro indicando a cor do rei.

Retorna

Retorna um ponteiro para um tabuleiro com o campo check da cor analisada, atualizado.

Parâmetros

board	Ponteiro para um tabuleiro.
color	Inteiro indicando a cor do rei.

Retorna

Retorna um ponteiro para uma lista de movimentos contendo os movimentos possíveis para sair do xeque ou NULL em caso de xeque mate.

5.12.1.13 VerifyCheckMate()

```
ListOfMoves* VerifyCheckMate (
             TBoard * board,
            int color )
```

5.12.1.14 VerifyValidMovement()

```
int VerifyValidMovement (
              TBoard * board,
             int originx,
             int originy,
             int destinyx,
             int destinyy )
```

Verifica com base nas coordenadas de origem e destino do movimento, se ele é válido.

Parâmetros

board	Ponteiro para um tabuleiro.
originx	Inteiro representando a coordenada x da posição (x,y) a ser verificada.
originy	Inteiro representando a coordenada y da posição (x,y) a ser verificada.
destinyx	Inteiro representando a coordenada x da posição de destino (x,y) a ser verificada.
destinyy	Inteiro representando a coordenada y da posição de destubi (x,y) a ser verificada.

Retorna

Retorna um inteiro 0 ou 1 indicando movimento inválido ou válido respectivamente.

5.12.1.15 WhitePawnMovements()

```
ListOfMoves * WhitePawnMovements (
              TBoard * board,
              ListOfMoves * AllMoves,
             int originx,
             int originy )
```

Adiciona os movimentos possíveis de um peão branco, dada uma configuração de tabuleiro, em uma lista de movimentos.

Parâmetros

board	Ponteiro para um tabuleiro.
AllMoves	Ponteiro para uma lista de movimentos.
originx	Inteiro referente a coordenada x da peça no tabuleiro.
originy	Inteiro referente a coordenada y da peça no tabuleiro.

Retorna

Retorna uma lista de movimentos e em caso de parâmetros inválidos retorna NULL.

5.13 Referência do Arquivo main.c

```
#include "../include/interface.h"
```

```
#include "stdlib.h"
```

Funções

- int **main** ()

5.13.1 Funções

5.13.1.1 main()

```
int main ( )
```

5.14 Referência do Arquivo README.md

5.15 Referência do Arquivo tabuleiro.c

```
#include <stdio.h>
#include <stdlib.h>
#include "../include/tabuleiro.h"
```

Funções

- **TBoard** * **AlocateBoard** (void)

  Aloca espaço em memória para um elemento de tabuleiro.
- int **StartEmptyBoard** (**TBoard** * **board**)

  Inicializa um tabuleiro sem nenhuma peça em todas as posições.
- int **StartStandardBoard** (**TBoard** * **board**)

  Inicializa um tabuleiro com as peças na posição padrão de um jogo de xadrez.
- int **ColorPiece** (char piece)

  Dada uma determinada peça, verifica qual a cor dela.
- char **WhatPiece** (**TBoard** * **board**, int line, int column)

  Verifica qual peça se encontra na posição dada.
- int **GetValue** (char piece)

  Verifica o valor de uma peça dada ou espaço vazio.
- void **RemovePiece** (**TBoard** * **board**, int line, int column)

  Remove uma peça de uma posição e modifica o peso do tabuleiro.
- int **InsertPiece** (**TBoard** * **board**, char piece, int line, int column)
- int **ValidBoard** (**TBoard** * **board**)

  Verifica se um tabuleiro é válido para ser jogado ou não.
- int **ChangePiece** (**TBoard** * **board**, char piece, int line, int column)

  Muda a peça em uma posição por outra.

- int **MovePiece** (**TBoard** ∗ **board**, int origin_line, int origin_column, int dest_line, int dest_column)

  Move *(*pag. **??***) a peça na posição (xo,yo) para a posição (x,y).*
- int **copy_boards** (**TBoard** ∗copy, **TBoard** ∗ **board**)

  Copia todas as peças de um tabuleiro para outro e também os pesos.
- int **valid_piece** (char piece)

  Verifica se uma peça dada é uma peça de xadrez ou não.
- int **HaveMinimun** (**TBoard** ∗ **board**)

  Verifica se um tabuleiro é no máximo o mínimo para ser válido.

5.15.1 Funções

5.15.1.1 AlocateBoard()

```
TBoard* AlocateBoard (
            void  )
```

Aloca espaço em memória para um elemento de tabuleiro.

Retorna

Um elemento de tabuleiro diferente de nulo alocado dinamicamente inicializado com peso igual a zero

5.15.1.2 ChangePiece()

```
int ChangePiece (
             TBoard * board,
            char piece,
            int line,
            int column )
```

Muda a peça em uma posição por outra.

Parâmetros

board	Ponteiro para um tabuleiro.
piece	É um caractere representado por alguma das constantes de peças definidas.
line	Inteiro indicando a linha x da posição (x,y) da peça a ser modificada. Deve ser um número de 0 a 7.
column	Inteiro indicando a coluna y da posição (x,y) da peça a ser modificada. Deve ser um número de 0 a 7.

Retorna

Retorna 0 para caso seja válido ou -1, caso contrário. Por parâmetro, retorna o tabuleiro modificado.

5.15.1.3 ColorPiece()

```
int ColorPiece (
            char piece )
```

Dada uma determinada peça, verifica qual a cor dela.

Parâmetros

board	Ponteiro para um tabuleiro.
line	Inteiro indicando a linha x da posição (x,y) a ser verificada. Deve ser um número de 0 a 7.
column	Inteiro indicando a coluna y da posição (x,y) a ser verificada. Deve ser um número de 0 a 7.

Retorna

Retorna o caractere correspondente da peça na posição (x,y) ou OUT_OF_RANGE para posições não existentes.

5.15.1.4 copy_boards()

```
int copy_boards (
            TBoard * copy,
            TBoard * board )
```

Copia todas as peças de um tabuleiro para outro e também os pesos.

A função não possui retorno

Parâmetros

copy	Tabuleiro que será a cópia. Não deve ser nulo.
board	Tabuleiro que será copiado. Não deve ser nulo.

Retorna

0, caso funcione adequadamente ou 1, caso contrário

5.15.1.5 GetValue()

```
int GetValue (
            char piece )
```

Verifica o valor de uma peça dada ou espaço vazio.

Os valores, em módulo, para as peças são: Peões - 1; Cavalos e Bispos - 3; Torres - 5; Rainhas - 9; Reis - 200; Espaço em branco - 0; Sendo que as peças pretas assumem valores negativos e as brancas, positivos.

Parâmetros

piece	É um caractere representado por alguma das constantes de peças definidas.

Retorna

Retorna o valor da peça requisitada ou 0 para peças inválidas.

5.15.1.6 HaveMinimun()

```
int HaveMinimun (
              TBoard * board )
```

Verifica se um tabuleiro é no máximo o mínimo para ser válido.

Por exemplo, o tabuleiro mínimo é um rei de cada cor, se tiver qualquer outra peça que não seja um rei ela retorna false, mas ela também retorna true para um tabuleiro com menos que o mínimo para ser válido como um tabuleiro vazio

Parâmetros

board	Tabuleiro que será verificado

Retorna

true Se o tabuleiro obedece as condições estabelecidas
false Se o tabuleiro não obedece as condições estabelecidas

5.15.1.7 InsertPiece()

```
int InsertPiece (
              TBoard * board,
             char piece,
             int line,
             int column )
```

5.15.1.8 MovePiece()

```
int MovePiece (
              TBoard * board,
             int origin_line,
             int origin_column,
             int dest_line,
             int dest_column )
```

Move (pag. **??**) a peça na posição (xo,yo) para a posição (x,y).

Parâmetros

board	Ponteiro para um tabuleiro.
origin_line	Inteiro indicando a linha xo da posição (xo,yo) da peça a ser movida. Deve ser um número de 0 a 7.
origin_column	Inteiro indicando a coluna yo da posição (xo,yo) da peça a ser movida. Deve ser um número de 0 a 7.
dest_line	Inteiro indicando a linha x da posição (x,y) da peça a ser movida. Deve ser um número de 0 a 7.
dest_column	Inteiro indicando a coluna y da posição (x,y) da peça a ser movida. Deve ser um número de 0 a 7.

Retorna

Retorna 0 para caso seja válido ou -1, caso contrário. Por parâmetro, retorna o tabuleiro modificado.

5.15.1.9 RemovePiece()

```
void RemovePiece (
              TBoard * board,
             int line,
             int column )
```

Remove uma peça de uma posição e modifica o peso do tabuleiro.

Insere uma peça válida no tabuleiro.

Parâmetros

board	Ponteiro para um tabuleiro.
line	Inteiro indicando a linha x da posição (x,y) da peça a ser removida. Deve ser um número de 0 a 7.
column	Inteiro indicando a coluna y da posição (x,y) da peça a ser removida. Deve ser um número de 0 a 7.

Retorna

Retorna, por parâmetro, o tabuleiro sem a peça e com seu peso modificado e um inteiro indicando seu funcionamento.

Parâmetros

board	Pontereiro para um tabuleiro.
piece	É um caractere representado por alguma das constantes de peças definidas.
line	Inteiro indicando a linha x da posição (x,y) da peça a ser inserida. Deve ser um número de 0 a 7.
column	Inteiro indicando a coluna y da posição (x,y) da peça a ser inserida. Deve ser um número de 0 a 7.

Retorna

Retorna um inteiro indicando a falha ou sucesso da operação.

5.15.1.10 StartEmptyBoard()

```
int StartEmptyBoard (
              TBoard * board )
```

Inicializa um tabuleiro sem nenhuma peça em todas as posições.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Por parâmetro, retorna o tabuleiro vazio e um inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.15.1.11 StartStandardBoard()

```
int StartStandardBoard (
              TBoard * board )
```

Inicializa um tabuleiro com as peças na posição padrão de um jogo de xadrez.

Peças pretas correspondem a parte "de cima" do tabuleiro (posições de (0.0) a (1,7)). Peças brancas correspondem a parte de "baixo" do tabuleiro (posiços de (6,0) a (7,7)). As outras posições são vazias.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Por parâmetro, retorna o tabuleiro com as peças em posições padrões e um inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.15.1.12 valid_piece()

```
int valid_piece (
             char piece )
```

Verifica se uma peça dada é uma peça de xadrez ou não.

Parâmetros

piece	Peça a ser verificada

Retorna

true Se a peça é uma peça de xadrez
false Se ela não for uma peça de xadrez

5.15.1.13 ValidBoard()

```
int ValidBoard (
              TBoard * board )
```

Verifica se um tabuleiro é válido para ser jogado ou não.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Retorna 1 para caso seja válido ou 0, caso contrário.

5.15.1.14 WhatPiece()

```
char WhatPiece (
              TBoard * board,
             int line,
             int column )
```

Verifica qual peça se encontra na posição dada.

Parâmetros

piece	É um caractere representado por alguma das constantes de peças definidas.

Retorna

Retorna a constante definida BLACK ou WHITE para a cor da peça passada ou -1 se não for uma peça.

5.16 Referência do Arquivo tabuleiro.h

Estruturas de Dados

- struct **board**

Definições de Tipos

- typedef struct **board** **TBoard**

Funções

- **TBoard** * **AlocateBoard** (void)
  Aloca espaço em memória para um elemento de tabuleiro.
- int **StartEmptyBoard** (**TBoard** * **board**)
  Inicializa um tabuleiro sem nenhuma peça em todas as posições.
- int **StartStandardBoard** (**TBoard** * **board**)
  Inicializa um tabuleiro com as peças na posição padrão de um jogo de xadrez.
- int **ColorPiece** (char piece)
  Dada uma determinada peça, verifica qual a cor dela.
- char **WhatPiece** (**TBoard** * **board**, int line, int column)
  Verifica qual peça se encontra na posição dada.
- int **GetValue** (char piece)
  Verifica o valor de uma peça dada ou espaço vazio.
- void **RemovePiece** (**TBoard** * **board**, int line, int column)
  Remove uma peça de uma posição e modifica o peso do tabuleiro.
- int **InsertPiece** (**TBoard** * **board**, char piece, int line, int column)
- int **ValidBoard** (**TBoard** * **board**)
  Verifica se um tabuleiro é válido para ser jogado ou não.
- int **ChangePiece** (**TBoard** * **board**, char piece, int line, int column)
  Muda a peça em uma posição por outra.
- int **MovePiece** (**TBoard** * **board**, int origin_line, int origin_column, int dest_line, int dest_column)
  ***Move** (*pag. **??**) *a peça na posição (xo,yo) para a posição (x,y).*
- int **copy_boards** (**TBoard** *copy, **TBoard** * **board**)
  Copia todas as peças de um tabuleiro para outro e também os pesos.
- int **valid_piece** (char piece)
  Verifica se uma peça dada é uma peça de xadrez ou não.
- int **HaveMinimun** (**TBoard** * **board**)
  Verifica se um tabuleiro é no máximo o mínimo para ser válido.

Variáveis

- const int **WHITE** = 1
- const int **BLACK** = 0
- const int **CHECK** = 1
- const char **W_KING** = 'k'
- const char **W_QUEEN** = 'q'
- const char **W_TOWER** = 'r'
- const char **W_BISHOP** = 'b'
- const char **W_HORSE** = 'n'
- const char **W_PAWN** = 'p'
- const char **BLANK** = '\\'

- const char **B_KING** = 'K'
- const char **B_QUEEN** = 'Q'
- const char **B_TOWER** = 'R'
- const char **B_BISHOP** = 'B'
- const char **B_HORSE** = 'N'
- const char **B_PAWN** = 'P'
- const char **OUT_OF_RANGE** = '∼'

5.16.1 Definições dos tipos

5.16.1.1 TBoard

```
typedef struct board TBoard
```

5.16.2 Funções

5.16.2.1 AlocateBoard()

```
TBoard * AlocateBoard (
            void )
```

Aloca espaço em memória para um elemento de tabuleiro.

Retorna

Um elemento de tabuleiro diferente de nulo alocado dinamicamente inicializado com peso igual a zero

5.16.2.2 ChangePiece()

```
int ChangePiece (
             TBoard * board,
            char piece,
            int line,
            int column )
```

Muda a peça em uma posição por outra.

Parâmetros

board	Ponteiro para um tabuleiro.
piece	É um caractere representado por alguma das constantes de peças definidas.
line	Inteiro indicando a linha x da posição (x,y) da peça a ser modificada. Deve ser um número de 0 a 7.
column	Inteiro indicando a coluna y da posição (x,y) da peça a ser modificada. Deve ser um número de 0 a 7.

Retorna

Retorna 0 para caso seja válido ou -1, caso contrário. Por parâmetro, retorna o tabuleiro modificado.

5.16.2.3 ColorPiece()

```
int ColorPiece (
            char piece )
```

Dada uma determinada peça, verifica qual a cor dela.

Parâmetros

board	Ponteiro para um tabuleiro.
line	Inteiro indicando a linha x da posição (x,y) a ser verificada. Deve ser um número de 0 a 7.
column	Inteiro indicando a coluna y da posição (x,y) a ser verificada. Deve ser um número de 0 a 7.

Retorna

Retorna o caractere correspondente da peça na posição (x,y) ou OUT_OF_RANGE para posições não existentes.

5.16.2.4 copy_boards()

```
void copy_boards (
             TBoard * copy,
             TBoard * board )
```

Copia todas as peças de um tabuleiro para outro e também os pesos.

A função não possui retorno

Parâmetros

copy	Tabuleiro que será a cópia. Não deve ser nulo.
board	Tabuleiro que será copiado. Não deve ser nulo.

Retorna

0, caso funcione adequadamente ou 1, caso contrário

5.16.2.5 GetValue()

```
int GetValue (
            char piece )
```

Verifica o valor de uma peça dada ou espaço vazio.

Os valores, em módulo, para as peças são: Peões - 1; Cavalos e Bispos - 3; Torres - 5; Rainhas - 9; Reis - 200; Espaço em branco - 0; Sendo que as peças pretas assumem valores negativos e as brancas, positivos.

Parâmetros

piece	É um caractere representado por alguma das constantes de peças definidas.

Retorna

Retorna o valor da peça requisitada ou 0 para peças inválidas.

5.16.2.6 HaveMinimun()

```
int HaveMinimun (
             TBoard * board )
```

Verifica se um tabuleiro é no máximo o mínimo para ser válido.

Por exemplo, o tabuleiro mínimo é um rei de cada cor, se tiver qualquer outra peça que não seja um rei ela retorna false, mas ela também retorna true para um tabuleiro com menos que o mínimo para ser válido como um tabuleiro vazio

Parâmetros

board	Tabuleiro que será verificado

Retorna

true Se o tabuleiro obedece as condições estabelecidas
false Se o tabuleiro não obedece as condições estabelecidas

5.16.2.7 InsertPiece()

```
int InsertPiece (
             TBoard * board,
            char piece,
            int line,
            int column )
```

5.16.2.8 MovePiece()

```
int MovePiece (
              TBoard * board,
             int origin_line,
             int origin_column,
             int dest_line,
             int dest_column )
```

Move (pag. **??**) a peça na posição (xo,yo) para a posição (x,y).

Parâmetros

board	Ponteiro para um tabuleiro.
origin_line	Inteiro indicando a linha xo da posição (xo,yo) da peça a ser movida. Deve ser um número de 0 a 7.
origin_column	Inteiro indicando a coluna yo da posição (xo,yo) da peça a ser movida. Deve ser um número de 0 a 7.
dest_line	Inteiro indicando a linha x da posição (x,y) da peça a ser movida. Deve ser um número de 0 a 7.
dest_column	Inteiro indicando a coluna y da posição (x,y) da peça a ser movida. Deve ser um número de 0 a 7.

Retorna

Retorna 0 para caso seja válido ou -1, caso contrário. Por parâmetro, retorna o tabuleiro modificado.

5.16.2.9 RemovePiece()

```
void RemovePiece (
              TBoard * board,
             int line,
             int column )
```

Remove uma peça de uma posição e modifica o peso do tabuleiro.

Insere uma peça válida no tabuleiro.

Parâmetros

board	Ponteiro para um tabuleiro.
line	Inteiro indicando a linha x da posição (x,y) da peça a ser removida. Deve ser um número de 0 a 7.
column	Inteiro indicando a coluna y da posição (x,y) da peça a ser removida. Deve ser um número de 0 a 7.

Retorna

Retorna, por parâmetro, o tabuleiro sem a peça e com seu peso modificado e um inteiro indicando seu funcionamento.

Parâmetros

board	Pontereiro para um tabuleiro.
piece	É um caractere representado por alguma das constantes de peças definidas.
line	Inteiro indicando a linha x da posição (x,y) da peça a ser inserida. Deve ser um número de 0 a 7.
column	Inteiro indicando a coluna y da posição (x,y) da peça a ser inserida. Deve ser um número de 0 a 7.

Retorna

Retorna um inteiro indicando a falha ou sucesso da operação.

5.16.2.10 StartEmptyBoard()

```
int StartEmptyBoard (
              TBoard * board )
```

Inicializa um tabuleiro sem nenhuma peça em todas as posições.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Por parâmetro, retorna o tabuleiro vazio e um inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.16.2.11 StartStandardBoard()

```
void StartStandardBoard (
              TBoard * board )
```

Inicializa um tabuleiro com as peças na posição padrão de um jogo de xadrez.

Peças pretas correspondem a parte "de cima" do tabuleiro (posições de (0.0) a (1,7)). Peças brancas correspondem a parte de "baixo" do tabuleiro (posiços de (6,0) a (7,7)). As outras posições são vazias.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Por parâmetro, retorna o tabuleiro com as peças em posições padrões e um inteiro indicando o funcionamento da função (0, caso funcione e 1 caso contrário).

5.16.2.12 valid_piece()

```
int valid_piece (
            char piece )
```

Verifica se uma peça dada é uma peça de xadrez ou não.

Parâmetros

piece	Peça a ser verificada

Retorna

true Se a peça é uma peça de xadrez
false Se ela não for uma peça de xadrez

5.16.2.13 ValidBoard()

```
int ValidBoard (
              TBoard * board )
```

Verifica se um tabuleiro é válido para ser jogado ou não.

Parâmetros

board	Ponteiro para um tabuleiro.

Retorna

Retorna 1 para caso seja válido ou 0, caso contrário.

5.16.2.14 WhatPiece()

```
char WhatPiece (
              TBoard * board,
             int line,
             int column )
```

Verifica qual peça se encontra na posição dada.

Parâmetros

piece	É um caractere representado por alguma das constantes de peças definidas.

Retorna

Retorna a constante definida BLACK ou WHITE para a cor da peça passada ou -1 se não for uma peça.

5.16.3 Variáveis

5.16.3.1 B_BISHOP

```
const char B_BISHOP = 'B'
```

5.16.3.2 B_HORSE

```
const char B_HORSE = 'N'
```

5.16.3.3 B_KING

```
const char B_KING = 'K'
```

5.16.3.4 B_PAWN

```
const char B_PAWN = 'P'
```

5.16.3.5 B_QUEEN

```
const char B_QUEEN = 'Q'
```

5.16.3.6 B_TOWER

```
const char B_TOWER = 'R'
```

5.16.3.7 BLACK

```
const int BLACK = 0
```

5.16.3.8 BLANK

```
const char BLANK = '\\'
```

5.16.3.9 CHECK

```
const int CHECK = 1
```

5.16.3.10 OUT_OF_RANGE

```
const char OUT_OF_RANGE = '~'
```

5.16.3.11 W_BISHOP

```
const char W_BISHOP = 'b'
```

5.16.3.12 W_HORSE

```
const char W_HORSE = 'n'
```

5.16.3.13 W_KING

```
const char W_KING = 'k'
```

5.16.3.14 W_PAWN

```
const char W_PAWN = 'p'
```

5.16.3.15 W_QUEEN

```
const char W_QUEEN = 'q'
```

5.16.3.16 W_TOWER

```
const char W_TOWER = 'r'
```

5.16.3.17 WHITE

```
const int WHITE = 1
```

5.17 Referência do Arquivo TEST_arv_decisoes.c

```
#include "gtest/gtest.h"
#include "../include/arv_decisoes.h"
```

Funções

- **TEST** (Test_AlocateTree, Verify_Alocation_Tree)
- **TEST** (Test_AlocateNodeTree, Verify_Alocation_Node)
- **TEST** (Test_AddChildNode, Verify_Insertion_Sucess)
- **TEST** (Test_AddChildNode, Verify_Insertion_Failure)
- **TEST** (Test_FreeTreeNodes, Verify_Free_OneNode)
- **TEST** (Test_FreeTreeNodes, Verify_Free_NodewithChild)
- int **main** (int argc, char ∗∗argv)

5.17.1 Funções

5.17.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

5.17.1.2 TEST() [1/6]

```
TEST (
            Test_AlocateTree ,
            Verify_Alocation_Tree  )
```

5.17.1.3 TEST() [2/6]

```
TEST (
            Test_AlocateNodeTree ,
            Verify_Alocation_Node  )
```

5.17.1.4 TEST() [3/6]

```
TEST (
            Test_AddChildNode ,
            Verify_Insertion_Sucess  )
```

5.17.1.5 TEST() [4/6]

```
TEST (
            Test_AddChildNode ,
            Verify_Insertion_Failure  )
```

5.17.1.6 TEST() [5/6]

```
TEST (
            Test_FreeTreeNodes ,
            Verify_Free_OneNode  )
```

5.17.1.7 TEST() [6/6]

```
TEST (
            Test_FreeTreeNodes ,
            Verify_Free_NodewithChild  )
```

5.18 Referência do Arquivo TEST_ia.c

```
#include "gtest/gtest.h"
#include "../include/ia.h"
```

Funções

- **TEST** (Test_CreateMovesTree, Verify_Creation_Tree)
- **TEST** (Test_CreateMovesTree, Verify_Alocation_Tree)
- **TEST** (Test_CreateMovesTree, Verify_Root_Information)
- **TEST** (Test_SortTree, Verify_SortTree)
- **TEST** (Test_SortTree, Verify_InvalidValues_SortTree)
- **TEST** (Test_SortTree, Verify_DontSucicide)
- **TEST** (Test_SortTree, Verify_PiecePreference)
- **TEST** (Test_SortTree, Verify_PieceSacrifice)
- **TEST** (Test_SortTree, Verify_PieceWhiteCheck)
- **TEST** (Test_SortTree, Verify_PieceBlackCheck)
- **TEST** (TEST_Best_Plays, VerifyListCreation)
- int **main** (int argc, char ∗∗argv)

5.18.1 Funções

5.18.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

5.18.1.2 TEST() [1/11]

```
TEST (
            Test_CreateMovesTree ,
            Verify_Creation_Tree  )
```

5.18.1.3 TEST() [2/11]

```
TEST (
            Test_CreateMovesTree ,
            Verify_Alocation_Tree  )
```

5.18.1.4 TEST() [3/11]

```
TEST (
            Test_CreateMovesTree ,
            Verify_Root_Information  )
```

5.18.1.5 TEST() [4/11]

```
TEST (
            Test_SortTree ,
            Verify_SortTree  )
```

5.18.1.6 TEST() [5/11]

```
TEST (
            Test_SortTree ,
            Verify_InvalidValues_SortTree  )
```

5.18.1.7 TEST() [6/11]

```
TEST (
            Test_SortTree ,
            Verify_DontSucicide  )
```

5.18.1.8 TEST() [7/11]

```
TEST (
            Test_SortTree ,
            Verify_PiecePreference  )
```

5.18.1.9 TEST() [8/11]

```
TEST (
            Test_SortTree ,
            Verify_PieceSacrifice  )
```

5.18.1.10 TEST() [9/11]

```
TEST (
            Test_SortTree ,
            Verify_PieceWhiteCheck  )
```

5.18.1.11 TEST() [10/11]

```
TEST (
            Test_SortTree ,
            Verify_PieceBlackCheck  )
```

5.18.1.12 TEST() [11/11]

```
TEST (
            TEST_Best_Plays ,
            VerifyListCreation  )
```

5.19 Referência do Arquivo TEST_in-out.c

```
#include "gtest/gtest.h"
#include "../include/in-out.h"
```

Funções

- **TEST** (Test_Verify_Start_List, Verify_Funcionality)
- **TEST** (Test_Verify_Add_Move, Verify_Invalid_Entries)
- **TEST** (Test_Verify_Add_Move, Verify_Function)
- **TEST** (Test_Verify_Remove_Last, Verify_Invalid_Entries)
- **TEST** (Test_Verify_Remove_Last, Verify_One_Item)
- **TEST** (Test_Verify_Remove_Last, Verify_Two_Itens)
- **TEST** (Test_Verify_Free_List, Verify_Invalid_Entries)
- **TEST** (Test_Verify_Free_List, Verify_Function)
- **TEST** (Test_SaveBoardFile, Verify_Function)
- **TEST** (Test_SaveBoardFile, Verify_InvalidValues)
- **TEST** (Test_RecoverBoardFromFile, Verify_Function)
- **TEST** (Test_RecoverBoardFromFile, Verify_Invalidvalues)
- **TEST** (Test_SavePGNFile, Verify_Function)
- **TEST** (Test_SavePGNFile, Verify_Invalidvalues)
- **TEST** (Test_RecoverMoveListFromFile, Verify_Function)
- **TEST** (Test_RecoverMoveListFromFile, Verify_Invalidvalues)
- int **main** (int argc, char ∗∗argv)

5.19.1 Funções

5.19.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

5.19.1.2 TEST() [1/16]

```
TEST (
            Test_Verify_Start_List ,
            Verify_Funcionality  )
```

5.19.1.3 TEST() [2/16]

```
TEST (
            Test_Verify_Add_Move ,
            Verify_Invalid_Entries  )
```

5.19.1.4 TEST() [3/16]

```
TEST (
            Test_Verify_Add_Move ,
            Verify_Function  )
```

5.19.1.5 TEST() [4/16]

```
TEST (
            Test_Verify_Remove_Last ,
            Verify_Invalid_Entries  )
```

5.19.1.6 TEST() [5/16]

```
TEST (
            Test_Verify_Remove_Last ,
            Verify_One_Item  )
```

5.19.1.7 TEST() [6/16]

```
TEST (
            Test_Verify_Remove_Last ,
            Verify_Two_Itens  )
```

5.19.1.8 TEST() [7/16]

```
TEST (
            Test_Verify_Free_List ,
            Verify_Invalid_Entries  )
```

5.19.1.9 TEST() [8/16]

```
TEST (
            Test_Verify_Free_List ,
            Verify_Function  )
```

5.19.1.10 TEST() [9/16]

```
TEST (
            Test_SaveBoardFile ,
            Verify_Function  )
```

5.19.1.11 TEST() [10/16]

```
TEST (
            Test_SaveBoardFile ,
            Verify_InvalidValues  )
```

5.19.1.12 TEST() [11/16]

```
TEST (
            Test_RecoverBoardFromFile ,
            Verify_Function  )
```

5.19.1.13 TEST() [12/16]

```
TEST (
            Test_RecoverBoardFromFile ,
            Verify_Invalidvalues  )
```

5.19.1.14 TEST() [13/16]

```
TEST (
            Test_SavePGNFile ,
            Verify_Function  )
```

5.19.1.15 TEST() [14/16]

```
TEST (
            Test_SavePGNFile ,
            Verify_Invalidvalues  )
```

5.19.1.16 TEST() [15/16]

```
TEST (
            Test_RecoverMoveListFromFile ,
            Verify_Function  )
```

5.19.1.17 TEST() [16/16]

```
TEST (
            Test_RecoverMoveListFromFile ,
            Verify_Invalidvalues  )
```

5.20 Referência do Arquivo TEST_list.c

```
#include "gtest/gtest.h"
#include "../include/list_of_moves.h"
```

Funções

- **TEST** (Test_CreateListOfMoves, Verify_Correct_Allocation)
- **TEST** (Test_InsertMove, Verify_Invalid_Entries)
- **TEST** (Test_InsertMove, Verify_Valid_Entries)
- **TEST** (Test_InsertMove, Verify_Correct_Insert)
- **TEST** (Test_DeleteListOfMoves, Verify_Invalid_Entrie)
- **TEST** (Test_DeleteListOfMoves, Verify_Valid_Entrie)
- **TEST** (Test_SearchListOfMoves, Verify_Succesfull_Search)
- **TEST** (Test_SearchListOfMoves, Verify_Failure_Search)
- int **main** (int argc, char ∗∗argv)

5.20.1 Funções

5.20.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

5.20.1.2 TEST() [1/8]

```
TEST (
            Test_CreateListOfMoves ,
            Verify_Correct_Allocation  )
```

5.20.1.3 TEST() [2/8]

```
TEST (
            Test_InsertMove ,
            Verify_Invalid_Entries  )
```

5.20.1.4 TEST() [3/8]

```
TEST (
            Test_InsertMove ,
            Verify_Valid_Entries  )
```

5.20.1.5 TEST() [4/8]

```
TEST (
            Test_InsertMove ,
            Verify_Correct_Insert  )
```

5.20.1.6 TEST() [5/8]

```
TEST (
            Test_DeleteListOfMoves ,
            Verify_Invalid_Entrie  )
```

5.20.1.7 TEST() [6/8]

```
TEST (
            Test_DeleteListOfMoves ,
            Verify_Valid_Entrie  )
```

5.20.1.8 TEST() [7/8]

```
TEST (
            Test_SearchListOfMoves ,
            Verify_Succesfull_Search  )
```

5.20.1.9 TEST() [8/8]

```
TEST (
            Test_SearchListOfMoves ,
            Verify_Failure_Search  )
```

5.21 Referência do Arquivo TEST_logica.c

```
#include "gtest/gtest.h"
#include "../include/tabuleiro.h"
#include "../include/logica.h"
```

Funções

- **TEST** (Algebraic_verification, Named_movement)
- **TEST** (Algebraic_verification, Castling_or_winner)
- **TEST** (Algebraic_verification, Queenside_castling)
- **TEST** (Algebraic_verification, Draw)
- **TEST** (Algebraic_verification, Unnamed_movement)
- **TEST** (Algebraic_translation, Named_movement)
- **TEST** (Algebraic_translation, Unnamed_movement)
- **TEST** (Test_WhitePawnMovements, Verify_NULL_Variables)
- **TEST** (Test_BlackPawnMovements, Verify_NULL_Variables)
- **TEST** (Test_HorseMovements, Verify_NULL_Variables)
- **TEST** (Test_TowerMovements, Verify_NULL_Variables)
- **TEST** (Test_BishopMovements, Verify_NULL_Variables)
- **TEST** (Test_QueenMovements, Verify_NULL_Variables)
- **TEST** (Test_KingMovements, Verify_NULL_Variables)
- **TEST** (Test_WhitePawnMovements, Verify_Movements_EmptyBoard)
- **TEST** (Test_BlackPawnMovements, Verify_Movements_EmptyBoard)
- **TEST** (Test_HorseMovements, Verify_Movements_EmptyBoard)
- **TEST** (Test_TowerMovements, Verify_Movements_EmptyBoard)
- **TEST** (Test_BishopMovements, Verify_Movements_EmptyBoard)
- **TEST** (Test_QueenMovements, Verify_Movements_EmptyBoard)
- **TEST** (Test_KingMovements, Verify_Movements_EmptyBoard)
- **TEST** (Test_WhitePawnMovements, Verify_Movements_RivalPieces)
- **TEST** (Test_BlackPawnMovements, Verify_Movements_RivalPieces)
- **TEST** (Test_HorseMovements, Verify_Movements_RivalPieces)
- **TEST** (Test_TowerMovements, Verify_Movements_RivalPieces)
- **TEST** (Test_BishopMovements, Verify_Movements_RivalPieces)
- **TEST** (Test_QueenMovements, Verify_Movements_RivalPieces)
- **TEST** (Test_KingMovements, Verify_Movements_RivalPieces)
- **TEST** (Test_KingMovements, Verify_Addict_Roque_Movement)
- **TEST** (Test_AllMovements, Verify_NULL_Board)
- **TEST** (Test_AllMovements, Verify_Movements_EmptyBoard)
- **TEST** (Test_AllMovements, Verify_Movements_StandardBoard)
- **TEST** (Test_VerifyValidMovement, Veirfy_Invalid_Entries)
- **TEST** (Test_VerifyValidMovement, Verify_Valid_Movements)
- **TEST** (Test_VerifyValidMovement, Verify_Invalid_Movements)
- **TEST** (Test_VerifyCheck, Verify_NULL_Variables)
- **TEST** (Test_VerifyCheck, Verify_RealBlackCheck)
- **TEST** (Test_VerifyCheck, Verify_RealWhiteCheck)
- **TEST** (Test_VerifyCheck, Verify_FakeBlackCheck)
- **TEST** (Test_VerifyCheck, Verify_FakeWhiteCheck)
- **TEST** (Test_VerifyCheck, Verify_ChangeBlackCheck)
- **TEST** (Test_VerifyCheck, Verify_ChangeWhiteCheck)
- **TEST** (Test_VerifyCheckMate, Veirfy_Invalid_Entries)
- **TEST** (Test_VerifyCheckMate, Verify_Correct_CheckMate)
- **TEST** (Test_VerifyCheckMate, Verify_Not_CheckMate)
- **TEST** (MovementTranslation, VariusMovements)
- int **main** (int argc, char ∗∗argv)

5.21.1 Funções

5.21.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

5.21.1.2 TEST() [1/46]

```
TEST (
            Algebraic_verification ,
            Named_movement  )
```

5.21.1.3 TEST() [2/46]

```
TEST (
            Algebraic_verification ,
            Castling_or_winner  )
```

5.21.1.4 TEST() [3/46]

```
TEST (
            Algebraic_verification ,
            Queenside_castling  )
```

5.21.1.5 TEST() [4/46]

```
TEST (
            Algebraic_verification ,
            Draw  )
```

5.21.1.6 TEST() [5/46]

```
TEST (
            Algebraic_verification ,
            Unnamed_movement  )
```

5.21.1.7 TEST() [6/46]

```
TEST (
            Algebraic_translation ,
            Named_movement  )
```

5.21.1.8 TEST() [7/46]

```
TEST (
            Algebraic_translation ,
            Unnamed_movement  )
```

5.21.1.9 TEST() [8/46]

```
TEST (
            Test_WhitePawnMovements ,
            Verify_NULL_Variables  )
```

5.21.1.10 TEST() [9/46]

```
TEST (
            Test_BlackPawnMovements ,
            Verify_NULL_Variables  )
```

5.21.1.11 TEST() [10/46]

```
TEST (
            Test_HorseMovements ,
            Verify_NULL_Variables  )
```

5.21.1.12 TEST() [11/46]

```
TEST (
            Test_TowerMovements ,
            Verify_NULL_Variables  )
```

5.21.1.13 TEST() [12/46]

```
TEST (
            Test_BishopMovements ,
            Verify_NULL_Variables  )
```

5.21.1.14 TEST() [13/46]

```
TEST (
            Test_QueenMovements ,
            Verify_NULL_Variables  )
```

5.21.1.15 TEST() [14/46]

```
TEST (
            Test_KingMovements ,
            Verify_NULL_Variables  )
```

5.21.1.16 TEST() [15/46]

```
TEST (
            Test_WhitePawnMovements ,
            Verify_Movements_EmptyBoard  )
```

5.21.1.17 TEST() [16/46]

```
TEST (
            Test_BlackPawnMovements ,
            Verify_Movements_EmptyBoard  )
```

5.21.1.18 TEST() [17/46]

```
TEST (
            Test_HorseMovements ,
            Verify_Movements_EmptyBoard  )
```

5.21.1.19 TEST() [18/46]

```
TEST (
            Test_TowerMovements ,
            Verify_Movements_EmptyBoard  )
```

5.21.1.20 TEST() [19/46]

```
TEST (
            Test_BishopMovements ,
            Verify_Movements_EmptyBoard  )
```

5.21.1.21 TEST() [20/46]

```
TEST (
            Test_QueenMovements ,
            Verify_Movements_EmptyBoard  )
```

5.21.1.22 TEST() [21/46]

```
TEST (
            Test_KingMovements ,
            Verify_Movements_EmptyBoard  )
```

5.21.1.23 TEST() [22/46]

```
TEST (
            Test_WhitePawnMovements ,
            Verify_Movements_RivalPieces  )
```

5.21.1.24 TEST() [23/46]

```
TEST (
            Test_BlackPawnMovements ,
            Verify_Movements_RivalPieces  )
```

5.21.1.25 TEST() [24/46]

```
TEST (
            Test_HorseMovements ,
            Verify_Movements_RivalPieces  )
```

5.21.1.26 TEST() [25/46]

```
TEST (
            Test_TowerMovements ,
            Verify_Movements_RivalPieces  )
```

5.21.1.27 TEST() [26/46]

```
TEST (
            Test_BishopMovements ,
            Verify_Movements_RivalPieces  )
```

5.21.1.28 TEST() [27/46]

```
TEST (
            Test_QueenMovements ,
            Verify_Movements_RivalPieces  )
```

5.21.1.29 TEST() [28/46]

```
TEST (
            Test_KingMovements ,
            Verify_Movements_RivalPieces  )
```

5.21.1.30 TEST() [29/46]

```
TEST (
            Test_KingMovements ,
            Verify_Addict_Roque_Movement  )
```

5.21.1.31 TEST() [30/46]

```
TEST (
            Test_AllMovements ,
            Verify_NULL_Board  )
```

5.21.1.32 TEST() [31/46]

```
TEST (
            Test_AllMovements ,
            Verify_Movements_EmptyBoard  )
```

5.21.1.33 TEST() [32/46]

```
TEST (
            Test_AllMovements ,
            Verify_Movements_StandardBoard  )
```

5.21.1.34 TEST() [33/46]

```
TEST (
            Test_VerifyValidMovement ,
            Veirfy_Invalid_Entries  )
```

5.21.1.35 TEST() [34/46]

```
TEST (
            Test_VerifyValidMovement ,
            Verify_Valid_Movements  )
```

5.21.1.36 TEST() [35/46]

```
TEST (
            Test_VerifyValidMovement ,
            Verify_Invalid_Movements  )
```

5.21.1.37 TEST() [36/46]

```
TEST (
            Test_VerifyCheck ,
            Verify_NULL_Variables  )
```

5.21.1.38 TEST() [37/46]

```
TEST (
            Test_VerifyCheck ,
            Verify_RealBlackCheck  )
```

5.21.1.39 TEST() [38/46]

```
TEST (
            Test_VerifyCheck ,
            Verify_RealWhiteCheck  )
```

5.21.1.40 TEST() [39/46]

```
TEST (
            Test_VerifyCheck ,
            Verify_FakeBlackCheck  )
```

5.21.1.41 TEST() [40/46]

```
TEST (
            Test_VerifyCheck ,
            Verify_FakeWhiteCheck  )
```

5.21.1.42 TEST() [41/46]

```
TEST (
            Test_VerifyCheck ,
            Verify_ChangeBlackCheck  )
```

5.21.1.43 TEST() [42/46]

```
TEST (
            Test_VerifyCheck ,
            Verify_ChangeWhiteCheck  )
```

5.21.1.44 TEST() [43/46]

```
TEST (
            Test_VerifyCheckMate ,
            Veirfy_Invalid_Entries  )
```

5.21.1.45 TEST() [44/46]

```
TEST (
            Test_VerifyCheckMate ,
            Verify_Correct_CheckMate  )
```

5.21.1.46 TEST() [45/46]

```
TEST (
            Test_VerifyCheckMate ,
            Verify_Not_CheckMate  )
```

5.21.1.47 TEST() [46/46]

```
TEST (
            MovementTranslation ,
            VariusMovements  )
```

5.22 Referência do Arquivo TEST_tabuleiro.c

```
#include "gtest/gtest.h"
#include "../include/tabuleiro.h"
```

Funções

- **TEST** (Test_Verify_Empty_Board, Verify_If_Empty)
- **TEST** (Test_Verify_Empty_Board, Verify_NULL_Variables)
- **TEST** (Test_Verify_Standard_Board, Verify_If_Correct_Positions)
- **TEST** (Test_Verify_Standard_Board, Verify_NULL_Variables)
- **TEST** (Test_Color_Piece, Verify_Correct_Color)
- **TEST** (Test_What_Piece_in_Position, Verify_Empty_Boards)
- **TEST** (Test_What_Piece_in_Position, Verify_Standard_Boards)
- **TEST** (Test_What_Piece_in_Position, Verify_Out_of_Range)
- **TEST** (Test_Get_Value_of_Piece, Verify_Every_Piece)
- **TEST** (Test_Get_Value_of_Piece, Verify_Non_Pieces)
- **TEST** (Test_Remove_Piece, Verify_Remotion)
- **TEST** (Test_Remove_Piece, Verify_Invalid_Entries)
- **TEST** (Test_Insert_Piece, Verify_Invalid_Entries)
- **TEST** (Test_Insert_Piece, Verify_Correct_Insertion)
- **TEST** (Test_Valid_Board, Verify_Standard)
- **TEST** (Test_Valid_Board, Verify_No_Double_Pieces)
- **TEST** (Test_Valid_Board, Verify_Empty_and_One_King)
- **TEST** (Test_Valid_Board, Verify_Two_Kings)
- **TEST** (Test_Valid_Board, Verify_Queens)
- **TEST** (Test_Valid_Board, Verify_Horses)
- **TEST** (Test_Valid_Board, Verify_Bishops)
- **TEST** (Test_Valid_Board, Verify_Towers)
- **TEST** (Test_Valid_Board, Verify_Pawns)
- **TEST** (Test_Valid_Board, Verify_NULL)
- **TEST** (Test_Valid_Board, Verify_Equivalance)
- **TEST** (Test_Change_Piece, Verify_Invalid_Entries)
- **TEST** (Test_Change_Piece, Verify_Valid_Entries)
- **TEST** (Test_Move_Piece, Verify_Invalid_Entries)
- **TEST** (Test_Move_Piece, Test_Empty_Space)
- **TEST** (Test_Move_Piece, Test_Full_Space)
- **TEST** (Test_Move_Piece, Test_Roque)
- **TEST** (Test_alocate, Alocate_new_board)
- **TEST** (Copy_boards, CopyEmptyBoards)
- **TEST** (Copy_boards, CopySTDBoard)
- **TEST** (Copy_boards, NULL_Board)
- **TEST** (Valid_piece, Letters)
- **TEST** (Valid_piece, Numbers)
- **TEST** (Valid_piece, Other_chars)
- **TEST** (MinimumChessBoard, NULL_Board)
- **TEST** (MinimumChessBoard, EmptyBoard)
- **TEST** (MinimumChessBoard, BoardWithKings)
- **TEST** (MinimumChessBoard, BoardWithOtherPieces)
- int **main** (int argc, char ∗∗argv)

5.22.1 Funções

5.22.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

5.22.1.2 TEST() [1/42]

```
TEST (
            Test_Verify_Empty_Board ,
            Verify_If_Empty  )
```

5.22.1.3 TEST() [2/42]

```
TEST (
            Test_Verify_Empty_Board ,
            Verify_NULL_Variables  )
```

5.22.1.4 TEST() [3/42]

```
TEST (
            Test_Verify_Standard_Board ,
            Verify_If_Correct_Positions  )
```

5.22.1.5 TEST() [4/42]

```
TEST (
            Test_Verify_Standard_Board ,
            Verify_NULL_Variables  )
```

5.22.1.6 TEST() [5/42]

```
TEST (
            Test_Color_Piece ,
            Verify_Correct_Color  )
```

5.22.1.7 TEST() [6/42]

```
TEST (
            Test_What_Piece_in_Position ,
            Verify_Empty_Boards  )
```

5.22.1.8 TEST() [7/42]

```
TEST (
            Test_What_Piece_in_Position ,
            Verify_Standard_Boards  )
```

5.22.1.9 TEST() [8/42]

```
TEST (
            Test_What_Piece_in_Position ,
            Verify_Out_of_Range  )
```

5.22.1.10 TEST() [9/42]

```
TEST (
            Test_Get_Value_of_Piece ,
            Verify_Every_Piece  )
```

5.22.1.11 TEST() [10/42]

```
TEST (
            Test_Get_Value_of_Piece ,
            Verify_Non_Pieces  )
```

5.22.1.12 TEST() [11/42]

```
TEST (
            Test_Remove_Piece ,
            Verify_Remotion  )
```

5.22.1.13 TEST() [12/42]

```
TEST (
            Test_Remove_Piece ,
            Verify_Invalid_Entries  )
```

5.22.1.14 TEST() [13/42]

```
TEST (
            Test_Insert_Piece ,
            Verify_Invalid_Entries  )
```

5.22.1.15 TEST() [14/42]

```
TEST (
            Test_Insert_Piece ,
            Verify_Correct_Insertion  )
```

5.22.1.16 TEST() [15/42]

```
TEST (
            Test_Valid_Board ,
            Verify_Standard  )
```

5.22.1.17 TEST() [16/42]

```
TEST (
            Test_Valid_Board ,
            Verify_No_Double_Pieces  )
```

5.22.1.18 TEST() [17/42]

```
TEST (
            Test_Valid_Board ,
            Verify_Empty_and_One_King  )
```

5.22.1.19 TEST() [18/42]

```
TEST (
            Test_Valid_Board ,
            Verify_Two_Kings  )
```

5.22.1.20 TEST() [19/42]

```
TEST (
            Test_Valid_Board ,
            Verify_Queens  )
```

5.22.1.21 TEST() [20/42]

```
TEST (
            Test_Valid_Board ,
            Verify_Horses  )
```

5.22.1.22 TEST() [21/42]

```
TEST (
            Test_Valid_Board ,
            Verify_Bishops  )
```

5.22.1.23 TEST() [22/42]

```
TEST (
            Test_Valid_Board ,
            Verify_Towers  )
```

5.22.1.24 TEST() [23/42]

```
TEST (
            Test_Valid_Board ,
            Verify_Pawns  )
```

5.22.1.25 TEST() [24/42]

```
TEST (
            Test_Valid_Board ,
            Verify_NULL  )
```

5.22.1.26 TEST() [25/42]

```
TEST (
            Test_Valid_Board ,
            Verify_Equivalance  )
```

5.22.1.27 TEST() [26/42]

```
TEST (
            Test_Change_Piece ,
            Verify_Invalid_Entries  )
```

5.22.1.28 TEST() [27/42]

```
TEST (
            Test_Change_Piece ,
            Verify_Valid_Entries  )
```

5.22.1.29 TEST() [28/42]

```
TEST (
            Test_Move_Piece ,
            Verify_Invalid_Entries  )
```

5.22.1.30 TEST() [29/42]

```
TEST (
            Test_Move_Piece ,
            Test_Empty_Space  )
```

5.22.1.31 TEST() [30/42]

```
TEST (
            Test_Move_Piece ,
            Test_Full_Space  )
```

5.22.1.32 TEST() [31/42]

```
TEST (
            Test_Move_Piece ,
            Test_Roque  )
```

5.22.1.33 TEST() [32/42]

```
TEST (
            Test_alocate ,
            Alocate_new_board  )
```

5.22.1.34 TEST() [33/42]

```
TEST (
            Copy_boards ,
            CopyEmptyBoards  )
```

5.22.1.35 TEST() [34/42]

```
TEST (
            Copy_boards ,
            CopySTDBoard  )
```

5.22.1.36 TEST() [35/42]

```
TEST (
            Copy_boards ,
            NULL_Board  )
```

5.22.1.37 TEST() [36/42]

```
TEST (
            Valid_piece ,
            Letters  )
```

5.22.1.38 TEST() [37/42]

```
TEST (
            Valid_piece ,
            Numbers  )
```

5.22.1.39 TEST() [38/42]

```
TEST (
            Valid_piece ,
            Other_chars  )
```

5.22.1.40 TEST() [39/42]

```
TEST (
            MinimumChessBoard ,
            NULL_Board  )
```

5.22.1.41 TEST() [40/42]

```
TEST (
            MinimumChessBoard ,
            EmptyBoard  )
```

5.22.1.42 TEST() [41/42]

```
TEST (
            MinimumChessBoard ,
            BoardWithKings  )
```

5.22.1.43 TEST() [42/42]

```
TEST (
            MinimumChessBoard ,
            BoardWithOtherPieces  )
```