

## 2nd Fase EDD

1.0

Generated by Doxygen 1.9.1



<b>1 Modules Index</b>	<b>1</b>
1.1 Modules List	1
<b>2 Data Type Index</b>	<b>3</b>
2.1 Data Types List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Module Documentation</b>	<b>7</b>
4.1 albums Module Reference	7
4.1.1 Function/Subroutine Documentation	8
4.1.1.1 add_image()	8
4.1.1.2 add_image_to_album()	8
4.1.1.3 gen_album_graph()	8
4.1.1.4 get_album()	9
4.1.1.5 get_image()	9
4.1.1.6 new_album()	10
4.1.1.7 remove_album()	10
4.1.1.8 remove_image()	10
4.1.1.9 remove_image_from_album()	11
4.1.1.10 search_in_album()	11
4.1.1.11 show_album_images()	12
4.1.1.12 show_albums()	12
4.1.1.13 show_images()	12
4.1.2 Variable Documentation	13
4.1.2.1 album_id	13
4.2 clients Module Reference	13
4.2.1 Function/Subroutine Documentation	14
4.2.1.1 add_client()	14
4.2.1.2 amplitude_traversal()	15
4.2.1.3 clients_dot()	15
4.2.1.4 create_node()	15
4.2.1.5 delete_client()	16
4.2.1.6 delete_client_rec()	16
4.2.1.7 dequeue()	17
4.2.1.8 enqueue()	17
4.2.1.9 find()	18
4.2.1.10 insert_node()	18
4.2.1.11 is_empty()	19
4.2.1.12 search_client()	19
4.2.1.13 set_value()	19
4.2.1.14 split_node()	20

4.2.1.15 traversal()	21
4.2.2 Variable Documentation	22
4.2.2.1 g_id	22
4.3 filehandler Module Reference	22
4.3.1 Function/Subroutine Documentation	22
4.3.1.1 initialize_admin()	22
4.3.1.2 initialize_user()	23
4.3.1.3 read_albums()	23
4.3.1.4 read_clients()	23
4.3.1.5 read_imgs()	24
4.3.1.6 read_layers()	24
4.3.1.7 set_user()	24
4.4 images Module Reference	25
4.4.1 Function/Subroutine Documentation	26
4.4.1.1 add_img()	26
4.4.1.2 add_img_rec()	26
4.4.1.3 add_layer()	26
4.4.1.4 delete_img()	27
4.4.1.5 delete_img_rec()	27
4.4.1.6 drl()	28
4.4.1.7 drr()	28
4.4.1.8 gen_img_traversal()	28
4.4.1.9 gen_layer_subtree()	29
4.4.1.10 gen_tree_subtree()	29
4.4.1.11 get_dot()	30
4.4.1.12 get_dot_rec()	30
4.4.1.13 get_height()	31
4.4.1.14 get_max()	31
4.4.1.15 min_child()	32
4.4.1.16 print_images()	32
4.4.1.17 search_img()	32
4.4.1.18 srl()	33
4.4.1.19 srr()	33
4.5 layers Module Reference	34
4.5.1 Function/Subroutine Documentation	35
4.5.1.1 add()	35
4.5.1.2 add_copied_val()	35
4.5.1.3 add_recursive()	36
4.5.1.4 dequeue()	36
4.5.1.5 enqueue()	36
4.5.1.6 gen_dot()	37
4.5.1.7 gen_dot_recursive()	37

4.5.1.8 <code>gen_inorder()</code>	37
4.5.1.9 <code>gen_postorder()</code>	38
4.5.1.10 <code>gen_preorder()</code>	38
4.5.1.11 <code>inorder()</code>	39
4.5.1.12 <code>inorder_print()</code>	39
4.5.1.13 <code>is_empty()</code>	39
4.5.1.14 <code>leaf_layers()</code>	40
4.5.1.15 <code>leaf_layers_rec()</code>	40
4.5.1.16 <code>list_layers()</code>	40
4.5.1.17 <code>max_depth()</code>	41
4.5.1.18 <code>max_depth_rec()</code>	41
4.5.1.19 <code>postorder()</code>	41
4.5.1.20 <code>preorder()</code>	43
4.5.1.21 <code>search()</code>	43
4.5.1.22 <code>traverse_limited()</code>	44
4.5.1.23 <code>traverse_matrix()</code>	44
4.6 pixels Module Reference	44
4.6.1 Function/Subroutine Documentation	45
4.6.1.1 <code>gen_matrix()</code>	45
4.6.1.2 <code>get_node()</code>	46
4.6.1.3 <code>get_value()</code>	46
4.6.1.4 <code>global_m_dot()</code>	47
4.6.1.5 <code>graph_pixels()</code>	47
4.6.1.6 <code>insert()</code>	47
4.6.1.7 <code>insert_column_header()</code>	48
4.6.1.8 <code>insert_in_column()</code>	48
4.6.1.9 <code>insert_in_row()</code>	48
4.6.1.10 <code>insert_row_header()</code>	49
4.6.1.11 <code>node_exists()</code>	49
4.6.1.12 <code>print_headers()</code>	50
4.6.1.13 <code>search_column()</code>	50
4.6.1.14 <code>search_row()</code>	50
4.6.1.15 <code>self_print()</code>	51
4.6.2 Variable Documentation	51
4.6.2.1 <code>id</code>	51
<b>5 Data Type Documentation</b>	<b>53</b>
5.1 <code>albums::album</code> Type Reference	53
5.1.1 Detailed Description	55
5.1.2 Member Function/Subroutine Documentation	55
5.1.2.1 <code>add_image()</code>	55
5.1.2.2 <code>get_image()</code>	55

5.1.2.3 remove_image()	55
5.1.2.4 show_images()	56
5.1.3 Member Data Documentation	56
5.1.3.1 head	56
5.1.3.2 id	56
5.1.3.3 name	56
5.1.3.4 next	56
5.1.3.5 prev	57
5.1.3.6 size	57
5.1.3.7 tail	57
5.2 albums::album_list Type Reference	57
5.2.1 Detailed Description	59
5.2.2 Member Function/Subroutine Documentation	59
5.2.2.1 add_image_to_album()	59
5.2.2.2 gen_album_graph()	59
5.2.2.3 get_album()	59
5.2.2.4 new_album()	60
5.2.2.5 remove_album()	60
5.2.2.6 remove_image_from_album()	60
5.2.2.7 search_in_album()	60
5.2.2.8 show_album_images()	60
5.2.2.9 show_albums()	60
5.2.3 Member Data Documentation	61
5.2.3.1 head	61
5.2.3.2 size	61
5.2.3.3 tail	61
5.3 clients::btree_clients Type Reference	62
5.3.1 Detailed Description	63
5.3.2 Member Function/Subroutine Documentation	63
5.3.2.1 add_client()	63
5.3.2.2 amplitude_traversal()	63
5.3.2.3 clients_dot()	63
5.3.2.4 create_node()	64
5.3.2.5 delete_client()	64
5.3.2.6 delete_client_rec()	64
5.3.2.7 search_client()	64
5.3.2.8 traversal()	64
5.3.3 Member Data Documentation	64
5.3.3.1 root	65
5.4 clients::btreenode Type Reference	65
5.4.1 Detailed Description	67
5.4.2 Member Function/Subroutine Documentation	67

5.4.2.1 find()	67
5.4.3 Member Data Documentation	67
5.4.3.1 clients	67
5.4.3.2 id	67
5.4.3.3 links	68
5.4.3.4 num	68
5.5 clients::client Type Reference	68
5.5.1 Detailed Description	70
5.5.2 Member Data Documentation	70
5.5.2.1 all_images	70
5.5.2.2 all_layers	70
5.5.2.3 dpi	70
5.5.2.4 list_albums	71
5.5.2.5 name	71
5.5.2.6 password	71
5.6 clients::client_queue Type Reference	71
5.6.1 Detailed Description	73
5.6.2 Member Function/Subroutine Documentation	73
5.6.2.1 dequeue()	73
5.6.2.2 enqueue()	73
5.6.2.3 is_empty()	73
5.6.3 Member Data Documentation	73
5.6.3.1 head	74
5.7 filehandler::fhandler Type Reference	74
5.7.1 Detailed Description	76
5.7.2 Member Function/Subroutine Documentation	76
5.7.2.1 initialize_admin()	76
5.7.2.2 initialize_user()	76
5.7.2.3 read_albums()	76
5.7.2.4 read_clients()	77
5.7.2.5 read_imgs()	77
5.7.2.6 read_layers()	77
5.7.2.7 set_user()	77
5.7.3 Member Data Documentation	77
5.7.3.1 albums_db	77
5.7.3.2 clients_db	78
5.7.3.3 images_db	78
5.7.3.4 layers_db	78
5.8 images::image Type Reference	78
5.8.1 Detailed Description	80
5.8.2 Member Function/Subroutine Documentation	80
5.8.2.1 add_layer()	80

5.8.3 Member Data Documentation	80
5.8.3.1 height	80
5.8.3.2 id	80
5.8.3.3 layers	80
5.8.3.4 layers_count	81
5.8.3.5 left	81
5.8.3.6 right	81
5.9 images::image_avl Type Reference	81
5.9.1 Detailed Description	83
5.9.2 Member Function/Subroutine Documentation	83
5.9.2.1 add_img()	83
5.9.2.2 add_img_rec()	83
5.9.2.3 delete_img()	83
5.9.2.4 delete_img_rec()	84
5.9.2.5 drl()	84
5.9.2.6 drr()	84
5.9.2.7 gen_img_traversal()	84
5.9.2.8 gen_tree_subtree()	84
5.9.2.9 get_dot()	84
5.9.2.10 get_dot_rec()	85
5.9.2.11 get_height()	85
5.9.2.12 get_max()	85
5.9.2.13 min_child()	85
5.9.2.14 print_images()	85
5.9.2.15 search_img()	85
5.9.2.16 srl()	86
5.9.2.17 srr()	86
5.9.3 Member Data Documentation	86
5.9.3.1 root	86
5.9.3.2 total	86
5.10 albums::img_node Type Reference	87
5.10.1 Detailed Description	88
5.10.2 Member Data Documentation	88
5.10.2.1 id	88
5.10.2.2 img_pointer	88
5.10.2.3 next	88
5.11 layers::layer Type Reference	89
5.11.1 Detailed Description	89
5.11.2 Member Data Documentation	90
5.11.2.1 id	90
5.11.2.2 layer_pixels	90
5.11.2.3 left	90



5.11.2.4 pixels_count . . . . .	90
5.11.2.5 right . . . . .	91
5.12 layers::layers_tree Type Reference . . . . .	91
5.12.1 Detailed Description . . . . .	92
5.12.2 Member Function/Subroutine Documentation . . . . .	92
5.12.2.1 add() . . . . .	92
5.12.2.2 add_copied_val() . . . . .	93
5.12.2.3 add_recursive() . . . . .	93
5.12.2.4 gen_dot() . . . . .	93
5.12.2.5 gen_dot_recursive() . . . . .	93
5.12.2.6 gen_inorder() . . . . .	93
5.12.2.7 gen_postorder() . . . . .	93
5.12.2.8 gen_preorder() . . . . .	94
5.12.2.9 inorder() . . . . .	94
5.12.2.10 inorder_print() . . . . .	94
5.12.2.11 leaf_layers() . . . . .	94
5.12.2.12 leaf_layers_rec() . . . . .	94
5.12.2.13 list_layers() . . . . .	94
5.12.2.14 max_depth() . . . . .	95
5.12.2.15 max_depth_rec() . . . . .	95
5.12.2.16 postorder() . . . . .	95
5.12.2.17 preorder() . . . . .	95
5.12.2.18 search() . . . . .	95
5.12.2.19 traverse_limited() . . . . .	95
5.12.2.20 traverse_matrix() . . . . .	96
5.12.3 Member Data Documentation . . . . .	96
5.12.3.1 global_matrix . . . . .	96
5.12.3.2 root . . . . .	96
5.12.3.3 total . . . . .	96
5.13 layers::node_layer Type Reference . . . . .	97
5.13.1 Detailed Description . . . . .	97
5.13.2 Member Data Documentation . . . . .	97
5.13.2.1 next . . . . .	98
5.13.2.2 value . . . . .	98
5.14 clients::nodeptr Type Reference . . . . .	98
5.14.1 Detailed Description . . . . .	100
5.14.2 Member Data Documentation . . . . .	100
5.14.2.1 ptr . . . . .	100
5.15 pixels::pixel Type Reference . . . . .	100
5.15.1 Detailed Description . . . . .	101
5.15.2 Member Data Documentation . . . . .	101
5.15.2.1 color . . . . .	101

5.15.2.2 down	101
5.15.2.3 id	102
5.15.2.4 left	102
5.15.2.5 on	102
5.15.2.6 right	102
5.15.2.7 up	102
5.15.2.8 x	103
5.15.2.9 y	103
5.16 pixels::pixel_matrix Type Reference	103
5.16.1 Detailed Description	104
5.16.2 Member Function/Subroutine Documentation	104
5.16.2.1 gen_matrix()	104
5.16.2.2 get_node()	105
5.16.2.3 get_value()	105
5.16.2.4 global_m_dot()	105
5.16.2.5 graph_pixels()	105
5.16.2.6 insert()	105
5.16.2.7 insert_column_header()	105
5.16.2.8 insert_in_column()	106
5.16.2.9 insert_in_row()	106
5.16.2.10 insert_row_header()	106
5.16.2.11 node_exists()	106
5.16.2.12 print_headers()	106
5.16.2.13 search_column()	106
5.16.2.14 search_row()	107
5.16.2.15 self_print()	107
5.16.3 Member Data Documentation	107
5.16.3.1 height	107
5.16.3.2 root	107
5.16.3.3 width	107
5.17 clients::q_node Type Reference	108
5.17.1 Detailed Description	109
5.17.2 Member Data Documentation	109
5.17.2.1 next	109
5.17.2.2 value	109
5.18 layers::queue Type Reference	110
5.18.1 Detailed Description	111
5.18.2 Member Function/Subroutine Documentation	111
5.18.2.1 dequeue()	111
5.18.2.2 enqueue()	111
5.18.2.3 is_empty()	111
5.18.3 Member Data Documentation	111

---

5.18.3.1 head . . . . .	111
<b>6 File Documentation</b>	<b>113</b>
6.1 /home/diego/Documents/EDD/Fase 2 Documentation/src/albums.f90 File Reference . . . . .	113
6.1.1 Detailed Description . . . . .	114
6.2 /home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90 File Reference . . . . .	114
6.2.1 Detailed Description . . . . .	116
6.3 /home/diego/Documents/EDD/Fase 2 Documentation/src/filehandler.f90 File Reference . . . . .	116
6.3.1 Detailed Description . . . . .	117
6.4 /home/diego/Documents/EDD/Fase 2 Documentation/src/images.f90 File Reference . . . . .	117
6.4.1 Detailed Description . . . . .	119
6.5 /home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90 File Reference . . . . .	119
6.5.1 Detailed Description . . . . .	121
6.6 /home/diego/Documents/EDD/Fase 2 Documentation/src/pixels.f90 File Reference . . . . .	121
6.6.1 Detailed Description . . . . .	122
<b>Index</b>	<b>123</b>



# Chapter 1

## Modules Index

### 1.1 Modules List

Here is a list of all modules with brief descriptions:

<a href="#">albums</a>	7
<a href="#">clients</a>	13
<a href="#">filehandler</a>	22
<a href="#">images</a>	25
<a href="#">layers</a>	34
<a href="#">pixels</a>	44



## Chapter 2

# Data Type Index

### 2.1 Data Types List

Here are the data types with brief descriptions:

<a href="#">albums::album</a>	Type to store the album . . . . .	53
<a href="#">albums::album_list</a>	Type to store the album list . . . . .	57
<a href="#">clients::btree_clients</a>	Type to represent a B-tree . . . . .	62
<a href="#">clients::breenode</a>	Type to represent a B-tree node . . . . .	65
<a href="#">clients::client</a>	Type to represent a client . . . . .	68
<a href="#">clients::client_queue</a>	Type to represent a client queue . . . . .	71
<a href="#">filehandler::fhandler</a>	Type that handles the reading of the json files and the initialization of the data structures . . . .	74
<a href="#">images::image</a>	Type that represents a node in the AVL tree . . . . .	78
<a href="#">images::image_avl</a>	Type that represents an AVL tree of images . . . . .	81
<a href="#">albums::img_node</a>	Type to store the image node . . . . .	87
<a href="#">layers::layer</a>	Type to represent a pixel matrix 89	
<a href="#">layers::layers_tree</a>	Type to represent a binary tree of layers . . . . .	91
<a href="#">layers::node_layer</a>	Type to represent a node of a linked list of layers . . . . .	97
<a href="#">clients::nodeptr</a>	Type to represent a node pointer . . . . .	98
<a href="#">pixels::pixel</a>	Type to represent a pixel . . . . .	100
<a href="#">pixels::pixel_matrix</a>	Type to represent a matrix of pixels . . . . .	103
<a href="#">clients::q_node</a>	Type to represent a queue node . . . . .	108
<a href="#">layers::queue</a>	Type to represent a queue of layers . . . . .	110





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

/home/diego/Documents/EDD/Fase 2 Documentation/src/ <a href="#">albums.f90</a>	
Module for albums and images management . . . . .	113
/home/diego/Documents/EDD/Fase 2 Documentation/src/ <a href="#">clients.f90</a>	
Module to manage clients in a B-tree . . . . .	114
/home/diego/Documents/EDD/Fase 2 Documentation/src/ <a href="#">filehandler.f90</a>	
Module that handles the reading of the json files and the initialization of the data structures . .	116
/home/diego/Documents/EDD/Fase 2 Documentation/src/ <a href="#">images.f90</a>	
Module that contains the image and image_avl types and their methods . . . . .	117
/home/diego/Documents/EDD/Fase 2 Documentation/src/ <a href="#">layers.f90</a>	
Module to handle layers of a neural network . . . . .	119
/home/diego/Documents/EDD/Fase 2 Documentation/src/ <a href="#">pixels.f90</a>	
Module to handle pixels in a matrix . . . . .	121



## Chapter 4

# Module Documentation

### 4.1 albums Module Reference

#### Data Types

- type [img\\_node](#)  
*Type to store the image node.*
- type [album](#)  
*Type to store the album.*
- type [album\\_list](#)  
*Type to store the album list.*

#### Functions/Subroutines

- subroutine [gen\\_album\\_graph](#) (this, unit)  
*Generates the graph of the albums.*
- subroutine [show\\_album\\_images](#) (this, id\_album)  
*Shows the images of an album.*
- subroutine [show\\_albums](#) (this)  
*Shows the albums.*
- subroutine [remove\\_image\\_from\\_album](#) (this, ralbum\_id, image\_id)  
*Removes an image from an album.*
- subroutine [add\\_image\\_to\\_album](#) (this, ialbum\_id, image\_node)  
*Adds an image to an album.*
- type([image](#)) function, pointer [search\\_in\\_album](#) (this, salbum\_id, image\_id)  
*Searches an image in an album.*
- type([album](#)) function, pointer [get\\_album](#) (this, id)  
*Gets an album.*
- subroutine [remove\\_album](#) (this, id)  
*Removes an album.*
- subroutine [new\\_album](#) (this, album\_name)  
*Creates a new album.*
- subroutine [add\\_image](#) (this, image\_node)  
*Adds an image to an album.*
- subroutine [remove\\_image](#) (this, id)  
*Removes an image from an album.*
- type([image](#)) function, pointer [get\\_image](#) (this, id)  
*Gets an image.*
- subroutine [show\\_images](#) (this)  
*Shows the images of an album.*

## Variables

- integer `album_id` = 0  
*Global variable to store the album id.*

### 4.1.1 Function/Subroutine Documentation

#### 4.1.1.1 `add_image()`

```
subroutine albums::add_image (
    class(album), intent(inout) this,
    type(image), intent(in), pointer image_node )
```

Adds an image to an album.

##### Parameters

<i>this</i>	Album
<i>image_node</i>	Image node

Definition at line 253 of file albums.f90.

#### 4.1.1.2 `add_image_to_album()`

```
subroutine albums::add_image_to_album (
    class(album_list), intent(inout) this,
    integer, intent(in) ialbum_id,
    type(image), intent(in), pointer image_node )
```

Adds an image to an album.

##### Parameters

<i>this</i>	Album list
<i>ialbum_id</i>	Album id
<i>image_node</i>	Image node

Definition at line 139 of file albums.f90.

#### 4.1.1.3 `gen_album_graph()`

```
subroutine albums::gen_album_graph (
```

```
class(album_list), intent(in) this,  
integer, intent(in) unit )
```

Generates the graph of the albums.

#### Parameters

<i>this</i>	Album list
<i>unit</i>	Unit to write the graph

Definition at line 52 of file albums.f90.

#### 4.1.1.4 get\_album()

```
type(album) function, pointer albums::get_album (  
    class(album_list), intent(in) this,  
    integer, intent(in) id )
```

Gets an album.

#### Parameters

<i>this</i>	Album list
<i>id</i>	Album id

#### Returns

Album node

Definition at line 173 of file albums.f90.

#### 4.1.1.5 get\_image()

```
type(image) function, pointer albums::get_image (  
    class(album), intent(in) this,  
    integer, intent(in) id )
```

Gets an image.

#### Parameters

<i>this</i>	Album
<i>id</i>	Image id

**Returns**

Image node

Definition at line 310 of file albums.f90.

**4.1.1.6 new\_album()**

```
subroutine albums::new_album (  
    class(album_list), intent(inout) this,  
    character(*), intent(in) album_name )
```

Creates a new album.

**Parameters**

<i>this</i>	Album list
<i>album_name</i>	Album name

Definition at line 232 of file albums.f90.

**4.1.1.7 remove\_album()**

```
subroutine albums::remove_album (  
    class(album_list), intent(inout) this,  
    integer, intent(in) id )
```

Removes an album.

**Parameters**

<i>this</i>	Album list
<i>id</i>	Album id

Definition at line 194 of file albums.f90.

**4.1.1.8 remove\_image()**

```
subroutine albums::remove_image (  
    class(album), intent(inout) this,  
    integer, intent(in) id )
```

Removes an image from an album.

## Parameters

<i>this</i>	Album
<i>id</i>	Image id

Definition at line 272 of file albums.f90.

**4.1.1.9 remove\_image\_from\_album()**

```
subroutine albums::remove_image_from_album (
    class(album_list), intent(inout) this,
    integer, intent(in) ralbum_id,
    integer, intent(in) image_id )
```

Removes an image from an album.

## Parameters

<i>this</i>	Album list
<i>ralbum_id</i>	Album id
<i>image_id</i>	Image id

Definition at line 123 of file albums.f90.

**4.1.1.10 search\_in\_album()**

```
type(image) function, pointer albums::search_in_album (
    class(album_list), intent(in) this,
    integer, intent(in) salbum_id,
    integer, intent(in) image_id )
```

Searches an image in an album.

## Parameters

<i>this</i>	Album list
<i>salbum_id</i>	Album id
<i>image_id</i>	Image id

## Returns

Image node

Definition at line 156 of file albums.f90.

#### 4.1.1.11 show\_album\_images()

```
subroutine albums::show_album_images (  
    class(album_list), intent(in) this,  
    integer, intent(in) id_album )
```

Shows the images of an album.

##### Parameters

<i>this</i>	Album list
<i>id_album</i>	Album id

Definition at line 92 of file albums.f90.

#### 4.1.1.12 show\_albums()

```
subroutine albums::show_albums (  
    class(album_list), intent(in) this )
```

Shows the albums.

##### Parameters

<i>this</i>	Album list
-------------	------------

Definition at line 106 of file albums.f90.

#### 4.1.1.13 show\_images()

```
subroutine albums::show_images (  
    class(album), intent(in) this )
```

Shows the images of an album.

##### Parameters

<i>this</i>	Album
-------------	-------

Definition at line 332 of file albums.f90.



## 4.1.2 Variable Documentation

### 4.1.2.1 album\_id

```
integer albums::album_id = 0
```

Global variable to store the album id.

Definition at line 10 of file albums.f90.

## 4.2 clients Module Reference

### Data Types

- type [client](#)  
*Type to represent a client.*
- type [nodeptr](#)  
*Type to represent a node pointer.*
- type [btreenode](#)  
*Type to represent a B-tree node.*
- type [btree\\_clients](#)  
*Type to represent a B-tree.*
- type [q\\_node](#)  
*Type to represent a queue node.*
- type [client\\_queue](#)  
*Type to represent a client queue.*

### Functions/Subroutines

- subroutine [amplitude\\_traversal](#) (this)  
*Subroutine to traverse the tree in amplitude.*
- logical function [is\\_empty](#) (this)  
*Function to check if the queue is empty.*
- type([btreenode](#)) function, pointer [dequeue](#) (this)  
*Function to dequeue a node from the queue.*
- subroutine [enqueue](#) (this, tree\_node)  
*Subroutine to enqueue a node in the queue.*
- subroutine [add\\_client](#) (this, new\_client)  
*Subroutine to add a client to the tree.*
- recursive logical function [set\\_value](#) (new\_client, pclient, node, child)  
*Recursive function to set the value of the tree.*
- type([btreenode](#)) function, pointer [create\\_node](#) (this, new\_client, child)  
*Function to create a node.*
- subroutine [insert\\_node](#) (pclient, pos, node, child)  
*Subroutine to insert a node.*
- subroutine [split\\_node](#) (new\_client, pclient, pos, node, child, new\_node)

- Subroutine to split a node.*
  - subroutine `traversal` (`this`, `node`)
    - Subroutine to traverse the tree.*
  - subroutine `clients_dot` (`this`, `node`, `unit`)
    - Subroutine to print the tree in dot format.*
  - recursive type(`client`) function, pointer `search_client` (`this`, `node`, `client_id`)
    - Function to search a client in the tree.*
  - subroutine `delete_client` (`this`, `client_id`)
    - Subroutine to delete a client from the tree.*
  - subroutine `delete_client_rec` (`this`, `father`, `temp`, `client_id`)
    - Recursive subroutine to delete a client from the tree.*
  - logical function `find` (`this`, `client_id`)
    - Function to find a client in the node.*

## Variables

- integer `g_id` = 1
  - Global variable to assign the id of the nodes.*

## 4.2.1 Function/Subroutine Documentation

### 4.2.1.1 `add_client()`

```
subroutine clients::add_client (
    class(btrees_clients), intent(inout) this,
    type(client), intent(in) new_client )
```

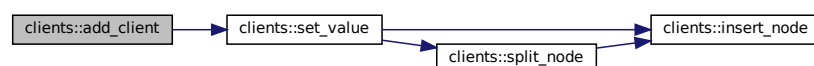
Subroutine to add a client to the tree.

#### Parameters

<i>this</i>	B-tree to add the client
<i>new_client</i>	Client to add

Definition at line 133 of file `clients.f90`.

Here is the call graph for this function:



#### 4.2.1.2 amplitude\_traversal()

```
subroutine clients::amplitude_traversal (
    class(btrees_clients), intent(inout) this )
```

Subroutine to traverse the tree in amplitude.

##### Parameters

<i>this</i>	B-tree to traverse
-------------	--------------------

Definition at line 65 of file clients.f90.

#### 4.2.1.3 clients\_dot()

```
subroutine clients::clients_dot (
    class(btrees_clients), intent(inout) this,
    type(btreenode), intent(in), pointer node,
    integer, intent(in) unit )
```

Subroutine to print the tree in dot format.

##### Parameters

<i>this</i>	B-tree to print
<i>node</i>	Node to print
<i>unit</i>	Unit to print

Definition at line 303 of file clients.f90.

#### 4.2.1.4 create\_node()

```
type(btreenode) function, pointer clients::create_node (
    class(btrees_clients), intent(inout) this,
    type(client), intent(in) new_client,
    type(btreenode), intent(in), pointer child )
```

Function to create a node.

##### Parameters

<i>this</i>	B-tree to create the node
<i>new_client</i>	Client to add
<i>child</i>	Child node

**Returns**

New node created

Definition at line 198 of file clients.f90.

**4.2.1.5 delete\_client()**

```
subroutine clients::delete_client (
    class(btrees_clients), intent(inout) this,
    integer(kind=8), intent(in) client_id )
```

Subroutine to delete a client from the tree.

**Parameters**

<i>this</i>	B-tree to delete the client
<i>client_id</i>	Id of the client to delete

Definition at line 358 of file clients.f90.

**4.2.1.6 delete\_client\_rec()**

```
subroutine clients::delete_client_rec (
    class(btrees_clients), intent(inout) this,
    type(btreenode), intent(inout), pointer father,
    type(btreenode), intent(inout), pointer temp,
    integer(kind=8), intent(in) client_id )
```

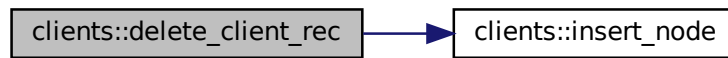
Recursive subroutine to delete a client from the tree.

**Parameters**

<i>this</i>	B-tree to delete the client
<i>father</i>	Father node
<i>temp</i>	Node to delete the client
<i>client_id</i>	Id of the client to delete

Definition at line 372 of file clients.f90.

Here is the call graph for this function:



#### 4.2.1.7 dequeue()

```

type(btreenode) function, pointer clients::dequeue (
    class(client_queue), intent(inout) this )
  
```

Function to dequeue a node from the queue.

##### Parameters

<i>this</i>	Queue to dequeue
-------------	------------------

##### Returns

Node dequeued

Definition at line 97 of file clients.f90.

#### 4.2.1.8 enqueue()

```

subroutine clients::enqueue (
    class(client_queue), intent(inout) this,
    type(btreenode), intent(in), pointer tree_node )
  
```

Subroutine to enqueue a node in the queue.

##### Parameters

<i>this</i>	Queue to enqueue
<i>tree_node</i>	Node to enqueue

Definition at line 112 of file clients.f90.

#### 4.2.1.9 find()

```
logical function clients::find (
    class(btreenode), intent(in) this,
    integer(kind=8), intent(in) client_id )
```

Function to find a client in the node.

##### Parameters

<i>this</i>	Node to search
<i>client_id</i>	Id of the client to search

##### Returns

True if the client was found, false otherwise

Definition at line 428 of file clients.f90.

#### 4.2.1.10 insert\_node()

```
subroutine clients::insert_node (
    type(client), intent(in) pclient,
    integer, intent(in) pos,
    type(btreenode), intent(inout), pointer node,
    type(btreenode), intent(in), pointer child )
```

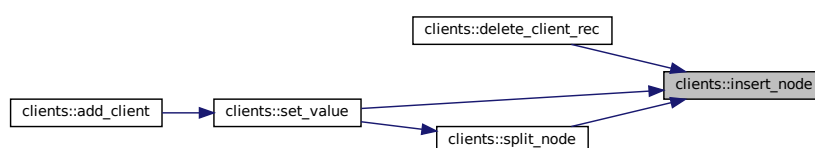
Subroutine to insert a node.

##### Parameters

<i>pclient</i>	Client to add
<i>pos</i>	Position to add the client
<i>node</i>	Node to add the client
<i>child</i>	Child node

Definition at line 220 of file clients.f90.

Here is the caller graph for this function:



#### 4.2.1.11 is\_empty()

```
logical function clients::is_empty (
    class(client_queue), intent(in) this )
```

Function to check if the queue is empty.

##### Parameters

<i>this</i>	Queue to check
-------------	----------------

##### Returns

True if the queue is empty, false otherwise

Definition at line 89 of file clients.f90.

#### 4.2.1.12 search\_client()

```
recursive type(client) function, pointer clients::search_client (
    class(btrees_clients), intent(inout) this,
    type(btreenode), intent(in), pointer node,
    integer(kind=8), intent(in) client_id )
```

Function to search a client in the tree.

##### Parameters

<i>this</i>	B-tree to search
<i>node</i>	Node to search
<i>client_id</i>	Id of the client to search

##### Returns

Client found

Definition at line 331 of file clients.f90.

#### 4.2.1.13 set\_value()

```
recursive logical function clients::set_value (
    type(client), intent(in) new_client,
```

```

type(client), intent(inout) pclient,
type(btreenode), intent(inout), pointer node,
type(btreenode), intent(inout), pointer child )

```

Recursive function to set the value of the tree.

#### Parameters

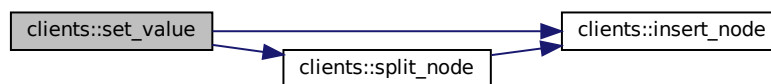
<i>new_client</i>	Client to add
<i>pclient</i>	Client to add
<i>node</i>	Node to add the client
<i>child</i>	Child node

#### Returns

True if the client was added, false otherwise

Definition at line 151 of file clients.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.2.1.14 split\_node()

```

subroutine clients::split_node (
    type(client), intent(in) new_client,
    type(client), intent(inout) pclient,
    integer, intent(in) pos,
    type(btreenode), intent(inout), pointer node,
    type(btreenode), intent(in), pointer child,
    type(btreenode), intent(inout), pointer new_node )

```

Subroutine to split a node.



## Parameters

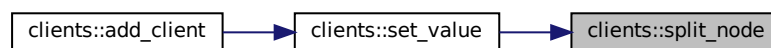
<i>new_client</i>	Client to add
<i>pclient</i>	Client to add
<i>pos</i>	Position to add the client
<i>node</i>	Node to split
<i>child</i>	Child node
<i>new_node</i>	New node

Definition at line 243 of file clients.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.2.1.15 traversal()

```

subroutine clients::traversal (
    class(btrees_clients), intent(inout) this,
    type(btreenode), intent(in), pointer node )
  
```

Subroutine to traverse the tree.

## Parameters

<i>this</i>	B-tree to traverse
<i>node</i>	Node to traverse

Definition at line 281 of file clients.f90.

## 4.2.2 Variable Documentation

### 4.2.2.1 g\_id

```
integer clients::g_id = 1
```

Global variable to assign the id of the nodes.

Definition at line 13 of file clients.f90.

## 4.3 filehandler Module Reference

### Data Types

- type [fhandler](#)

*Type that handles the reading of the json files and the initialization of the data structures.*

### Functions/Subroutines

- subroutine [initialize\\_admin](#) (this)  
*Initializes the data structures for the admin user.*
- subroutine [initialize\\_user](#) (this)  
*Initializes the data structures for the user.*
- subroutine [set\\_user](#) (this, user)  
*Sets the user data structures to the fhandler object.*
- subroutine [read\\_imgs](#) (this)  
*Reads the images from the json file.*
- subroutine [read\\_layers](#) (this)  
*Reads the layers from the json file.*
- subroutine [read\\_clients](#) (this)  
*Reads the clients from the json file.*
- subroutine [read\\_albums](#) (this)  
*Reads the albums from the json file.*

### 4.3.1 Function/Subroutine Documentation

#### 4.3.1.1 initialize\_admin()

```
subroutine filehandler::initialize_admin (
    class(fhandler), intent(inout) this )
```

Initializes the data structures for the admin user.

## Parameters

<i>this</i>	The fhandler object
-------------	---------------------

Definition at line 34 of file filehandler.f90.

#### 4.3.1.2 initialize\_user()

```
subroutine filehandler::initialize_user (  
    class(fhandler), intent(inout) this )
```

Initializes the data structures for the user.

## Parameters

<i>this</i>	The fhandler object
-------------	---------------------

Definition at line 40 of file filehandler.f90.

#### 4.3.1.3 read\_albums()

```
subroutine filehandler::read_albums (  
    class(fhandler), intent(inout) this )
```

Reads the albums from the json file.

## Parameters

<i>this</i>	The fhandler object
-------------	---------------------

Definition at line 189 of file filehandler.f90.

#### 4.3.1.4 read\_clients()

```
subroutine filehandler::read_clients (  
    class(fhandler), intent(inout) this )
```

Reads the clients from the json file.

## Parameters

<i>this</i>	The fhandler object
-------------	---------------------

Definition at line 153 of file filehandler.f90.

#### 4.3.1.5 read\_imgs()

```
subroutine filehandler::read_imgs (  
    class(fhandler), intent(inout) this )
```

Reads the images from the json file.

##### Parameters

<i>this</i>	The fhandler object
-------------	---------------------

Definition at line 59 of file filehandler.f90.

#### 4.3.1.6 read\_layers()

```
subroutine filehandler::read_layers (  
    class(fhandler), intent(inout) this )
```

Reads the layers from the json file.

##### Parameters

<i>this</i>	The fhandler object
-------------	---------------------

Definition at line 100 of file filehandler.f90.

#### 4.3.1.7 set\_user()

```
subroutine filehandler::set_user (  
    class(fhandler), intent(inout) this,  
    type(client), intent(in), pointer user )
```

Sets the user data structures to the handler object.

##### Parameters

<i>this</i>	The fhandler object
<i>user</i>	The user object

Definition at line 50 of file filehandler.f90.

## 4.4 images Module Reference

### Data Types

- type `image`  
*Type that represents a node in the AVL tree.*
- type `image_avl`  
*Type that represents an AVL tree of images.*

### Functions/Subroutines

- subroutine `delete_img` (this, img\_id)  
*Method that deletes an image from the tree.*
- recursive type(`image`) function, pointer `delete_img_rec` (this, temp, img\_id)  
*Recursive method that deletes an image from the tree.*
- recursive type(`image`) function, pointer `min_child` (this, temp)  
*Method that returns the node with the minimum value in the tree.*
- subroutine `add_layer` (this, new\_layer)  
*Method that adds a layer to the image.*
- subroutine `add_img` (this, new\_image)  
*Method that adds an image to the tree.*
- subroutine `add_img_rec` (this, new\_image, tmp)  
*Recursive method that adds an image to the tree.*
- recursive type(`image`) function, pointer `search_img` (this, temp, img\_id)  
*Method that searches for an image in the tree.*
- type(`image`) function, pointer `srl` (this, t1)  
*Method that performs a single right rotation.*
- type(`image`) function, pointer `srr` (this, t1)  
*Method that performs a single left rotation.*
- type(`image`) function, pointer `drl` (this, tmp)  
*Method that performs a double right rotation.*
- type(`image`) function, pointer `drr` (this, tmp)  
*Method that performs a double left rotation.*
- integer function `get_max` (this, val1, val2)  
*Method that returns the maximum value between two integers.*
- integer function `get_height` (this, tmp)  
*Method that returns the height of a node.*
- subroutine `get_dot` (this, tmp, unit)  
*Method that generates the dot representation of the tree.*
- subroutine `get_dot_rec` (this, tmp, unit)  
*Recursive method that generates the dot representation of the tree.*
- subroutine `print_images` (this, temp)  
*Method that prints the images in the tree.*
- subroutine `gen_tree_subtree` (this, id\_img, unit)  
*Method that generates the dot representation of the tree and its subtree.*
- recursive subroutine `gen_layer_subtree` (current\_layer, unit)  
*Recursive method that generates the dot representation of the layers subtree.*
- subroutine `gen_img_traversal` (this, unit, id\_img)  
*Method that generates the dot representation of the image traversal.*

## 4.4.1 Function/Subroutine Documentation

### 4.4.1.1 add\_img()

```
subroutine images::add_img (
    class(image_avl), intent(inout) this,
    type(image), intent(in), pointer new_image )
```

Method that adds an image to the tree.

#### Parameters

<i>this</i>	The <code>image_avl</code> object
<i>new_image</i>	The image to add

Definition at line 145 of file images.f90.

### 4.4.1.2 add\_img\_rec()

```
subroutine images::add_img_rec (
    class(image_avl), intent(inout) this,
    type(image), intent(in), pointer new_image,
    type(image), pointer tmp )
```

Recursive method that adds an image to the tree.

#### Parameters

<i>this</i>	The <code>image_avl</code> object
<i>new_image</i>	The image to add
<i>tmp</i>	The current node

Definition at line 159 of file images.f90.

### 4.4.1.3 add\_layer()

```
subroutine images::add_layer (
    class(image), intent(inout) this,
    type(layer), intent(in), pointer new_layer )
```

Method that adds a layer to the image.

## Parameters

<i>this</i>	The image object
<i>new_layer</i>	The layer to add

Definition at line 136 of file images.f90.

**4.4.1.4 delete\_img()**

```
subroutine images::delete_img (
    class(image_avl), intent(inout) this,
    integer, intent(in) img_id )
```

Method that deletes an image from the tree.

## Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>img_id</i>	The id of the image to delete

Definition at line 48 of file images.f90.

**4.4.1.5 delete\_img\_rec()**

```
recursive type(image) function, pointer images::delete_img_rec (
    class(image_avl), intent(inout) this,
    type(image), intent(inout), pointer temp,
    integer, intent(in) img_id )
```

Recursive method that deletes an image from the tree.

## Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>temp</i>	The current node
<i>img_id</i>	The id of the image to delete

## Returns

The new subtree

Definition at line 62 of file images.f90.

#### 4.4.1.6 `drl()`

```
type(image) function, pointer images::drl (  
    class(image_avl), intent(in) this,  
    type(image), intent(in), pointer tmp )
```

Method that performs a double right rotation.

##### Parameters

<i>this</i>	The <i>image_avl</i> object
<i>tmp</i>	The node to rotate

##### Returns

The new subtree

Definition at line 244 of file `images.f90`.

#### 4.4.1.7 `drr()`

```
type(image) function, pointer images::drr (  
    class(image_avl), intent(in) this,  
    type(image), intent(in), pointer tmp )
```

Method that performs a double left rotation.

##### Parameters

<i>this</i>	The <i>image_avl</i> object
<i>tmp</i>	The node to rotate

##### Returns

The new subtree

Definition at line 255 of file `images.f90`.

#### 4.4.1.8 `gen_img_traversal()`

```
subroutine images::gen_img_traversal (  
    class(image_avl), intent(inout) this,  
    integer, intent(in) unit,  
    integer, intent(in) id_img )
```

Method that generates the dot representation of the image traversal.



## Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>unit</i>	The unit to write the dot representation
<i>id_img</i>	The id of the image to generate the traversal

Definition at line 400 of file images.f90.

#### 4.4.1.9 gen\_layer\_subtree()

```
recursive subroutine images::gen_layer_subtree (
    type(layer), intent(in), pointer current_layer,
    integer, intent(in) unit )
```

Recursive method that generates the dot representation of the layers subtree.

## Parameters

<i>current_layer</i>	The current layer
<i>unit</i>	The unit to write the dot representation

Definition at line 383 of file images.f90.

Here is the caller graph for this function:



#### 4.4.1.10 gen\_tree\_subtree()

```
subroutine images::gen_tree_subtree (
    class(image_avl), intent(inout) this,
    integer, intent(in) id_img,
    integer, intent(in) unit )
```

Method that generates the dot representation of the tree and its subtree.

## Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>id_img</i>	The id of the image to generate the subtree
<i>unit</i>	The unit to write the dot representation

Definition at line 337 of file images.f90.

Here is the call graph for this function:



#### 4.4.1.11 get\_dot()

```

subroutine images::get_dot (
    class(image_avl), intent(in) this,
    type(image), intent(in), pointer tmp,
    integer, intent(in) unit )
  
```

Method that generates the dot representation of the tree.

##### Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>tmp</i>	The current node
<i>unit</i>	The unit to write the dot representation

Definition at line 291 of file images.f90.

#### 4.4.1.12 get\_dot\_rec()

```

subroutine images::get_dot_rec (
    class(image_avl), intent(in) this,
    type(image), intent(in), pointer tmp,
    integer, intent(in) unit )
  
```

Recursive method that generates the dot representation of the tree.

##### Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>tmp</i>	The current node
<i>unit</i>	The unit to write the dot representation

Definition at line 303 of file images.f90.

#### 4.4.1.13 get\_height()

```
integer function images::get_height (  
    class(image_avl), intent(in) this,  
    type(image), intent(in), pointer tmp )
```

Method that returns the height of a node.

##### Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>tmp</i>	The node

##### Returns

The height of the node

Definition at line 277 of file images.f90.

#### 4.4.1.14 get\_max()

```
integer function images::get_max (  
    class(image_avl), intent(in) this,  
    integer, intent(in) val1,  
    integer, intent(in) val2 )
```

Method that returns the maximum value between two integers.

##### Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>val1</i>	The first value
<i>val2</i>	The second value

##### Returns

The maximum value

Definition at line 267 of file images.f90.

#### 4.4.1.15 min\_child()

```
recursive type(image) function, pointer images::min_child (
    class(image\_avl), intent(inout) this,
    type(image), intent(in), pointer temp )
```

Method that returns the node with the minimum value in the tree.

##### Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>temp</i>	The current node

##### Returns

The node with the minimum value

Definition at line 124 of file images.f90.

#### 4.4.1.16 print\_images()

```
subroutine images::print_images (
    class(image\_avl), intent(in) this,
    type(image), intent(in), pointer temp )
```

Method that prints the images in the tree.

##### Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>temp</i>	The current node

Definition at line 323 of file images.f90.

#### 4.4.1.17 search\_img()

```
recursive type(image) function, pointer images::search_img (
    class(image\_avl), intent(in) this,
    type(image), intent(in), pointer temp,
    integer, intent(in) img_id )
```

Method that searches for an image in the tree.

##### Parameters

<i>this</i>	The <a href="#">image_avl</a> object
<i>img_id</i>	The id of the image to search
<i>_ig</i>	
<i>temp</i>	The current node

**Returns**

The image with the id `img_ig`

Definition at line 195 of file `images.f90`.

**4.4.1.18 srl()**

```
type(image) function, pointer images::srl (  
    class(image_avl), intent(in) this,  
    type(image), intent(in), pointer t1 )
```

Method that performs a single right rotation.

**Parameters**

<i>this</i>	The <code>image_avl</code> object
<i>t1</i>	The node to rotate

**Returns**

The new subtree

Definition at line 216 of file `images.f90`.

**4.4.1.19 srr()**

```
type(image) function, pointer images::srr (  
    class(image_avl), intent(in) this,  
    type(image), intent(in), pointer t1 )
```

Method that performs a single left rotation.

**Parameters**

<i>this</i>	The <code>image_avl</code> object
<i>t1</i>	The node to rotate

**Returns**

The new subtree

Definition at line 230 of file `images.f90`.

## 4.5 layers Module Reference

### Data Types

- type `layer`  
*Type to represent a pixel matrix*
- type `layers_tree`  
*Type to represent a binary tree of layers.*
- type `node_layer`  
*Type to represent a node of a linked list of layers.*
- type `queue`  
*Type to represent a queue of layers.*

### Functions/Subroutines

- subroutine `gen_inorder` (this, tmp, limit, unit)  
*Subroutine to generate the matrix of pixels of a layer.*
- subroutine `gen_postorder` (this, tmp, limit, unit)  
*Subroutine to generate the matrix of pixels of a layer.*
- subroutine `gen_preorder` (this, tmp, limit, unit)  
*Subroutine to generate the matrix of pixels of a layer.*
- subroutine `traverse_limited` (this, order, limit, unit)  
*Subroutine to traverse the layers tree in a limited way.*
- subroutine `enqueue` (this, layer\_val)  
*Subroutine to add a layer to the layers tree.*
- type(`layer`) function, pointer `dequeue` (this)  
*Subroutine to remove a layer from the queue.*
- logical function `is_empty` (this)  
*Function to check if the queue is empty.*
- subroutine `traverse_matrix` (this)  
*Subroutine to traverse the layers tree.*
- subroutine `add` (this, new\_layer)  
*Subroutine to add a layer to the layers tree.*
- subroutine `add_recursive` (this, new\_layer, tmp)  
*Subroutine to add a layer to the layers tree recursively.*
- subroutine `add_copied_val` (this, new\_layer)  
*Subroutine to add a layer to the layers tree.*
- type(`layer`) function, pointer `search` (this, id\_searched)  
*Function to search a layer in the layers tree.*
- subroutine `preorder` (this, tmp)  
*Subroutine to traverse the layers tree in preorder.*
- subroutine `inorder` (this, tmp)  
*Subroutine to traverse the layers tree in inorder.*
- subroutine `postorder` (this, tmp)  
*Subroutine to traverse the layers tree in postorder.*
- subroutine `gen_dot` (this, tmp, unit)  
*Subroutine to generate the dot file of the layers tree.*
- subroutine `gen_dot_recursive` (this, tmp, unit)  
*Subroutine to generate the dot file of the layers tree recursively.*

- subroutine `max_depth` (this)  
*Subroutine to calculate the max depth of the layers tree.*
- recursive integer function `max_depth_rec` (this, root)  
*Function to calculate the max depth of the layers tree recursively.*
- subroutine `leaf_layers` (this)  
*Subroutine to traverse the leaf layers of the layers tree.*
- subroutine `leaf_layers_rec` (this, tmp)  
*Subroutine to traverse the leaf layers of the layers tree recursively.*
- subroutine `list_layers` (this, option)  
*Subroutine to list the layers of the layers tree.*
- subroutine `inorder_print` (this, tmp)  
*Subroutine to print the layers of the layers tree in inorder.*

## 4.5.1 Function/Subroutine Documentation

### 4.5.1.1 `add()`

```
subroutine layers::add (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer new_layer )
```

Subroutine to add a layer to the layers tree.

#### Parameters

<i>this</i>	Layers tree
<i>new_layer</i>	Layer to add

Definition at line 229 of file layers.f90.

### 4.5.1.2 `add_copied_val()`

```
subroutine layers::add_copied_val (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer new_layer )
```

Subroutine to add a layer to the layers tree.

#### Parameters

<i>this</i>	Layers tree
<i>new_layer</i>	Layer to add

Definition at line 264 of file layers.f90.

#### 4.5.1.3 add\_recursive()

```
subroutine layers::add_recursive (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer new_layer,
    type(layer), intent(inout) tmp )
```

Subroutine to add a layer to the layers tree recursively.

##### Parameters

<i>this</i>	Layers tree
<i>new_layer</i>	Layer to add
<i>tmp</i>	Temporary layer

Definition at line 243 of file layers.f90.

#### 4.5.1.4 dequeue()

```
type(layer) function, pointer layers::dequeue (
    class(queue), intent(inout) this )
```

Subroutine to remove a layer from the queue.

##### Parameters

<i>this</i>	Queue
<i>layer_val</i>	Layer removed

Definition at line 180 of file layers.f90.

#### 4.5.1.5 enqueue()

```
subroutine layers::enqueue (
    class(queue), intent(inout) this,
    type(layer), intent(in), pointer layer_val )
```

Subroutine to add a layer to the layers tree.

##### Parameters

<i>this</i>	Layers tree
<i>new_layer</i>	Layer to add



Definition at line 160 of file layers.f90.

#### 4.5.1.6 gen\_dot()

```
subroutine layers::gen_dot (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer tmp,
    integer, intent(in) unit )
```

Subroutine to generate the dot file of the layers tree.

##### Parameters

<i>this</i>	Layers tree
<i>tmp</i>	Layer
<i>unit</i>	Unit to write the dot file

Definition at line 351 of file layers.f90.

#### 4.5.1.7 gen\_dot\_recursive()

```
subroutine layers::gen_dot_recursive (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer tmp,
    integer, intent(in) unit )
```

Subroutine to generate the dot file of the layers tree recursively.

##### Parameters

<i>this</i>	Layers tree
<i>tmp</i>	Layer
<i>unit</i>	Unit to write the dot file

Definition at line 363 of file layers.f90.

#### 4.5.1.8 gen\_inorder()

```
subroutine layer::gen_inorder (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer tmp,
    integer, intent(inout) limit,
    integer, intent(in) unit )
```

Subroutine to generate the matrix of pixels of a layer.

**Parameters**

<i>this</i>	Layers tree
<i>tmp</i>	Layer
<i>limit</i>	Limit of layers to traverse
<i>unit</i>	Unit to write the matrix

Definition at line 65 of file layers.f90.

**4.5.1.9 gen\_postorder()**

```
subroutine layers::gen_postorder (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer tmp,
    integer, intent(inout) limit,
    integer, intent(in) unit )
```

Subroutine to generate the matrix of pixels of a layer.

**Parameters**

<i>this</i>	Layers tree
<i>tmp</i>	Layer
<i>limit</i>	Limit of layers to traverse
<i>unit</i>	Unit to write the matrix

Definition at line 91 of file layers.f90.

**4.5.1.10 gen\_preorder()**

```
subroutine layers::gen_preorder (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer tmp,
    integer, intent(inout) limit,
    integer, intent(in) unit )
```

Subroutine to generate the matrix of pixels of a layer.

**Parameters**

<i>this</i>	Layers tree
<i>tmp</i>	Layer
<i>limit</i>	Limit of layers to traverse
<i>unit</i>	Unit to write the matrix

Definition at line 117 of file layers.f90.

#### 4.5.1.11 inorder()

```
subroutine layers::inorder (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer tmp )
```

Subroutine to traverse the layers tree in inorder.

##### Parameters

<i>this</i>	Layers tree
<i>tmp</i>	Layer

Definition at line 323 of file layers.f90.

#### 4.5.1.12 inorder\_print()

```
subroutine layers::inorder_print (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer tmp )
```

Subroutine to print the layers of the layers tree in inorder.

##### Parameters

<i>this</i>	Layers tree
<i>tmp</i>	Layer

Definition at line 460 of file layers.f90.

#### 4.5.1.13 is\_empty()

```
logical function layers::is_empty (
    class(queue), intent(in) this )
```

Function to check if the queue is empty.

##### Parameters

<i>this</i>	Queue
-------------	-------

**Returns**

res Logical value  
 Logical value

Definition at line 195 of file layers.f90.

**4.5.1.14 leaf\_layers()**

```
subroutine layers::leaf_layers (
    class(layers_tree), intent(inout) this )
```

Subroutine to traverse the leaf layers of the layers tree.

**Parameters**

<i>this</i>	Layers tree
-------------	-------------

Definition at line 416 of file layers.f90.

**4.5.1.15 leaf\_layers\_rec()**

```
subroutine layers::leaf_layers_rec (
    class(layers_tree), intent(inout) this,
    type(layer), intent(in), pointer tmp )
```

Subroutine to traverse the leaf layers of the layers tree recursively.

**Parameters**

<i>this</i>	Layers tree
<i>tmp</i>	Layer

Definition at line 428 of file layers.f90.

**4.5.1.16 list\_layers()**

```
subroutine layers::list_layers (
    class(layers_tree), intent(inout) this,
    integer, intent(in) option )
```

Subroutine to list the layers of the layers tree.

## Parameters

<i>this</i>	Layers tree
<i>option</i>	Option to list the layers

Definition at line 443 of file layers.f90.

**4.5.1.17 max\_depth()**

```
subroutine layers::max_depth (  
    class(layers_tree), intent(inout) this )
```

Subroutine to calculate the max depth of the layers tree.

## Parameters

<i>this</i>	Layers tree
-------------	-------------

Definition at line 382 of file layers.f90.

**4.5.1.18 max\_depth\_rec()**

```
recursive integer function layers::max_depth_rec (  
    class(layers_tree), intent(inout) this,  
    type(layer), intent(in), pointer root )
```

Function to calculate the max depth of the layers tree recursively.

## Parameters

<i>this</i>	Layers tree
<i>root</i>	Root of the tree

## Returns

depth Depth of the tree

Definition at line 397 of file layers.f90.

**4.5.1.19 postorder()**

```
subroutine layers::postorder (  
    class(layers_tree), intent(inout) this,  
    type(layer), intent(in), pointer tmp )
```

Subroutine to traverse the layers tree in postorder.

**Parameters**

<i>this</i>	Layers tree
<i>tmp</i>	Layer

Definition at line 336 of file layers.f90.

**4.5.1.20 preorder()**

```
subroutine layers::preorder (  
    class(layers_tree), intent(inout) this,  
    type(layer), intent(in), pointer tmp )
```

Subroutine to traverse the layers tree in preorder.

**Parameters**

<i>this</i>	Layers tree
<i>tmp</i>	Layer

Definition at line 309 of file layers.f90.

**4.5.1.21 search()**

```
type(layer) function, pointer layers::search (  
    class(layers_tree), intent(inout) this,  
    integer, intent(in) id_searched )
```

Function to search a layer in the layers tree.

**Parameters**

<i>this</i>	Layers tree
<i>id_searched</i>	Id of the layer to search

**Returns**

tmp Layer found

Definition at line 286 of file layers.f90.

#### 4.5.1.22 `traverse_limited()`

```
subroutine layers::traverse_limited (
    class(layers_tree), intent(inout) this,
    integer, intent(in) order,
    integer, intent(inout) limit,
    integer, intent(in) unit )
```

Subroutine to traverse the layers tree in a limited way.

##### Parameters

<i>this</i>	Layers tree
<i>order</i>	Order of traversal
<i>limit</i>	Limit of layers to traverse
<i>unit</i>	Unit to write the matrix

Definition at line 142 of file layers.f90.

#### 4.5.1.23 `traverse_matrix()`

```
subroutine layers::traverse_matrix (
    class(layers_tree), intent(inout) this )
```

Subroutine to traverse the layers tree.

##### Parameters

<i>this</i>	Layers tree
-------------	-------------

Definition at line 202 of file layers.f90.

## 4.6 pixels Module Reference

### Data Types

- type `pixel`  
*Type to represent a pixel.*
- type `pixel_matrix`  
*Type to represent a matrix of pixels.*

### Functions/Subroutines

- subroutine `insert` (this, x, y, value, color)  
*Insert a pixel in the matrix.*



- type([pixel](#)) function, pointer [search\\_row](#) (this, y)  
*Search a row in the matrix.*
- type([pixel](#)) function, pointer [search\\_column](#) (this, x)  
*Search a column in the matrix.*
- logical function [node\\_exists](#) (this, new\_node)  
*Check if a node exists in the matrix.*
- type([pixel](#)) function, pointer [insert\\_row\\_header](#) (this, y)  
*Insert a row header in the matrix.*
- subroutine [insert\\_in\\_row](#) (this, new\_node, row\_header)  
*Insert a pixel in a row.*
- type([pixel](#)) function, pointer [insert\\_column\\_header](#) (this, x)  
*Insert a column header in the matrix.*
- subroutine [insert\\_in\\_column](#) (this, new\_node, column\_header)  
*Insert a pixel in a column.*
- subroutine [print\\_headers](#) (this)  
*Print the headers of the matrix.*
- logical function [get\\_value](#) (this, x, y)  
*Get the value of a pixel.*
- type([pixel](#)) function, pointer [get\\_node](#) (this, x, y)  
*Get the node in the matrix.*
- subroutine [self\\_print](#) (this)  
*Print the matrix.*
- subroutine [graph\\_pixels](#) (this, unit)  
*Generate a graph of the pixels.*
- subroutine [gen\\_matrix](#) (this, g\_matrix)  
*Generate a matrix of pixels.*
- subroutine [global\\_m\\_dot](#) (this, unit)  
*Generate a graph of the pixels.*

## Variables

- integer [id](#) = 0  
*Unique identifier for each pixel.*

### 4.6.1 Function/Subroutine Documentation

#### 4.6.1.1 [gen\\_matrix\(\)](#)

```
subroutine pixels::gen_matrix (
    class(pixel\_matrix), intent(inout) this,
    type(pixel\_matrix), intent(inout) g_matrix )
```

Generate a matrix of pixels.

##### Parameters

<i>this</i>	The matrix to be converted
<i>g_matrix</i>	The matrix to be generated

Definition at line 385 of file pixels.f90.

#### 4.6.1.2 get\_node()

```
type(pixel) function, pointer pixels::get_node (
    class(pixel_matrix), intent(inout) this,
    integer, intent(in) x,
    integer, intent(in) y )
```

Get the node in the matrix.

##### Parameters

<i>this</i>	The matrix where the node is located
<i>x</i>	The x coordinate of the node
<i>y</i>	The y coordinate of the node

##### Returns

The pointer to the node

Definition at line 257 of file pixels.f90.

#### 4.6.1.3 get\_value()

```
logical function pixels::get_value (
    class(pixel_matrix), intent(inout) this,
    integer, intent(in) x,
    integer, intent(in) y )
```

Get the value of a pixel.

##### Parameters

<i>this</i>	The matrix where the pixel is located
<i>x</i>	The x coordinate of the pixel
<i>y</i>	The y coordinate of the pixel

##### Returns

The value of the pixel

The value of the pixel

Definition at line 230 of file pixels.f90.

#### 4.6.1.4 global\_m\_dot()

```
subroutine pixels::global_m_dot (
    class(pixel_matrix), intent(inout) this,
    integer, intent(in) unit )
```

Generate a graph of the pixels.

##### Parameters

<i>this</i>	The matrix to be graphed
<i>unit</i>	The unit where the graph will be written

Definition at line 409 of file pixels.f90.

#### 4.6.1.5 graph\_pixels()

```
subroutine pixels::graph_pixels (
    class(pixel_matrix), intent(inout) this,
    integer, intent(in) unit )
```

Generate a graph of the pixels.

##### Parameters

<i>this</i>	The matrix to be graphed
<i>unit</i>	The unit where the graph will be written

Definition at line 303 of file pixels.f90.

#### 4.6.1.6 insert()

```
subroutine pixels::insert (
    class(pixel_matrix), intent(inout) this,
    integer, intent(in) x,
    integer, intent(in) y,
    logical, intent(in) value,
    character(len=7), intent(in) color )
```

Insert a pixel in the matrix.

##### Parameters

<i>this</i>	The matrix where the pixel will be inserted
<i>x</i>	The x coordinate of the pixel
<i>y</i>	The y coordinate of the pixel
<i>value</i>	The value of the pixel
<i>color</i>	The color of the pixel

Definition at line 51 of file pixels.f90.

#### 4.6.1.7 insert\_column\_header()

```
type(pixel) function, pointer pixels::insert_column_header (
    class(pixel_matrix), intent(inout) this,
    integer, intent(in) x )
```

Insert a column header in the matrix.

##### Parameters

<i>this</i>	The matrix where the column header will be inserted
<i>x</i>	The x coordinate of the column header

##### Returns

The pointer to the column header

Definition at line 181 of file pixels.f90.

#### 4.6.1.8 insert\_in\_column()

```
subroutine pixels::insert_in_column (
    class(pixel_matrix), intent(inout) this,
    type(pixel), intent(in), pointer new_node,
    type(pixel), intent(in), pointer column_header )
```

Insert a pixel in a column.

##### Parameters

<i>this</i>	The matrix where the pixel will be inserted
<i>new_node</i>	The pixel to be inserted
<i>column_header</i>	The column header where the pixel will be inserted

Definition at line 195 of file pixels.f90.

#### 4.6.1.9 insert\_in\_row()

```
subroutine pixels::insert_in_row (
    class(pixel_matrix), intent(inout) this,
```

```
type(pixel), intent(in), pointer new_node,  
type(pixel), intent(in), pointer row_header )
```

Insert a pixel in a row.

#### Parameters

<i>this</i>	The matrix where the pixel will be inserted
<i>new_node</i>	The pixel to be inserted
<i>row_header</i>	The row header where the pixel will be inserted

Definition at line 156 of file pixels.f90.

#### 4.6.1.10 insert\_row\_header()

```
type(pixel) function, pointer pixels::insert_row_header (  
    class(pixel_matrix), intent(inout) this,  
    integer, intent(in) y )
```

Insert a row header in the matrix.

#### Parameters

<i>this</i>	The matrix where the row header will be inserted
<i>y</i>	The y coordinate of the row header

#### Returns

The pointer to the row header

Definition at line 142 of file pixels.f90.

#### 4.6.1.11 node\_exists()

```
logical function pixels::node_exists (  
    class(pixel_matrix), intent(inout) this,  
    type(pixel), intent(in), pointer new_node )
```

Check if a node exists in the matrix.

#### Parameters

<i>this</i>	The matrix where the node will be searched
<i>new_node</i>	The node to be searched

**Returns**

True if the node exists, False otherwise

Definition at line 116 of file pixels.f90.

**4.6.1.12 print\_headers()**

```
subroutine pixels::print_headers (
    class(pixel_matrix), intent(inout) this )
```

Print the headers of the matrix.

**Parameters**

<i>this</i>	The matrix where the headers will be printed
-------------	--

Definition at line 218 of file pixels.f90.

**4.6.1.13 search\_column()**

```
type(pixel) function, pointer pixels::search_column (
    class(pixel_matrix), intent(inout) this,
    integer, intent(in) x )
```

Search a column in the matrix.

**Parameters**

<i>this</i>	The matrix where the column will be searched
<i>x</i>	The x coordinate of the column

**Returns**

The pointer to the column header

Definition at line 102 of file pixels.f90.

**4.6.1.14 search\_row()**

```
type(pixel) function, pointer pixels::search_row (
    class(pixel_matrix), intent(inout) this,
    integer, intent(in) y )
```

Search a row in the matrix.

**Parameters**

<i>this</i>	The matrix where the row will be searched
<i>y</i>	The y coordinate of the row

**Returns**

The pointer to the row header

Definition at line 88 of file pixels.f90.

**4.6.1.15 self\_print()**

```
subroutine pixels::self_print (  
    class(pixel_matrix), intent(inout) this )
```

Print the matrix.

**Parameters**

<i>this</i>	The matrix to be printed
-------------	--------------------------

Definition at line 281 of file pixels.f90.

**4.6.2 Variable Documentation****4.6.2.1 id**

```
integer pixels::id = 0
```

Unique identifier for each pixel.

Definition at line 10 of file pixels.f90.





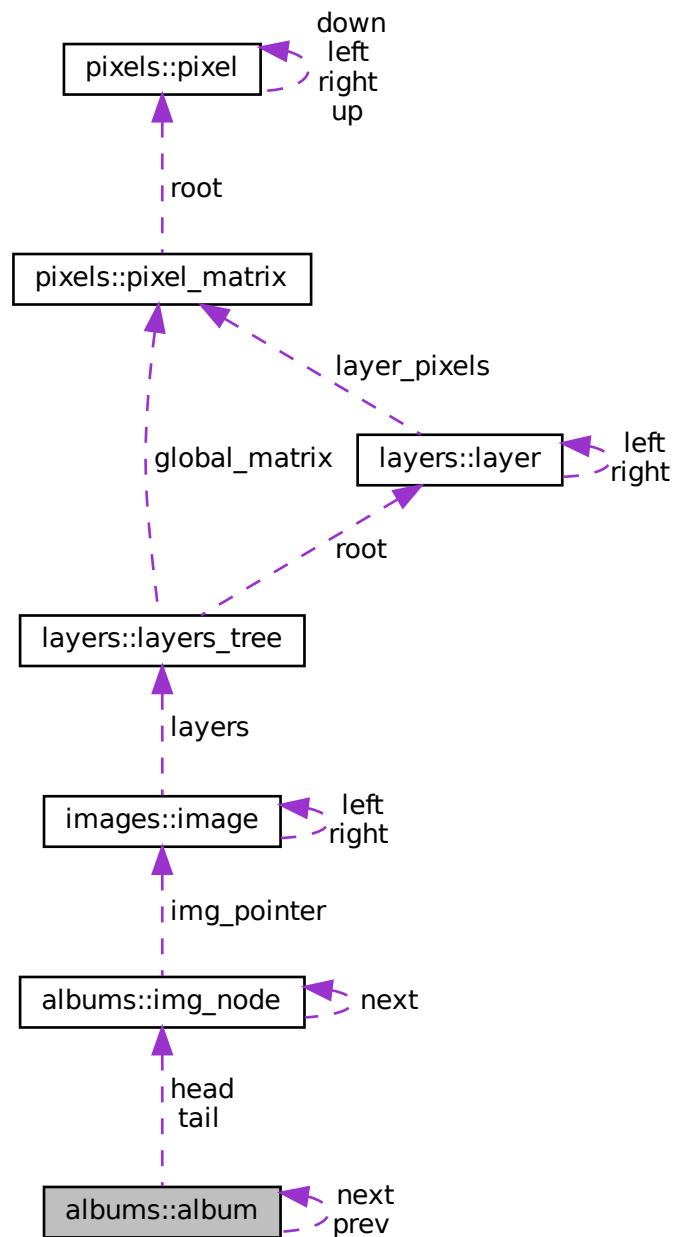
## Chapter 5

# Data Type Documentation

### 5.1 `albums::album` Type Reference

Type to store the album.

Collaboration diagram for albums::album:



## Public Member Functions

- procedure [add\\_image](#)
- procedure [remove\\_image](#)
- procedure [get\\_image](#)
- procedure [show\\_images](#)

## Public Attributes

- integer `id`  
*Album id.*
- character(:), allocatable `name`  
*Album name.*
- type(`album`), pointer `next` => null()  
*Pointer to the next album.*
- type(`album`), pointer `prev` => null()  
*Pointer to the previous album.*
- type(`img_node`), pointer `head` => null()  
*Pointer to the first image.*
- type(`img_node`), pointer `tail` => null()  
*Pointer to the last image.*
- integer `size` = 0  
*Number of images in the album.*

### 5.1.1 Detailed Description

Type to store the album.

Definition at line 18 of file albums.f90.

### 5.1.2 Member Function/Subroutine Documentation

#### 5.1.2.1 add\_image()

```
procedure albums::album::add_image
```

Definition at line 27 of file albums.f90.

#### 5.1.2.2 get\_image()

```
procedure albums::album::get_image
```

Definition at line 29 of file albums.f90.

#### 5.1.2.3 remove\_image()

```
procedure albums::album::remove_image
```

Definition at line 28 of file albums.f90.

#### 5.1.2.4 show\_images()

```
procedure albums::album::show_images
```

Definition at line 30 of file albums.f90.

### 5.1.3 Member Data Documentation

#### 5.1.3.1 head

```
type(img_node), pointer albums::album::head => null()
```

Pointer to the first image.

Definition at line 23 of file albums.f90.

#### 5.1.3.2 id

```
integer albums::album::id
```

Album id.

Definition at line 19 of file albums.f90.

#### 5.1.3.3 name

```
character(:), allocatable albums::album::name
```

Album name.

Definition at line 20 of file albums.f90.

#### 5.1.3.4 next

```
type(album), pointer albums::album::next => null()
```

Pointer to the next album.

Definition at line 21 of file albums.f90.

### 5.1.3.5 prev

```
type(album), pointer albums::album::prev => null()
```

Pointer to the previous album.

Definition at line 22 of file albums.f90.

### 5.1.3.6 size

```
integer albums::album::size = 0
```

Number of images in the album.

Definition at line 25 of file albums.f90.

### 5.1.3.7 tail

```
type(img_node), pointer albums::album::tail => null()
```

Pointer to the last image.

Definition at line 24 of file albums.f90.

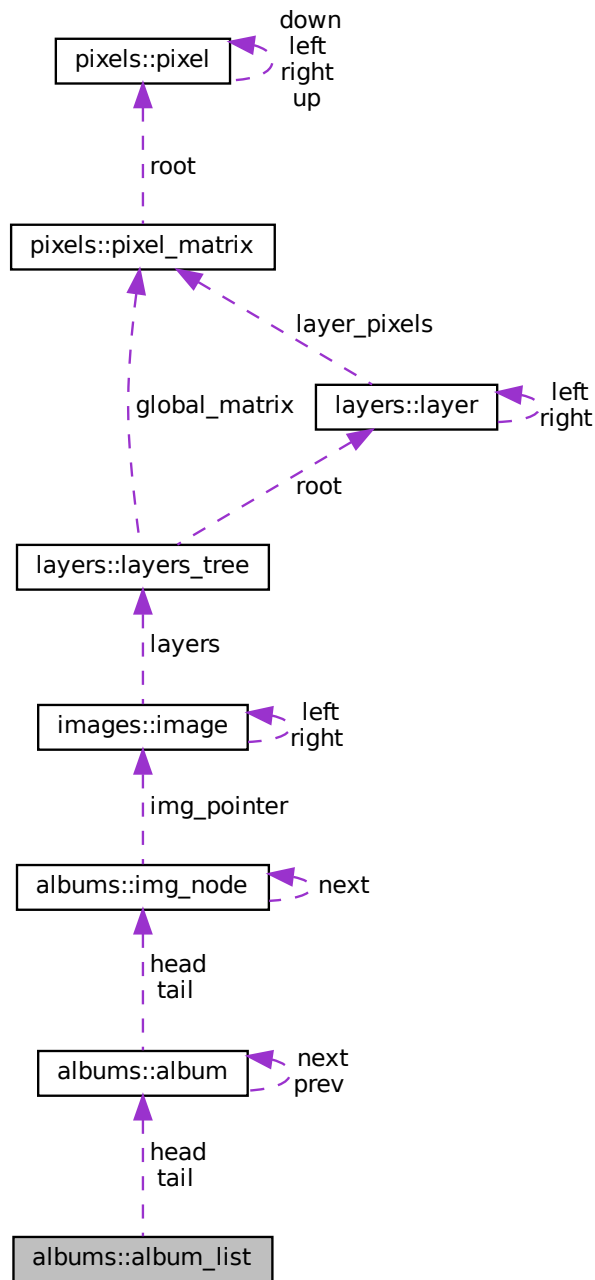
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/albums.f90](#)

## 5.2 albums::album\_list Type Reference

Type to store the album list.

Collaboration diagram for albums::album\_list:



## Public Member Functions

- procedure [new\\_album](#)
- procedure [remove\\_album](#)
- procedure [get\\_album](#)
- procedure [search\\_in\\_album](#)
- procedure [add\\_image\\_to\\_album](#)

- procedure [remove\\_image\\_from\\_album](#)
- procedure [show\\_albums](#)
- procedure [show\\_album\\_images](#)
- procedure [gen\\_album\\_graph](#)

## Public Attributes

- type([album](#)), pointer [head](#) => null()  
*Pointer to the first album.*
- type([album](#)), pointer [tail](#) => null()  
*Pointer to the last album.*
- integer [size](#) = 0  
*Number of albums in the list.*

### 5.2.1 Detailed Description

Type to store the album list.

Definition at line 33 of file albums.f90.

### 5.2.2 Member Function/Subroutine Documentation

#### 5.2.2.1 add\_image\_to\_album()

```
procedure albums::album_list::add_image_to_album
```

Definition at line 42 of file albums.f90.

#### 5.2.2.2 gen\_album\_graph()

```
procedure albums::album_list::gen_album_graph
```

Definition at line 46 of file albums.f90.

#### 5.2.2.3 get\_album()

```
procedure albums::album_list::get_album
```

Definition at line 40 of file albums.f90.

#### 5.2.2.4 new\_album()

```
procedure albums::album_list::new_album
```

Definition at line 38 of file albums.f90.

#### 5.2.2.5 remove\_album()

```
procedure albums::album_list::remove_album
```

Definition at line 39 of file albums.f90.

#### 5.2.2.6 remove\_image\_from\_album()

```
procedure albums::album_list::remove_image_from_album
```

Definition at line 43 of file albums.f90.

#### 5.2.2.7 search\_in\_album()

```
procedure albums::album_list::search_in_album
```

Definition at line 41 of file albums.f90.

#### 5.2.2.8 show\_album\_images()

```
procedure albums::album_list::show_album_images
```

Definition at line 45 of file albums.f90.

#### 5.2.2.9 show\_albums()

```
procedure albums::album_list::show_albums
```

Definition at line 44 of file albums.f90.



## 5.2.3 Member Data Documentation

### 5.2.3.1 head

```
type(album), pointer albums::album_list::head => null()
```

Pointer to the first album.

Definition at line 34 of file albums.f90.

### 5.2.3.2 size

```
integer albums::album_list::size = 0
```

Number of albums in the list.

Definition at line 36 of file albums.f90.

### 5.2.3.3 tail

```
type(album), pointer albums::album_list::tail => null()
```

Pointer to the last album.

Definition at line 35 of file albums.f90.

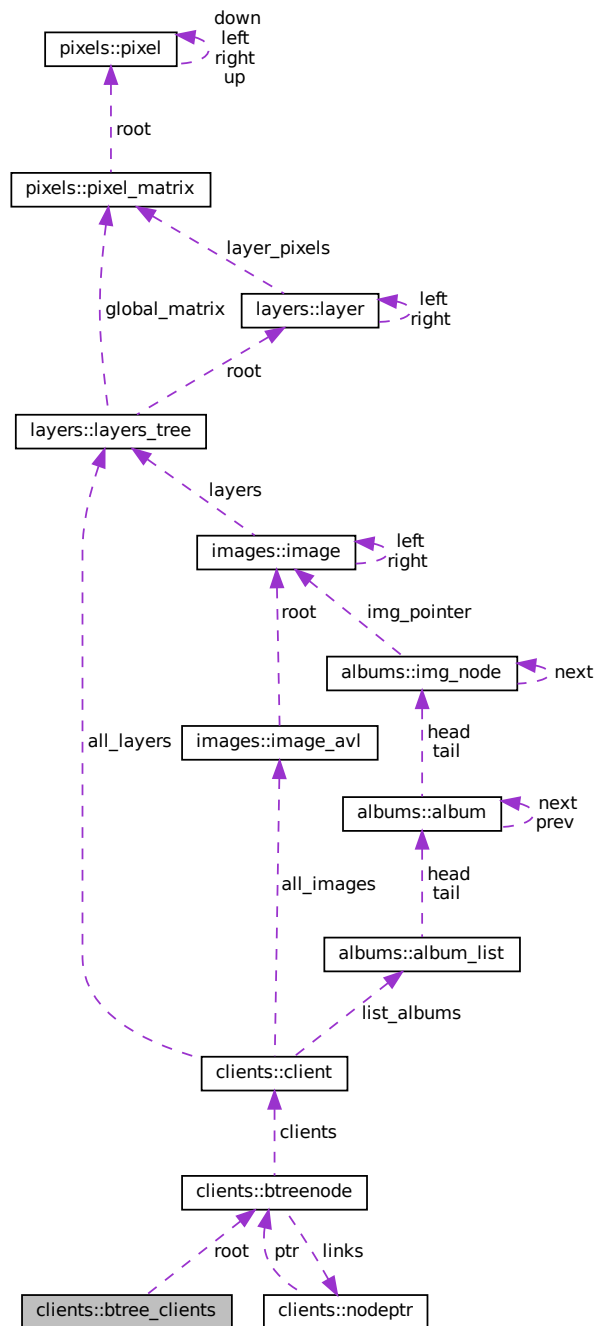
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/[albums.f90](#)

### 5.3 clients::btree\_clients Type Reference

Type to represent a B-tree.

Collaboration diagram for clients::btree\_clients:



#### Public Member Functions

- procedure [add\\_client](#)

- procedure [search\\_client](#)
- procedure [delete\\_client](#)
- procedure [delete\\_client\\_rec](#)
- procedure [create\\_node](#)
- procedure [traversal](#)
- procedure [clients\\_dot](#)
- procedure [amplitude\\_traversal](#)

## Public Attributes

- type([btreenode](#)), pointer [root](#) => null()  
*Pointer to the root of the tree.*

### 5.3.1 Detailed Description

Type to represent a B-tree.

Definition at line 37 of file clients.f90.

### 5.3.2 Member Function/Subroutine Documentation

#### 5.3.2.1 add\_client()

```
procedure clients::btree_clients::add_client
```

Definition at line 40 of file clients.f90.

#### 5.3.2.2 amplitude\_traversal()

```
procedure clients::btree_clients::amplitude_traversal
```

Definition at line 47 of file clients.f90.

#### 5.3.2.3 clients\_dot()

```
procedure clients::btree_clients::clients_dot
```

Definition at line 46 of file clients.f90.

#### 5.3.2.4 create\_node()

```
procedure clients::btree_clients::create_node
```

Definition at line 44 of file clients.f90.

#### 5.3.2.5 delete\_client()

```
procedure clients::btree_clients::delete_client
```

Definition at line 42 of file clients.f90.

#### 5.3.2.6 delete\_client\_rec()

```
procedure clients::btree_clients::delete_client_rec
```

Definition at line 43 of file clients.f90.

#### 5.3.2.7 search\_client()

```
procedure clients::btree_clients::search_client
```

Definition at line 41 of file clients.f90.

#### 5.3.2.8 traversal()

```
procedure clients::btree_clients::traversal
```

Definition at line 45 of file clients.f90.

### 5.3.3 Member Data Documentation

### 5.3.3.1 root

```
type(btreenode), pointer clients::btree_clients::root => null()
```

Pointer to the root of the tree.

Definition at line 38 of file clients.f90.

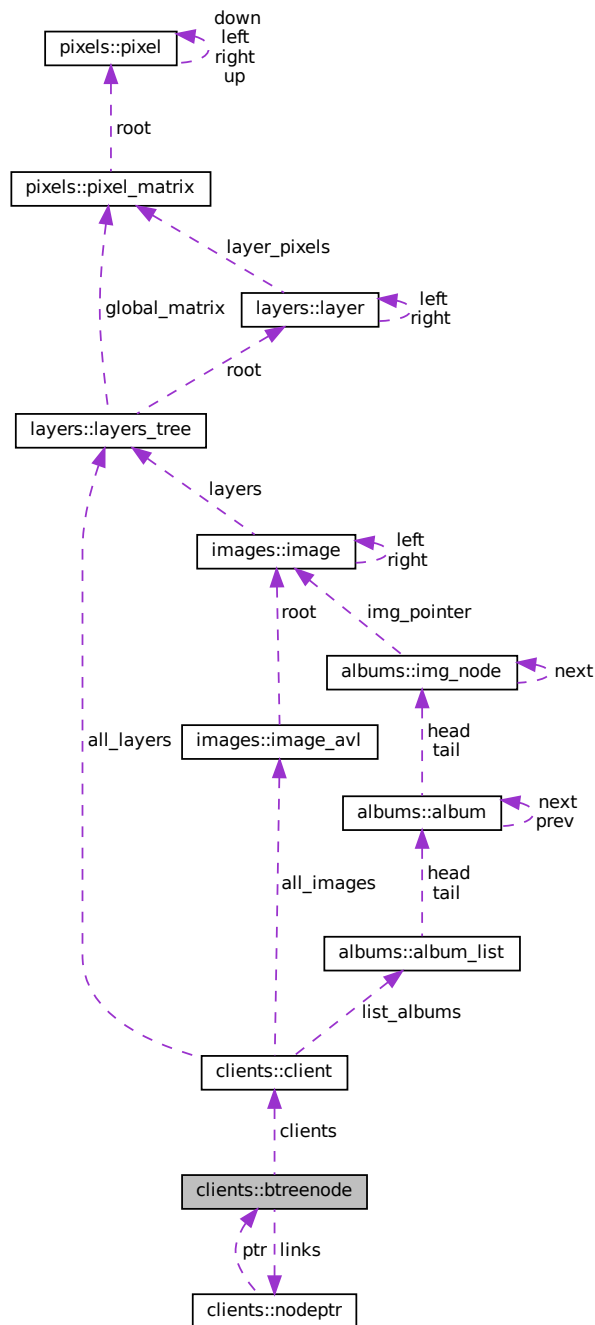
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90](#)

## 5.4 clients::btreenode Type Reference

Type to represent a B-tree node.

Collaboration diagram for `clients::btreeode`:



## Public Member Functions

- procedure [find](#)

## Public Attributes

- integer [id](#)

- Id of the node.*
- type([client](#)), dimension(0:5) [clients](#)  
*Array to store the clients of the node.*
- integer [num](#) = 0  
*Number of clients in the node.*
- type([nodeptr](#)), dimension(0:5) [links](#)  
*Array to store the links to other nodes.*

### 5.4.1 Detailed Description

Type to represent a B-tree node.

Definition at line 28 of file clients.f90.

### 5.4.2 Member Function/Subroutine Documentation

#### 5.4.2.1 find()

```
procedure clients::btreenode::find
```

Definition at line 34 of file clients.f90.

### 5.4.3 Member Data Documentation

#### 5.4.3.1 clients

```
type(client), dimension(0:5) clients::btreenode::clients
```

Array to store the clients of the node.

Definition at line 30 of file clients.f90.

#### 5.4.3.2 id

```
integer clients::btreenode::id
```

Id of the node.

Definition at line 29 of file clients.f90.

#### 5.4.3.3 links

```
type(nodeptr), dimension(0:5) clients::btreeNode::links
```

Array to store the links to other nodes.

Definition at line 32 of file clients.f90.

#### 5.4.3.4 num

```
integer clients::btreeNode::num = 0
```

Number of clients in the node.

Definition at line 31 of file clients.f90.

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

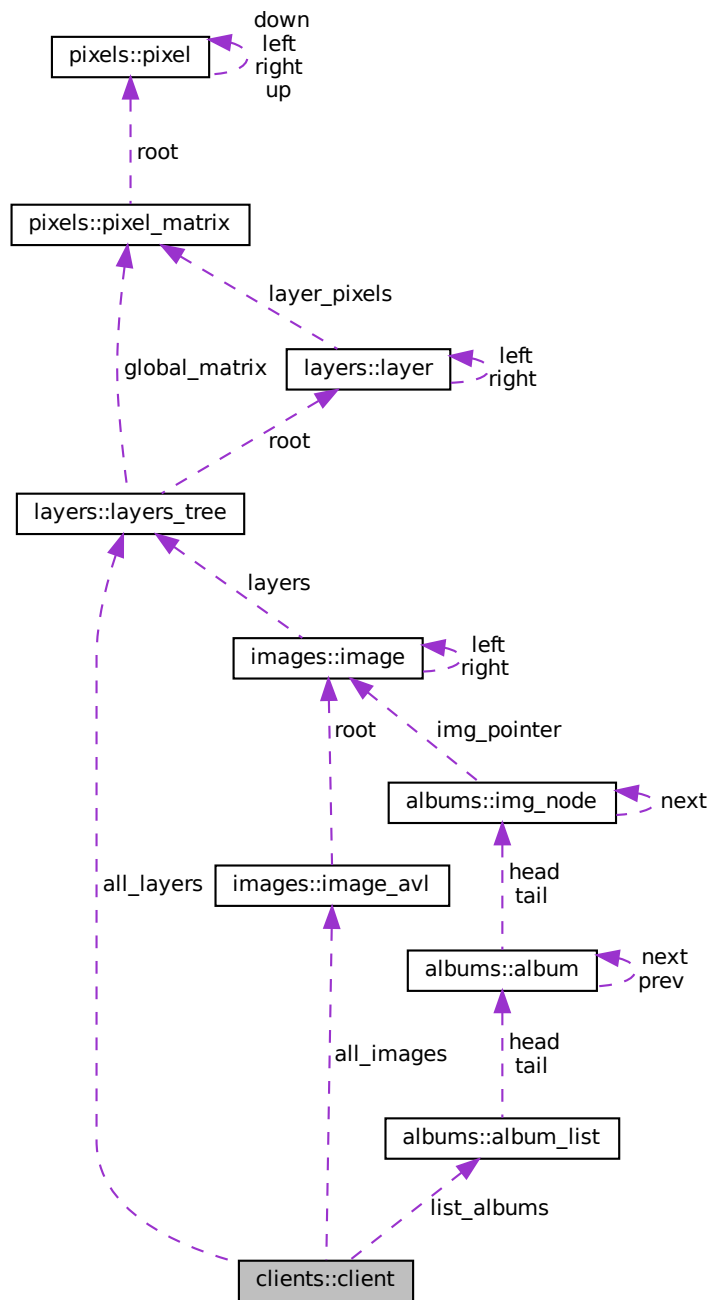
- [/home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90](#)

## 5.5 clients::client Type Reference

Type to represent a client.



Collaboration diagram for clients::client:



## Public Attributes

- `character(:)`, allocatable [name](#)  
*Name of the client.*
- `integer(kind=8)` [dpi](#)  
*DPI of the client.*
- `character(:)`, allocatable [password](#)

*Password of the client.*

- `type(image_avl)` `all_images`

*AVL tree to store the images of the client.*

- `type(album_list)` `list_albums`

*List to store the albums of the client.*

- `type(layers_tree)` `all_layers`

*Tree to store the layers of the client.*

### 5.5.1 Detailed Description

Type to represent a client.

Definition at line 15 of file clients.f90.

### 5.5.2 Member Data Documentation

#### 5.5.2.1 `all_images`

```
type(image_avl) clients::client::all_images
```

AVL tree to store the images of the client.

Definition at line 19 of file clients.f90.

#### 5.5.2.2 `all_layers`

```
type(layers_tree) clients::client::all_layers
```

Tree to store the layers of the client.

Definition at line 21 of file clients.f90.

#### 5.5.2.3 `dpi`

```
integer(kind=8) clients::client::dpi
```

DPI of the client.

Definition at line 17 of file clients.f90.

#### 5.5.2.4 list\_albums

```
type(album_list) clients::client::list_albums
```

List to store the albums of the client.

Definition at line 20 of file clients.f90.

#### 5.5.2.5 name

```
character(:), allocatable clients::client::name
```

Name of the client.

Definition at line 16 of file clients.f90.

#### 5.5.2.6 password

```
character(:), allocatable clients::client::password
```

Password of the client.

Definition at line 18 of file clients.f90.

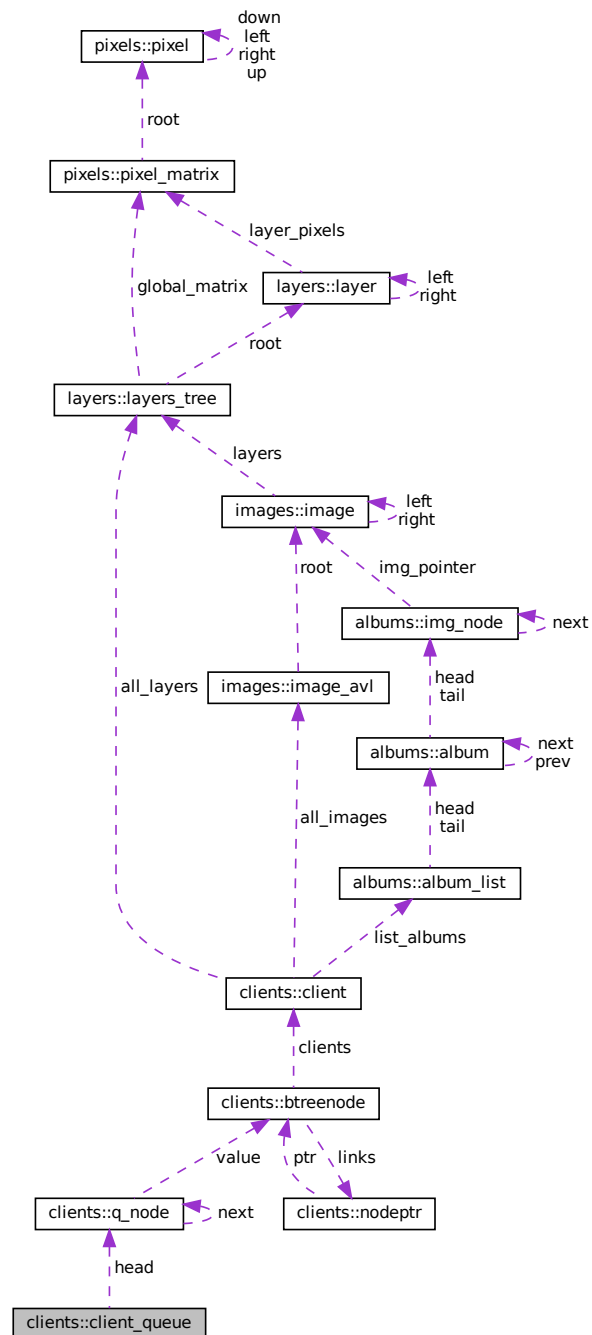
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90](#)

## 5.6 clients::client\_queue Type Reference

Type to represent a client queue.

Collaboration diagram for `clients::client_queue`:



## Public Member Functions

- procedure [enqueue](#)
- procedure [dequeue](#)
- procedure [is\\_empty](#)

## Public Attributes

- type([q\\_node](#)), pointer [head](#) => null()  
*Pointer to the head of the queue.*

### 5.6.1 Detailed Description

Type to represent a client queue.

Definition at line 55 of file clients.f90.

### 5.6.2 Member Function/Subroutine Documentation

#### 5.6.2.1 dequeue()

```
procedure clients::client_queue::dequeue
```

Definition at line 59 of file clients.f90.

#### 5.6.2.2 enqueue()

```
procedure clients::client_queue::enqueue
```

Definition at line 58 of file clients.f90.

#### 5.6.2.3 is\_empty()

```
procedure clients::client_queue::is_empty
```

Definition at line 60 of file clients.f90.

### 5.6.3 Member Data Documentation

### 5.6.3.1 head

```
type(q_node), pointer clients::client_queue::head => null()
```

Pointer to the head of the queue.

Definition at line 56 of file clients.f90.

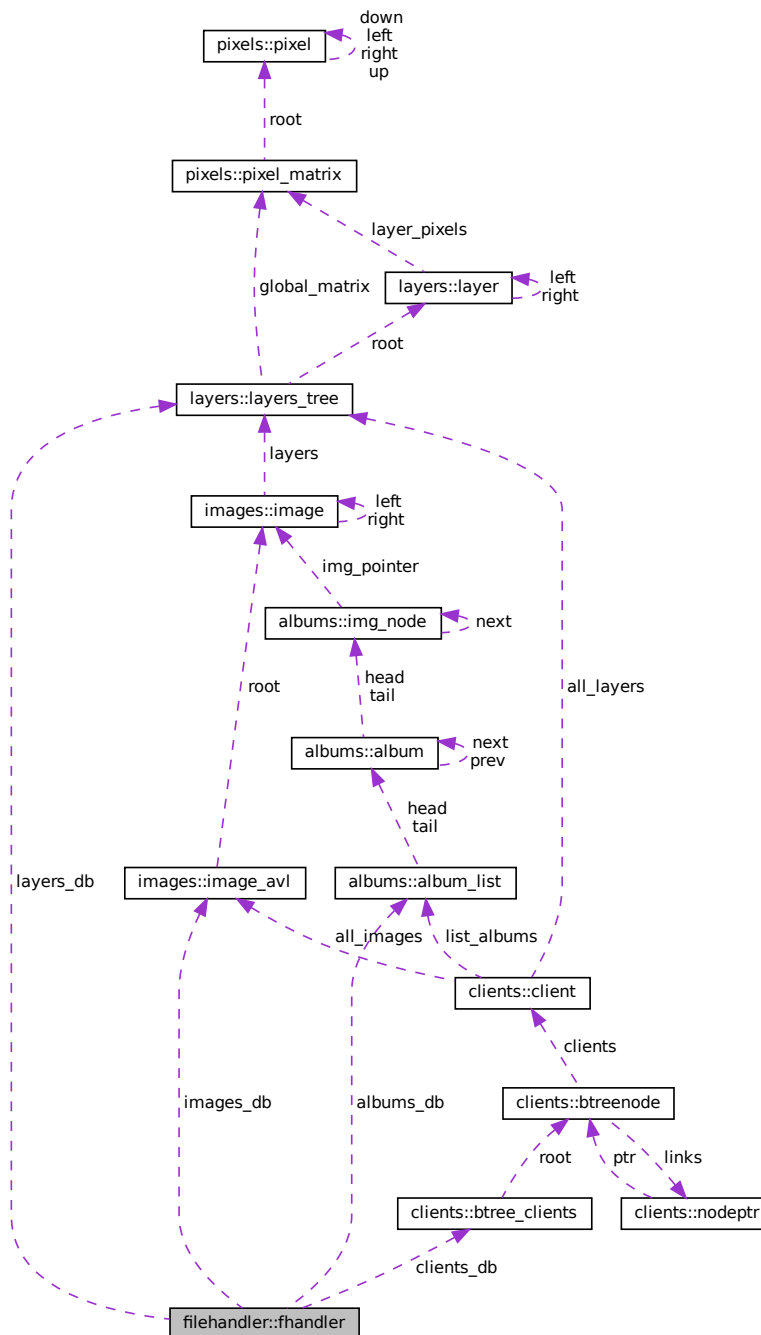
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90](#)

## 5.7 filehandler::fhandler Type Reference

Type that handles the reading of the json files and the initialization of the data structures.

Collaboration diagram for filehandler::fhandler:



## Public Member Functions

- procedure [initialize\\_admin](#)
- procedure [initialize\\_user](#)
- procedure [set\\_user](#)
- procedure [read\\_imgs](#)
- procedure [read\\_layers](#)

- procedure [read\\_albums](#)
- procedure [read\\_clients](#)

## Public Attributes

- type([album\\_list](#)), pointer [albums\\_db](#) => null()  
*Pointer to the albums database.*
- type([image\\_avl](#)), pointer [images\\_db](#) => null()  
*Pointer to the images database.*
- type([btree\\_clients](#)), pointer [clients\\_db](#) => null()  
*Pointer to the clients database.*
- type([layers\\_tree](#)), pointer [layers\\_db](#) => null()  
*Pointer to the layers database.*

### 5.7.1 Detailed Description

Type that handles the reading of the json files and the initialization of the data structures.

Definition at line 17 of file `filehandler.f90`.

### 5.7.2 Member Function/Subroutine Documentation

#### 5.7.2.1 `initialize_admin()`

```
procedure filehandler::fhandler::initialize_admin
```

Definition at line 23 of file `filehandler.f90`.

#### 5.7.2.2 `initialize_user()`

```
procedure filehandler::fhandler::initialize_user
```

Definition at line 24 of file `filehandler.f90`.

#### 5.7.2.3 `read_albums()`

```
procedure filehandler::fhandler::read_albums
```

Definition at line 28 of file `filehandler.f90`.



#### 5.7.2.4 read\_clients()

```
procedure filehandler::fhandler::read_clients
```

Definition at line 29 of file filehandler.f90.

#### 5.7.2.5 read\_imgs()

```
procedure filehandler::fhandler::read_imgs
```

Definition at line 26 of file filehandler.f90.

#### 5.7.2.6 read\_layers()

```
procedure filehandler::fhandler::read_layers
```

Definition at line 27 of file filehandler.f90.

#### 5.7.2.7 set\_user()

```
procedure filehandler::fhandler::set_user
```

Definition at line 25 of file filehandler.f90.

### 5.7.3 Member Data Documentation

#### 5.7.3.1 albums\_db

```
type(album_list), pointer filehandler::fhandler::albums_db => null()
```

Pointer to the albums database.

Definition at line 18 of file filehandler.f90.

### 5.7.3.2 clients\_db

```
type(btrees_clients), pointer filehandler::fhandler::clients_db => null()
```

Pointer to the clients database.

Definition at line 20 of file filehandler.f90.

### 5.7.3.3 images\_db

```
type(image_avl), pointer filehandler::fhandler::images_db => null()
```

Pointer to the images database.

Definition at line 19 of file filehandler.f90.

### 5.7.3.4 layers\_db

```
type(layers_tree), pointer filehandler::fhandler::layers_db => null()
```

Pointer to the layers database.

Definition at line 21 of file filehandler.f90.

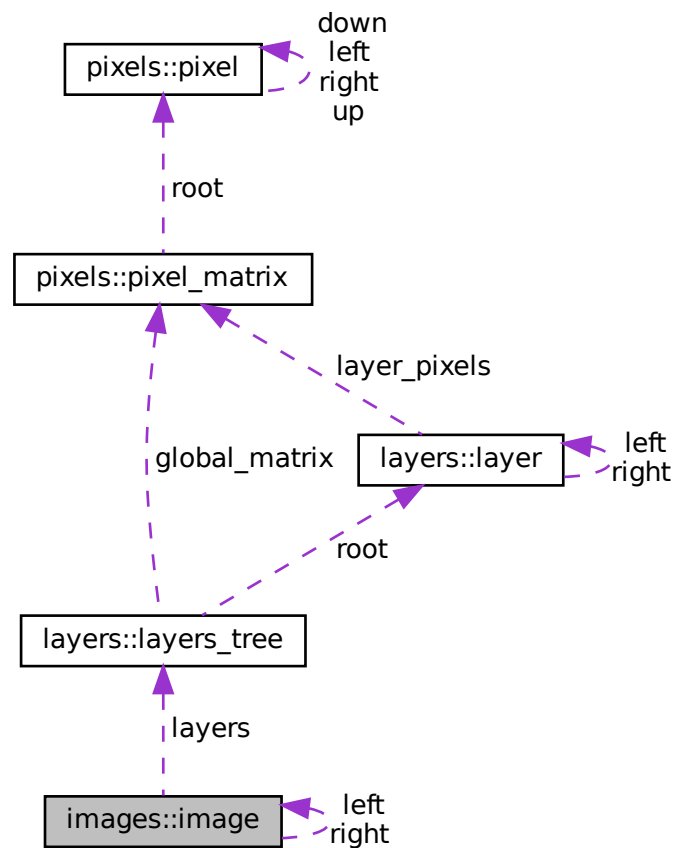
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/filehandler.f90](#)

## 5.8 images::image Type Reference

Type that represents a node in the AVL tree.

Collaboration diagram for images::image:



## Public Member Functions

- procedure [add\\_layer](#)

## Public Attributes

- integer [id](#)  
*The id of the image.*
- integer [height](#)  
*The height of the node.*
- integer [layers\\_count](#) = 0  
*The number of layers in the image.*
- type([layers\\_tree](#)) [layers](#)  
*The layers of the image.*
- type([image](#)), pointer [left](#) => null()  
*The left child of the node.*
- type([image](#)), pointer [right](#) => null()  
*The right child of the nodes.*

### 5.8.1 Detailed Description

Type that represents a node in the AVL tree.

Definition at line 11 of file images.f90.

### 5.8.2 Member Function/Subroutine Documentation

#### 5.8.2.1 add\_layer()

```
procedure images::image::add_layer
```

Definition at line 19 of file images.f90.

### 5.8.3 Member Data Documentation

#### 5.8.3.1 height

```
integer images::image::height
```

The height of the node.

Definition at line 13 of file images.f90.

#### 5.8.3.2 id

```
integer images::image::id
```

The id of the image.

Definition at line 12 of file images.f90.

#### 5.8.3.3 layers

```
type(layers_tree) images::image::layers
```

The layers of the image.

Definition at line 15 of file images.f90.

#### 5.8.3.4 layers\_count

```
integer images::image::layers_count = 0
```

The number of layers in the image.

Definition at line 14 of file images.f90.

#### 5.8.3.5 left

```
type(image), pointer images::image::left => null()
```

The left child of the node.

Definition at line 16 of file images.f90.

#### 5.8.3.6 right

```
type(image), pointer images::image::right => null()
```

The right child of the nodes.

Definition at line 17 of file images.f90.

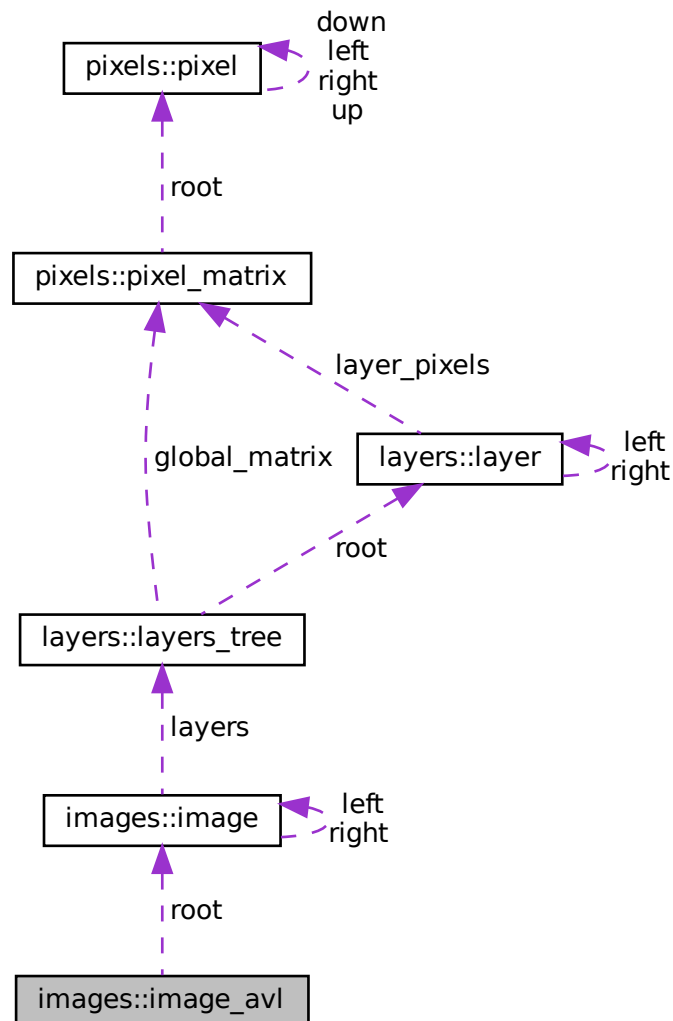
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/images.f90](#)

## 5.9 images::image\_avl Type Reference

Type that represents an AVL tree of images.

Collaboration diagram for `images::image_avl`:



## Public Member Functions

- procedure [add\\_img](#)
- procedure [add\\_img\\_rec](#)
- procedure [search\\_img](#)
- procedure [srl](#)
- procedure [srr](#)
- procedure [drl](#)
- procedure [drr](#)
- procedure [get\\_max](#)
- procedure [min\\_child](#)
- procedure [get\\_height](#)
- procedure [get\\_dot](#)
- procedure [get\\_dot\\_rec](#)

- procedure [delete\\_img](#)
- procedure [delete\\_img\\_rec](#)
- procedure [print\\_images](#)
- procedure [gen\\_tree\\_subtree](#)
- procedure [gen\\_img\\_traversal](#)

## Public Attributes

- type([image](#)), pointer [root](#) => null()  
*The root of the tree.*
- integer [total](#) = 0  
*The total number of images in the tree.*

### 5.9.1 Detailed Description

Type that represents an AVL tree of images.

Definition at line 22 of file images.f90.

### 5.9.2 Member Function/Subroutine Documentation

#### 5.9.2.1 add\_img()

```
procedure images::image_avl::add_img
```

Definition at line 26 of file images.f90.

#### 5.9.2.2 add\_img\_rec()

```
procedure images::image_avl::add_img_rec
```

Definition at line 27 of file images.f90.

#### 5.9.2.3 delete\_img()

```
procedure images::image_avl::delete_img
```

Definition at line 38 of file images.f90.

#### 5.9.2.4 delete\_img\_rec()

```
procedure images::image_avl::delete_img_rec
```

Definition at line 39 of file images.f90.

#### 5.9.2.5 drl()

```
procedure images::image_avl::drl
```

Definition at line 31 of file images.f90.

#### 5.9.2.6 drr()

```
procedure images::image_avl::drr
```

Definition at line 32 of file images.f90.

#### 5.9.2.7 gen\_img\_traversal()

```
procedure images::image_avl::gen_img_traversal
```

Definition at line 42 of file images.f90.

#### 5.9.2.8 gen\_tree\_subtree()

```
procedure images::image_avl::gen_tree_subtree
```

Definition at line 41 of file images.f90.

#### 5.9.2.9 get\_dot()

```
procedure images::image_avl::get_dot
```

Definition at line 36 of file images.f90.



#### 5.9.2.10 get\_dot\_rec()

```
procedure images::image_avl::get_dot_rec
```

Definition at line 37 of file images.f90.

#### 5.9.2.11 get\_height()

```
procedure images::image_avl::get_height
```

Definition at line 35 of file images.f90.

#### 5.9.2.12 get\_max()

```
procedure images::image_avl::get_max
```

Definition at line 33 of file images.f90.

#### 5.9.2.13 min\_child()

```
procedure images::image_avl::min_child
```

Definition at line 34 of file images.f90.

#### 5.9.2.14 print\_images()

```
procedure images::image_avl::print_images
```

Definition at line 40 of file images.f90.

#### 5.9.2.15 search\_img()

```
procedure images::image_avl::search_img
```

Definition at line 28 of file images.f90.

#### 5.9.2.16 srl()

```
procedure images::image_avl::srl
```

Definition at line 29 of file images.f90.

#### 5.9.2.17 srr()

```
procedure images::image_avl::srr
```

Definition at line 30 of file images.f90.

### 5.9.3 Member Data Documentation

#### 5.9.3.1 root

```
type(image), pointer images::image_avl::root => null()
```

The root of the tree.

Definition at line 23 of file images.f90.

#### 5.9.3.2 total

```
integer images::image_avl::total = 0
```

The total number of images in the tree.

Definition at line 24 of file images.f90.

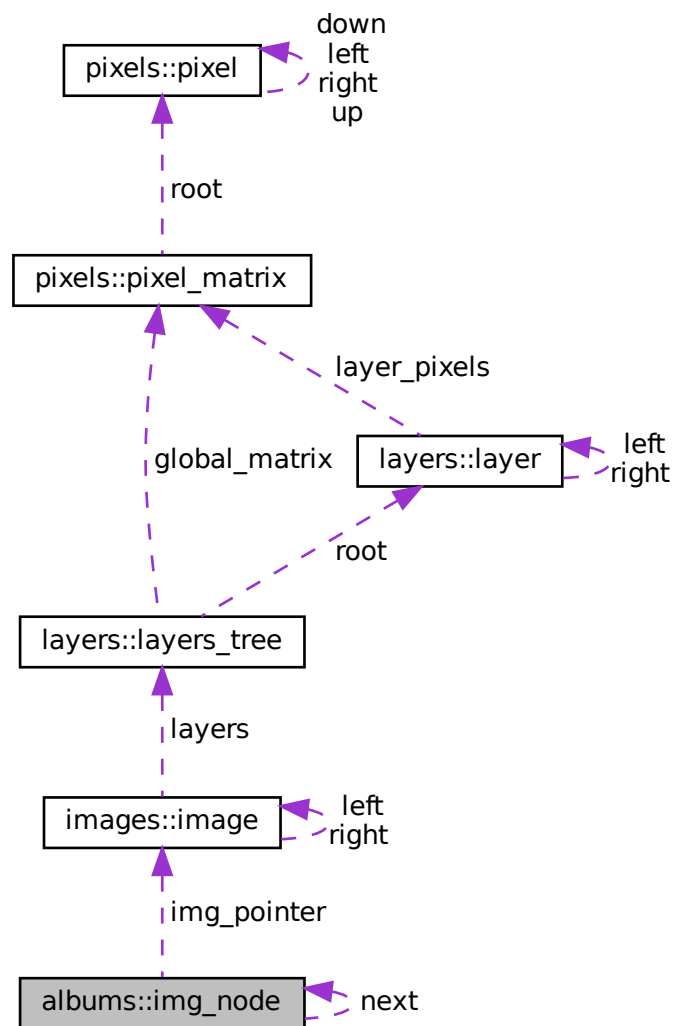
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/images.f90](#)

## 5.10 albums::img\_node Type Reference

Type to store the image node.

Collaboration diagram for albums::img\_node:



### Public Attributes

- integer `id`  
*Image id.*
- type(`image`), pointer `img_pointer` => `null()`  
*Pointer to the image.*
- type(`img_node`), pointer `next` => `null()`  
*Pointer to the next image.*

### 5.10.1 Detailed Description

Type to store the image node.

Definition at line 12 of file albums.f90.

### 5.10.2 Member Data Documentation

#### 5.10.2.1 id

```
integer albums::img_node::id
```

Image id.

Definition at line 13 of file albums.f90.

#### 5.10.2.2 img\_pointer

```
type(image), pointer albums::img_node::img_pointer => null()
```

Pointer to the image.

Definition at line 14 of file albums.f90.

#### 5.10.2.3 next

```
type(img\_node), pointer albums::img_node::next => null()
```

Pointer to the next image.

Definition at line 15 of file albums.f90.

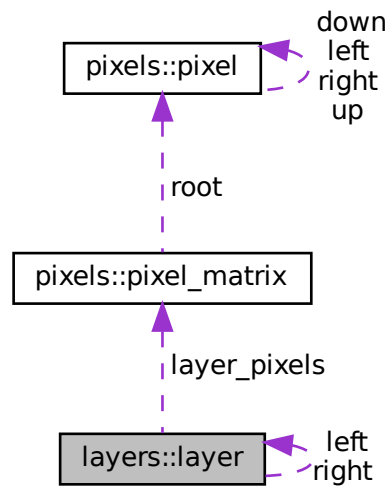
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/albums.f90](#)

## 5.11 layers::layer Type Reference

Type to represent a pixel matrix

Collaboration diagram for layers::layer:



### Public Attributes

- integer `id`  
*Layer id.*
- integer `pixels_count`  
*Number of pixels in the layer.*
- type(`pixel_matrix`) `layer_pixels`  
*Matrix of pixels in the layer.*
- type(`layer`), pointer `left` => null()  
*Pointer to the left child layer.*
- type(`layer`), pointer `right` => null()  
*Pointer to the right child layer.*

### 5.11.1 Detailed Description

Type to represent a pixel matrix

Definition at line 12 of file layers.f90.

## 5.11.2 Member Data Documentation

### 5.11.2.1 id

```
integer layers::layer::id
```

Layer id.

Definition at line 13 of file layers.f90.

### 5.11.2.2 layer\_pixels

```
type(pixel_matrix) layers::layer::layer_pixels
```

Matrix of pixels in the layer.

Definition at line 15 of file layers.f90.

### 5.11.2.3 left

```
type(layer), pointer layers::layer::left => null()
```

Pointer to the left child layer.

Definition at line 16 of file layers.f90.

### 5.11.2.4 pixels\_count

```
integer layers::layer::pixels_count
```

Number of pixels in the layer.

Definition at line 14 of file layers.f90.

### 5.11.2.5 right

```
type(layer), pointer layers::layer::right => null()
```

Pointer to the right child layer.

Definition at line 17 of file layers.f90.

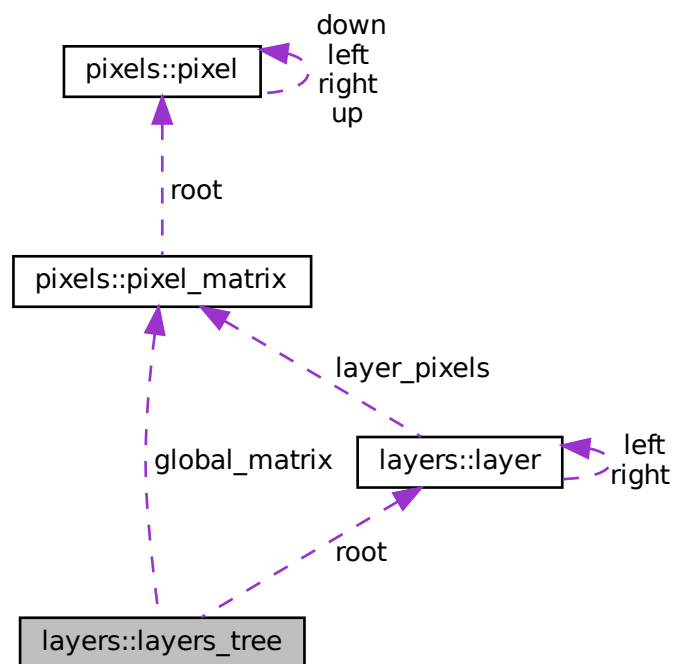
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90

## 5.12 layers::layers\_tree Type Reference

Type to represent a binary tree of layers.

Collaboration diagram for layers::layers\_tree:



## Public Member Functions

- procedure [add](#)
- procedure [add\\_recursive](#)
- procedure [add\\_copied\\_val](#)
- procedure [preorder](#)
- procedure [inorder](#)
- procedure [postorder](#)
- procedure [gen\\_dot](#)
- procedure [gen\\_dot\\_recursive](#)
- procedure [search](#)
- procedure [traverse\\_matrix](#)
- procedure [max\\_depth](#)
- procedure [max\\_depth\\_rec](#)
- procedure [leaf\\_layers](#)
- procedure [leaf\\_layers\\_rec](#)
- procedure [list\\_layers](#)
- procedure [inorder\\_print](#)
- procedure [traverse\\_limited](#)
- procedure [gen\\_preorder](#)
- procedure [gen\\_inorder](#)
- procedure [gen\\_postorder](#)

## Public Attributes

- type([layer](#)), pointer [root](#) => null()  
*Pointer to the root layer.*
- type([pixel\\_matrix](#)) [global\\_matrix](#)  
*Global matrix of pixels.*
- integer [total](#) = 0  
*Total number of layers in the tree.*

### 5.12.1 Detailed Description

Type to represent a binary tree of layers.

Definition at line 20 of file layers.f90.

### 5.12.2 Member Function/Subroutine Documentation

#### 5.12.2.1 [add\(\)](#)

```
procedure layers::layers_tree::add
```

Definition at line 25 of file layers.f90.



#### 5.12.2.2 add\_copied\_val()

```
procedure layers::layers_tree::add_copied_val
```

Definition at line 27 of file layers.f90.

#### 5.12.2.3 add\_recursive()

```
procedure layers::layers_tree::add_recursive
```

Definition at line 26 of file layers.f90.

#### 5.12.2.4 gen\_dot()

```
procedure layers::layers_tree::gen_dot
```

Definition at line 31 of file layers.f90.

#### 5.12.2.5 gen\_dot\_recursive()

```
procedure layers::layers_tree::gen_dot_recursive
```

Definition at line 32 of file layers.f90.

#### 5.12.2.6 gen\_inorder()

```
procedure layers::layers_tree::gen_inorder
```

Definition at line 43 of file layers.f90.

#### 5.12.2.7 gen\_postorder()

```
procedure layers::layers_tree::gen_postorder
```

Definition at line 44 of file layers.f90.

#### 5.12.2.8 `gen_preorder()`

```
procedure layers::layers_tree::gen_preorder
```

Definition at line 42 of file layers.f90.

#### 5.12.2.9 `inorder()`

```
procedure layers::layers_tree::inorder
```

Definition at line 29 of file layers.f90.

#### 5.12.2.10 `inorder_print()`

```
procedure layers::layers_tree::inorder_print
```

Definition at line 40 of file layers.f90.

#### 5.12.2.11 `leaf_layers()`

```
procedure layers::layers_tree::leaf_layers
```

Definition at line 37 of file layers.f90.

#### 5.12.2.12 `leaf_layers_rec()`

```
procedure layers::layers_tree::leaf_layers_rec
```

Definition at line 38 of file layers.f90.

#### 5.12.2.13 `list_layers()`

```
procedure layers::layers_tree::list_layers
```

Definition at line 39 of file layers.f90.

**5.12.2.14 max\_depth()**

```
procedure layers::layers_tree::max_depth
```

Definition at line 35 of file layers.f90.

**5.12.2.15 max\_depth\_rec()**

```
procedure layers::layers_tree::max_depth_rec
```

Definition at line 36 of file layers.f90.

**5.12.2.16 postorder()**

```
procedure layers::layers_tree::postorder
```

Definition at line 30 of file layers.f90.

**5.12.2.17 preorder()**

```
procedure layers::layers_tree::preorder
```

Definition at line 28 of file layers.f90.

**5.12.2.18 search()**

```
procedure layers::layers_tree::search
```

Definition at line 33 of file layers.f90.

**5.12.2.19 traverse\_limited()**

```
procedure layers::layers_tree::traverse_limited
```

Definition at line 41 of file layers.f90.

#### 5.12.2.20 `traverse_matrix()`

```
procedure layers::layers_tree::traverse_matrix
```

Definition at line 34 of file layers.f90.

### 5.12.3 Member Data Documentation

#### 5.12.3.1 `global_matrix`

```
type(pixel_matrix) layers::layers_tree::global_matrix
```

Global matrix of pixels.

Definition at line 22 of file layers.f90.

#### 5.12.3.2 `root`

```
type(layer), pointer layers::layers_tree::root => null()
```

Pointer to the root layer.

Definition at line 21 of file layers.f90.

#### 5.12.3.3 `total`

```
integer layers::layers_tree::total = 0
```

Total number of layers in the tree.

Definition at line 23 of file layers.f90.

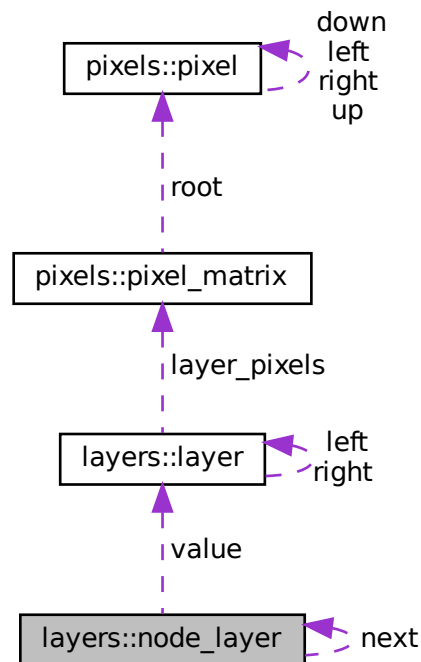
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90](#)

## 5.13 layers::node\_layer Type Reference

Type to represent a node of a linked list of layers.

Collaboration diagram for layers::node\_layer:



### Public Attributes

- `type(layer)`, pointer `value`  
*Pointer to a layer.*
- `type(node_layer)`, pointer `next` => `null()`  
*Pointer to the next node.*

#### 5.13.1 Detailed Description

Type to represent a node of a linked list of layers.

Definition at line 47 of file `layers.f90`.

#### 5.13.2 Member Data Documentation

#### 5.13.2.1 next

```
type(node\_layer), pointer layers::node_layer::next => null()
```

Pointer to the next node.

Definition at line 49 of file layers.f90.

#### 5.13.2.2 value

```
type(layer), pointer layers::node_layer::value
```

Pointer to a layer.

Definition at line 48 of file layers.f90.

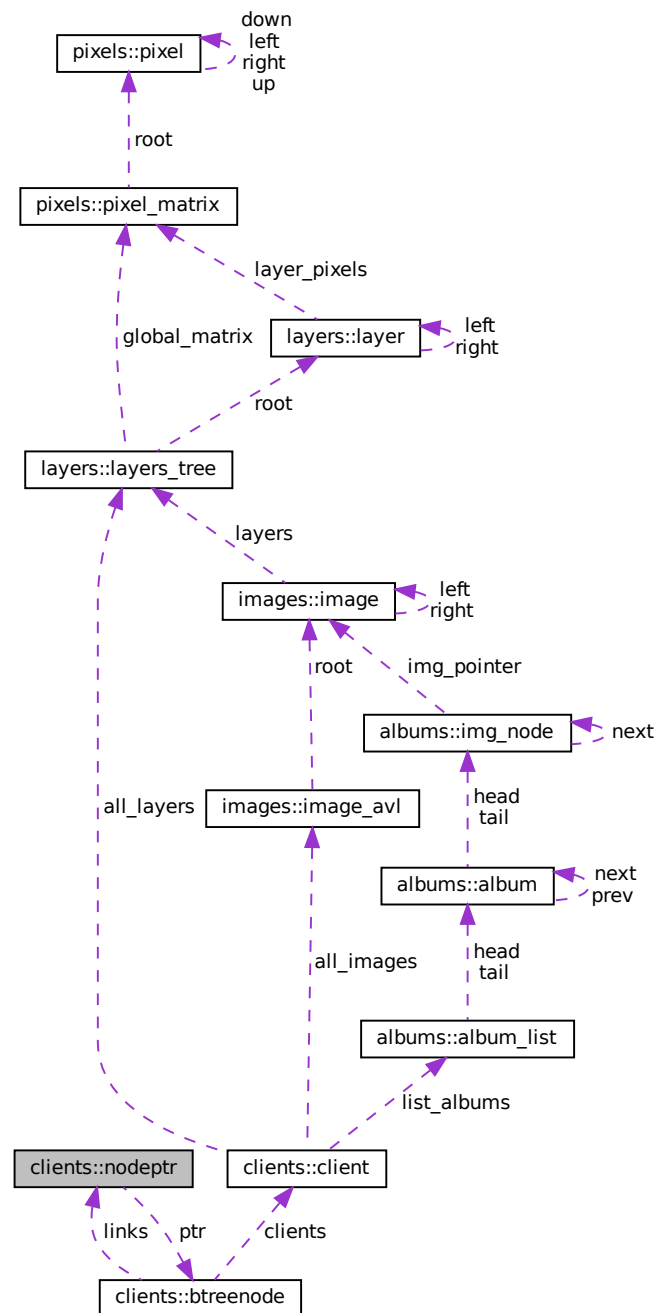
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90](#)

## 5.14 clients::nodeptr Type Reference

Type to represent a node pointer.

Collaboration diagram for clients::nodeptr:



## Public Attributes

- type([btree\\_node](#)), pointer [ptr](#) => null()  
*Pointer to a B-tree node.*

### 5.14.1 Detailed Description

Type to represent a node pointer.

Definition at line 24 of file clients.f90.

### 5.14.2 Member Data Documentation

#### 5.14.2.1 ptr

```
type(btreenode), pointer clients::nodeptr::ptr => null()
```

Pointer to a B-tree node.

Definition at line 25 of file clients.f90.

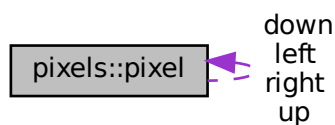
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/[clients.f90](#)

## 5.15 pixels::pixel Type Reference

Type to represent a pixel.

Collaboration diagram for pixels::pixel:





## Public Attributes

- integer `id`  
*Unique identifier for the pixel.*
- integer `x`
- integer `y`  
*Coordinates of the pixel.*
- logical `on` = .FALSE.  
*Value of the pixel.*
- character(len=7) `color`  
*Color of the pixel.*
- type(`pixel`), pointer `up` => null()  
*Pointer to the pixel above.*
- type(`pixel`), pointer `down` => null()  
*Pointer to the pixel below.*
- type(`pixel`), pointer `right` => null()  
*Pointer to the pixel on the right.*
- type(`pixel`), pointer `left` => null()  
*Pointer to the pixel on the left.*

### 5.15.1 Detailed Description

Type to represent a pixel.

Definition at line 12 of file pixels.f90.

### 5.15.2 Member Data Documentation

#### 5.15.2.1 color

```
character(len=7) pixels::pixel::color
```

Color of the pixel.

Definition at line 16 of file pixels.f90.

#### 5.15.2.2 down

```
type(pixel), pointer pixels::pixel::down => null()
```

Pointer to the pixel below.

Definition at line 18 of file pixels.f90.

### 5.15.2.3 id

```
integer pixels::pixel::id
```

Unique identifier for the pixel.

Definition at line 13 of file pixels.f90.

### 5.15.2.4 left

```
type(pixel), pointer pixels::pixel::left => null()
```

Pointer to the pixel on the left.

Definition at line 20 of file pixels.f90.

### 5.15.2.5 on

```
logical pixels::pixel::on = .FALSE.
```

Value of the pixel.

Definition at line 15 of file pixels.f90.

### 5.15.2.6 right

```
type(pixel), pointer pixels::pixel::right => null()
```

Pointer to the pixel on the right.

Definition at line 19 of file pixels.f90.

### 5.15.2.7 up

```
type(pixel), pointer pixels::pixel::up => null()
```

Pointer to the pixel above.

Definition at line 17 of file pixels.f90.

### 5.15.2.8 x

```
integer pixels::pixel::x
```

Definition at line 14 of file pixels.f90.

### 5.15.2.9 y

```
integer pixels::pixel::y
```

Coordinates of the pixel.

Definition at line 14 of file pixels.f90.

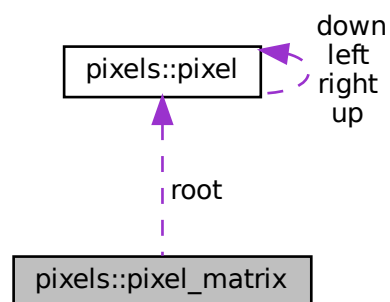
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/pixels.f90

## 5.16 pixels::pixel\_matrix Type Reference

Type to represent a matrix of pixels.

Collaboration diagram for pixels::pixel\_matrix:



## Public Member Functions

- procedure [insert](#)
- procedure [self\\_print](#)
- procedure [search\\_row](#)
- procedure [search\\_column](#)
- procedure [node\\_exists](#)
- procedure [insert\\_column\\_header](#)
- procedure [insert\\_row\\_header](#)
- procedure [insert\\_in\\_row](#)
- procedure [insert\\_in\\_column](#)
- procedure [print\\_headers](#)
- procedure [get\\_value](#)
- procedure [gen\\_matrix](#)
- procedure [get\\_node](#)
- procedure [graph\\_pixels](#)
- procedure [global\\_m\\_dot](#)

## Public Attributes

- type([pixel](#)), pointer [root](#) => null()  
*Pointer to the root of the matrix.*
- integer [width](#) = 0  
*Width of the matrix.*
- integer [height](#) = 0  
*Height of the matrix.*

### 5.16.1 Detailed Description

Type to represent a matrix of pixels.

Definition at line 23 of file pixels.f90.

### 5.16.2 Member Function/Subroutine Documentation

#### 5.16.2.1 [gen\\_matrix\(\)](#)

```
procedure pixels::pixel_matrix::gen_matrix
```

Definition at line 39 of file pixels.f90.

### 5.16.2.2 get\_node()

```
procedure pixels::pixel_matrix::get_node
```

Definition at line 40 of file pixels.f90.

### 5.16.2.3 get\_value()

```
procedure pixels::pixel_matrix::get_value
```

Definition at line 38 of file pixels.f90.

### 5.16.2.4 global\_m\_dot()

```
procedure pixels::pixel_matrix::global_m_dot
```

Definition at line 42 of file pixels.f90.

### 5.16.2.5 graph\_pixels()

```
procedure pixels::pixel_matrix::graph_pixels
```

Definition at line 41 of file pixels.f90.

### 5.16.2.6 insert()

```
procedure pixels::pixel_matrix::insert
```

Definition at line 28 of file pixels.f90.

### 5.16.2.7 insert\_column\_header()

```
procedure pixels::pixel_matrix::insert_column_header
```

Definition at line 33 of file pixels.f90.

**5.16.2.8 insert\_in\_column()**

```
procedure pixels::pixel_matrix::insert_in_column
```

Definition at line 36 of file pixels.f90.

**5.16.2.9 insert\_in\_row()**

```
procedure pixels::pixel_matrix::insert_in_row
```

Definition at line 35 of file pixels.f90.

**5.16.2.10 insert\_row\_header()**

```
procedure pixels::pixel_matrix::insert_row_header
```

Definition at line 34 of file pixels.f90.

**5.16.2.11 node\_exists()**

```
procedure pixels::pixel_matrix::node_exists
```

Definition at line 32 of file pixels.f90.

**5.16.2.12 print\_headers()**

```
procedure pixels::pixel_matrix::print_headers
```

Definition at line 37 of file pixels.f90.

**5.16.2.13 search\_column()**

```
procedure pixels::pixel_matrix::search_column
```

Definition at line 31 of file pixels.f90.

#### 5.16.2.14 search\_row()

```
procedure pixels::pixel_matrix::search_row
```

Definition at line 30 of file pixels.f90.

#### 5.16.2.15 self\_print()

```
procedure pixels::pixel_matrix::self_print
```

Definition at line 29 of file pixels.f90.

### 5.16.3 Member Data Documentation

#### 5.16.3.1 height

```
integer pixels::pixel_matrix::height = 0
```

Height of the matrix.

Definition at line 26 of file pixels.f90.

#### 5.16.3.2 root

```
type(pixel), pointer pixels::pixel_matrix::root => null()
```

Pointer to the root of the matrix.

Definition at line 24 of file pixels.f90.

#### 5.16.3.3 width

```
integer pixels::pixel_matrix::width = 0
```

Width of the matrix.

Definition at line 25 of file pixels.f90.

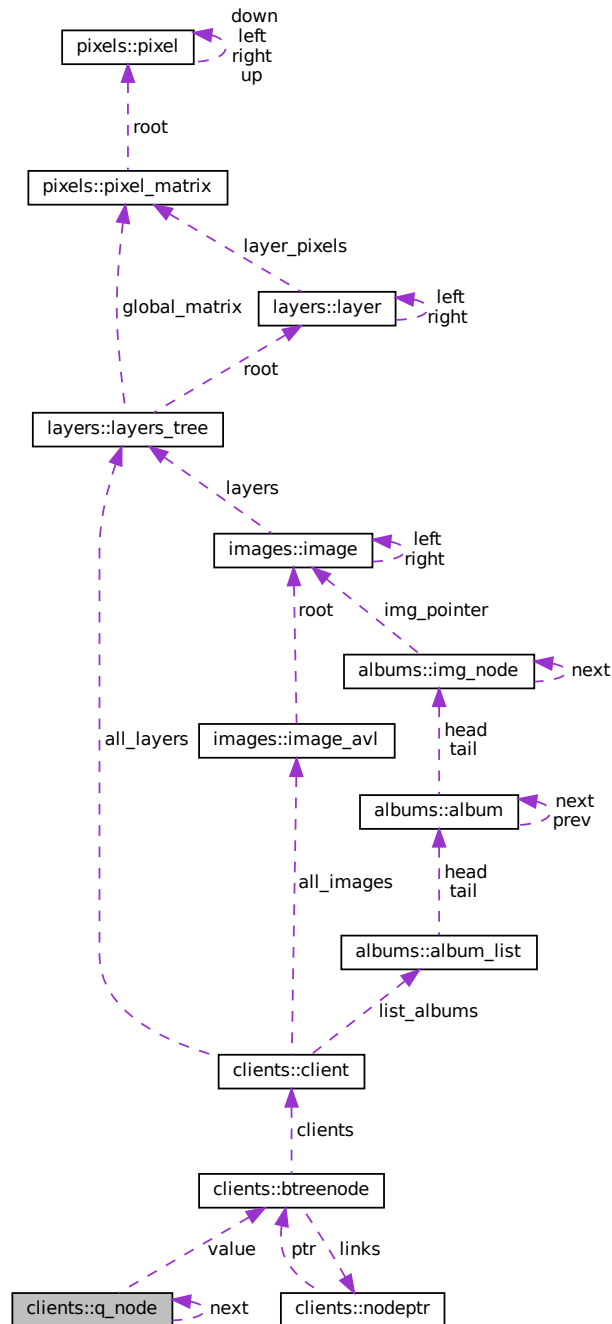
The documentation for this type was generated from the following file:

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/pixels.f90](#)

## 5.17 clients::q\_node Type Reference

Type to represent a queue node.

Collaboration diagram for clients::q\_node:



### Public Attributes

- `type(btreenode)`, pointer `value` => `null()`



*Value of the node.*

- `type(q_node)`, pointer `next` => `null()`

*Pointer to the next node.*

### 5.17.1 Detailed Description

Type to represent a queue node.

Definition at line 50 of file `clients.f90`.

### 5.17.2 Member Data Documentation

#### 5.17.2.1 next

```
type(q_node), pointer clients::q_node::next => null()
```

Pointer to the next node.

Definition at line 52 of file `clients.f90`.

#### 5.17.2.2 value

```
type(btreenode), pointer clients::q_node::value => null()
```

Value of the node.

Definition at line 51 of file `clients.f90`.

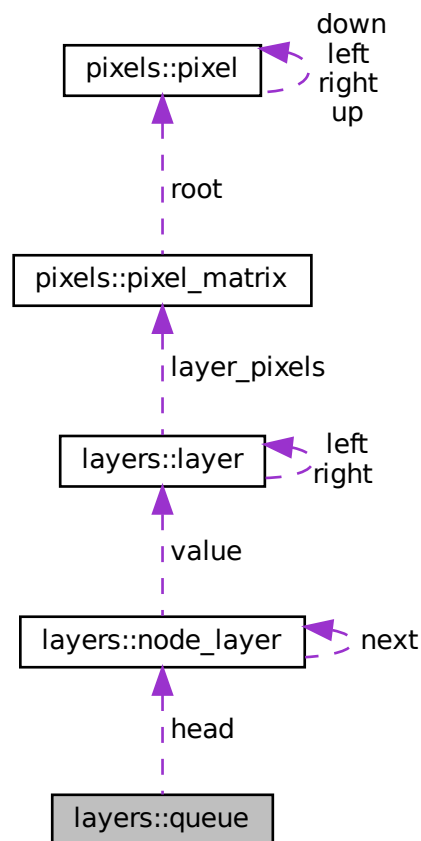
The documentation for this type was generated from the following file:

- `/home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90`

## 5.18 layers::queue Type Reference

Type to represent a queue of layers.

Collaboration diagram for layers::queue:



### Public Member Functions

- procedure [enqueue](#)
- procedure [dequeue](#)
- procedure [is\\_empty](#)

### Public Attributes

- type([node\\_layer](#)), pointer [head](#) => null()  
*Pointer to the head of the queue.*

### 5.18.1 Detailed Description

Type to represent a queue of layers.

Definition at line 52 of file layers.f90.

### 5.18.2 Member Function/Subroutine Documentation

#### 5.18.2.1 dequeue()

```
procedure layers::queue::dequeue
```

Definition at line 56 of file layers.f90.

#### 5.18.2.2 enqueue()

```
procedure layers::queue::enqueue
```

Definition at line 55 of file layers.f90.

#### 5.18.2.3 is\_empty()

```
procedure layers::queue::is_empty
```

Definition at line 57 of file layers.f90.

### 5.18.3 Member Data Documentation

#### 5.18.3.1 head

```
type(node\_layer), pointer layers::queue::head => null()
```

Pointer to the head of the queue.

Definition at line 53 of file layers.f90.

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

- [/home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90](#)



## Chapter 6

# File Documentation

### 6.1 /home/diego/Documents/EDD/Fase 2 Documentation/src/albums.f90 File Reference

Module for albums and images management.

#### Data Types

- type `albums::img_node`  
*Type to store the image node.*
- type `albums::album`  
*Type to store the album.*
- type `albums::album_list`  
*Type to store the album list.*

#### Modules

- module `albums`

#### Functions/Subroutines

- subroutine `albums::gen_album_graph` (this, unit)  
*Generates the graph of the albums.*
- subroutine `albums::show_album_images` (this, id\_album)  
*Shows the images of an album.*
- subroutine `albums::show_albums` (this)  
*Shows the albums.*
- subroutine `albums::remove_image_from_album` (this, ralbum\_id, image\_id)  
*Removes an image from an album.*
- subroutine `albums::add_image_to_album` (this, ialbum\_id, image\_node)  
*Adds an image to an album.*
- type(image) function, pointer `albums::search_in_album` (this, salbum\_id, image\_id)  
*Searches an image in an album.*

- type(album) function, pointer `albums::get_album` (this, id)  
*Gets an album.*
- subroutine `albums::remove_album` (this, id)  
*Removes an album.*
- subroutine `albums::new_album` (this, album\_name)  
*Creates a new album.*
- subroutine `albums::add_image` (this, image\_node)  
*Adds an image to an album.*
- subroutine `albums::remove_image` (this, id)  
*Removes an image from an album.*
- type(image) function, pointer `albums::get_image` (this, id)  
*Gets an image.*
- subroutine `albums::show_images` (this)  
*Shows the images of an album.*

## Variables

- integer `albums::album_id` = 0  
*Global variable to store the album id.*

### 6.1.1 Detailed Description

Module for albums and images management.

Author

Diego Cali

Date

2024

Version

1.0

This module contains the definition of the albums and images management

## 6.2 /home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90 File Reference

Module to manage clients in a B-tree.

## Data Types

- type `clients::client`  
*Type to represent a client.*
- type `clients::nodeptr`  
*Type to represent a node pointer.*
- type `clients::breenode`  
*Type to represent a B-tree node.*
- type `clients::btree_clients`  
*Type to represent a B-tree.*
- type `clients::q_node`  
*Type to represent a queue node.*
- type `clients::client_queue`  
*Type to represent a client queue.*

## Modules

- module `clients`

## Functions/Subroutines

- subroutine `clients::amplitude_traversal` (this)  
*Subroutine to traverse the tree in amplitude.*
- logical function `clients::is_empty` (this)  
*Function to check if the queue is empty.*
- type(breenode) function, pointer `clients::dequeue` (this)  
*Function to dequeue a node from the queue.*
- subroutine `clients::enqueue` (this, tree\_node)  
*Subroutine to enqueue a node in the queue.*
- subroutine `clients::add_client` (this, new\_client)  
*Subroutine to add a client to the tree.*
- recursive logical function `clients::set_value` (new\_client, pclient, node, child)  
*Recursive function to set the value of the tree.*
- type(breenode) function, pointer `clients::create_node` (this, new\_client, child)  
*Function to create a node.*
- subroutine `clients::insert_node` (pclient, pos, node, child)  
*Subroutine to insert a node.*
- subroutine `clients::split_node` (new\_client, pclient, pos, node, child, new\_node)  
*Subroutine to split a node.*
- subroutine `clients::traversal` (this, node)  
*Subroutine to traverse the tree.*
- subroutine `clients::clients_dot` (this, node, unit)  
*Subroutine to print the tree in dot format.*
- recursive type(client) function, pointer `clients::search_client` (this, node, client\_id)  
*Function to search a client in the tree.*
- subroutine `clients::delete_client` (this, client\_id)  
*Subroutine to delete a client from the tree.*
- subroutine `clients::delete_client_rec` (this, father, temp, client\_id)  
*Recursive subroutine to delete a client from the tree.*
- logical function `clients::find` (this, client\_id)  
*Function to find a client in the node.*

## Variables

- integer `clients::g_id` = 1

*Global variable to assign the id of the nodes.*

### 6.2.1 Detailed Description

Module to manage clients in a B-tree.

#### Author

Diego Cali

#### Date

2024

#### Version

1.0

This module contains the definition of the clients module, which is used to manage clients in a B-tree. The module contains the definition of the client type, which contains the name, dpi, password, and the albums and layers of the client. The module also contains the definition of the B-tree node type, which contains the clients and links to other nodes. The module contains the definition of the B-tree type, which contains the root of the tree. The module contains the definition of the node pointer type, which is used to traverse the tree. The module contains the definition of the queue node type, which is used to traverse the tree in amplitude. The module contains the definition of the client queue type, which is used to traverse the tree in amplitude. The module contains the definition of the functions and subroutines to add, search, delete, traverse, and print the tree. The module contains the definition of the functions and subroutines to insert, split, and set the value of the tree. The module contains the definition of the functions and subroutines to enqueue, dequeue, and check if the queue is empty

## 6.3 /home/diego/Documents/EDD/Fase 2 Documentation/src/filehandler.f90 File Reference

Module that handles the reading of the json files and the initialization of the data structures.

### Data Types

- type `filehandler::fhandler`

*Type that handles the reading of the json files and the initialization of the data structures.*

### Modules

- module `filehandler`



## Functions/Subroutines

- subroutine `filehandler::initialize_admin` (this)  
*Initializes the data structures for the admin user.*
- subroutine `filehandler::initialize_user` (this)  
*Initializes the data structures for the user.*
- subroutine `filehandler::set_user` (this, user)  
*Sets the user data structures to the fhandler object.*
- subroutine `filehandler::read_imgs` (this)  
*Reads the images from the json file.*
- subroutine `filehandler::read_layers` (this)  
*Reads the layers from the json file.*
- subroutine `filehandler::read_clients` (this)  
*Reads the clients from the json file.*
- subroutine `filehandler::read_albums` (this)  
*Reads the albums from the json file.*

### 6.3.1 Detailed Description

Module that handles the reading of the json files and the initialization of the data structures.

#### Author

Diego Cali

#### Date

2024

#### Version

1.0

This module is in charge of reading the json files and initializing the data structures that will be used in the program

## 6.4 /home/diego/Documents/EDD/Fase 2 Documentation/src/images.f90 File Reference

Module that contains the image and image\_avl types and their methods.

### Data Types

- type `images::image`  
*Type that represents a node in the AVL tree.*
- type `images::image_avl`  
*Type that represents an AVL tree of images.*

## Modules

- module [images](#)

## Functions/Subroutines

- subroutine [images::delete\\_img](#) (this, img\_id)  
*Method that deletes an image from the tree.*
- recursive type(image) function, pointer [images::delete\\_img\\_rec](#) (this, temp, img\_id)  
*Recursive method that deletes an image from the tree.*
- recursive type(image) function, pointer [images::min\\_child](#) (this, temp)  
*Method that returns the node with the minimum value in the tree.*
- subroutine [images::add\\_layer](#) (this, new\_layer)  
*Method that adds a layer to the image.*
- subroutine [images::add\\_img](#) (this, new\_image)  
*Method that adds an image to the tree.*
- subroutine [images::add\\_img\\_rec](#) (this, new\_image, tmp)  
*Recursive method that adds an image to the tree.*
- recursive type(image) function, pointer [images::search\\_img](#) (this, temp, img\_id)  
*Method that searches for an image in the tree.*
- type(image) function, pointer [images::srl](#) (this, t1)  
*Method that performs a single right rotation.*
- type(image) function, pointer [images::srr](#) (this, t1)  
*Method that performs a single left rotation.*
- type(image) function, pointer [images::drl](#) (this, tmp)  
*Method that performs a double right rotation.*
- type(image) function, pointer [images::drr](#) (this, tmp)  
*Method that performs a double left rotation.*
- integer function [images::get\\_max](#) (this, val1, val2)  
*Method that returns the maximum value between two integers.*
- integer function [images::get\\_height](#) (this, tmp)  
*Method that returns the height of a node.*
- subroutine [images::get\\_dot](#) (this, tmp, unit)  
*Method that generates the dot representation of the tree.*
- subroutine [images::get\\_dot\\_rec](#) (this, tmp, unit)  
*Recursive method that generates the dot representation of the tree.*
- subroutine [images::print\\_images](#) (this, temp)  
*Method that prints the images in the tree.*
- subroutine [images::gen\\_tree\\_subtree](#) (this, id\_img, unit)  
*Method that generates the dot representation of the tree and its subtree.*
- recursive subroutine [images::gen\\_layer\\_subtree](#) (current\_layer, unit)  
*Recursive method that generates the dot representation of the layers subtree.*
- subroutine [images::gen\\_img\\_traversal](#) (this, unit, id\_img)  
*Method that generates the dot representation of the image traversal.*

### 6.4.1 Detailed Description

Module that contains the image and image\_avl types and their methods.

Author

Diego Cali

Date

2024

Version

1.0

This module contains the image and image\_avl types and their methods

## 6.5 /home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90 File Reference

Module to handle layers of a neural network.

### Data Types

- type `layers::layer`  
*Type to represent a pixel matrix*
- type `layers::layers_tree`  
*Type to represent a binary tree of layers.*
- type `layers::node_layer`  
*Type to represent a node of a linked list of layers.*
- type `layers::queue`  
*Type to represent a queue of layers.*

### Modules

- module `layers`

## Functions/Subroutines

- subroutine `layers::gen_inorder` (this, tmp, limit, unit)  
*Subroutine to generate the matrix of pixels of a layer.*
- subroutine `layers::gen_postorder` (this, tmp, limit, unit)  
*Subroutine to generate the matrix of pixels of a layer.*
- subroutine `layers::gen_preorder` (this, tmp, limit, unit)  
*Subroutine to generate the matrix of pixels of a layer.*
- subroutine `layers::traverse_limited` (this, order, limit, unit)  
*Subroutine to traverse the layers tree in a limited way.*
- subroutine `layers::enqueue` (this, layer\_val)  
*Subroutine to add a layer to the layers tree.*
- type(layer) function, pointer `layers::dequeue` (this)  
*Subroutine to remove a layer from the queue.*
- logical function `layers::is_empty` (this)  
*Function to check if the queue is empty.*
- subroutine `layers::traverse_matrix` (this)  
*Subroutine to traverse the layers tree.*
- subroutine `layers::add` (this, new\_layer)  
*Subroutine to add a layer to the layers tree.*
- subroutine `layers::add_recursive` (this, new\_layer, tmp)  
*Subroutine to add a layer to the layers tree recursively.*
- subroutine `layers::add_copied_val` (this, new\_layer)  
*Subroutine to add a layer to the layers tree.*
- type(layer) function, pointer `layers::search` (this, id\_searched)  
*Function to search a layer in the layers tree.*
- subroutine `layers::preorder` (this, tmp)  
*Subroutine to traverse the layers tree in preorder.*
- subroutine `layers::inorder` (this, tmp)  
*Subroutine to traverse the layers tree in inorder.*
- subroutine `layers::postorder` (this, tmp)  
*Subroutine to traverse the layers tree in postorder.*
- subroutine `layers::gen_dot` (this, tmp, unit)  
*Subroutine to generate the dot file of the layers tree.*
- subroutine `layers::gen_dot_recursive` (this, tmp, unit)  
*Subroutine to generate the dot file of the layers tree recursively.*
- subroutine `layers::max_depth` (this)  
*Subroutine to calculate the max depth of the layers tree.*
- recursive integer function `layers::max_depth_rec` (this, root)  
*Function to calculate the max depth of the layers tree recursively.*
- subroutine `layers::leaf_layers` (this)  
*Subroutine to traverse the leaf layers of the layers tree.*
- subroutine `layers::leaf_layers_rec` (this, tmp)  
*Subroutine to traverse the leaf layers of the layers tree recursively.*
- subroutine `layers::list_layers` (this, option)  
*Subroutine to list the layers of the layers tree.*
- subroutine `layers::inorder_print` (this, tmp)  
*Subroutine to print the layers of the layers tree in inorder.*

### 6.5.1 Detailed Description

Module to handle layers of a neural network.

Author

Diego Cali

Date

2024

Version

1.0

This module contains the definition of the `layers_tree` type, which is a binary tree of layers. Each layer has an id, a number of pixels, a `pixel_matrix` and two pointers to other layers. The module also contains the definition of the `pixel_matrix` type, which is a matrix of pixels. The module also contains the definition of the `node_layer` type, which is a node of a linked list of layers. The module also contains the definition of the `queue` type, which is a queue of layers. The module contains the following subroutines and functions:

## 6.6 /home/diego/Documents/EDD/Fase 2 Documentation/src/pixels.f90 File Reference

Module to handle pixels in a matrix.

### Data Types

- type `pixels::pixel`  
*Type to represent a pixel.*
- type `pixels::pixel_matrix`  
*Type to represent a matrix of pixels.*

### Modules

- module `pixels`

## Functions/Subroutines

- subroutine `pixels::insert` (this, x, y, value, color)  
*Insert a pixel in the matrix.*
- type(pixel) function, pointer `pixels::search_row` (this, y)  
*Search a row in the matrix.*
- type(pixel) function, pointer `pixels::search_column` (this, x)  
*Search a column in the matrix.*
- logical function `pixels::node_exists` (this, new\_node)  
*Check if a node exists in the matrix.*
- type(pixel) function, pointer `pixels::insert_row_header` (this, y)  
*Insert a row header in the matrix.*
- subroutine `pixels::insert_in_row` (this, new\_node, row\_header)  
*Insert a pixel in a row.*
- type(pixel) function, pointer `pixels::insert_column_header` (this, x)  
*Insert a column header in the matrix.*
- subroutine `pixels::insert_in_column` (this, new\_node, column\_header)  
*Insert a pixel in a column.*
- subroutine `pixels::print_headers` (this)  
*Print the headers of the matrix.*
- logical function `pixels::get_value` (this, x, y)  
*Get the value of a pixel.*
- type(pixel) function, pointer `pixels::get_node` (this, x, y)  
*Get the node in the matrix.*
- subroutine `pixels::self_print` (this)  
*Print the matrix.*
- subroutine `pixels::graph_pixels` (this, unit)  
*Generate a graph of the pixels.*
- subroutine `pixels::gen_matrix` (this, g\_matrix)  
*Generate a matrix of pixels.*
- subroutine `pixels::global_m_dot` (this, unit)  
*Generate a graph of the pixels.*

## Variables

- integer `pixels::id` = 0  
*Unique identifier for each pixel.*

### 6.6.1 Detailed Description

Module to handle pixels in a matrix.

#### Author

Diego Cali

#### Date

2024

#### Version

1.0

This module is used to handle pixels in a matrix. It is possible to insert, search, print and generate a graph of the pixels.

# Index

/home/diego/Documents/EDD/Fase 2 Documenta-  
tion/src/albums.f90, [113](#)  
/home/diego/Documents/EDD/Fase 2 Documenta-  
tion/src/clients.f90, [114](#)  
/home/diego/Documents/EDD/Fase 2 Documenta-  
tion/src/filehandler.f90, [116](#)  
/home/diego/Documents/EDD/Fase 2 Documenta-  
tion/src/images.f90, [117](#)  
/home/diego/Documents/EDD/Fase 2 Documenta-  
tion/src/layers.f90, [119](#)  
/home/diego/Documents/EDD/Fase 2 Documenta-  
tion/src/pixels.f90, [121](#)

add  
  layers, [35](#)  
  layers::layers\_tree, [92](#)  
add\_client  
  clients, [14](#)  
  clients::btree\_clients, [63](#)  
add\_copied\_val  
  layers, [35](#)  
  layers::layers\_tree, [92](#)  
add\_image  
  albums, [8](#)  
  albums::album, [55](#)  
add\_image\_to\_album  
  albums, [8](#)  
  albums::album\_list, [59](#)  
add\_img  
  images, [26](#)  
  images::image\_avl, [83](#)  
add\_img\_rec  
  images, [26](#)  
  images::image\_avl, [83](#)  
add\_layer  
  images, [26](#)  
  images::image, [80](#)  
add\_recursive  
  layers, [36](#)  
  layers::layers\_tree, [93](#)  
album\_id  
  albums, [13](#)  
albums, [7](#)  
  add\_image, [8](#)  
  add\_image\_to\_album, [8](#)  
  album\_id, [13](#)  
  gen\_album\_graph, [8](#)  
  get\_album, [9](#)  
  get\_image, [9](#)  
  new\_album, [10](#)  
  remove\_album, [10](#)  
  remove\_image, [10](#)  
  remove\_image\_from\_album, [11](#)  
  search\_in\_album, [11](#)  
  show\_album\_images, [12](#)  
  show\_albums, [12](#)  
  show\_images, [12](#)  
albums::album, [53](#)  
  add\_image, [55](#)  
  get\_image, [55](#)  
  head, [56](#)  
  id, [56](#)  
  name, [56](#)  
  next, [56](#)  
  prev, [56](#)  
  remove\_image, [55](#)  
  show\_images, [55](#)  
  size, [57](#)  
  tail, [57](#)  
albums::album\_list, [57](#)  
  add\_image\_to\_album, [59](#)  
  gen\_album\_graph, [59](#)  
  get\_album, [59](#)  
  head, [61](#)  
  new\_album, [59](#)  
  remove\_album, [60](#)  
  remove\_image\_from\_album, [60](#)  
  search\_in\_album, [60](#)  
  show\_album\_images, [60](#)  
  show\_albums, [60](#)  
  size, [61](#)  
  tail, [61](#)  
albums::img\_node, [87](#)  
  id, [88](#)  
  img\_pointer, [88](#)  
  next, [88](#)  
albums\_db  
  filehandler::fhandler, [77](#)  
all\_images  
  clients::client, [70](#)  
all\_layers  
  clients::client, [70](#)  
amplitude\_traversal  
  clients, [14](#)  
  clients::btree\_clients, [63](#)  
  
clients, [13](#)  
  add\_client, [14](#)  
  amplitude\_traversal, [14](#)  
  clients::breenode, [67](#)

- clients\_dot, 15
- create\_node, 15
- delete\_client, 16
- delete\_client\_rec, 16
- dequeue, 17
- enqueue, 17
- find, 17
- g\_id, 22
- insert\_node, 18
- is\_empty, 19
- search\_client, 19
- set\_value, 19
- split\_node, 20
- traversal, 21
- clients::btree\_clients, 62
  - add\_client, 63
  - amplitude\_traversal, 63
  - clients\_dot, 63
  - create\_node, 63
  - delete\_client, 64
  - delete\_client\_rec, 64
  - root, 64
  - search\_client, 64
  - traversal, 64
- clients::btreenode, 65
  - clients, 67
  - find, 67
  - id, 67
  - links, 67
  - num, 68
- clients::client, 68
  - all\_images, 70
  - all\_layers, 70
  - dpi, 70
  - list\_albums, 70
  - name, 71
  - password, 71
- clients::client\_queue, 71
  - dequeue, 73
  - enqueue, 73
  - head, 73
  - is\_empty, 73
- clients::nodeptr, 98
  - ptr, 100
- clients::q\_node, 108
  - next, 109
  - value, 109
- clients\_db
  - filehandler::fhandler, 77
- clients\_dot
  - clients, 15
  - clients::btree\_clients, 63
- color
  - pixels::pixel, 101
- create\_node
  - clients, 15
  - clients::btree\_clients, 63
- delete\_client
  - clients, 16
  - clients::btree\_clients, 64
- delete\_client\_rec
  - clients, 16
  - clients::btree\_clients, 64
- delete\_img
  - images, 27
  - images::image\_avl, 83
- delete\_img\_rec
  - images, 27
  - images::image\_avl, 83
- dequeue
  - clients, 17
  - clients::client\_queue, 73
  - layers, 36
  - layers::queue, 111
- down
  - pixels::pixel, 101
- dpi
  - clients::client, 70
- drl
  - images, 27
  - images::image\_avl, 84
- drr
  - images, 28
  - images::image\_avl, 84
- enqueue
  - clients, 17
  - clients::client\_queue, 73
  - layers, 36
  - layers::queue, 111
- filehandler, 22
  - initialize\_admin, 22
  - initialize\_user, 23
  - read\_albums, 23
  - read\_clients, 23
  - read\_imgs, 24
  - read\_layers, 24
  - set\_user, 24
- filehandler::fhandler, 74
  - albums\_db, 77
  - clients\_db, 77
  - images\_db, 78
  - initialize\_admin, 76
  - initialize\_user, 76
  - layers\_db, 78
  - read\_albums, 76
  - read\_clients, 76
  - read\_imgs, 77
  - read\_layers, 77
  - set\_user, 77
- find
  - clients, 17
  - clients::btreenode, 67
- g\_id
  - clients, 22



- gen\_album\_graph
  - albums, 8
  - albums::album\_list, 59
- gen\_dot
  - layers, 37
  - layers::layers\_tree, 93
- gen\_dot\_recursive
  - layers, 37
  - layers::layers\_tree, 93
- gen\_img\_traversal
  - images, 28
  - images::image\_avl, 84
- gen\_inorder
  - layers, 37
  - layers::layers\_tree, 93
- gen\_layer\_subtree
  - images, 29
- gen\_matrix
  - pixels, 45
  - pixels::pixel\_matrix, 104
- gen\_postorder
  - layers, 38
  - layers::layers\_tree, 93
- gen\_preorder
  - layers, 38
  - layers::layers\_tree, 93
- gen\_tree\_subtree
  - images, 29
  - images::image\_avl, 84
- get\_album
  - albums, 9
  - albums::album\_list, 59
- get\_dot
  - images, 30
  - images::image\_avl, 84
- get\_dot\_rec
  - images, 30
  - images::image\_avl, 84
- get\_height
  - images, 31
  - images::image\_avl, 85
- get\_image
  - albums, 9
  - albums::album, 55
- get\_max
  - images, 31
  - images::image\_avl, 85
- get\_node
  - pixels, 46
  - pixels::pixel\_matrix, 104
- get\_value
  - pixels, 46
  - pixels::pixel\_matrix, 105
- global\_m\_dot
  - pixels, 46
  - pixels::pixel\_matrix, 105
- global\_matrix
  - layers::layers\_tree, 96
- graph\_pixels
  - pixels, 47
  - pixels::pixel\_matrix, 105
- head
  - albums::album, 56
  - albums::album\_list, 61
  - clients::client\_queue, 73
  - layers::queue, 111
- height
  - images::image, 80
  - pixels::pixel\_matrix, 107
- id
  - albums::album, 56
  - albums::img\_node, 88
  - clients::btree\_node, 67
  - images::image, 80
  - layers::layer, 90
  - pixels, 51
  - pixels::pixel, 101
- images, 25
  - add\_img, 26
  - add\_img\_rec, 26
  - add\_layer, 26
  - delete\_img, 27
  - delete\_img\_rec, 27
  - drl, 27
  - drr, 28
  - gen\_img\_traversal, 28
  - gen\_layer\_subtree, 29
  - gen\_tree\_subtree, 29
  - get\_dot, 30
  - get\_dot\_rec, 30
  - get\_height, 31
  - get\_max, 31
  - min\_child, 31
  - print\_images, 32
  - search\_img, 32
  - srl, 33
  - srr, 33
- images::image, 78
  - add\_layer, 80
  - height, 80
  - id, 80
  - layers, 80
  - layers\_count, 80
  - left, 81
  - right, 81
- images::image\_avl, 81
  - add\_img, 83
  - add\_img\_rec, 83
  - delete\_img, 83
  - delete\_img\_rec, 83
  - drl, 84
  - drr, 84
  - gen\_img\_traversal, 84
  - gen\_tree\_subtree, 84
  - get\_dot, 84

- get\_dot\_rec, 84
- get\_height, 85
- get\_max, 85
- min\_child, 85
- print\_images, 85
- root, 86
- search\_img, 85
- srl, 85
- srr, 86
- total, 86
- images\_db
  - filehandler::fhandler, 78
- img\_pointer
  - albums::img\_node, 88
- initialize\_admin
  - filehandler, 22
  - filehandler::fhandler, 76
- initialize\_user
  - filehandler, 23
  - filehandler::fhandler, 76
- inorder
  - layers, 39
  - layers::layers\_tree, 94
- inorder\_print
  - layers, 39
  - layers::layers\_tree, 94
- insert
  - pixels, 47
  - pixels::pixel\_matrix, 105
- insert\_column\_header
  - pixels, 48
  - pixels::pixel\_matrix, 105
- insert\_in\_column
  - pixels, 48
  - pixels::pixel\_matrix, 105
- insert\_in\_row
  - pixels, 48
  - pixels::pixel\_matrix, 106
- insert\_node
  - clients, 18
- insert\_row\_header
  - pixels, 49
  - pixels::pixel\_matrix, 106
- is\_empty
  - clients, 19
  - clients::client\_queue, 73
  - layers, 39
  - layers::queue, 111
- layer\_pixels
  - layers::layer, 90
- layers, 34
  - add, 35
  - add\_copied\_val, 35
  - add\_recursive, 36
  - dequeue, 36
  - enqueue, 36
  - gen\_dot, 37
  - gen\_dot\_recursive, 37
  - gen\_inorder, 37
  - gen\_postorder, 38
  - gen\_preorder, 38
  - images::image, 80
  - inorder, 39
  - inorder\_print, 39
  - is\_empty, 39
  - leaf\_layers, 40
  - leaf\_layers\_rec, 40
  - list\_layers, 40
  - max\_depth, 41
  - max\_depth\_rec, 41
  - postorder, 41
  - preorder, 43
  - search, 43
  - traverse\_limited, 43
  - traverse\_matrix, 44
- layers::layer, 89
  - id, 90
  - layer\_pixels, 90
  - left, 90
  - pixels\_count, 90
  - right, 90
- layers::layers\_tree, 91
  - add, 92
  - add\_copied\_val, 92
  - add\_recursive, 93
  - gen\_dot, 93
  - gen\_dot\_recursive, 93
  - gen\_inorder, 93
  - gen\_postorder, 93
  - gen\_preorder, 93
  - global\_matrix, 96
  - inorder, 94
  - inorder\_print, 94
  - leaf\_layers, 94
  - leaf\_layers\_rec, 94
  - list\_layers, 94
  - max\_depth, 94
  - max\_depth\_rec, 95
  - postorder, 95
  - preorder, 95
  - root, 96
  - search, 95
  - total, 96
  - traverse\_limited, 95
  - traverse\_matrix, 95
- layers::node\_layer, 97
  - next, 97
  - value, 98
- layers::queue, 110
  - dequeue, 111
  - enqueue, 111
  - head, 111
  - is\_empty, 111
- layers\_count
  - images::image, 80
- layers\_db

- filehandler::fhandler, 78
- leaf\_layers
  - layers, 40
  - layers::layers\_tree, 94
- leaf\_layers\_rec
  - layers, 40
  - layers::layers\_tree, 94
- left
  - images::image, 81
  - layers::layer, 90
  - pixels::pixel, 102
- links
  - clients::btree\_node, 67
- list\_albums
  - clients::client, 70
- list\_layers
  - layers, 40
  - layers::layers\_tree, 94
- max\_depth
  - layers, 41
  - layers::layers\_tree, 94
- max\_depth\_rec
  - layers, 41
  - layers::layers\_tree, 95
- min\_child
  - images, 31
  - images::image\_avl, 85
- name
  - albums::album, 56
  - clients::client, 71
- new\_album
  - albums, 10
  - albums::album\_list, 59
- next
  - albums::album, 56
  - albums::img\_node, 88
  - clients::q\_node, 109
  - layers::node\_layer, 97
- node\_exists
  - pixels, 49
  - pixels::pixel\_matrix, 106
- num
  - clients::btree\_node, 68
- on
  - pixels::pixel, 102
- password
  - clients::client, 71
- pixels, 44
  - gen\_matrix, 45
  - get\_node, 46
  - get\_value, 46
  - global\_m\_dot, 46
  - graph\_pixels, 47
  - id, 51
  - insert, 47
  - insert\_column\_header, 48
  - insert\_in\_column, 48
  - insert\_in\_row, 48
  - insert\_row\_header, 49
  - node\_exists, 49
  - print\_headers, 50
  - search\_column, 50
  - search\_row, 50
  - self\_print, 51
- pixels::pixel, 100
  - color, 101
  - down, 101
  - id, 101
  - left, 102
  - on, 102
  - right, 102
  - up, 102
  - x, 102
  - y, 103
- pixels::pixel\_matrix, 103
  - gen\_matrix, 104
  - get\_node, 104
  - get\_value, 105
  - global\_m\_dot, 105
  - graph\_pixels, 105
  - height, 107
  - insert, 105
  - insert\_column\_header, 105
  - insert\_in\_column, 105
  - insert\_in\_row, 106
  - insert\_row\_header, 106
  - node\_exists, 106
  - print\_headers, 106
  - root, 107
  - search\_column, 106
  - search\_row, 106
  - self\_print, 107
  - width, 107
- pixels\_count
  - layers::layer, 90
- postorder
  - layers, 41
  - layers::layers\_tree, 95
- preorder
  - layers, 43
  - layers::layers\_tree, 95
- prev
  - albums::album, 56
- print\_headers
  - pixels, 50
  - pixels::pixel\_matrix, 106
- print\_images
  - images, 32
  - images::image\_avl, 85
- ptr
  - clients::node\_ptr, 100
- read\_albums
  - filehandler, 23

- filehandler::fhandler, 76
- read\_clients
  - filehandler, 23
  - filehandler::fhandler, 76
- read\_imgs
  - filehandler, 24
  - filehandler::fhandler, 77
- read\_layers
  - filehandler, 24
  - filehandler::fhandler, 77
- remove\_album
  - albums, 10
  - albums::album\_list, 60
- remove\_image
  - albums, 10
  - albums::album, 55
- remove\_image\_from\_album
  - albums, 11
  - albums::album\_list, 60
- right
  - images::image, 81
  - layers::layer, 90
  - pixels::pixel, 102
- root
  - clients::btree\_clients, 64
  - images::image\_avl, 86
  - layers::layers\_tree, 96
  - pixels::pixel\_matrix, 107
- search
  - layers, 43
  - layers::layers\_tree, 95
- search\_client
  - clients, 19
  - clients::btree\_clients, 64
- search\_column
  - pixels, 50
  - pixels::pixel\_matrix, 106
- search\_img
  - images, 32
  - images::image\_avl, 85
- search\_in\_album
  - albums, 11
  - albums::album\_list, 60
- search\_row
  - pixels, 50
  - pixels::pixel\_matrix, 106
- self\_print
  - pixels, 51
  - pixels::pixel\_matrix, 107
- set\_user
  - filehandler, 24
  - filehandler::fhandler, 77
- set\_value
  - clients, 19
- show\_album\_images
  - albums, 12
  - albums::album\_list, 60
- show\_albums
  - albums, 12
  - albums::album\_list, 60
- show\_images
  - albums, 12
  - albums::album, 55
- size
  - albums::album, 57
  - albums::album\_list, 61
- split\_node
  - clients, 20
- srl
  - images, 33
  - images::image\_avl, 85
- srr
  - images, 33
  - images::image\_avl, 86
- tail
  - albums::album, 57
  - albums::album\_list, 61
- total
  - images::image\_avl, 86
  - layers::layers\_tree, 96
- traversal
  - clients, 21
  - clients::btree\_clients, 64
- traverse\_limited
  - layers, 43
  - layers::layers\_tree, 95
- traverse\_matrix
  - layers, 44
  - layers::layers\_tree, 95
- up
  - pixels::pixel, 102
- value
  - clients::q\_node, 109
  - layers::node\_layer, 98
- width
  - pixels::pixel\_matrix, 107
- x
  - pixels::pixel, 102
- y
  - pixels::pixel, 103