

Jogo da velha

Generated by Doxygen 1.8.11

Contents

1	Main Page	1
2	README	3
3	Data Structure Index	5
3.1	Data Structures	5
4	File Index	7
4.1	File List	7
5	Data Structure Documentation	9
5.1	node Struct Reference	9
5.1.1	Field Documentation	9
5.1.1.1	altura	9
5.1.1.2	chave	10
5.1.1.3	dir	10
5.1.1.4	esq	10
5.1.1.5	jogador	10
5.2	struct Struct Reference	10
5.2.1	Detailed Description	10

6	File Documentation	11
6.1	avl_jogo_da_velha.c File Reference	11
6.1.1	Detailed Description	12
6.1.2	Function Documentation	12
6.1.2.1	altura(Arvore *a)	12
6.1.2.2	atualizar_altura(Arvore *a)	13
6.1.2.3	atualizar_fb_dir(Arvore *r)	14
6.1.2.4	atualizar_fb_esq(Arvore *r)	14
6.1.2.5	balanceamento(Arvore *a)	15
6.1.2.6	buscar(Arvore *a, int chave)	16
6.1.2.7	inserir(Arvore *a, int chave, int jogador)	17
6.1.2.8	maior(int esq, int dir)	18
6.1.2.9	remover(Arvore *a, int chave)	19
6.1.2.10	rotacao_dupla_dir(Arvore *r)	19
6.1.2.11	rotacao_dupla_esq(Arvore *r)	20
6.1.2.12	rotacao_simples_dir(Arvore *r)	21
6.1.2.13	rotacao_simples_esq(Arvore *r)	21
6.1.2.14	verifica_ganhador(Arvore *a, int jogador)	22
6.2	avl_jogo_da_velha.h File Reference	23
6.2.1	Detailed Description	25
6.2.2	Function Documentation	25
6.2.2.1	altura(Arvore *a)	25
6.2.2.2	atualizar_altura(Arvore *a)	26
6.2.2.3	atualizar_fb_dir(Arvore *r)	26
6.2.2.4	atualizar_fb_esq(Arvore *r)	27
6.2.2.5	balanceamento(Arvore *a)	28
6.2.2.6	buscar(Arvore *a, int chave)	29
6.2.2.7	inserir(Arvore *a, int chave, int jogador)	30
6.2.2.8	maior(int esq, int dir)	31
6.2.2.9	remover(Arvore *a, int chave)	32
6.2.2.10	rotacao_dupla_dir(Arvore *r)	32
6.2.2.11	rotacao_dupla_esq(Arvore *r)	33
6.2.2.12	rotacao_simples_dir(Arvore *r)	34
6.2.2.13	rotacao_simples_esq(Arvore *r)	34
6.2.2.14	verifica_ganhador(Arvore *a, int jogador)	35
6.3	jogo_da_velha_main.c File Reference	36
6.3.1	Detailed Description	37
6.3.2	Function Documentation	37
6.3.2.1	jogar_contra_computador(Arvore *a)	37
6.3.2.2	jogar_contra_jogador(Arvore *a)	38
6.3.2.3	main()	38

Chapter 1

Main Page

Código fonte do jogo da velha.

Esse código, juntamente com a Main, representa uma série de funções que controlam o jogo da velha!

Author

Lucas Ribeiro

Date

25/05/2017

Chapter 2

README

Jogo da Velha I.A

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

<code>node</code>	9
<code>struct</code>	Essa estrutura representa um nó que irá formar a árvore AVL	10

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

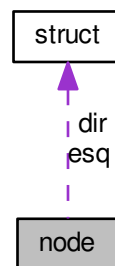
avl_jogo_da_velha.c	Implementação das funções descritas na biblioteca avl_jogo_da_velha.h	11
avl_jogo_da_velha.h	É uma biblioteca para manipulação de AVL	23
jogo_da_velha_main.c	Código fonte do jogo da velha	36

Chapter 5

Data Structure Documentation

5.1 node Struct Reference

Collaboration diagram for node:



Data Fields

- int `chave`
- int `jogador`
- int `altura`
- `struct node * esq`
- `struct node * dir`

5.1.1 Field Documentation

5.1.1.1 int altura

Variável inteiro que armazena a altura do nó na árvore

5.1.1.2 `int chave`

Variável inteira que armazena uma chave para reconhecer o nó

5.1.1.3 `struct node* dir`

Ponteiro que aponta para o nó á direita do nó

5.1.1.4 `struct node* esq`

Ponteiro que aponta para o nó á esquerda do nó

5.1.1.5 `int jogador`

Variável inteiro que armazena qual jogador fez a jogada

The documentation for this struct was generated from the following file:

- [avl_jogo_da_velha.h](#)

5.2 `struct Struct Reference`

Essa estrutura representa um nó que irá formar a árvore AVL.

5.2.1 Detailed Description

Essa estrutura representa um nó que irá formar a árvore AVL.

The documentation for this struct was generated from the following file:

- [avl_jogo_da_velha.h](#)

Chapter 6

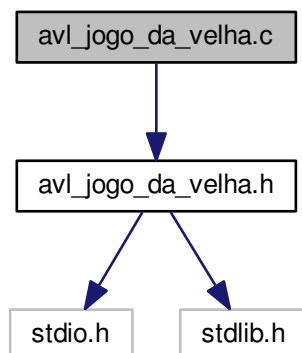
File Documentation

6.1 avl_jogo_da_velha.c File Reference

Implementação das funções descritas na biblioteca [avl_jogo_da_velha.h](#).

```
#include "avl_jogo_da_velha.h"
```

Include dependency graph for avl_jogo_da_velha.c:



Functions

- void **clear** ()
- int **maior** (int esq, int dir)
Função que ao receber 2 inteiro, retorna o maior deles.
- int **altura** (**Arvore** *a)
Função retorna a altura de uma árvore.
- int **atualizar_altura** (**Arvore** *a)
Função recursiva que atualiza as altura da árvore.
- int **balanceamento** (**Arvore** *a)

- Função que retorna a diferença das alturas do nós.*
- [Arvore * rotacao_simples_esq](#) ([Arvore *r](#))
- Função que realiza uma rotação simples a esquerda na AVL.*
- [Arvore * rotacao_simples_dir](#) ([Arvore *r](#))
- Função que realiza uma rotação simples a direita na AVL.*
- [Arvore * rotacao_dupla_esq](#) ([Arvore *r](#))
- Função que realiza uma rotação dupla a esquerda na AVL.*
- [Arvore * rotacao_dupla_dir](#) ([Arvore *r](#))
- Função que realiza uma rotação dupla a direita na AVL.*
- [Arvore * atualizar_fb_dir](#) ([Arvore *r](#))
- [Arvore * atualizar_fb_esq](#) ([Arvore *r](#))
- [Arvore * inserir](#) ([Arvore *a](#), int chave, int jogador)
- Insere um novo nó na árvore AVL.*
- [Arvore * remover](#) ([Arvore *a](#), int chave)
- Remove um nó da árvore AVL baseado na sua chave.*
- int [buscar](#) ([Arvore *a](#), int chave)
- Função que encontra um nó específico baseado em sua chave e retorna o valor da variável jogador.*
- void [imprimir_in_order](#) ([Arvore *a](#), int nivel)
- void [arvore_para_vetor](#) ([Arvore *a](#), int *tabuleiro)
- void [imprimir_tabuleiro](#) ([Arvore *a](#))
- int [verifica_ganhador](#) ([Arvore *a](#), int jogador)
- Essa função verifica se alguém ganhou o jogo e retorna um inteiro.*
- int [conta_nos](#) ([Arvore *a](#))
- void [calcula_passos](#) ([Arvore **verificacao](#), [Arvore *a](#), [Arvore *b](#), int *passos, int index)
- [Arvore * jogada_computador](#) ([Arvore *a](#), [Arvore *verificacao](#))
- int [jogada_jogador](#) ()
- int [menu_principal](#) ()

6.1.1 Detailed Description

Implementação das funções descritas na biblioteca [avl_jogo_da_velha.h](#).

Author

Lucas Ribeiro

Date

25/05/2017

6.1.2 Function Documentation

6.1.2.1 int altura ([Arvore * a](#))

Função retorna a altura de uma árvore.

Precondition

O ponteiro para a árvore não deve ter valor NULL

Warning

Caso o pré requisito não seja atendido a função retornará NULL

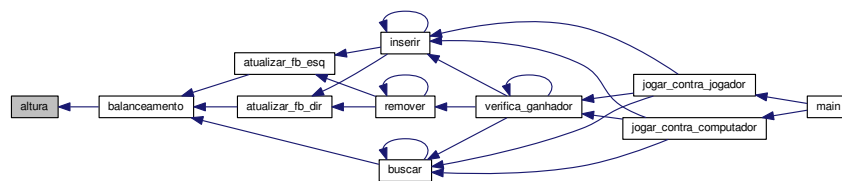
Parameters

<i>a</i>	Ponteiro que aponta para a árvore AVL.
----------	--

Returns

Retorna um inteiro que representa

Here is the caller graph for this function:

6.1.2.2 int atualizar_altura (**Arvore** * *a*)

Função recursiva que atualiza as altura da árvore.

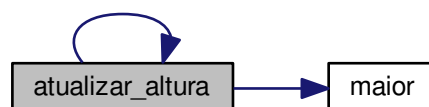
Parameters

<i>a</i>	Ponteiro que aponta para o início da AVL
----------	--

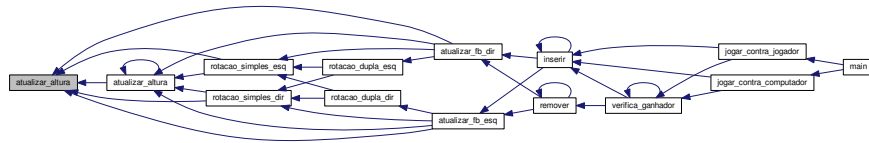
Returns

O retorno inteiro serve para o funcionamento da recursividade.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.3 Arvore * atualizar_fb_dir (Arvore * r)

Função que atualiza a altura e faz o balanceamento dos nós casos os dá direita estejam desbalanceados.

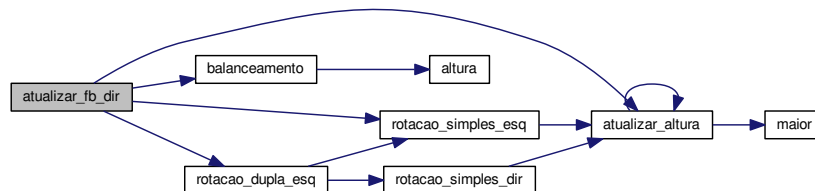
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

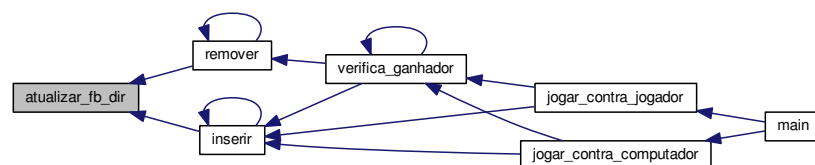
Returns

Retorna o ponteiro para o nó após as devidas atualizações.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.4 Arvore * atualizar_fb_esq (Arvore * r)

Função que atualiza a altura e faz o balanceamento dos nós casos os daá esquerda estejam desbalanceados.

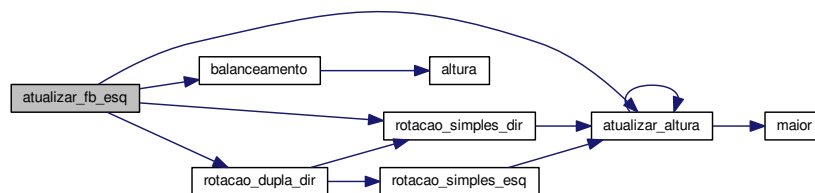
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

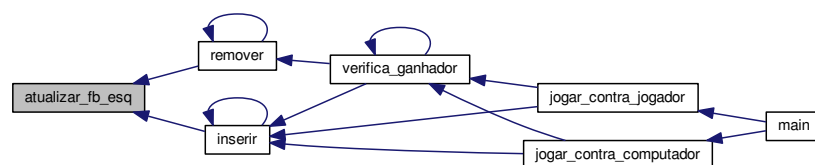
Returns

Retorna o ponteiro para o nó após as devidas alterações feitas.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.5 int balanceamento (Arvore * a)

Função que retorna a diferença das alturas do nós.

A função de balanceamento serve para retornar um valor que indica se o nó está balanceado ou não, retornando um valor que varia de -2 a 2.

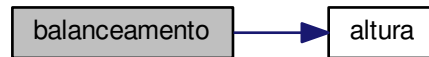
Parameters

<i>a</i>	Ponteiro que aponta para um nó da árvore AVL;
----------	---

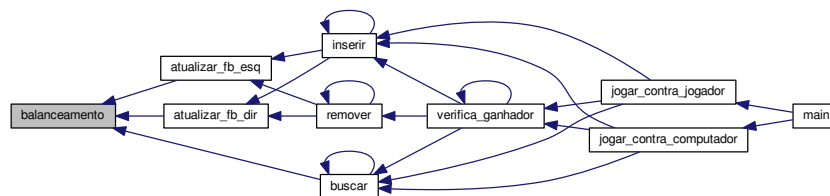
Returns

Retorna o valor do balanceamento

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.6 int buscar (*Arvore* * *a*, int *chave*)

Função que encontra um nó específico baseado em sua chave e retorna o valor da variável jogador.

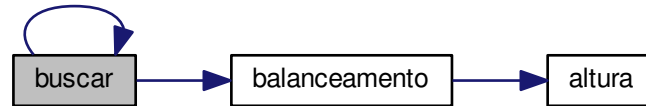
Parameters

<i>a</i>	Ponteiro que aponta para a árvore em que se deseja buscar.
<i>chave</i>	Inteiro que representa a chave do nó que se deseja achar.

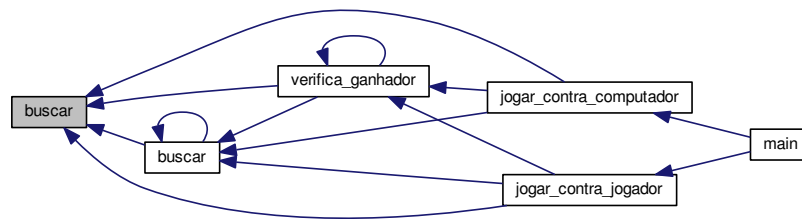
Returns

Retorna um inteiro com o mesmo valor da variável jogador do nó que se buscava.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.1.2.7 Arvore * inserir (Arvore * a, int chave, int jogador)**

Insere um novo nó na árvore AVL.

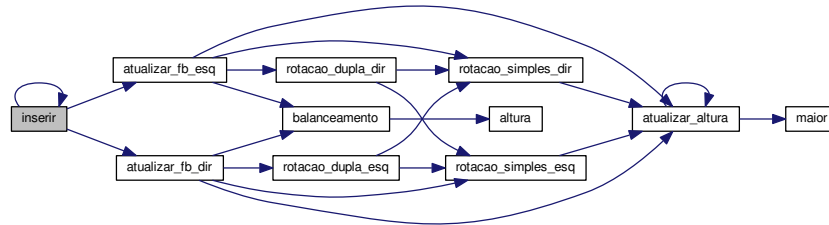
Parameters

<i>a</i>	É um ponteiro que representa a árvore onde se quer inserir.
<i>chave</i>	É um inteiro que representa o valor do nó.
<i>jogador</i>	É um inteiro que representa o jogador que fez a jogada.

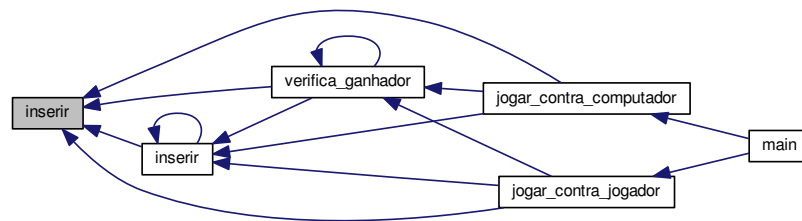
Returns

Retorna o ponteiro para árvore após a inserção ter sido feita.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.1.2.8 int maior (int *esq*, int *dir*)**

Função que ao receber 2 inteiro, retorna o maior deles.

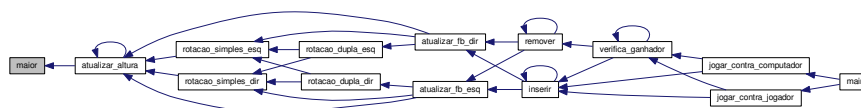
Parameters

<i>esq</i>	Inteiro que representa o valor a esquerda da AVL.
<i>dir</i>	Inteiro que representa o valor a direita da AVL.

Returns

Retorno inteiro que representa o maior de ambos os valores.

Here is the caller graph for this function:



6.1.2.9 Arvore * remover (Arvore * a, int chave)

Remove um nó da árvore AVL baseado na sua chave.

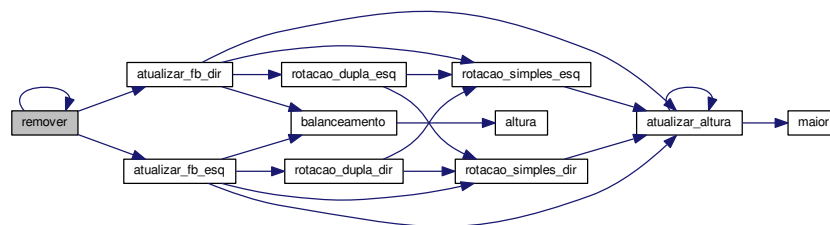
Parameters

<i>a</i>	É um ponteiro que aponta para a árvore ao qual se deseja remover o nó.
<i>chave</i>	É um inteiro que representa a chave do nó que se pretende remover

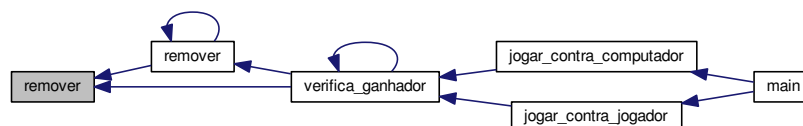
Returns

Retorna o ponteiro para a árvore, após ter sido removido o nó.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.10 Arvore * rotacao_dupla_dir (Arvore * r)

Função que realiza uma rotação dupla a direita na AVL.

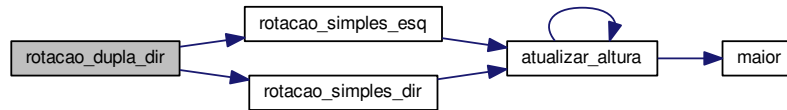
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

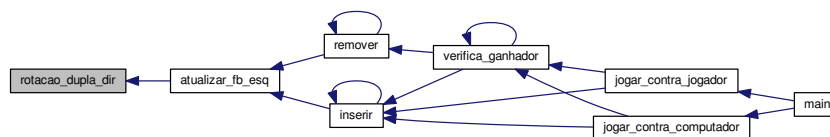
Returns

Retorna o ponteiro do nó da AVL após a rotação feita.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.11 `Arvore * rotacao_dupla_esq (Arvore * r)`

Função que realiza uma rotação dupla a esquerda na AVL.

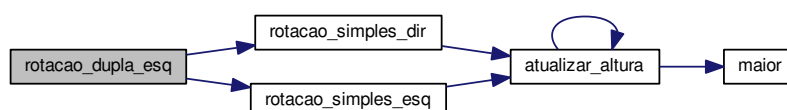
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

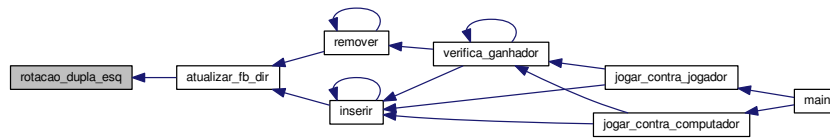
Returns

Retorna o ponteiro do nó da AVL após a rotação feita.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.12 `Arvore * rotacao_simples_dir (Arvore * r)`

Função que realiza uma rotação simples a direita na AVL.

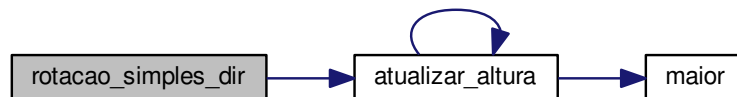
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

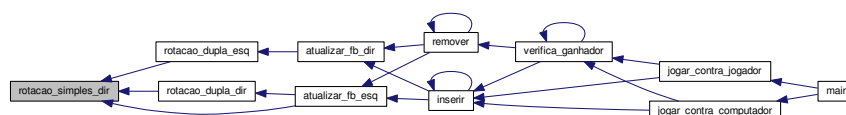
Returns

Retorna o ponteiro do nó da AVL após a rotação feita

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.13 `Arvore * rotacao_simples_esq (Arvore * r)`

Função que realiza uma rotação simples a esquerda na AVL.

Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL!
----------	---

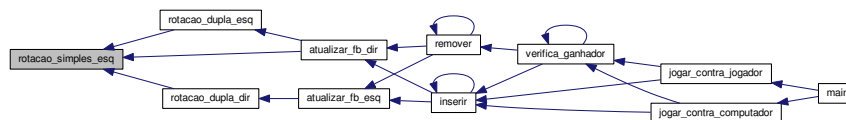
Returns

Retorna o ponteiro do nó da AVL após a rotação feita

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.14 int verifica_ganhador (Arvore * a, int jogador)

Essa função verifica se alguém ganhou o jogo e retorna um inteiro.

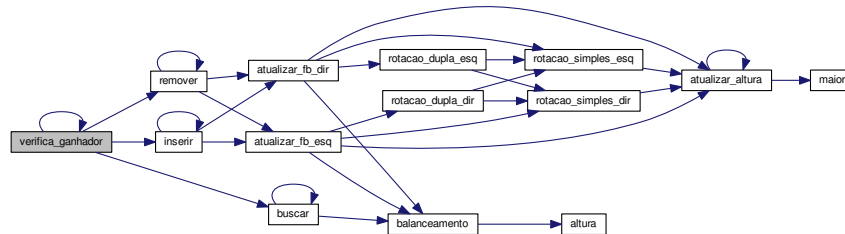
Parameters

<i>a</i>	É um ponteiro para a árvore AVL.
<i>jogador</i>	É um inteiro que representa qual jogador se está verificando.

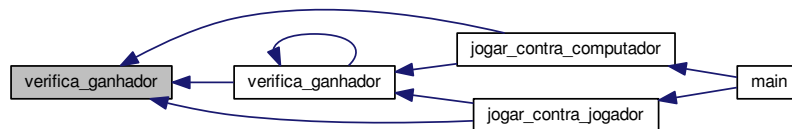
Returns

Inteiro de retorno

Here is the call graph for this function:



Here is the caller graph for this function:



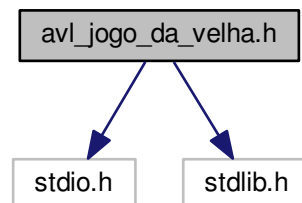
6.2 avl_jogo_da_velha.h File Reference

É uma biblioteca para manipulação de AVL.

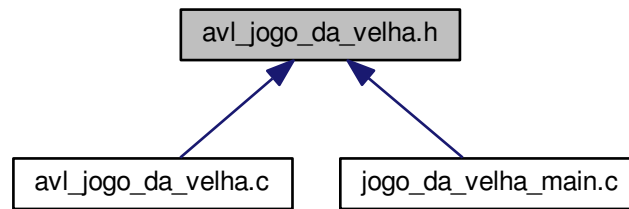
```
#include <stdio.h>
```

```
#include <stdlib.h>
```

Include dependency graph for avl_jogo_da_velha.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [node](#)

Typedefs

- typedef struct node **No**
- typedef struct node **Arvore**

Functions

- int [maior](#) (int esq, int dir)
Função que ao receber 2 inteiro, retorna o maior deles.
- int [altura](#) (Arvore *a)
Função retorna a altura de uma árvore.
- int [atualizar_altura](#) (Arvore *a)
Função recursiva que atualiza as altura da árvore.
- int [balanceamento](#) (Arvore *a)
Função que retorna a diferença das alturas do nós.
- Arvore * [rotacao_simples_esq](#) (Arvore *r)
Função que realiza uma rotação simples a esquerda na AVL.
- Arvore * [rotacao_simples_dir](#) (Arvore *r)
Função que realiza uma rotação simples a direita na AVL.
- Arvore * [rotacao_dupla_esq](#) (Arvore *r)
Função que realiza uma rotação dupla a esquerda na AVL.
- Arvore * [rotacao_dupla_dir](#) (Arvore *r)
Função que realiza uma rotação dupla a direita na AVL.
- Arvore * [atualizar_fb_dir](#) (Arvore *r)
- Arvore * [atualizar_fb_esq](#) (Arvore *r)
- Arvore * [inserir](#) (Arvore *a, int chave, int jogador)
Insere um novo nó na árvore AVL.
- Arvore * [remover](#) (Arvore *a, int chave)
Remove um nó da árvore AVL baseado na sua chave.
- int [buscar](#) (Arvore *a, int chave)

Função que encontra um nó específico baseado em sua chave e retorna o valor da variável jogador.

- void **imprimir_in_order** ([Arvore](#) *a, int nivel)
- void **arvore_para_vetor** ([Arvore](#) *a, int *tabuleiro)
- void **imprimir_tabuleiro** ([Arvore](#) *a)
- int **verifica_ganhador** ([Arvore](#) *a, int jogador)

Essa função verifica se alguém ganhou o jogo e retorna um inteiro.

- int **conta_nos** ([Arvore](#) *a)
- void **calcula_passos** ([Arvore](#) **verificacao, [Arvore](#) *a, [Arvore](#) *b, int *passos, int index)
- [Arvore](#) * **jogada_computador** ([Arvore](#) *a, [Arvore](#) *verificacao)
- int **jogada_jogador** ()
- int **menu_principal** ()

6.2.1 Detailed Description

É uma biblioteca para manipulação de AVL.

Essa biblioteca para manipulação de uma AVL especial que guarda e gerência dados para o funcionamento do jogo da velha.

Author

Lucas Ribeiro

Date

25/05/2017

6.2.2 Function Documentation

6.2.2.1 int altura ([Arvore](#) * a)

Função retorna a altura de uma árvore.

Precondition

O ponteiro para a árvore não deve ter valor NULL

Warning

Caso o pré requisito não seja atendido a função retornará NULL

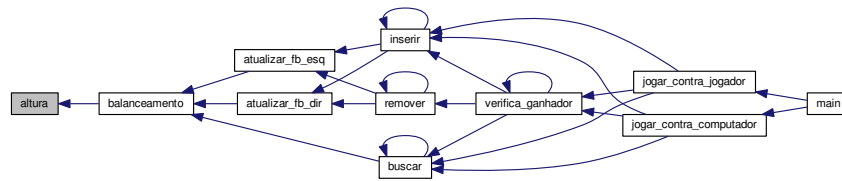
Parameters

<i>a</i>	Ponteiro que aponta para a árvore AVL.
----------	--

Returns

Retorna um inteiro que representa

Here is the caller graph for this function:



6.2.2.2 int atualizar_altura (Arvore * a)

Função recursiva que atualiza as altura da árvore.

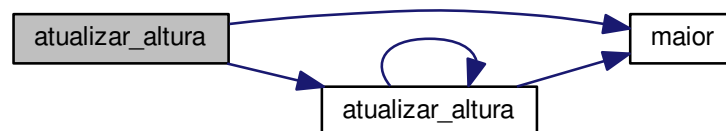
Parameters

<i>a</i>	Ponteiro que aponta para o início da AVL
----------	--

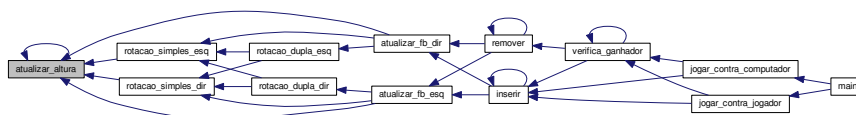
Returns

O retorno inteiro serve para o funcionamento da recursividade.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.3 Arvore* atualizar_fb_dir (Arvore * r)

Função que atualiza a altura e faz o balanceamento dos nós casos os da direita estejam desbalanceados.

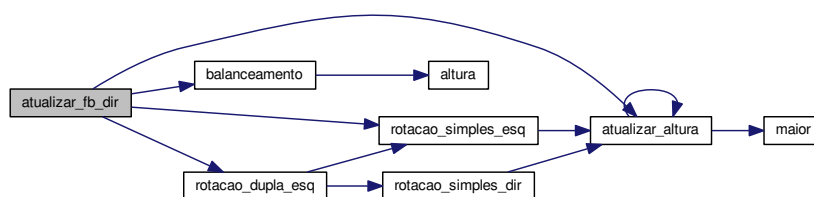
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

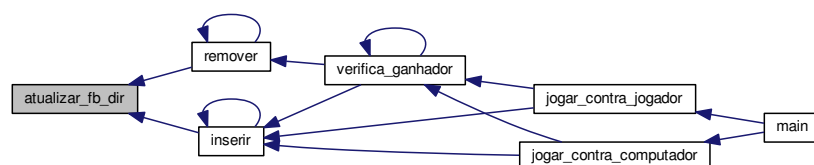
Returns

Retorna o ponteiro para o nó após as devidas atualizações.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.4 Arvore* atualizar_fb_esq (Arvore * r)

Função que atualiza a altura e faz o balanceamento dos nós casos os daá esquerda estejam desbalanceados.

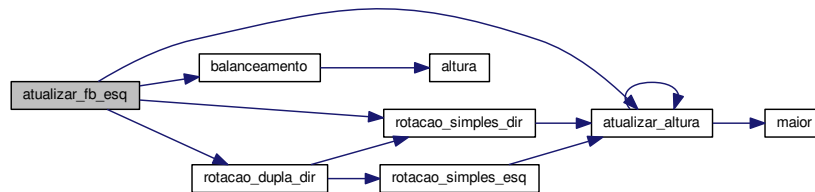
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

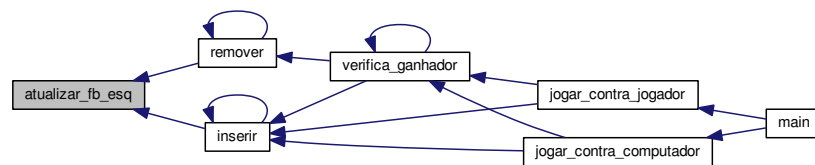
Returns

Retorna o ponteiro para o nó após as devidas alterações feitas.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.5 int balanceamento (Arvore * a)

Função que retorna a diferença das alturas do nós.

A função de balanceamento serve para retornar um valor que indica se o nó está balanceado ou não, retornando um valor que varia de -2 a 2.

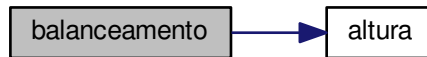
Parameters

<i>a</i>	Ponteiro que aponta para um nó da árvore AVL;
----------	---

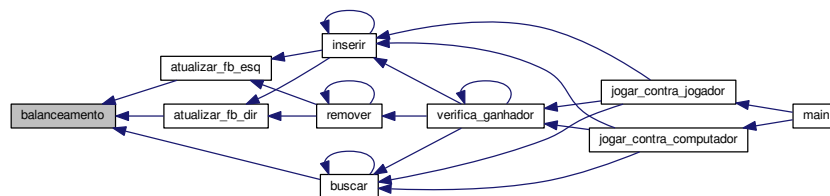
Returns

Retorna o valor do balanceamento

Here is the call graph for this function:



Here is the caller graph for this function:

**6.2.2.6 int buscar (Arvore * a, int chave)**

Função que encontra um nó específico baseado em sua chave e retorna o valor da variável jogador.

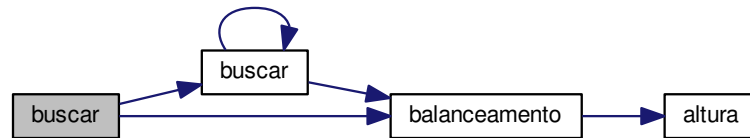
Parameters

<i>a</i>	Ponteiro que aponta para a árvore em que se deseja buscar.
<i>chave</i>	Inteiro que representa a chave do nó que se deseja achar.

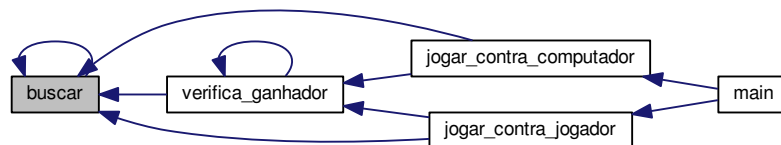
Returns

Retorna um inteiro com o mesmo valor da variável jogador do nó que se buscava.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.7 Arvore* inserir (Arvore * a, int chave, int jogador)

Insere um novo nó na árvore AVL.

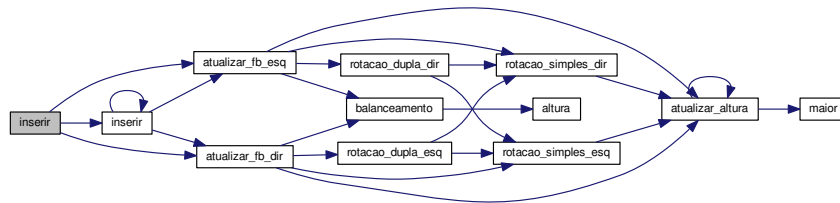
Parameters

<i>a</i>	É um ponteiro que representa a árvore onde se quer inserir.
<i>chave</i>	É um inteiro que representa o valor do nó.
<i>jogador</i>	É um inteiro que representa o jogador que fez a jogada.

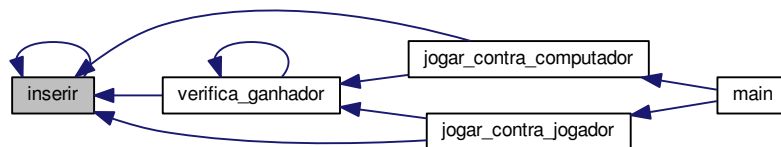
Returns

Retorna o ponteiro para árvore após a inserção ter sido feita.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.2.2.8 int maior (int *esq*, int *dir*)**

Função que ao receber 2 inteiro, retorna o maior deles.

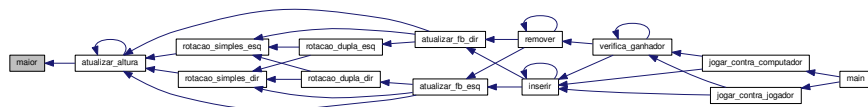
Parameters

<i>esq</i>	Inteiro que representa o valor a esquerda da AVL.
<i>dir</i>	Inteiro que representa o valor a direita da AVL.

Returns

Retorno inteiro que representa o maior de ambos os valores.

Here is the caller graph for this function:



6.2.2.9 Arvore* remover (Arvore * a, int chave)

Remove um nó da árvore AVL baseado na sua chave.

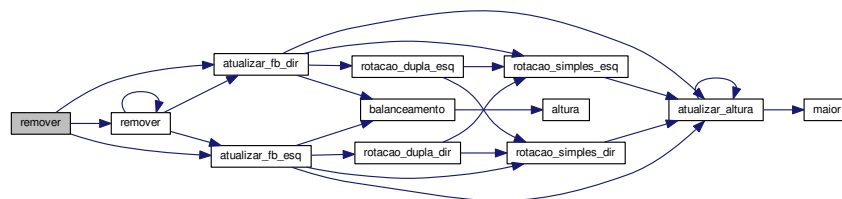
Parameters

<i>a</i>	É um ponteiro que aponta para a árvore ao qual se deseja remover o nó.
<i>chave</i>	É um inteiro que representa a chave do nó que se pretende remover

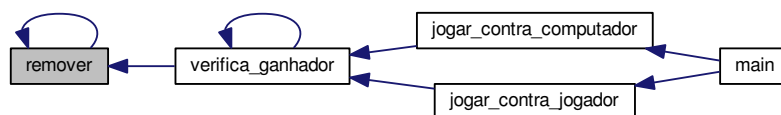
Returns

Retorna o ponteiro para a árvore, após ter sido removido o nó.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.10 Arvore* rotacao_dupla_dir (Arvore * r)

Função que realiza uma rotação dupla a direita na AVL.

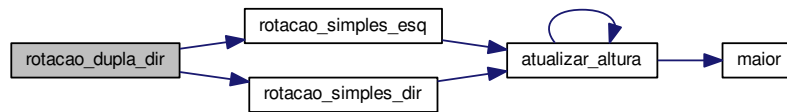
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

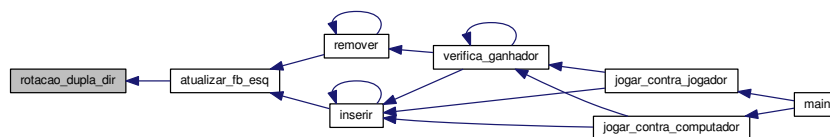
Returns

Retorna o ponteiro do nó da AVL após a rotação feita.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.11 Arvore* rotacao_dupla_esq (Arvore * r)

Função que realiza uma rotação dupla a esquerda na AVL.

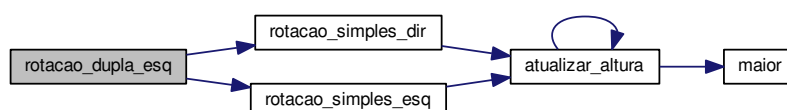
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

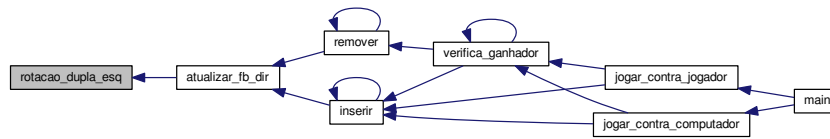
Returns

Retorna o ponteiro do nó da AVL após a rotação feita.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.12 `Arvore* rotacao_simples_dir (Arvore * r)`

Função que realiza uma rotação simples a direita na AVL.

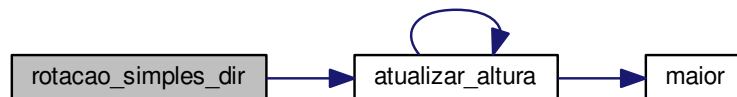
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL.
----------	---

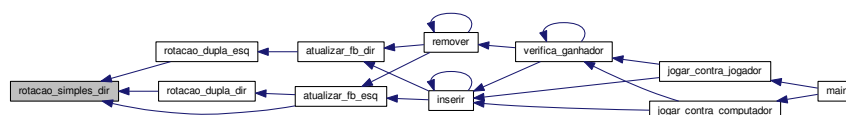
Returns

Retorna o ponteiro do nó da AVL após a rotação feita

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.13 `Arvore* rotacao_simples_esq (Arvore * r)`

Função que realiza uma rotação simples a esquerda na AVL.

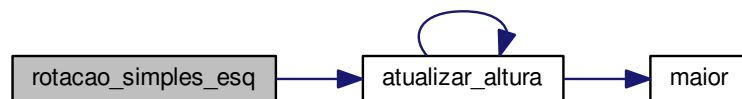
Parameters

<i>r</i>	Ponteiro que aponta para um nó da árvore AVL!
----------	---

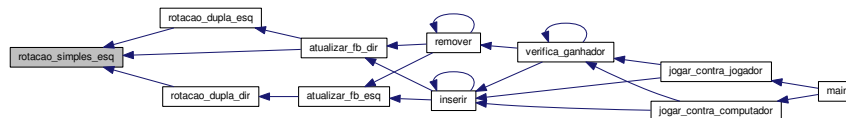
Returns

Retorna o ponteiro do nó da AVL após a rotação feita

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.2.14 int verifica_ganhador (Arvore * a, int jogador)

Essa função verifica se alguém ganhou o jogo e retorna um inteiro.

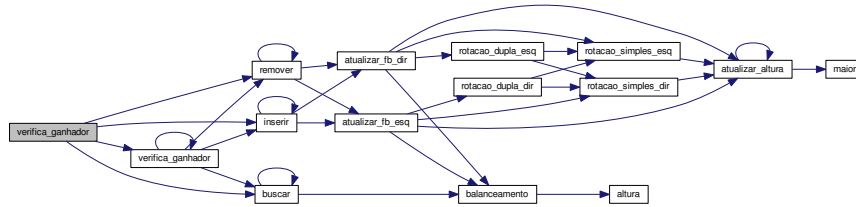
Parameters

<i>a</i>	É um ponteiro para a árvore AVL.
<i>jogador</i>	É um inteiro que representa qual jogador se está verificando.

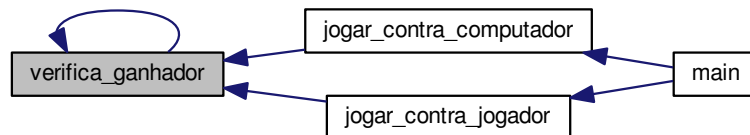
Returns

Inteiro de retorno

Here is the call graph for this function:



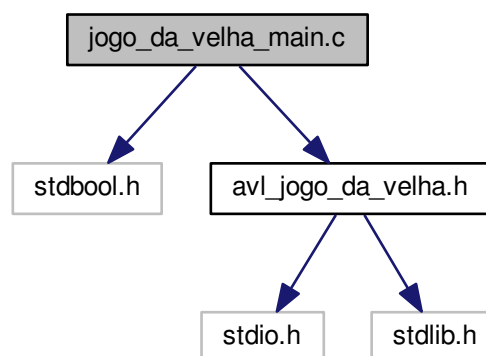
Here is the caller graph for this function:



6.3 jogo_da_velha_main.c File Reference

Código fonte do jogo da velha.

```
#include <stdbool.h>
#include "avl_jogo_da_velha.h"
Include dependency graph for jogo_da_velha_main.c:
```



Functions

- void `jogar_contra_computador` (`Arvore` *a)
Função que controla o jogo quando selecionado a opção Player vs IA.
- void `jogar_contra_jogador` (`Arvore` *a)
Função que controla o jogo quando é selecionado a opção de Player vc Player.
- int `main` ()
Função Main do programa C.

6.3.1 Detailed Description

Código fonte do jogo da velha.

Author

Lucas Ribeiro

Date

25/05/2017

6.3.2 Function Documentation

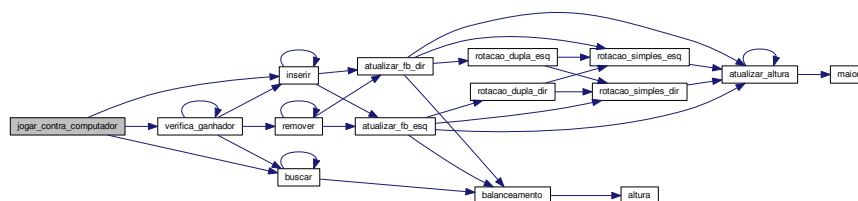
6.3.2.1 void jogar_contra_computador (`Arvore` * a)

Função que controla o jogo quando selecionado a opção Player vs IA.

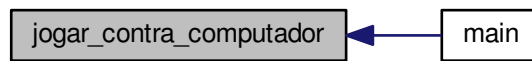
Parameters

<code>a</code>	Ponteiro para a árvore AVL
----------------	----------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



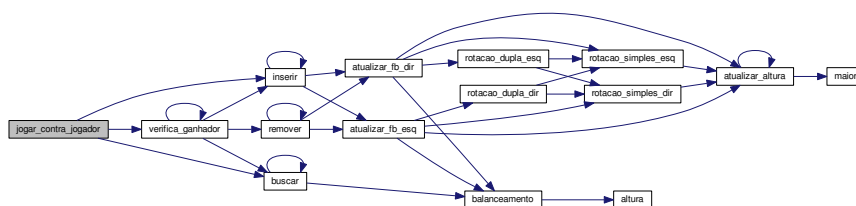
6.3.2.2 void jogar_contra_jogador (**Arvore** * *a*)

Função que controla o jogo quando é selecionado a opção de Player vc Player.

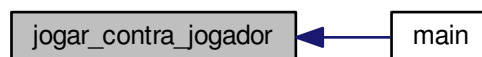
Parameters

<i>a</i>	Ponteiro para a árvore AVL.
----------	-----------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



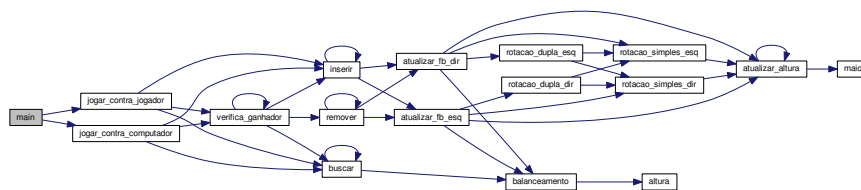
6.3.2.3 int main ()

Função Main do programa C.

Returns

Retorno da função Main

Here is the call graph for this function:



Index

altura

avl_jogo_da_velha.c, [12](#)
avl_jogo_da_velha.h, [25](#)
node, [9](#)

atualizar_altura

avl_jogo_da_velha.c, [13](#)
avl_jogo_da_velha.h, [26](#)

atualizar_fb_dir

avl_jogo_da_velha.c, [14](#)
avl_jogo_da_velha.h, [26](#)

atualizar_fb_esq

avl_jogo_da_velha.c, [14](#)
avl_jogo_da_velha.h, [27](#)

avl_jogo_da_velha.c, [11](#)

altura, [12](#)
atualizar_altura, [13](#)
atualizar_fb_dir, [14](#)
atualizar_fb_esq, [14](#)
balanceamento, [15](#)
buscar, [16](#)
inserir, [17](#)
maior, [18](#)
remover, [19](#)
rotacao_dupla_dir, [19](#)
rotacao_dupla_esq, [20](#)
rotacao_simples_dir, [21](#)
rotacao_simples_esq, [21](#)
verifica_ganhador, [22](#)

avl_jogo_da_velha.h, [23](#)

altura, [25](#)
atualizar_altura, [26](#)
atualizar_fb_dir, [26](#)
atualizar_fb_esq, [27](#)
balanceamento, [28](#)
buscar, [29](#)
inserir, [30](#)
maior, [31](#)
remover, [31](#)
rotacao_dupla_dir, [32](#)
rotacao_dupla_esq, [33](#)
rotacao_simples_dir, [34](#)
rotacao_simples_esq, [34](#)
verifica_ganhador, [35](#)

balanceamento

avl_jogo_da_velha.c, [15](#)
avl_jogo_da_velha.h, [28](#)

buscar

avl_jogo_da_velha.c, [16](#)
avl_jogo_da_velha.h, [29](#)

chave

node, [9](#)

dir

node, [10](#)

esq

node, [10](#)

inserir

avl_jogo_da_velha.c, [17](#)
avl_jogo_da_velha.h, [30](#)

jogador

node, [10](#)

jogar_contra_computador

jogo_da_velha_main.c, [37](#)

jogar_contra_jogador

jogo_da_velha_main.c, [38](#)

jogo_da_velha_main.c, [36](#)

jogar_contra_computador, [37](#)
jogar_contra_jogador, [38](#)
main, [38](#)

main

jogo_da_velha_main.c, [38](#)

maior

avl_jogo_da_velha.c, [18](#)
avl_jogo_da_velha.h, [31](#)

node, [9](#)

altura, [9](#)
chave, [9](#)
dir, [10](#)
esq, [10](#)
jogador, [10](#)

remover

avl_jogo_da_velha.c, [19](#)
avl_jogo_da_velha.h, [31](#)

rotacao_dupla_dir

avl_jogo_da_velha.c, [19](#)
avl_jogo_da_velha.h, [32](#)

rotacao_dupla_esq

avl_jogo_da_velha.c, [20](#)
avl_jogo_da_velha.h, [33](#)

rotacao_simples_dir

avl_jogo_da_velha.c, [21](#)
avl_jogo_da_velha.h, [34](#)

rotacao_simples_esq

avl_jogo_da_velha.c, [21](#)

avl_jogo_da_velha.h, [34](#)

struct, [10](#)

verifica_ganhador

avl_jogo_da_velha.c, [22](#)

avl_jogo_da_velha.h, [35](#)