

## Tetris - PS Edition

Gerado por Doxygen 1.8.11

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

## Sumário

<b>1</b>	<b>Índice das Estruturas de Dados</b>	<b>2</b>
1.1	Estruturas de Dados . . . . .	2
<b>2</b>	<b>Índice dos Arquivos</b>	<b>2</b>
2.1	Lista de Arquivos . . . . .	2
<b>3</b>	<b>Estruturas</b>	<b>3</b>
3.1	Referência da Estrutura Bloco . . . . .	3
3.1.1	Campos . . . . .	3
3.2	Referência da Estrutura Peca . . . . .	4
3.2.1	Campos . . . . .	5
3.3	Referência da Estrutura Placar . . . . .	6
3.3.1	Campos . . . . .	6
3.4	Referência da Estrutura tela . . . . .	7
3.4.1	Descrição Detalhada . . . . .	8
3.4.2	Campos . . . . .	8
3.5	Referência da Estrutura timeb . . . . .	9
<b>4</b>	<b>Arquivos</b>	<b>9</b>
4.1	Referência do Arquivo bloco.h . . . . .	9
4.2	Referência do Arquivo engine.c . . . . .	10
4.2.1	Funções . . . . .	10
4.3	Referência do Arquivo engine.h . . . . .	11
4.3.1	Funções . . . . .	11
4.4	Referência do Arquivo engine_teste.c . . . . .	12
4.4.1	Funções . . . . .	12
4.5	Referência do Arquivo engine_teste.h . . . . .	13
4.5.1	Funções . . . . .	13
4.6	Referência do Arquivo main_teste.c . . . . .	14
4.7	Referência do Arquivo pecas.c . . . . .	14

4.7.1	Funções . . . . .	15
4.7.2	Variáveis . . . . .	18
4.8	Referência do Arquivo pecas.h . . . . .	18
4.8.1	Enumerações . . . . .	20
4.8.2	Funções . . . . .	20
4.9	Referência do Arquivo pecas_teste.c . . . . .	22
4.9.1	Funções . . . . .	23
4.10	Referência do Arquivo pecas_teste.h . . . . .	24
4.10.1	Funções . . . . .	24
4.11	Referência do Arquivo placar.c . . . . .	24
4.11.1	Funções . . . . .	25
4.12	Referência do Arquivo placar.h . . . . .	26
4.12.1	Funções . . . . .	27
4.13	Referência do Arquivo placar_teste.c . . . . .	28
4.13.1	Funções . . . . .	29
4.14	Referência do Arquivo placar_teste.h . . . . .	29
4.14.1	Funções . . . . .	30
4.15	Referência do Arquivo tela.c . . . . .	30
4.15.1	Funções . . . . .	31
4.16	Referência do Arquivo tela.h . . . . .	33
4.16.1	Definições dos tipos . . . . .	34
4.16.2	Enumerações . . . . .	34
4.16.3	Funções . . . . .	34
4.17	Referência do Arquivo tela_teste.c . . . . .	37
4.17.1	Funções . . . . .	37
4.18	Referência do Arquivo tela_teste.h . . . . .	38
4.18.1	Funções . . . . .	38

## 1 Índice das Estruturas de Dados

### 1.1 Estruturas de Dados

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

<a href="#">Bloco</a>	3
<a href="#">Peca</a>	4
<a href="#">Placar</a>	6
<a href="#">tela</a>	7
<a href="#">timeb</a>	9

## 2 Índice dos Arquivos

### 2.1 Lista de Arquivos

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

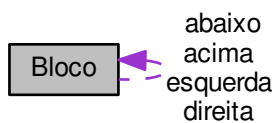
<a href="#">bloco.h</a>	9
<a href="#">cdefs.h</a>	??
<a href="#">engine.c</a>	10
<a href="#">engine.h</a>	11
<a href="#">engine_teste.c</a>	12
<a href="#">engine_teste.h</a>	13
<a href="#">features.h</a>	??
<a href="#">main_teste.c</a>	14
<a href="#">pecas.c</a>	14
<a href="#">pecas.h</a>	18
<a href="#">pecas_teste.c</a>	22
<a href="#">pecas_teste.h</a>	24
<a href="#">placar.c</a>	24
<a href="#">placar.h</a>	26
<a href="#">placar_teste.c</a>	28
<a href="#">placar_teste.h</a>	29
<a href="#">stubs-64.h</a>	??
<a href="#">stubs.h</a>	??

tela.c	30
tela.h	33
tela_teste.c	37
tela_teste.h	38
testes.h	??
timeb.h	??
wordsize.h	??

## 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

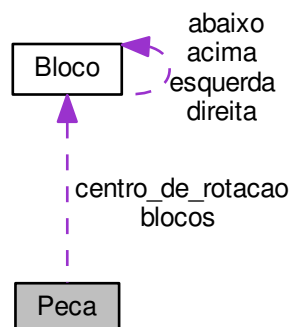
Coordenada cartesiana vertical do bloco.

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\\_pecas](#)
- unsigned short int [move\\_pecas](#)
- float [velocidade](#)
- tp\_pecas [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\\_pecas](#)

Cor da peça.

##### 3.2.1.4 unsigned short int Peca::[move\\_pecas](#)

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\_pecas 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.

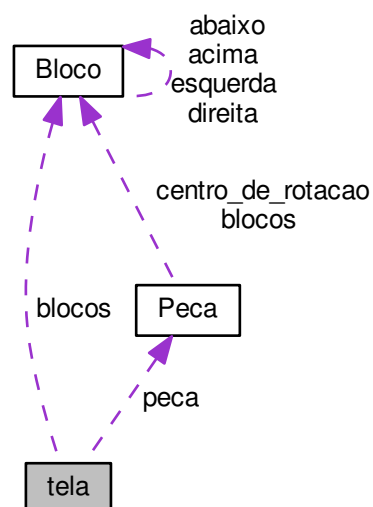
A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- [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**

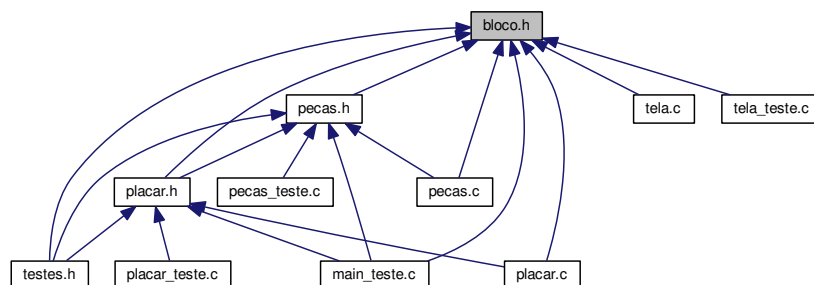
A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- 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

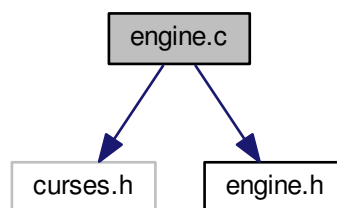
- typedef struct [Bloco](#) **bloco**

## 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

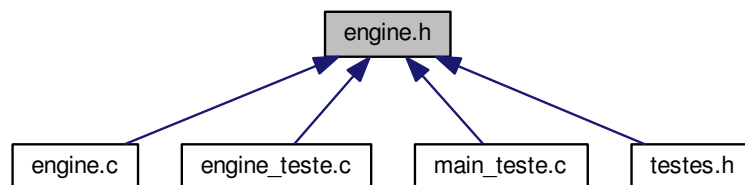
<i>input</i>	Entrada.
--------------	----------

## 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**

<i>input</i>	Entrada.
--------------	----------

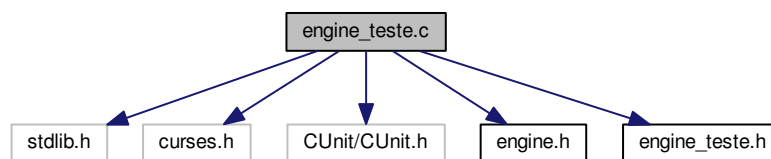
**Retorna**

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_pegar_input` ()

**4.4.1 Funções****4.4.1.1 void test\_ncurses ( )**

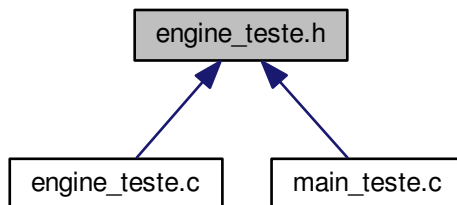
Verifica se é possível inicializar e finalizar as funções do curses corretamente.

**4.4.1.2 void test\_pegar\_input ( )**

Testa a função `pegar_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\\_pegar\\_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.

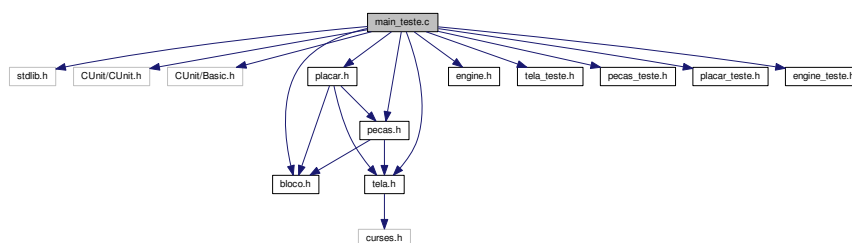
##### 4.5.1.2 ENGINE\_TESTE\_EXT void test\_pegar\_input ( )

Testa a função `pegar_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

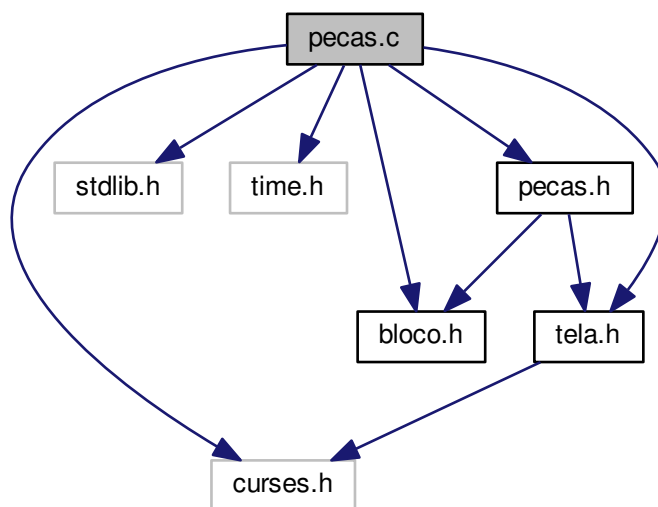
- int **main** ()

#### 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_pecas` (`Tela *tela`, int teste, int valor)
- `peca` \* `cria_pecas_I` (`Tela *tela`)
- `peca` \* `cria_pecas_Z` (`Tela *tela`)
- `peca` \* `cria_pecas_T` (`Tela *tela`)
- `peca` \* `cria_pecas_O` (`Tela *tela`)
- `peca` \* `cria_pecas_L` (`Tela *tela`)
- void `move_pecas_x` (`peca` \*p, int x)
- void `move_pecas_y` (`peca` \*p, int y)
- void `rotaciona_pecas` (`peca` \*peca)
- void `speed_up` (`peca` \*peca, int y)
- void `libera_pecas` (`peca` \*p)

### Variáveis

- unsigned short int `cor_nova_pecas` = 4
- unsigned short int `speed_ups` = 0

#### 4.7.1 Funções

##### 4.7.1.1 `peca* cria_pecas_I ( Tela * tela )`

Cria uma peça do tipo I na tela.

**Parâmetros**

<i>tela</i>	Ponteiro para tela.
-------------	---------------------

**Retorna**

Ponteiro para a peça.

**4.7.1.2 peca\* cria\_peca\_L ( Tela \* tela )**

Cria peça L na tela.

**Parâmetros**

<i>tela</i>	Ponteiro para a tela.
-------------	-----------------------

**Retorna**

Ponteiro para peça.

**4.7.1.3 peca\* cria\_peca\_O ( Tela \* tela )**

Cria peça O na tela.

**Parâmetros**

<i>tela</i>	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**

<i>tela</i>	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

<i>tela</i>	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

<i>p</i>	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

<i>p</i>	A peça a ser movimentada.
<i>x</i>	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

<i>p</i>	Peça a ser movida.
<i>y</i>	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

<i>tela</i>	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

<i>peca</i>	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

<i>peca</i>	Ponteiro para peça.
<i>y</i>	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.

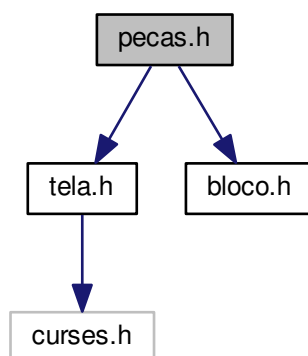
## 4.7.2.2 unsigned short int speed\_ups = 0

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

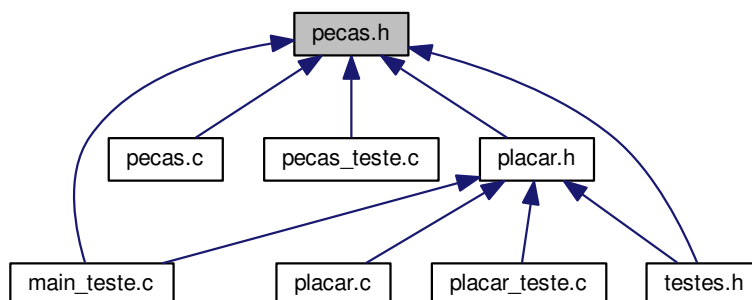
## 4.8 Referência do Arquivo pecas.h

```
#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\_pec**
- typedef struct [Peca](#) **peca**

#### Enumerações

- enum [Tipo\\_Peca](#) {  
**Tipo\_I**, **Tipo\_Z**, **Tipo\_T**, **Tipo\_O**,  
**Tipo\_L** }

#### Funções

- `PECAS_EXT void nova\_pec (Tela *tela, int teste, int valor)`
- `PECAS_EXT void move\_pec_x (peca *peca, int x)`
- `PECAS_EXT void move\_pec_y (peca *peca, int y)`
- `PECAS_EXT void rotaciona\_pec (peca *peca)`
- `PECAS_EXT void speed\_up (peca *peca, int y)`
- `PECAS_EXT void libera\_pec (peca *p)`
- `PECAS_EXT peca * cria\_pec_I (Tela *tela)`
- `PECAS_EXT peca * cria\_pec_Z (Tela *tela)`
- `PECAS_EXT peca * cria\_pec_T (Tela *tela)`
- `PECAS_EXT peca * cria\_pec_O (Tela *tela)`
- `PECAS_EXT peca * cria\_pec_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

<i>tela</i>	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

<i>tela</i>	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

<i>tela</i>	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

<i>tela</i>	Ponteiro para tela de jogo.
-------------	-----------------------------

## Retorna

Ponteiro para a peça.

**4.8.2.5 PECAS\_EXT peca\* cria\_pecas\_Z ( Tela \* tela )**

Cria uma peça do tipo Z na tela.

## Parâmetros

<i>tela</i>	Ponteiro para a tela de jogo.
-------------	-------------------------------

## Retorna

Ponteiro para a peça.

**4.8.2.6 PECAS\_EXT void libera\_pecas ( peca \* p )**

Libera a memória alocada para a peça.

## Parâmetros

<i>p</i>	Ponteiro para a peça a ser liberada.
----------	--------------------------------------

**4.8.2.7 PECAS\_EXT void move\_pecas\_x ( peca \* p, int x )**

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

## Parâmetros

<i>p</i>	A peça a ser movimentada.
<i>x</i>	Indica a direção do movimento. Se positivo, para a direita. Se negativo, para a esquerda.

**4.8.2.8 PECAS\_EXT void move\_pecas\_y ( peca \* p, int y )**

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

## Parâmetros

<i>p</i>	Peça a ser movida.
<i>y</i>	Sempre deve ser positivo, pois a peça só pode se movimentar para baixo.

#### 4.8.2.9 PECAS\_EXT void nova\_pecas ( 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

<i>tela</i>	Ponteiro para tela de jogo.
-------------	-----------------------------

<Indica o tipo da peça

#### 4.8.2.10 PECAS\_EXT void rotaciona\_pecas ( pecas \* pecas )

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

##### Parâmetros

<i>pecas</i>	Ponteiro para a peça.
--------------	-----------------------

#### 4.8.2.11 PECAS\_EXT void speed\_up ( pecas \* pecas, int y )

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

##### Parâmetros

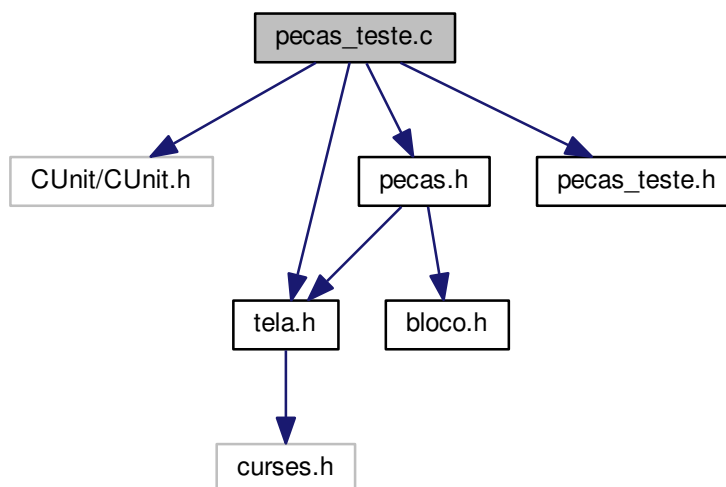
<i>pecas</i>	Ponteiro para peça.
<i>y</i>	Inteiro qualquer.

## 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_pecas ()`

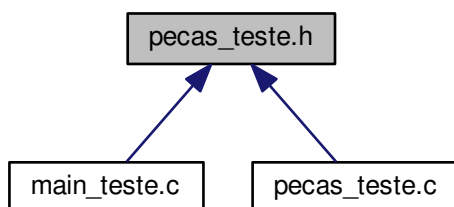
#### 4.9.1 Funções

##### 4.9.1.1 `void test_nova_pecas ( )`

Testa a função `nova_pecas(Tela*, int, int)`. Testa todos os casos do "switch case" da função `nova_pecas()`, chamando exatamente uma vez cada função de criação: `cria_pecas_I`, `cria_pecas_Z`, `cria_pecas_T`, `cria_pecas_O`, `cria_pecas_L`. Para cada peça criada, é testado se a peça é alocada e, além disso, 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

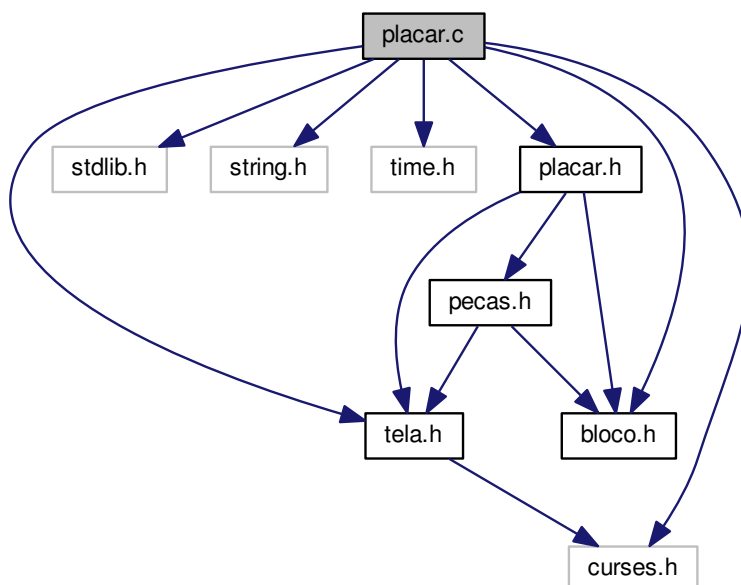
###### 4.10.1.1 **PECAS\_TESTE\_EXT** void [test\\_nova\\_peca](#) ( )

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 disso, 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

<i>pontuacao</i>	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.

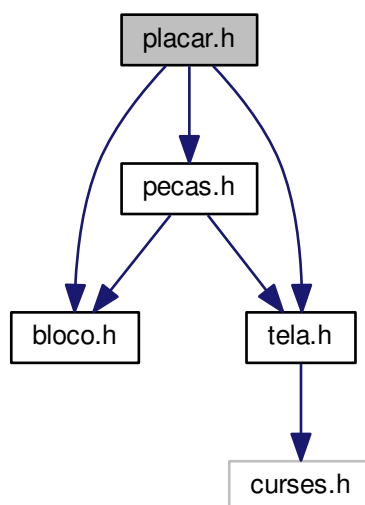
##### Parâmetros

<i>t</i>	Ponteiro para tela.
----------	---------------------

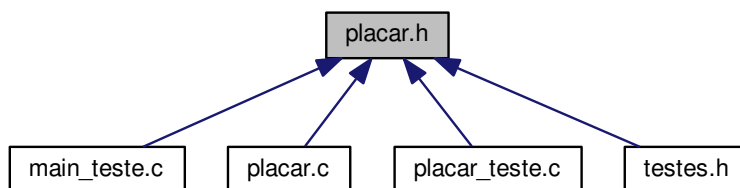
## 4.12 Referência do Arquivo placar.h

```
#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

<code>pontuacao</code>	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 score para a tela de execução.

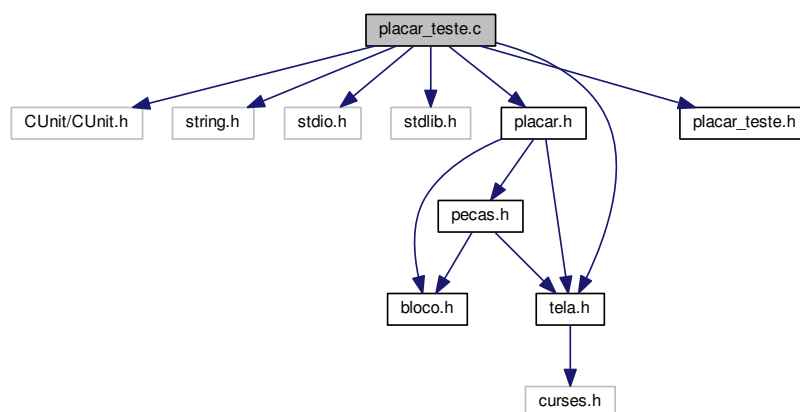
##### Parâmetros

<i>t</i>	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 ( )

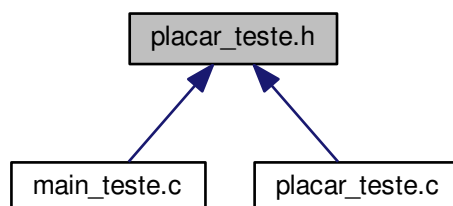
Verifica se a função [atualiza\\_placar\(\)](#) é capaz de verificar o caso de exceção em que o arquivo aberto é nulo. Verifica também, com base nos critérios de limite, como ele age na situação e que o arquivo é vazio, em que só tem um valor inicializado, tem o máximo de valores inicializados e tem 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 verificar 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.

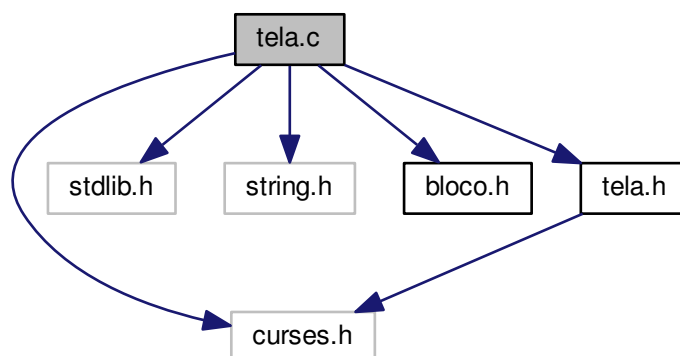
##### 4.14.1.2 PLACAR\_TESTE\_EXT void test\_cria\_placar ( )

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**

<i>t</i>	Ponteiro para a tela do jogo.
<i>y</i>	Posição da linha a ser eliminada.

**4.15.1.7 void mostra\_pontos ( int *pontos* )**

Mostra a pontuação do jogadores.

**Parâmetros**

<i>pontos</i>	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**

<i>t</i>	Ponteiro para tela a ser mostrada.
----------	------------------------------------

**4.15.1.9 void mostra\_tempo ( int *minutos*, int *segundos* )**

Mostra o tempo da partida.

**Parâmetros**

<i>minutos</i>	Tempo em minutos.
<i>segundos</i>	Tempo em segundos.

**4.15.1.10 void muda\_letra ( Tela \* *t*, int *valor* )**

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

**Parâmetros**

<i>t</i>	Ponteiro para tela.
<i>valor</i>	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

<i>t</i>	Ponteiro para tela.
<i>valor</i>	Quantidade a ser incrementada.

## 4.15.1.12 int verifica\_linha ( Tela \* t )

Verifica se uma linha horizontal do jogo está completamente preenchida.

## Parâmetros

<i>t</i>	Ponteiro para a tela do jogo.
----------	-------------------------------

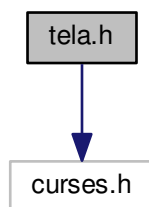
## Retorna

100 se a linha estiver preenchida. 0 caso contrário.

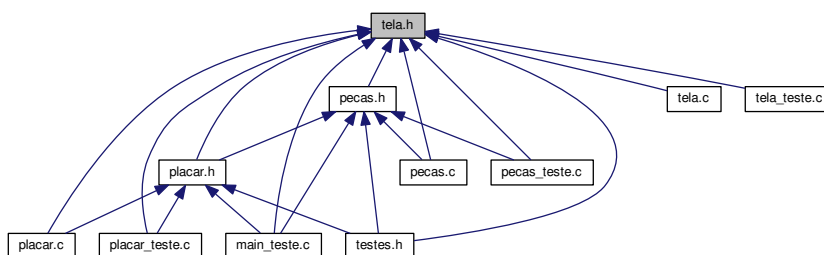
## 4.16 Referência do Arquivo tela.h

```
#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

<i>t</i>	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

<i>t</i>	Ponteiro para tela.
----------	---------------------

4.16.3.4 TELA\_EXT void desce\_linhas ( Tela \* *t*, int *y* )

Desce determinada linha da tela.

## Parâmetros

<i>t</i>	Ponteiro para a tela de jogo.
<i>y</i>	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

<i>t</i>	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

<i>t</i>	Ponteiro para a tela do jogo.
<i>y</i>	Posição da linha a ser eliminada.

4.16.3.7 TELA\_EXT void mostra\_pontos ( int *pontos* )

Mostra a pontuação do jogadores.

## Parâmetros

<i>pontos</i>	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

<i>t</i>	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

<i>minutos</i>	Tempo em minutos.
<i>segundos</i>	Tempo em segundos.

4.16.3.10 TELA\_EXT void muda\_letra ( Tela \* *t*, int *valor* )

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

## Parâmetros

<i>t</i>	Ponteiro para tela.
<i>valor</i>	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

<i>t</i>	Ponteiro para tela.
<i>valor</i>	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**

<i>t</i>	Ponteiro para a tela do jogo.
----------	-------------------------------

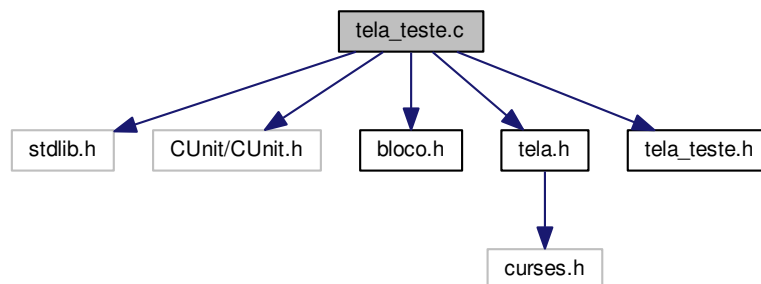
**Retorna**

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()`

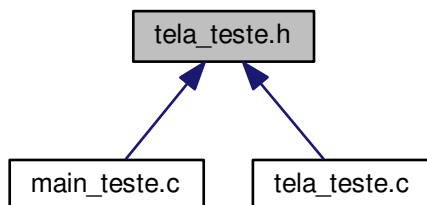
## 4.17.1 Funções

## 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

abaixo  
    Bloco, 3  
acima  
    Bloco, 3  
anos  
    Placar, 6  
arquivo  
    Placar, 6  
atualiza\_placar  
    placar.c, 25  
    placar.h, 27  
  
Bloco, 3  
    abaixo, 3  
    acima, 3  
    bolinha, 4  
    cor, 4  
    direita, 4  
    esquerda, 4  
    move, 4  
    pos\_x, 4  
    pos\_y, 4  
bloco.h, 9  
blocos  
    Peca, 5  
    tela, 8  
bolinha  
    Bloco, 4  
  
centro\_de\_rotacao  
    Peca, 5  
checa\_fim  
    tela.c, 31  
    tela.h, 34  
comprimento  
    tela, 8  
contador\_jogadores  
    Placar, 6  
cor  
    Bloco, 4  
cor\_nova\_pecas  
    pecas.c, 18  
cor\_pecas  
    Peca, 5  
cria\_pecas\_I  
    pecas.c, 15  
    pecas.h, 20  
cria\_pecas\_L  
    pecas.c, 16  
    pecas.h, 20  
cria\_pecas\_O  
    pecas.c, 16  
    pecas.h, 20  
cria\_pecas\_T  
    pecas.c, 16  
    pecas.h, 20  
cria\_pecas\_Z  
    pecas.c, 16  
    pecas.h, 21  
cria\_placar  
    placar.c, 25  
    placar.h, 27  
cria\_tela  
    tela.c, 31  
    tela.h, 35  
  
define\_jogador  
    tela.c, 31  
    tela.h, 35  
desce\_linhas  
    tela.c, 31  
    tela.h, 35  
destroi\_placar  
    placar.c, 25  
    placar.h, 28  
destroi\_tela  
    tela.c, 31  
    tela.h, 35  
dias  
    Placar, 6  
direita  
    Bloco, 4  
  
engine.c, 10  
    finaliza\_ncurses, 10  
    inicia\_ncurses, 10  
    pega\_input, 10  
engine.h, 11  
    finaliza\_ncurses, 11  
    inicia\_ncurses, 11  
    pega\_input, 11  
engine\_teste.c, 12  
    test\_ncurses, 12  
    test\_pega\_input, 12  
engine\_teste.h, 13  
    test\_ncurses, 13  
    test\_pega\_input, 13  
esquerda  
    Bloco, 4  
estado  
    tela, 8  
    tela.h, 34  
  
finaliza\_ncurses  
    engine.c, 10  
    engine.h, 11  
  
inicia\_ncurses  
    engine.c, 10  
    engine.h, 11

janela  
  tela, 8  
jogador  
  Placar, 6  
  tela, 8  
jogadores  
  Placar, 6  
  
largura  
  tela, 8  
letra  
  tela, 8  
libera\_pecas  
  pecas.c, 17  
  pecas.h, 21  
limpa\_linha  
  tela.c, 31  
  tela.h, 35  
  
main\_teste.c, 14  
meses  
  Placar, 6  
mostra\_placar  
  placar.c, 25  
  placar.h, 28  
mostra\_pontos  
  tela.c, 32  
  tela.h, 36  
mostra\_tela  
  tela.c, 32  
  tela.h, 36  
mostra\_tempo  
  tela.c, 32  
  tela.h, 36  
move  
  Bloco, 4  
move\_pecas  
  Peca, 5  
move\_pecas\_x  
  pecas.c, 17  
  pecas.h, 21  
move\_pecas\_y  
  pecas.c, 17  
  pecas.h, 21  
muda\_letra  
  tela.c, 32  
  tela.h, 36  
  
nova\_pecas  
  pecas.c, 17  
  pecas.h, 21  
  
Peca, 4  
  blocos, 5  
  centro\_de\_rotacao, 5  
  cor\_pecas, 5  
  move\_pecas, 5  
  tamanho, 5  
  tipo, 5  
  
velocidade, 5  
pecas  
  tela, 8  
pecas.c, 14  
  cor\_nova\_pecas, 18  
  cria\_pecas\_I, 15  
  cria\_pecas\_L, 16  
  cria\_pecas\_O, 16  
  cria\_pecas\_T, 16  
  cria\_pecas\_Z, 16  
  libera\_pecas, 17  
  move\_pecas\_x, 17  
  move\_pecas\_y, 17  
  nova\_pecas, 17  
  rotaciona\_pecas, 17  
  speed\_up, 18  
  speed\_ups, 18  
pecas.h, 18  
  cria\_pecas\_I, 20  
  cria\_pecas\_L, 20  
  cria\_pecas\_O, 20  
  cria\_pecas\_T, 20  
  cria\_pecas\_Z, 21  
  libera\_pecas, 21  
  move\_pecas\_x, 21  
  move\_pecas\_y, 21  
  nova\_pecas, 21  
  rotaciona\_pecas, 22  
  speed\_up, 22  
  Tipo\_Peca, 20  
pecas\_teste.c, 22  
  test\_nova\_pecas, 23  
pecas\_teste.h, 24  
  test\_nova\_pecas, 24  
pega\_input  
  engine.c, 10  
  engine.h, 11  
Placar, 6  
  anos, 6  
  arquivo, 6  
  contador\_jogadores, 6  
  dias, 6  
  jogador, 6  
  jogadores, 6  
  meses, 6  
  pontuacao, 6  
  pontuacoes, 7  
  tempo\_m, 7  
  tempo\_s, 7  
  tempos\_m, 7  
  tempos\_s, 7  
placar.c, 24  
  atualiza\_placar, 25  
  cria\_placar, 25  
  destroi\_placar, 25  
  mostra\_placar, 25  
  seta\_jogador, 26  
placar.h, 26

- atualiza\_placar, 27
- cria\_placar, 27
- destroi\_placar, 28
- mostra\_placar, 28
- seta\_jogador, 28
- placar\_teste.c, 28
  - test\_atualiza\_placar, 29
  - test\_cria\_placar, 29
- placar\_teste.h, 29
  - test\_atualiza\_placar, 30
  - test\_cria\_placar, 30
- pontos
  - tela, 9
- pontuacao
  - Placar, 6
- pontuacoes
  - Placar, 7
- pos\_x
  - Bloco, 4
- pos\_y
  - Bloco, 4
- rotaciona\_pecas
  - pecas.c, 17
  - pecas.h, 22
- seta\_jogador
  - placar.c, 26
  - placar.h, 28
- speed\_up
  - pecas.c, 18
  - pecas.h, 22
- speed\_ups
  - pecas.c, 18
- tamanho
  - Peca, 5
- Tela
  - tela.h, 34
- tela, 7
  - blocos, 8
  - comprimento, 8
  - estado, 8
  - janela, 8
  - jogador, 8
  - largura, 8
  - letra, 8
  - peca, 8
  - pontos, 9
  - tempo\_m, 9
  - tempo\_s, 9
- tela.c, 30
  - checa\_fim, 31
  - cria\_tela, 31
  - define\_jogador, 31
  - desce\_linhas, 31
  - destroi\_tela, 31
  - limpa\_linha, 31
  - mostra\_pontos, 32
  - mostra\_tela, 32
  - mostra\_tempo, 32
  - muda\_letra, 32
  - troca\_letra, 32
  - verifica\_linha, 33
- tela.h, 33
  - checa\_fim, 34
  - cria\_tela, 35
  - define\_jogador, 35
  - desce\_linhas, 35
  - destroi\_tela, 35
  - estado, 34
  - limpa\_linha, 35
  - mostra\_pontos, 36
  - mostra\_tela, 36
  - mostra\_tempo, 36
  - muda\_letra, 36
  - Tela, 34
  - troca\_letra, 36
  - verifica\_linha, 37
- tela\_teste.c, 37
  - test\_cria\_tela, 37
- tela\_teste.h, 38
  - test\_cria\_tela, 38
- tempo\_m
  - Placar, 7
  - tela, 9
- tempo\_s
  - Placar, 7
  - tela, 9
- tempos\_m
  - Placar, 7
- tempos\_s
  - Placar, 7
- test\_atualiza\_placar
  - placar\_teste.c, 29
  - placar\_teste.h, 30
- test\_cria\_placar
  - placar\_teste.c, 29
  - placar\_teste.h, 30
- test\_cria\_tela
  - tela\_teste.c, 37
  - tela\_teste.h, 38
- test\_ncurses
  - engine\_teste.c, 12
  - engine\_teste.h, 13
- test\_nova\_pecas
  - pecas\_teste.c, 23
  - pecas\_teste.h, 24
- test\_pegar\_input
  - engine\_teste.c, 12
  - engine\_teste.h, 13
- timeb, 9
- tipo
  - Peca, 5
- Tipo\_Peca
  - pecas.h, 20
- troca\_letra

tela.c, [32](#)  
tela.h, [36](#)

velocidade

Peca, [5](#)

verifica\_linha

tela.c, [33](#)

tela.h, [37](#)