# Tetris - PS Edition

Gerado por Doxygen 1.8.11

Quinta, 7 de Julho de 2016 22:26:24

# Sumário

# 1 Índice das Estruturas de Dados

## 1.1 Estruturas de Dados

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



# 2 Índice dos Arquivos

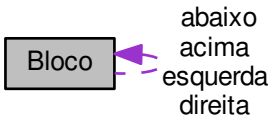
## 2.1 Lista de Arquivos

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

# 3 Estruturas

## 3.1 Referência da Estrutura Bloco

Diagrama de colaboração para Bloco:




### Campos de Dados

- char bolinha
- unsigned short int cor
- int pos_x
- int pos_y
- unsigned short int move
- struct Bloco ∗ esquerda
- struct Bloco ∗ direita
- struct Bloco ∗ abaixo
- struct Bloco ∗ acima

### 3.1.1 Campos

#### 3.1.1.1 struct Bloco∗ Bloco::abaixo

Ponteiro para vizinho abaixo.

#### 3.1.1.2 struct Bloco* Bloco::acima

Ponteiro para vizinho acima.

#### 3.1.1.3 char Bloco::bolinha

Caractere atual da peça.

#### 3.1.1.4 unsigned short int Bloco::cor

Cor da peça.

#### 3.1.1.5 struct Bloco* Bloco::direita

Ponteiro para vizinho à direita.

#### 3.1.1.6 struct Bloco* Bloco::esquerda

Ponteiro para vizinho à esquerda.

#### 3.1.1.7 unsigned short int Bloco::move

Valor booleano que indica se o bloco está em movimento ou não.

#### 3.1.1.8 int Bloco::pos_x

Coordenada cartesiana horizontal do bloco.

#### 3.1.1.9 int Bloco::pos_y

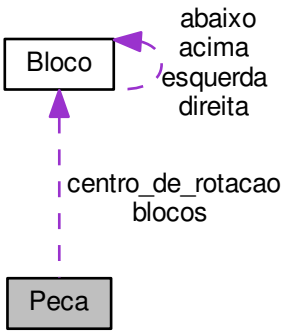
Coordenada cartesiana vertical do bloco.

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

- bloco.h

## 3.2 Referência da Estrutura Peca

Diagrama de colaboração para Peca:



#### Campos de Dados

- int tamanho
- unsigned short int cor_peca
- unsigned short int move_peca
- float velocidade
- tp_peca tipo
- bloco * centro_de_rotacao
- bloco ** blocos

### 3.2.1 Campos

#### 3.2.1.1 bloco** Peca::blocos

Referência para blocos na tela.

#### 3.2.1.2 bloco* Peca::centro_de_rotacao

Ponteiro para o bloco de centro de rotação.

#### 3.2.1.3 unsigned short int Peca::cor_peca

Cor da peça.

#### 3.2.1.4 unsigned short int Peca::move_peca

Booleano que checa se a peça está em movimento ou não.

#### 3.2.1.5 int Peca::tamanho

Tamanho da peça.

#### 3.2.1.6 tp_peca Peca::tipo

Qual o formato da peça.

#### 3.2.1.7 float Peca::velocidade

Velocidade na qual a peça está caindo.

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

- pecas.h

## 3.3 Referência da Estrutura Placar

### Campos de Dados

- char jogadores [5][4]
- int pontuacoes [5]
- int tempos_m [5]
- int tempos_s [5]
- int anos [5]
- int meses [5]
- int dias [5]
- int contador_jogadores
- FILE * arquivo
- char jogador [3]
- int pontuacao
- int tempo_m
- int tempo_s

### 3.3.1 Campos

#### 3.3.1.1 int Placar::anos[5]

Vetor de anos dos jogadores.

#### 3.3.1.2 FILE* Placar::arquivo

Ponteiro para o arquivo usado para abrir o placar (pontuacao.txt) para leitura e escrita.

#### 3.3.1.3 int Placar::contador_jogadores

Contador de jogadores presentes no placar (máximo 5).

#### 3.3.1.4 int Placar::dias[5]

Vetor de dias dos jogadores no placar.

#### 3.3.1.5 char Placar::jogador[3]

Nome do jogador atual.

#### 3.3.1.6 char Placar::jogadores[5][4]

Vetor de jogadores.

#### 3.3.1.7 int Placar::meses[5]

Vetor de meses dos jogadores no placar.

#### 3.3.1.8 int Placar::pontuacao

Pontuacao do jogador atual.

#### 3.3.1.9 int Placar::pontuacoes[5]

Vetor de pontuação dos jogadores no placar.

#### 3.3.1.10 int Placar::tempo_m

Tempo em minutos do jogador atual.

#### 3.3.1.11 int Placar::tempo_s

Tempo em segundos do jogador atual.

#### 3.3.1.12 int Placar::tempos_m[5]

Vetor de tempo em minutos dos jogadores.

#### 3.3.1.13 int Placar::tempos_s[5]

Vetor de tempo em segundos dos jogadores.

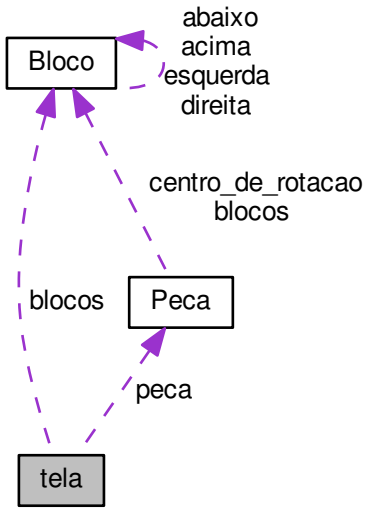
A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- placar.h

## 3.4 Referência da Estrutura tela

```
#include <tela.h>
```

Diagrama de colaboração para tela:



**Campos de Dados**

- char jogador [3]
- int letra
- int estado
- int pontos
- int tempo_m
- int tempo_s
- int comprimento
- int largura
- WINDOW ∗ janela
- struct Peca ∗ peca
- bloco blocos [ ]

### 3.4.1 Descrição Detalhada

/struct Define a tela do jogo.

### 3.4.2 Campos

#### 3.4.2.1 bloco tela::blocos[ ]

Matriz dos blocos na tela.

#### 3.4.2.2 int tela::comprimento

Comprimento da tela.

#### 3.4.2.3 int tela::estado

Estado atual do jogo.

#### 3.4.2.4 WINDOW∗ tela::janela

Ponteiro para a janela do jogo.

#### 3.4.2.5 char tela::jogador[3]

Nome do atual jogador.

#### 3.4.2.6 int tela::largura

Largura da tela.

#### 3.4.2.7 int tela::letra

Auxiliar na escolha de nome do jogador.

#### 3.4.2.8 struct Peca* tela::peca

Ponteiro para a peça em movimento.

#### 3.4.2.9 int tela::pontos

Pontuação do jogador.

#### 3.4.2.10 int tela::tempo_m

Tempo de execução em minutos.

#### 3.4.2.11 int tela::tempo_s

Tempo de execução em segundos.

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

- tela.h

## 3.5 Referência da Estrutura timeb

### Campos de Dados

- time_t **time**
- unsigned short int **millitm**
- short int **timezone**
- short int **dstflag**

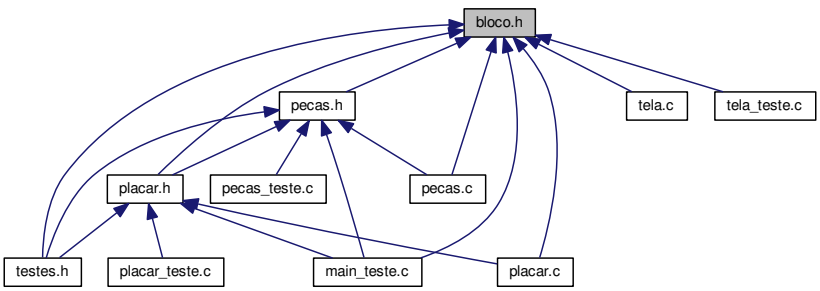
A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- timeb.h

# 4 Arquivos

## 4.1 Referência do Arquivo bloco.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



#### Estruturas de Dados

- struct Bloco

#### Definições e Macros

- #define **COMPRIMENTO** 15
- #define **LARGURA** 25

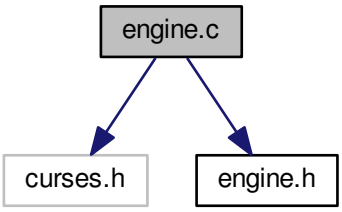
#### Definições de Tipos

- typedef struct Bloco **bloco**

## 4.2 Referência do Arquivo engine.c

```
#include <curses.h>
#include "engine.h"
```

Gráfico de dependência de inclusões para engine.c:




#### Funções

- void inicia_ncurses ()
- void finaliza_ncurses ()
- int pega_input (int input)

### 4.2.1 Funções

#### 4.2.1.1 void finaliza_ncurses ( )

Finaliza o modo ncurses.

#### 4.2.1.2 void inicia_ncurses ( )

Inicializa o modo ncurses e determina as funcionalidades dele que serão usadas.

#### 4.2.1.3 int pega_input ( int *input* )

Determina como interpretar a entrada do teclado.

**Parâmetros**

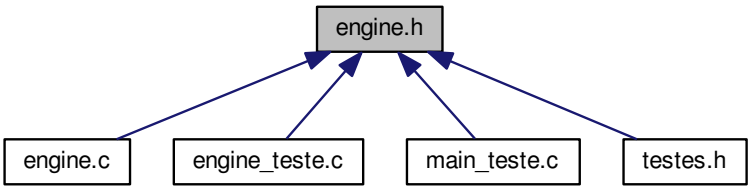
| | |
|---|---|
| *input* | Entrada. |

**Retorna**

Saída convertida.

## 4.3 Referência do Arquivo engine.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:




**Definições e Macros**

- #define **ENGINE_EXT** extern

**Funções**

- ENGINE_EXT void inicia_ncurses ()
- ENGINE_EXT void finaliza_ncurses ()
- ENGINE_EXT int pega_input (int input)

### 4.3.1 Funções

#### 4.3.1.1 ENGINE_EXT void finaliza_ncurses ( )

Finaliza o modo ncurses.

#### 4.3.1.2 ENGINE_EXT void inicia_ncurses ( )

Inicializa o modo ncurses e determina as funcionalidades dele que serão usadas.

#### 4.3.1.3 ENGINE_EXT int pega_input ( int *input* )

Determina como interpretar a entrada do teclado.

**Parâmetros**

| | |
|---|---|
| *input* | Entrada. |

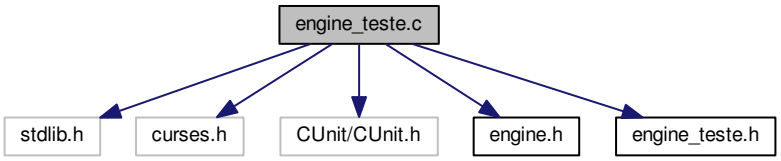
**Retorna**

Saída convertida.

## 4.4 Referência do Arquivo engine_teste.c

```
#include <stdlib.h>
#include <curses.h>
#include <CUnit/CUnit.h>
#include "engine.h"
#include "engine_teste.h"
```

Gráfico de dependência de inclusões para engine_teste.c:




**Funções**

- void test_ncurses ()
- void test_pega_input ()

### 4.4.1 Funções

#### 4.4.1.1 void test_ncurses ( )

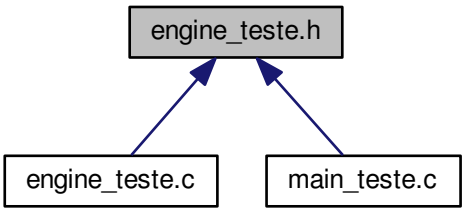
Verifica se é possível inicializar e finalizar as funções do curses corretamente.

#### 4.4.1.2 void test_pega_input ( )

Testa a função peca_input(). Garante que dentro da faixa de caracteres ascii que não são caracteres de controle são interpretados corretamente. Além disso, verifica alguns caracteres especiais do curses que serão usados no programa.

## 4.5 Referência do Arquivo engine_teste.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:




### Definições e Macros

- #define **ENGINE_TESTE_EXT** extern

### Funções

- ENGINE_TESTE_EXT void test_ncurses ()
- ENGINE_TESTE_EXT void test_pega_input ()

### 4.5.1 Funções

#### 4.5.1.1 ENGINE_TESTE_EXT void test_ncurses ( )

Verifica se é possível inicializar e finalizar as funções do curses corretamente.

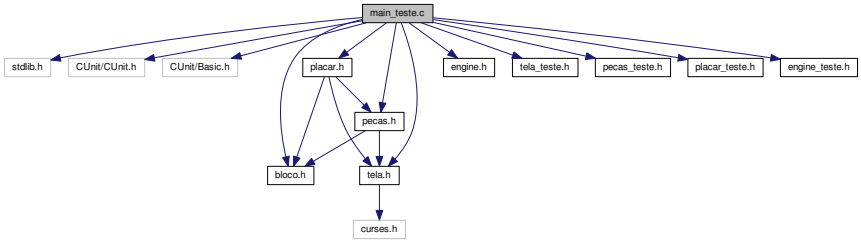
#### 4.5.1.2 ENGINE_TESTE_EXT void test_pega_input ( )

Testa a função peca_input(). Garante que dentro da faixa de caracteres ascii que não são caracteres de controle são interpretados corretamente. Além disso, verifica alguns caracteres especiais do curses que serão usados no programa.

## 4.6 Referência do Arquivo main_teste.c

```
#include <stdlib.h>
#include "CUnit/CUnit.h"
#include "CUnit/Basic.h"
#include "bloco.h"
#include "tela.h"
#include "pecas.h"
#include "placar.h"
#include "engine.h"
#include "tela_teste.h"
#include "pecas_teste.h"
#include "placar_teste.h"
#include "engine_teste.h"
```

Gráfico de dependência de inclusões para main_teste.c:



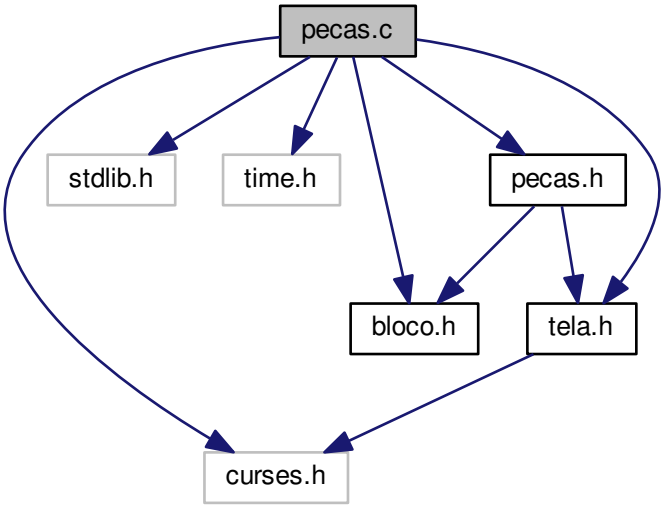

### Funções

- int **main** ()

## 4.7 Referência do Arquivo pecas.c

```
#include <curses.h>
#include <stdlib.h>
#include <time.h>
#include "bloco.h"
#include "tela.h"
#include "pecas.h"
```

Gráfico de dependência de inclusões para pecas.c:




### Funções

- void nova_peca (Tela *tela, int teste, int valor)
- peca * cria_peca_I (Tela *tela)
- peca * cria_peca_Z (Tela *tela)
- peca * cria_peca_T (Tela *tela)
- peca * cria_peca_O (Tela *tela)
- peca * cria_peca_L (Tela *tela)
- void move_peca_x (peca *p, int x)
- void move_peca_y (peca *p, int y)
- void rotaciona_peca (peca *peca)
- void speed_up (peca *peca, int y)
- void libera_peca (peca *p)

### Variáveis

- unsigned short int cor_nova_peca = 4
- unsigned short int speed_ups = 0

## 4.7.1 Funções

### 4.7.1.1 peca* cria_peca_I ( Tela * *tela* )

Cria uma peça do tipo I na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para tela. |

**Retorna**

Ponteiro para a peça.

#### 4.7.1.2 peca* cria_peca_L ( Tela * *tela* )

Cria peça L na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para a tela. |

**Retorna**

Ponteiro para peça.

#### 4.7.1.3 peca* cria_peca_O ( Tela * *tela* )

Cria peça O na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para a tela de jogo. |

**Retorna**

Ponteiro para a peça.

#### 4.7.1.4 peca* cria_peca_T ( Tela * *tela* )

Cria peça T na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para tela de jogo. |

**Retorna**

Ponteiro para a peça.

#### 4.7.1.5 peca* cria_peca_Z ( Tela * *tela* )

Cria uma peça do tipo Z na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para a tela de jogo. |

**Retorna**

Ponteiro para a peça.

#### 4.7.1.6 void libera_peca ( peca * p )

Libera a memória alocada para a peça.

**Parâmetros**

| | |
|---|---|
| *p* | Ponteiro para a peça a ser liberada. |

#### 4.7.1.7 void move_peca_x ( peca * p, int x )

Movimenta a peça no eixo x, ou seja no sentido horizontal.

**Parâmetros**

| | |
|---|---|
| *p* | A peça a ser movimentada. |
| *x* | Indica a direção do movimento. Se positivo, para a direita. Se negativo, para a esquerda. |

#### 4.7.1.8 void move_peca_y ( peca * p, int y )

Movimenta a peça no eixo y, ou seja, na direção vertical.

**Parâmetros**

| | |
|---|---|
| *p* | Peça a ser movida. |
| *y* | Sempre deve ser positivo, pois a peça só pode se movimentar para baixo. |

#### 4.7.1.9 void nova_peca ( Tela * tela, int teste, int valor )

Cria nova peça na tela do jogo. O tipo de peça a ser criado é randomizado. A coloração das peças se dá de forma cíclica.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para tela de jogo. |

<Indica o tipo da peça

#### 4.7.1.10 void rotaciona_peca ( peca * peca )

Rotaciona a peça no sentido horário em torno de seu centro de rotação.

**Parâmetros**

| | |
|---|---|
| *peca* | Ponteiro para a peça. |

#### 4.7.1.11 void speed_up ( peca * *peca,* int *y* )

Dobra a velocidade da peça. Pode ser chamada com sucesso no máximo 5 vezes para a mesma peça.

**Parâmetros**

| | |
|---|---|
| *peca* | Ponteiro para peça. |
| *y* | Inteiro qualquer. |

### 4.7.2 Variáveis

#### 4.7.2.1 unsigned short int cor_nova_peca = 4

Par de cores das peças variam entre 4 e 7.

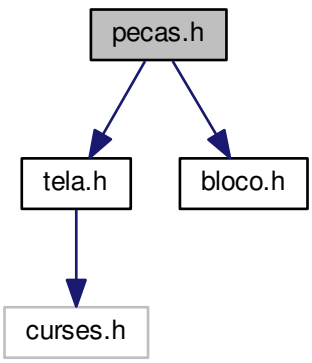
#### 4.7.2.2 unsigned short int speed_ups = 0

Indica quantas chamadas bem-sucedidas para a função speed_up foi feita para uma peça. Valor máximo de 5

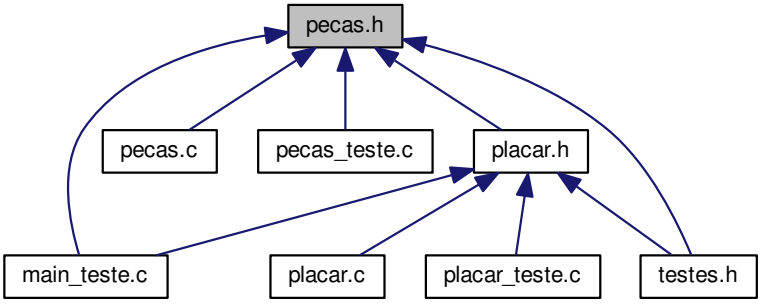
## 4.8 Referência do Arquivo pecas.h

```
#include "tela.h"
#include "bloco.h"
```

Gráfico de dependência de inclusões para pecas.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:




#### Estruturas de Dados

- struct Peca

#### Definições e Macros

- #define **PECAS_EXT** extern

#### Definições de Tipos

- typedef enum Tipo_Peca **tp_peca**
- typedef struct Peca **peca**

#### Enumerações

- enum Tipo_Peca {
  **Tipo_I**, **Tipo_Z**, **Tipo_T**, **Tipo_O**,
  **Tipo_L** }

#### Funções

- PECAS_EXT void nova_peca (Tela ∗tela, int teste, int valor)
- PECAS_EXT void move_peca_x (peca ∗peca, int x)
- PECAS_EXT void move_peca_y (peca ∗peca, int y)
- PECAS_EXT void rotaciona_peca (peca ∗peca)
- PECAS_EXT void speed_up (peca ∗peca, int y)
- PECAS_EXT void libera_peca (peca ∗p)
- PECAS_EXT peca ∗ cria_peca_I (Tela ∗tela)
- PECAS_EXT peca ∗ cria_peca_Z (Tela ∗tela)
- PECAS_EXT peca ∗ cria_peca_T (Tela ∗tela)
- PECAS_EXT peca ∗ cria_peca_O (Tela ∗tela)
- PECAS_EXT peca ∗ cria_peca_L (Tela ∗tela)

#### 4.8.1 Enumerações

##### 4.8.1.1 enum Tipo_Peca

Indica qual é o formato da peça a ser mostrada na tela.

#### 4.8.2 Funções

##### 4.8.2.1 PECAS_EXT peca* cria_peca_I ( Tela * *tela* )

Cria uma peça do tipo I na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para tela. |

**Retorna**

Ponteiro para a peça.

##### 4.8.2.2 PECAS_EXT peca* cria_peca_L ( Tela * *tela* )

Cria peça L na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para a tela. |

**Retorna**

Ponteiro para peça.

##### 4.8.2.3 PECAS_EXT peca* cria_peca_O ( Tela * *tela* )

Cria peça O na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para a tela de jogo. |

**Retorna**

Ponteiro para a peça.

##### 4.8.2.4 PECAS_EXT peca* cria_peca_T ( Tela * *tela* )

Cria peça T na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para tela de jogo. |

**Retorna**

Ponteiro para a peça.

#### 4.8.2.5 PECAS_EXT peca* cria_peca_Z ( Tela * *tela* )

Cria uma peça do tipo Z na tela.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para a tela de jogo. |

**Retorna**

Ponteiro para a peça.

#### 4.8.2.6 PECAS_EXT void libera_peca ( peca * *p* )

Libera a memória alocada para a peça.

**Parâmetros**

| | |
|---|---|
| *p* | Ponteiro para a peça a ser liberada. |

#### 4.8.2.7 PECAS_EXT void move_peca_x ( peca * *p*, int *x* )

Movimenta a peça no eixo x, ou seja no sentido horizontal.

**Parâmetros**

| | |
|---|---|
| *p* | A peça a ser movimentada. |
| *x* | Indica a direção do movimento. Se positivo, para a direita. Se negativo, para a esquerda. |

#### 4.8.2.8 PECAS_EXT void move_peca_y ( peca * *p*, int *y* )

Movimenta a peça no eixo y, ou seja, na direção vertical.

**Parâmetros**

| | |
|---|---|
| *p* | Peça a ser movida. |
| *y* | Sempre deve ser positivo, pois a peça só pode se movimentar para baixo. |

#### 4.8.2.9 PECAS_EXT void nova_peca ( Tela * *tela,* int *teste,* int *valor* )

Cria nova peça na tela do jogo. O tipo de peça a ser criado é randomizado. A coloração das peças se dá de forma cíclica.

**Parâmetros**

| | |
|---|---|
| *tela* | Ponteiro para tela de jogo. |

<Indica o tipo da peça

#### 4.8.2.10 PECAS_EXT void rotaciona_peca ( peca * *peca* )

Rotaciona a peça no sentido horário em torno de seu centro de rotação.

**Parâmetros**

| | |
|---|---|
| *peca* | Ponteiro para a peça. |

#### 4.8.2.11 PECAS_EXT void speed_up ( peca * *peca,* int *y* )

Dobra a velocidade da peça. Pode ser chamada com sucesso no máximo 5 vezes para a mesma peça.

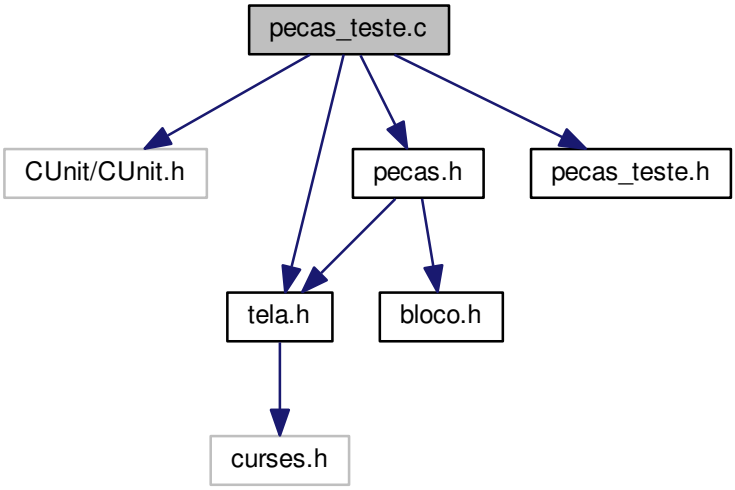
**Parâmetros**

| | |
|---|---|
| *peca* | Ponteiro para peça. |
| *y* | Inteiro qualquer. |

## 4.9 Referência do Arquivo pecas_teste.c

```
#include <CUnit/CUnit.h>
#include "pecas.h"
#include "tela.h"
#include "pecas_teste.h"
```

Gráfico de dependência de inclusões para pecas_teste.c:




#### Definições e Macros

- #define **TRUE** 1
- #define **FALSE** 0
- #define **PECAS_TESTE_OWN**

#### Funções

- void test_nova_peca ()

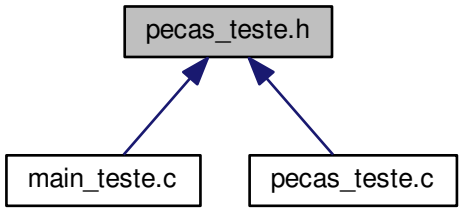
### 4.9.1 Funções

#### 4.9.1.1 void test_nova_peca ( )

Testa a função nova_peca(Tela∗, int, int). Testa todos os casos do "switch case" da função nova_peca(), chamando exatamente uma vez cada função de criação: cria_peca_I, cria_peca_Z, cria_peca_T, cria_peca_O, cria_peca_L. Para cada peça criada, é testado se a peça é alocada e, além disse, se ela é inicializada com os parâmetros esperados.

## 4.10 Referência do Arquivo pecas_teste.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:




### Definições e Macros

- #define **PECAS_TESTE_EXT** extern

### Funções

- PECAS_TESTE_EXT void test_nova_peca ()

### 4.10.1 Funções

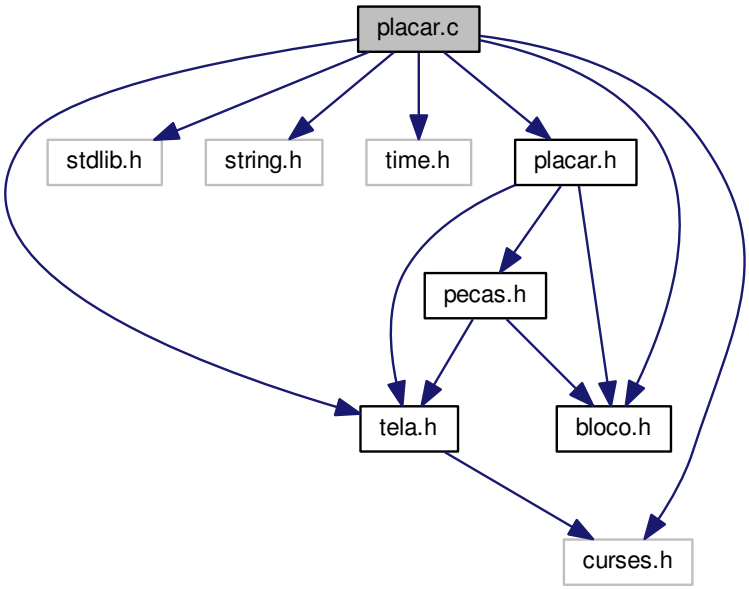
#### 4.10.1.1 PECAS_TESTE_EXT void test_nova_peca ( )

Testa a função nova_peca(Tela∗, int, int). Testa todos os casos do "switch case" da função nova_peca(), chamando exatamente uma vez cada função de criação: cria_peca_I, cria_peca_Z, cria_peca_T, cria_peca_O, cria_peca_L. Para cada peça criada, é testado se a peça é alocada e, além disse, se ela é inicializada com os parâmetros esperados.

## 4.11 Referência do Arquivo placar.c

```
#include <curses.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "bloco.h"
#include "tela.h"
#include "placar.h"
```

Gráfico de dependência de inclusões para placar.c:




**Funções**

- void cria_placar ()
- void atualiza_placar (int pontuacao)
- void mostra_placar ()
- void seta_jogador (Tela *t)
- void destroi_placar ()

### 4.11.1 Funções

#### 4.11.1.1 void atualiza_placar ( int *pontuacao* )

Ordena o placar por ordem de pontuação. O limite é de 5 jogadores, sendo excluída a menor pontuação para manter este padrão.

**Parâmetros**

| | |
|---|---|
| *pontuacao* | Pontuação do jogador atual. |

#### 4.11.1.2 void cria_placar ( )

Cria o placar, inicializando ou lendo o arquivo de pontuação, carregando pro programa os valores obtidos.

#### 4.11.1.3 void destroi_placar ( )

Libera a memória alocada para o placar.

#### 4.11.1.4 void mostra_placar ( )

Mostra o placar ao fim do jogo.

#### 4.11.1.5 void seta_jogador ( Tela * t )

Copia os dados da variável local de escore para a tela de execução.

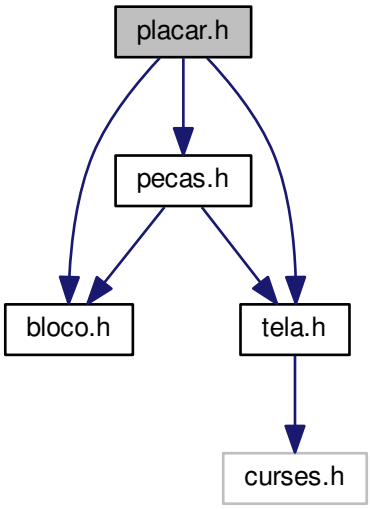
**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela. |

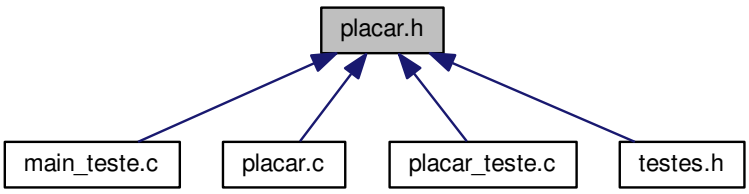
## 4.12 Referência do Arquivo placar.h

```
#include "bloco.h"
#include "pecas.h"
#include "tela.h"
```

Gráfico de dependência de inclusões para placar.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:




#### Estruturas de Dados

- struct Placar

#### Definições e Macros

- #define **PLACAR_EXT** extern

#### Definições de Tipos

- typedef struct Placar **placar**

#### Funções

- PLACAR_EXT void cria_placar ()
- PLACAR_EXT void atualiza_placar (int pontuacao)
- PLACAR_EXT void mostra_placar ()
- PLACAR_EXT void destroi_placar ()
- PLACAR_EXT void seta_jogador (Tela ∗t)

### 4.12.1 Funções

#### 4.12.1.1 PLACAR_EXT void atualiza_placar ( int *pontuacao* )

Ordena o placar por ordem de pontuação. O limite é de 5 jogadores, sendo excluída a menor pontuação para manter este padrão.

**Parâmetros**

| | |
|---|---|
| *pontuacao* | Pontuação do jogador atual. |

#### 4.12.1.2 PLACAR_EXT void cria_placar ( )

Cria o placar, inicializando ou lendo o arquivo de pontuação, carregando pro programa os valores obtidos.

#### 4.12.1.3 PLACAR_EXT void destroi_placar ( )

Libera a memória alocada para o placar.

#### 4.12.1.4 PLACAR_EXT void mostra_placar ( )

Mostra o placar ao fim do jogo.

#### 4.12.1.5 PLACAR_EXT void seta_jogador ( Tela * t )

Copia os dados da variável local de escore para a tela de execução.

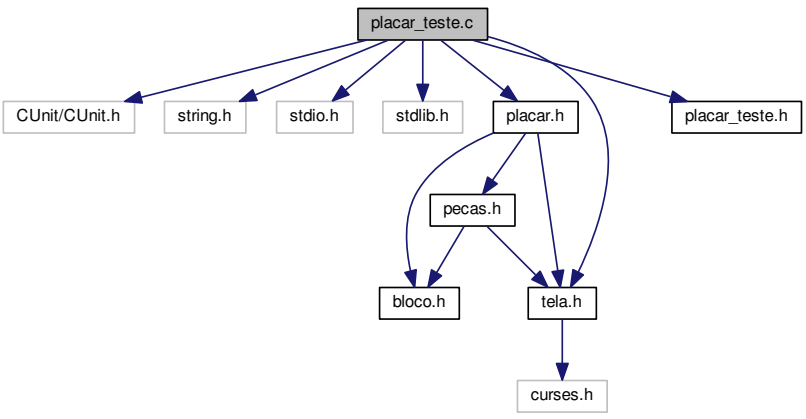
**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela. |

## 4.13 Referência do Arquivo placar_teste.c

```
#include "CUnit/CUnit.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "placar.h"
#include "tela.h"
#include "placar_teste.h"
```

Gráfico de dependência de inclusões para placar_teste.c:



#### Definições e Macros

- #define **TRUE** 1
- #define **FALSE** 0
- #define **PLACAR_TESTE_OWN**

#### Funções

- void test_cria_placar ()
- void test_atualiza_placar ()

### 4.13.1 Funções

#### 4.13.1.1 void test_atualiza_placar ( )

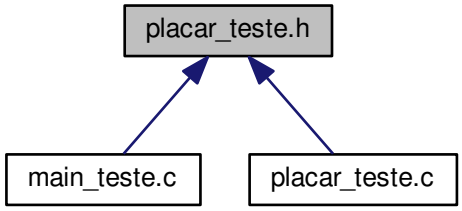
Verifica se a função atualiza_placar() é capaz de verificas o caso de exceção em que o arquivo aberto é nulo. Verifica também, com base nos critérios de limite, como ele age na situação e que o arquivo é vazio, em que só tem um valor inicializado, tem o máximo de valores inicializados e tem valores além do permitido.

#### 4.13.1.2 void test_cria_placar ( )

Verifica se a função de criar placar está gerando um arquivo não-nulo.

## 4.14 Referência do Arquivo placar_teste.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:




#### Definições e Macros

- #define **PLACAR_TESTE_EXT** extern

#### Funções

- PLACAR_TESTE_EXT void test_cria_placar ()
- PLACAR_TESTE_EXT void test_atualiza_placar ()

#### 4.14.1 Funções

##### 4.14.1.1 PLACAR_TESTE_EXT void test_atualiza_placar ( )

Verifica se a função atualiza_placar() é capaz de verificas o caso de exceção em que o arquivo aberto é nulo. Verifica também, com base nos critérios de limite, como ele age na situação e que o arquivo é vazio, em que só tem um valor inicializado, tem o máximo de valores inicializados e tem valores além do permitido.

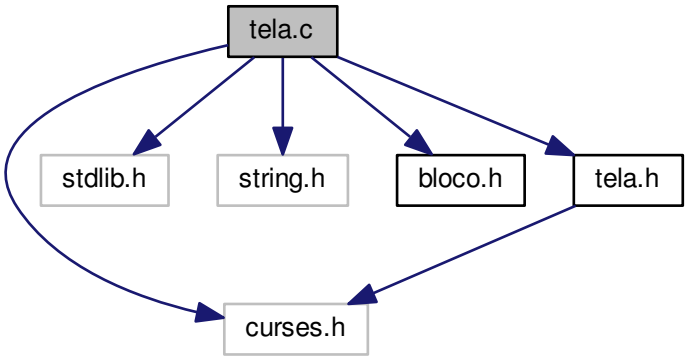
##### 4.14.1.2 PLACAR_TESTE_EXT void test_cria_placar ( )

Verifica se a função de criar placar está gerando um arquivo não-nulo.

### 4.15 Referência do Arquivo tela.c

```
#include <curses.h>
#include <stdlib.h>
#include <string.h>
#include "bloco.h"
#include "tela.h"
```

Gráfico de dependência de inclusões para tela.c:




#### Funções

- Tela ∗ cria_tela ()
- void mostra_tela (Tela ∗t)
- void mostra_pontos (int pontos)
- void mostra_tempo (int minutos, int segundos)
- int verifica_linha (Tela ∗t)
- void limpa_linha (Tela ∗t, int y)
- void desce_linhas (Tela ∗t, int y)
- int checa_fim (Tela ∗t)
- void destroi_tela (Tela ∗t)
- void define_jogador (Tela ∗t)
- void troca_letra (Tela ∗t, int valor)
- void muda_letra (Tela ∗t, int valor)

#### 4.15.1 Funções

##### 4.15.1.1 int checa_fim ( Tela * t )

Verifica se as peças ultrapassaram o limite superior do jogo.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela de jogo. |

**Retorna**

Verdadeiro se ultrapassou o limite. Falso caso contrário.

##### 4.15.1.2 Tela* cria_tela ( )

Cria uma tela de jogo com os parâmetros corretos.

**Retorna**

Retorna um ponteiro para tal tela.

##### 4.15.1.3 void define_jogador ( Tela * t )

Estado da tela onde o jogador escolhe o seu apelido.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela. |

##### 4.15.1.4 void desce_linhas ( Tela * t, int y )

Desce determinada linha da tela.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela de jogo. |
| *y* | Posição para a linha. |

##### 4.15.1.5 void destroi_tela ( Tela * t )

Libera o espaço de memória reservado para a tela de jogo.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela de jogo. |

#### 4.15.1.6 void limpa_linha ( Tela * t, int y )

Limpa uma determinada linha do jogo.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela do jogo. |
| *y* | Posição da linha a ser eliminada. |

#### 4.15.1.7 void mostra_pontos ( int pontos )

Mostra a pontuação do jogados.

**Parâmetros**

| | |
|---|---|
| *pontos* | O escore atual. |

#### 4.15.1.8 void mostra_tela ( Tela * t )

Mostra a tela de jogo, conforme seu atual estado. Também inicializa os pares de cores a serem utilizados.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela a ser mostrada. |

#### 4.15.1.9 void mostra_tempo ( int minutos, int segundos )

Mostra o tempo da partida.

**Parâmetros**

| | |
|---|---|
| *minutos* | Tempo em minutos. |
| *segundos* | Tempo em segundos. |

#### 4.15.1.10 void muda_letra ( Tela * t, int valor )

Função que alatera o caracter da atual letra selecionada para escolha do apelido.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela. |
| *valor* | Quantidade a ser incrementada. |

#### 4.15.1.11 void troca_letra ( Tela * t, int valor )

Função que altera a atual letra selecionada para escolha do apelido.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela. |
| *valor* | Quantidade a ser incrementada. |

#### 4.15.1.12 int verifica_linha ( Tela * t )

Verifica se uma linha horizontal do jogo está completamente preenchida.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela do jogo. |

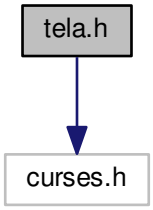
**Retorna**

100 se a linha estiver preenchida. 0 caso contrário.

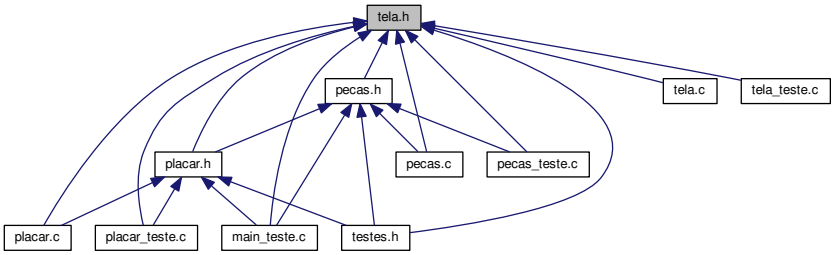
## 4.16 Referência do Arquivo tela.h

```
#include <curses.h>
```

Gráfico de dependência de inclusões para tela.h:




Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



**Estruturas de Dados**

- struct tela

**Definições e Macros**

- #define **TELA_EXT** extern

**Definições de Tipos**

- typedef struct tela Tela

**Enumerações**

- enum estado { **INICIO**, **JOGO**, **FINAL** }

**Funções**

- TELA_EXT Tela ∗ cria_tela ()
- TELA_EXT void mostra_tela (Tela ∗t)
- TELA_EXT void mostra_pontos (int pontos)
- TELA_EXT void mostra_tempo (int minutos, int segundos)
- TELA_EXT void destroi_tela (Tela ∗t)
- TELA_EXT void define_jogador (Tela ∗t)
- TELA_EXT void troca_letra (Tela ∗t, int valor)
- TELA_EXT void muda_letra (Tela ∗t, int valor)
- TELA_EXT void limpa_linha (Tela ∗t, int y)
- TELA_EXT void desce_linhas (Tela ∗t, int y)
- TELA_EXT int verifica_linha (Tela ∗t)
- TELA_EXT int checa_fim (Tela ∗t)

### 4.16.1 Definições dos tipos

#### 4.16.1.1 typedef struct tela Tela

/struct Define a tela do jogo.

### 4.16.2 Enumerações

#### 4.16.2.1 enum estado

Variável enumerada que indica o estado do jogo.

### 4.16.3 Funções

#### 4.16.3.1 TELA_EXT int checa_fim ( Tela ∗ *t* )

Verifica se as peças ultrapassaram o limite superior do jogo.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela de jogo. |

**Retorna**

Verdadeiro se ultrapassou o limite. Falso caso contrário.

#### 4.16.3.2 TELA_EXT Tela* cria_tela ( )

Cria uma tela de jogo com os parâmetros corretos.

**Retorna**

Retorna um ponteiro para tal tela.

#### 4.16.3.3 TELA_EXT void define_jogador ( Tela * *t* )

Estado da tela onde o jogador escolhe o seu apelido.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela. |

#### 4.16.3.4 TELA_EXT void desce_linhas ( Tela * *t*, int *y* )

Desce determinada linha da tela.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela de jogo. |
| *y* | Posição para a linha. |

#### 4.16.3.5 TELA_EXT void destroi_tela ( Tela * *t* )

Libera o espaço de memória reservado para a tela de jogo.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela de jogo. |

#### 4.16.3.6 TELA_EXT void limpa_linha ( Tela * *t*, int *y* )

Limpa uma determinada linha do jogo.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela do jogo. |
| *y* | Posição da linha a ser eliminada. |

#### 4.16.3.7 TELA_EXT void mostra_pontos ( int *pontos* )

Mostra a pontuação do jogados.

**Parâmetros**

| | |
|---|---|
| *pontos* | O escore atual. |

#### 4.16.3.8 TELA_EXT void mostra_tela ( Tela * *t* )

Mostra a tela de jogo, conforme seu atual estado. Também inicializa os pares de cores a serem utilizados.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela a ser mostrada. |

#### 4.16.3.9 TELA_EXT void mostra_tempo ( int *minutos,* int *segundos* )

Mostra o tempo da partida.

**Parâmetros**

| | |
|---|---|
| *minutos* | Tempo em minutos. |
| *segundos* | Tempo em segundos. |

#### 4.16.3.10 TELA_EXT void muda_letra ( Tela * *t,* int *valor* )

Função que alatera o caracter da atual letra selecionada para escolha do apelido.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela. |
| *valor* | Quantidade a ser incrementada. |

#### 4.16.3.11 TELA_EXT void troca_letra ( Tela * *t,* int *valor* )

Função que altera a atual letra selecionada para escolha do apelido.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para tela. |
| *valor* | Quantidade a ser incrementada. |

#### 4.16.3.12 TELA_EXT int verifica_linha ( Tela * t )

Verifica se uma linha horizontal do jogo está completamente preenchida.

**Parâmetros**

| | |
|---|---|
| *t* | Ponteiro para a tela do jogo. |

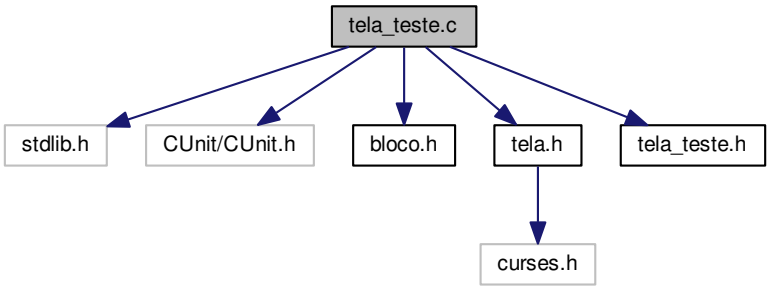
**Retorna**

100 se a linha estiver preenchida. 0 caso contrário.

## 4.17 Referência do Arquivo tela_teste.c

```
#include <stdlib.h>
#include <CUnit/CUnit.h>
#include "bloco.h"
#include "tela.h"
#include "tela_teste.h"
```

Gráfico de dependência de inclusões para tela_teste.c:




**Funções**

- void test_cria_tela ()

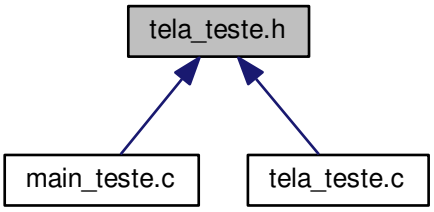
### 4.17.1 Funções

#### 4.17.1.1 void test_cria_tela ( )

Testa a função cria_tela(). Como critério de teste, foram avaliados o valor de saída, garantindo que a tela foi criada. Após isso, testou-se se a tela foi inicializada com os parâmetros iniciais esperados.

## 4.18 Referência do Arquivo tela_teste.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:




### Definições e Macros

- #define **TELA_TESTE_EXT** extern

### Funções

- TELA_TESTE_EXT void test_cria_tela ()

### 4.18.1 Funções

#### 4.18.1.1 TELA_TESTE_EXT void test_cria_tela ( )

Testa a função cria_tela(). Como critério de teste, foram avaliados o valor de saída, garantindo que a tela foi criada. Após isso, testou-se se a tela foi inicializada com os parâmetros iniciais esperados.

## Índice Remissivo