

File Organization Final Work

Generated by Doxygen 1.9.3

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 bNode Struct Reference	5
3.1.1 Detailed Description	5
3.2 bTree Struct Reference	5
3.2.1 Detailed Description	6
3.3 index_data_s1 Struct Reference	6
3.3.1 Detailed Description	6
3.4 index_data_s2 Struct Reference	6
3.4.1 Detailed Description	7
3.5 index_s1 Struct Reference	7
3.5.1 Detailed Description	7
3.6 index_s2 Struct Reference	7
3.6.1 Detailed Description	7
3.7 key Struct Reference	8
3.7.1 Detailed Description	8
3.8 registry_data_s Struct Reference	8
3.8.1 Detailed Description	9
3.9 WField Struct Reference	9
3.9.1 Detailed Description	9
4 File Documentation	11
4.1 includes/btree.h File Reference	11
4.1.1 Detailed Description	12
4.1.2 Function Documentation	12
4.1.2.1 compareKeys()	12
4.1.2.2 create_btree_t1()	13
4.1.2.3 create_btree_t2()	13
4.1.2.4 destroyTree()	13
4.1.2.5 insertKey()	13
4.1.2.6 read_tree_header()	14
4.1.2.7 readNode()	14
4.1.2.8 searchID()	15
4.2 btree.h	15
4.3 includes/csv.h File Reference	16
4.3.1 Detailed Description	16
4.3.2 Function Documentation	16
4.3.2.1 read_from_csv()	16

4.4 csv.h	17
4.5 includes/database.h File Reference	17
4.5.1 Detailed Description	18
4.5.2 Function Documentation	18
4.5.2.1 create_index()	18
4.5.2.2 create_index_b()	18
4.5.2.3 create_table()	18
4.5.2.4 delete_from_where()	19
4.5.2.5 insert_into()	19
4.5.2.6 insert_into_b()	19
4.5.2.7 search_where_b()	20
4.5.2.8 select_from_RRN()	20
4.5.2.9 select_from_where()	20
4.5.2.10 update_where()	22
4.6 database.h	22
4.7 includes/file_t1.h File Reference	22
4.7.1 Detailed Description	23
4.7.2 Function Documentation	23
4.7.2.1 add_registry_t1()	23
4.7.2.2 close_file_t1()	24
4.7.2.3 create_file_t1()	24
4.7.2.4 insert_into_t1()	24
4.7.2.5 push_deleted_stack()	25
4.7.2.6 read_file_t1()	25
4.7.2.7 read_registry_t1()	25
4.7.2.8 search_rrn()	26
4.7.2.9 search_where_t1()	26
4.7.2.10 update_registry_t1()	26
4.7.2.11 update_RRN()	27
4.8 file_t1.h	27
4.9 includes/file_t2.h File Reference	27
4.9.1 Detailed Description	28
4.9.2 Function Documentation	28
4.9.2.1 add_registry_t2()	28
4.9.2.2 calculate_registry_size()	29
4.9.2.3 close_file_t2()	29
4.9.2.4 create_file_t2()	29
4.9.2.5 insert_into_t2()	30
4.9.2.6 read_file_t2()	30
4.9.2.7 read_registry_t2()	30
4.9.2.8 search_byteOffset()	31
4.9.2.9 search_where_t2()	31

4.9.2.10 update_byte_offset()	31
4.10 file_t2.h	32
4.11 includes/index_t1.h File Reference	32
4.11.1 Detailed Description	33
4.11.2 Function Documentation	33
4.11.2.1 binary_search_t1()	33
4.11.2.2 create_index_t1()	33
4.11.2.3 delete_where_t1()	34
4.11.2.4 get_index_t1()	34
4.11.2.5 update_where_t1()	34
4.11.2.6 write_index_file_t1()	35
4.12 index_t1.h	35
4.13 includes/index_t2.h File Reference	36
4.13.1 Detailed Description	37
4.13.2 Function Documentation	37
4.13.2.1 binary_search_t2()	37
4.13.2.2 create_index_t2()	37
4.13.2.3 delete_where_t2()	37
4.13.2.4 get_index_t2()	38
4.13.2.5 update_registry_t2()	38
4.13.2.6 update_where_t2()	39
4.13.2.7 write_index_file_t2()	39
4.14 index_t2.h	39
4.15 includes/myString.h File Reference	40
4.15.1 Detailed Description	40
4.15.2 Function Documentation	40
4.15.2.1 readString()	40
4.15.2.2 readString2()	41
4.15.2.3 stringEnd()	41
4.16 myString.h	41
4.17 includes/utils.h File Reference	42
4.17.1 Detailed Description	43
4.17.2 Typedef Documentation	43
4.17.2.1 Field	43
4.17.3 Function Documentation	43
4.17.3.1 check_status()	43
4.17.3.2 compare_registry()	43
4.17.3.3 free_field()	44
4.17.3.4 get_field()	44
4.17.3.5 logical_deletion()	44
4.17.3.6 open_file_rb()	45
4.17.3.7 open_file_rplusb()	45

4.17.3.8 open_file_wb()	45
4.17.3.9 print_registry()	46
4.17.3.10 search_field_id()	46
4.18 utils.h	46
4.19 main.c File Reference	47
4.19.1 Detailed Description	48
Index	49

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

bNode	Struct that represents a B-Tree node. The node type has 3 possible values: ROOT(0), MID(1) and LEAF(2)	5
bTree	Struct that represents a B-Tree	5
index_data_s1	Struct that represents a single item data in the index	6
index_data_s2	Struct that represents a single item data in the index	6
index_s1	Struct that stores a array of index_data_st1 and it size	7
index_s2	Struct that stores a array of index_data_st1 and it size	7
key	Struct that represents a B-Tree key. Every key has a id, file type and a rrn or byteoffset. A union "value" is used to store different type values on the key	8
registry_data_s	Struct that stores registry data	8
WField	Struct that stores searching data	9

Chapter 2

File Index

2.1 File List

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

main.c	File Organization final assignment	47
includes/ btree.h	Functions for managing B-Tree	11
includes/ csv.h	Header responsible for handling interactions with csv files	16
includes/ database.h	Functions to pre process functions 1-11	17
includes/ file_t1.h	Functions related to type 1 registries	22
includes/ file_t2.h	Functions related to type 2 registries	27
includes/ index_t1.h	Functions to deal with type 1 index files	32
includes/ index_t2.h	Functions to deal with type 1 index files	36
includes/ myString.h	Functions related to reading of strings	40
includes/ utils.h	Common functions across files in project	42

Chapter 3

Class Documentation

3.1 bNode Struct Reference

Struct that represents a B-Tree node. The node type has 3 possible values: ROOT(0), MID(1) and LEAF(2)

```
#include <btree.h>
```

Public Attributes

- char **nodeType**
- int **numKeys**
- [Key](#) **key** [MAX_KEYS+1]
- int **desc** [MAX_DESC+1]
- int **rrn**

3.1.1 Detailed Description

Struct that represents a B-Tree node. The node type has 3 possible values: ROOT(0), MID(1) and LEAF(2)

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

- [includes/btree.h](#)

3.2 bTree Struct Reference

Struct that represents a B-Tree.

```
#include <btree.h>
```

Public Attributes

- char **status**
- struct [bNode](#) * **root**
- int **rootRRN**
- int **nextRRN**
- int **numNodes**

3.2.1 Detailed Description

Struct that represents a B-Tree.

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

- [includes/btree.h](#)

3.3 index_data_s1 Struct Reference

Struct that represents a single item data in the index.

```
#include <index_t1.h>
```

Public Attributes

- int **id**
- int **rrn**

3.3.1 Detailed Description

Struct that represents a single item data in the index.

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

- [includes/index_t1.h](#)

3.4 index_data_s2 Struct Reference

Struct that represents a single item data in the index.

```
#include <index_t2.h>
```

Public Attributes

- int **id**
- long long int **byteOffset**

3.4.1 Detailed Description

Struct that represents a single item data in the index.

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

- includes/[index_t2.h](#)

3.5 index_s1 Struct Reference

Struct that stores a array of index_data_st1 and it size.

```
#include <index_t1.h>
```

Public Attributes

- [index_data_st1](#) * **data**
- int **size**

3.5.1 Detailed Description

Struct that stores a array of index_data_st1 and it size.

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

- includes/[index_t1.h](#)

3.6 index_s2 Struct Reference

Struct that stores a array of index_data_st1 and it size.

```
#include <index_t2.h>
```

Public Attributes

- [index_data_st2](#) * **data**
- int **size**

3.6.1 Detailed Description

Struct that stores a array of index_data_st1 and it size.

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

- includes/[index_t2.h](#)

3.7 key Struct Reference

Struct that represents a B-Tree key. Every key has a id, file type and a rrn or byteoffset. A union "value" is used to store different type values on the key.

```
#include <btree.h>
```

Public Attributes

- int **id**
- char **fileType**
- union {
 long int **byteOffset**
 int **rrn**
} **value**

3.7.1 Detailed Description

Struct that represents a B-Tree key. Every key has a id, file type and a rrn or byteoffset. A union "value" is used to store different type values on the key.

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

- includes/[btree.h](#)

3.8 registry_data_s Struct Reference

Struct that stores registry data.

```
#include <utils.h>
```

Public Attributes

- int **id**
- int **year**
- char **city** [50]
- int **amount**
- char **initials** [5]
- char **brand** [50]
- char **model** [50]

3.8.1 Detailed Description

Struct that stores registry data.

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

- includes/[utils.h](#)

3.9 WField Struct Reference

Struct that stores searching data.

```
#include <utils.h>
```

Public Attributes

- char * **name**
- char * **value**

3.9.1 Detailed Description

Struct that stores searching data.

The fields of this struct are strings. Name stores name of the field and Value its value.

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

- includes/[utils.h](#)

Chapter 4

File Documentation

4.1 includes/btree.h File Reference

Functions for managing B-Tree.

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

Classes

- struct [key](#)
Struct that represents a B-Tree key. Every key has a id, file type and a rrn or byteoffset. A union "value" is used to store different type values on the key.
- struct [bNode](#)
Struct that represents a B-Tree node. The node type has 3 possible values: ROOT(0), MID(1) and LEAF(2)
- struct [bTree](#)
Struct that represents a B-Tree.

Macros

- #define **INF** 10e8
- #define **MAX_KEYS** 3
- #define **MAX_DESC** 4
- #define **ROOT** '0'
- #define **MID** '1'
- #define **LEAF** '2'

Typedefs

- typedef struct [key](#) **Key**
Struct that represents a B-Tree key. Every key has a id, file type and a rrn or byteoffset. A union "value" is used to store different type values on the key.

Functions

- int [compareKeys](#) (const void *x, const void *y)
Compare two values to make operations in the B-Tree.
- struct [bNode](#) * [readNode](#) (FILE *indexFile, int nodeRRN, char type)
Read a node from the indexFile with the specified RRN.
- struct [bTree](#) * [read_tree_header](#) (FILE *indexFile, char type)
Reads the B-Tree header.
- [Key](#) [searchID](#) (FILE *indexFile, struct [bNode](#) *node, int id, char type)
Search for a key in B-Tree with id.
- struct [bNode](#) * [insertKey](#) (FILE *indexFile, struct [bNode](#) *node, [Key](#))
Insert a new key in B-Tree.
- void [destroyTree](#) (struct [bTree](#) *tree)
Free all allocated memory for B-Tree.
- void [create_btree_t1](#) (FILE *binFile, FILE *indexFile)
Creates a type 1 B-Tree index file based on registries binary file.
- void [create_btree_t2](#) (FILE *binFile, FILE *indexFile)
Creates a type 2 B-Tree index file based on registries binary file.

4.1.1 Detailed Description

Functions for managing B-Tree.

4.1.2 Function Documentation

4.1.2.1 [compareKeys\(\)](#)

```
int compareKeys (
    const void * x,
    const void * y )
```

Compare two values to make operations in the B-Tree.

Parameters

<i>x</i>	id 1
<i>y</i>	id 2

Returns

int

4.1.2.2 create_btree_t1()

```
void create_btree_t1 (
    FILE * binFile,
    FILE * indexFile )
```

Creates a type 1 B-Tree index file based on registries binary file.

Parameters

<i>binFile</i>	binFile
<i>indexFile</i>	B-Tree index file

4.1.2.3 create_btree_t2()

```
void create_btree_t2 (
    FILE * binFile,
    FILE * indexFile )
```

Creates a type 2 B-Tree index file based on registries binary file.

Parameters

<i>binFile</i>	binFile
<i>indexFile</i>	B-Tree index file

4.1.2.4 destroyTree()

```
void destroyTree (
    struct bTree * tree )
```

Free all allocated memory for B-Tree.

Parameters

<i>tree</i>	Allocated B-Tree
-------------	------------------

4.1.2.5 insertKey()

```
struct bNode * insertKey (
    FILE * indexFile,
```

```
struct bNode * node,  
Key )
```

Insert a new key in B-Tree.

Parameters

<i>indexFile</i>	B-Tree index file
<i>node</i>	B-Tree root

Returns

struct bNode*

4.1.2.6 read_tree_header()

```
struct bTree * read_tree_header (  
    FILE * indexFile,  
    char type )
```

Reads the B-Tree header.

Parameters

<i>indexFile</i>	B-Tree index file
<i>type</i>	Type of file. 1 for fixed size and 2 for variable size

Returns

struct bTree* A struct that represents B-Tree

4.1.2.7 readNode()

```
struct bNode * readNode (  
    FILE * indexFile,  
    int nodeRRN,  
    char type )
```

Read a node from the indexFile with the specified RRN.

Parameters

<i>indexFile</i>	B-Tree index file
<i>nodeRRN</i>	Node RRN
<i>type</i>	Type of file. 1 for fixed size and 2 for variable size

Returns

struct bNode* A struct that represents found node

4.1.2.8 searchID()

```
Key searchID (
    FILE * indexFile,
    struct bNode * node,
    int id,
    char type )
```

Search for a key in B-Tree with id.

Parameters

<i>node</i>	B-Tree root
<i>id</i>	ID to be searched

Returns

Key Struct

4.2 btree.h

[Go to the documentation of this file.](#)

```
1
5 #ifndef BTREE_H
6 #define BTREE_H
7
8 #include <stdio.h>
9 #include <stdlib.h>
10
11 #define INF 10e8
12
13 #define MAX_KEYS 3
14 #define MAX_DESC 4
15
16 //cada nó possui um tipo
17 #define ROOT '0'
18 #define MID '1'
19 #define LEAF '2'
20
21 typedef struct key{
22     int id;
23     char fileType;
24     union{
25         long int byteOffset;
26         int rrn;
27     } value;
28 }Key;
29
30 struct bNode{
31     char nodeType;
32     int numKeys;
33     Key key[MAX_KEYS+1];
34     int desc[MAX_DESC+1];
35     int rrn;
36 };
37
38 struct bTree{
39     char status;
40     struct bNode *root;
```

```

53     int rootRRN;
54     int nextRRN;
55     int numNodes;
56 };
57
58
66 int compareKeys (const void *x, const void *y);
67
76 struct bNode *readNode(FILE *indexFile, int nodeRRN, char type);
77
85 struct bTree *read_tree_header(FILE *indexFile, char type);
86
94 Key searchID(FILE *indexFile, struct bNode *node, int id, char type);
95
103 struct bNode *insertKey(FILE *indexFile, struct bNode *node, Key);
104
110 void destroyTree(struct bTree *tree);
111
118 void create_btree_t1(FILE *binFile, FILE *indexFile);
119
126 void create_btree_t2(FILE *binFile, FILE *indexFile);
127
128 #endif

```

4.3 includes/csv.h File Reference

Header responsible for handling interactions with csv files.

```

#include <stdio.h>
#include "utils.h"

```

Functions

- int [read_from_csv](#) (FILE *file, [registry_data_st](#) *data)
Reads a single line from a csv file.

4.3.1 Detailed Description

Header responsible for handling interactions with csv files.

4.3.2 Function Documentation

4.3.2.1 read_from_csv()

```

int read_from_csv (
    FILE * file,
    registry_data_st * data )

```

Reads a single line from a csv file.

Parameters

<i>file</i>	File that will be read
<i>data</i>	Struct that will store the data

Returns

int Returns 1 if it detects a EOF, 0 otherwise

4.4 csv.h

[Go to the documentation of this file.](#)

```

1
6 #ifndef __CSV_H__
7 #define __CSV_H__
8 #include <stdio.h>
9 #include "utils.h"
10
18 int read_from_csv(FILE *file, registry_data_st *data);
19
20 #endif

```

4.5 includes/database.h File Reference

Functions to pre process functions 1-11.

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

Functions

- void [create_table](#) (char type, const char *CSVFileName, const char *binFileName)
Function 1. Reads a CSV and creates a new file with its content.
- void [select_from](#) (FILE *binFile, char type)
Function 2. Prints whole binary file.
- void [select_from_where](#) (FILE *binFile, char type)
Function 3. Select some registries from file and print them.
- void [select_from_RRN](#) (FILE *binFile)
Function 4. Select a registry with specified RRN and print it Type 1 function.
- void [create_index](#) (char type, FILE *binFile, FILE *indexFile)
Function 5. Create a index file based on binFile.
- void [delete_from_where](#) (FILE *binFile, const char *indexFilename, char type)
Function 6. Delete registries with specified fields.
- void [insert_into](#) (FILE *binFile, const char *indexFilename, char type)
Function 7. Insert a new registry in binFile.
- void [update_where](#) (FILE *binFile, const char *indexFilename, char type)
Function 8. Update fields of specified registries.
- void [search_where_b](#) (FILE *binFile, FILE *indexFile, char type)
Function 9. B-Tree index based search.
- void [insert_into_b](#) (FILE *binFile, FILE *indexFile, char type)
Function 10. Insert a new registry in binFile and B-Tree.
- void [create_index_b](#) (FILE *binFile, FILE *indexFile, char type)
Function 11. Create a B-Tree index based on binary file.

4.5.1 Detailed Description

Functions to pre process functions 1-11.

4.5.2 Function Documentation

4.5.2.1 create_index()

```
void create_index (
    char type,
    FILE * binFile,
    FILE * indexFile )
```

Function 5. Create a index file based on binFile.

Parameters

<i>type</i>	Type of file. 1 for fixed size and 2 for variable size
<i>binFile</i>	File with registries
<i>indexFile</i>	Index File

4.5.2.2 create_index_b()

```
void create_index_b (
    FILE * binFile,
    FILE * indexFile,
    char type )
```

Function 11. Create a B-Tree index based on binary file.

Parameters

<i>binFile</i>	binFile
<i>indexFile</i>	indexFile
<i>type</i>	Type of file. 1 for fixed size and 2 for variable size

4.5.2.3 create_table()

```
void create_table (
    char type,
```



```
const char * CSVFileName,  
const char * binFileName )
```

Function 1. Reads a CSV and creates a new file with its content.

Parameters

<i>type</i>	Type of file. 1 for fixed size and 2 for variable size
<i>CSVFilename</i>	Name of CSV file
<i>binFile</i>	Name of new file

4.5.2.4 delete_from_where()

```
void delete_from_where (  
    FILE * binFile,  
    const char * indexFilename,  
    char type )
```

Function 6. Delete registries with specified fields.

Parameters

<i>binFile</i>	binFile
<i>indexFilename</i>	indexFile
<i>type</i>	Type of file. 1 for fixed size and 2 for variable size

4.5.2.5 insert_into()

```
void insert_into (  
    FILE * binFile,  
    const char * indexFilename,  
    char type )
```

Function 7. Insert a new registry in binFile.

Parameters

<i>binFile</i>	binFile
<i>indexFilename</i>	indexFilename
<i>type</i>	Type of file. 1 for fixed size and 2 for variable size

4.5.2.6 insert_into_b()

```
void insert_into_b (  

```

```
FILE * binFile,
FILE * indexFile,
char type )
```

Function 10. Insert a new registry in binFile and B-Tree.

Parameters

<i>binFile</i>	binFile
<i>indexFile</i>	indexFile
<i>type</i>	Type of file. 1 for fixed size and 2 for variable size

4.5.2.7 search_where_b()

```
void search_where_b (
    FILE * binFile,
    FILE * indexFile,
    char type )
```

Function 9. B-Tree index based search.

Parameters

<i>binFile</i>	binFile
<i>indexFile</i>	indexFile
<i>type</i>	Type of file. 1 for fixed size and 2 for variable size

4.5.2.8 select_from_RRN()

```
void select_from_RRN (
    FILE * binFile )
```

Function 4. Select a registry with specified RRN and print it Type 1 function.

Parameters

<i>binFile</i>	
----------------	--

4.5.2.9 select_from_where()

```
void select_from_where (
    FILE * binFile,
    char type )
```

Function 3. Select some registries from file and print them.

Parameters

<i>binFile</i>	File
----------------	------

4.5.2.10 update_where()

```
void update_where (
    FILE * binFile,
    const char * indexFilename,
    char type )
```

Function 8. Update fields of specified registries.

Parameters

<i>binFile</i>	binFile
<i>indexFilename</i>	indexFilename
<i>type</i>	Type of file. 1 for fixed size and 2 for variable size

4.6 database.h

[Go to the documentation of this file.](#)

```
1
6 #ifndef __DATABASE_H__
7 #define __DATABASE_H__
8
9 #include <stdio.h>
10
18 void create_table(char type, const char *CSVFileName, const char *binFileName);
19
23 void select_from(FILE *binFile, char type);
24
30 void select_from_where(FILE *binFile, char type);
31
38 void select_from_RRN(FILE *binFile);
39
47 void create_index(char type, FILE *binFile, FILE *indexFile);
48
56 void delete_from_where(FILE *binFile, const char *indexFilename, char type);
57
65 void insert_into(FILE *binFile, const char *indexFilename, char type);
66
74 void update_where(FILE *binFile, const char *indexFilename, char type);
75
83 void search_where_b(FILE *binFile, FILE *indexFile, char type);
84
92 void insert_into_b(FILE *binFile, FILE *indexFile, char type);
93
101 void create_index_b(FILE *binFile, FILE *indexFile, char type);
102
103 #endif
```

4.7 includes/file_t1.h File Reference

Functions related to type 1 registries.

```
#include <stdio.h>
#include "csv.h"
#include "index_t1.h"
```

Functions

- FILE * [create_file_t1](#) (const char *filename)
Creates a file of type 1 and writes it's header.
- void [add_registry_t1](#) (FILE *file, [registry_data_st](#) data)
Adds a registry to a type 1 file.
- [registry_data_st](#) [read_registry_t1](#) (FILE *file)
Reads a variable size registry of specified size.
- void [read_file_t1](#) (FILE *file)
Read and print all registries in a type 1 file.
- void [search_where_t1](#) (FILE *file, [Field](#) *f, int n)
Search and print for registries in file with Field f equivalency.
- void [close_file_t1](#) (FILE *file)
Adequately closes a type 1 file.
- void [search_rrn](#) (FILE *file, int rrn)
Search and print registry with specified RRN.
- void [update_RRN](#) (FILE *file, int finalRRN)
Updates the RRN field of a header from a type 1 file.
- void [push_deleted_stack](#) (FILE *binFile, int rrn)
Insert new RRN in deleted stack.
- [registry_data_st](#) [update_registry_t1](#) ([registry_data_st](#) newReg, [Field](#) *update, int n)
Updates the fields of a registry.
- int [insert_into_t1](#) (FILE *binFile, [registry_data_st](#) data)
Inserts a new registry in file.

4.7.1 Detailed Description

Functions related to type 1 registries.

4.7.2 Function Documentation

4.7.2.1 [add_registry_t1\(\)](#)

```
void add_registry_t1 (
    FILE * file,
    registry\_data\_st data )
```

Adds a registry to a type 1 file.

Parameters

<i>file</i>	File of type 1
<i>data</i>	Registry data

4.7.2.2 close_file_t1()

```
void close_file_t1 (
    FILE * file )
```

Adequately closes a type 1 file.

Parameters

<i>file</i>	File to be closed
-------------	-------------------

4.7.2.3 create_file_t1()

```
FILE * create_file_t1 (
    const char * filename )
```

Creates a file of type 1 and writes it's header.

Parameters

<i>filename</i>	Name of file to be created
-----------------	----------------------------

Returns

FILE* Pointer to file

4.7.2.4 insert_into_t1()

```
int insert_into_t1 (
    FILE * binFile,
    registry_data_st data )
```

Inserts a new registry in file.

Parameters

<i>binFile</i>	Type 1 file
<i>data</i>	New registry data

4.7.2.5 push_deleted_stack()

```
void push_deleted_stack (
    FILE * binFile,
    int rrn )
```

Insert new RRN in deleted stack.

Parameters

<i>binFile</i>	Stack file
<i>rrn</i>	RRN to be deleted

4.7.2.6 read_file_t1()

```
void read_file_t1 (
    FILE * file )
```

Read and print all registries in a type 1 file.

Parameters

<i>file</i>	File to be read
-------------	-----------------

4.7.2.7 read_registry_t1()

```
registry_data_st read_registry_t1 (
    FILE * file )
```

Reads a variable size registry of specified size.

Parameters

<i>file</i>	File that contains the registry
<i>registrySize</i>	size of registry to be read

Returns

csv_data_st Read data of registry

4.7.2.8 search_rrn()

```
void search_rrn (
    FILE * file,
    int rrn )
```

Search and print registry with specified RRN.

Parameters

<i>file</i>	File to be searched
<i>rrn</i>	RRN of registry

4.7.2.9 search_where_t1()

```
void search_where_t1 (
    FILE * file,
    Field * f,
    int n )
```

Search and print for registries in file with Field *f* equivalency.

Parameters

<i>file</i>	File with registries
<i>f</i>	Array of structs to check
<i>n</i>	Size of array <i>f</i>

4.7.2.10 update_registry_t1()

```
registry_data_st update_registry_t1 (
    registry_data_st newReg,
    Field * update,
    int n )
```

Updates the fields of a registry.

Parameters

<i>newReg</i>	Registry to be updated
<i>update</i>	Array of new values
<i>n</i>	Size of array

Returns

registry_data_st New created registry

4.7.2.11 update_RRN()

```
void update_RRN (
    FILE * file,
    int finalRRN )
```

Updates the RRN field of a header from a type 1 file.

Parameters

<i>file</i>	File to be updated
<i>byteOffset</i>	New RRN

4.8 file_t1.h

[Go to the documentation of this file.](#)

```
1
5 #ifndef __FILE_T1_H__
6 #define __FILE_T1_H__
7 #include <stdio.h>
8 #include "csv.h"
9 #include "index_t1.h"
10
17 FILE *create_file_t1(const char *filename);
18
25 void add_registry_t1(FILE *file, registry_data_st data);
26
34 registry_data_st read_registry_t1(FILE *file);
35
41 void read_file_t1(FILE *file);
42
50 void search_where_t1(FILE *file, Field *f, int n);
51
57 void close_file_t1(FILE *file);
58
65 void search_rrn(FILE *file, int rrn);
66
73 void update_RRN(FILE *file, int finalRRN);
74
81 void push_deleted_stack(FILE *binFile, int rrn);
82
91 registry_data_st update_registry_t1(registry_data_st newReg, Field *update, int n);
92
99 int insert_into_t1(FILE *binFile, registry_data_st data);
100
101 #endif
```

4.9 includes/file_t2.h File Reference

Functions related to type 2 registries.

```
#include <stdio.h>
#include "../includes/csv.h"
```

Typedefs

- typedef long long int lli

Functions

- FILE * [create_file_t2](#) (const char *filename)
Creates a file of type 2 and writes it's header.
- int [calculate_registry_size](#) ([registry_data_st](#) data)
Calculate size of registry.
- void [add_registry_t2](#) (FILE *file, [registry_data_st](#) data, int additionalSize)
Adds a registry to a type 2 file.
- void [update_byte_offset](#) (FILE *file, lli byteOffset)
Updates the byte offset field of a header from a type 2 file.
- [registry_data_st](#) [read_registry_t2](#) (FILE *file)
Reads a variable size registry of specified size.
- void [read_file_t2](#) (FILE *file)
Read and print all registries in a type 2 file.
- void [close_file_t2](#) (FILE *file)
Adequately closes a type 2 file.
- void [search_where_t2](#) (FILE *file, [Field](#) *f, int n)
Search and print for registries in file with Field f equivalency.
- long int [insert_into_t2](#) (FILE *binFile, [registry_data_st](#) data)
Inserts a new registry in file.
- void [search_byteOffset](#) (FILE *file, long int byteOffset)
Search and print registry with specified byte offset.

4.9.1 Detailed Description

Functions related to type 2 registries.

4.9.2 Function Documentation

4.9.2.1 [add_registry_t2\(\)](#)

```
void add_registry_t2 (
    FILE * file,
    registry\_data\_st data,
    int additionalSize )
```

Adds a registry to a type 2 file.

Parameters

<i>file</i>	File of type 2
<i>data</i>	Registry data

4.9.2.2 calculate_registry_size()

```
int calculate_registry_size (
    registry_data_st data )
```

Calculate size of registry.

Parameters

<i>data</i>	Registry
-------------	----------

Returns

int Size of Registry

4.9.2.3 close_file_t2()

```
void close_file_t2 (
    FILE * file )
```

Adequately closes a type 2 file.

Parameters

<i>file</i>	File to be closed
-------------	-------------------

4.9.2.4 create_file_t2()

```
FILE * create_file_t2 (
    const char * filename )
```

Creates a file of type 2 and writes it's header.

Parameters

<i>filename</i>	Name of file to be created
-----------------	----------------------------

Returns

FILE* Pointer to file

4.9.2.5 insert_into_t2()

```
long int insert_into_t2 (
    FILE * binFile,
    registry_data_st data )
```

Inserts a new registry in file.

Parameters

<i>binFile</i>	Type 2 file
<i>data</i>	New registry data

4.9.2.6 read_file_t2()

```
void read_file_t2 (
    FILE * file )
```

Read and print all registries in a type 2 file.

Parameters

<i>file</i>	File to be read
-------------	-----------------

4.9.2.7 read_registry_t2()

```
registry_data_st read_registry_t2 (
    FILE * file )
```

Reads a variable size registry of specified size.

Parameters

<i>file</i>	File that contains the registry
<i>registrySize</i>	size of registry to be read

Returns

csv_data_st Read data of registry

4.9.2.8 search_byteOffset()

```
void search_byteOffset (
    FILE * file,
    long int byteOffset )
```

Search and print registry with specified byte offset.

Parameters

<i>file</i>	File to be searched
<i>byteOffset</i>	Byte offset of registry

4.9.2.9 search_where_t2()

```
void search_where_t2 (
    FILE * file,
    Field * f,
    int n )
```

Search and print for registries in file with Field f equivalency.

Parameters

<i>file</i>	File with registries
<i>f</i>	Array of structs to check
<i>n</i>	Size of array f

4.9.2.10 update_byte_offset()

```
void update_byte_offset (
    FILE * file,
    lli byteOffset )
```

Updates the byte offset field of a header from a type 2 file.

Parameters

<i>file</i>	File to be updated
<i>byteOffset</i>	New byte offset

4.10 file_t2.h

[Go to the documentation of this file.](#)

```

1
5 #ifndef __FILE_T2_H__
6 #define __FILE_T2_H__
7
8 #include <stdio.h>
9 #include "../includes/csv.h"
10
11 typedef long long int lli;
12
19 FILE *create_file_t2(const char *filename);
20
27 int calculate_registry_size(registry_data_st data);
28
35 void add_registry_t2(FILE *file, registry_data_st data, int additionalSize);
36
43 void update_byte_offset(FILE *file, lli byteOffset);
44
52 registry_data_st read_registry_t2(FILE* file);
53
59 void read_file_t2(FILE *file);
60
66 void close_file_t2(FILE *file);
67
75 void search_where_t2(FILE *file, Field *f, int n);
76
83 long int insert_into_t2(FILE *binFile, registry_data_st data);
84
91 void search_byteOffset(FILE *file, long int byteOffset);
92
93 #endif

```

4.11 includes/index_t1.h File Reference

Functions to deal with type 1 index files.

```

#include <stdlib.h>
#include "utils.h"
#include <stdio.h>

```

Classes

- struct [index_data_s1](#)
Struct that represents a single item data in the index.
- struct [index_s1](#)
Struct that stores a array of index_data_st1 and it size.

Macros

- #define **DATA_HEADER_SIZE** 182

Typedefs

- typedef struct [index_data_s1](#) **index_data_st1**
Struct that represents a single item data in the index.
- typedef struct [index_s1](#) **index_st1**
Struct that stores a array of index_data_st1 and it size.

Functions

- void `create_index_t1` (FILE *binFile, FILE *indexFile)
Create a index file based on binFile.
- void `write_index_file_t1` (FILE *indexFile, `index_st1` index)
Write index_st1 values at index file.
- `index_st1` `get_index_t1` (FILE *indexFile)
Read all index data in index file.
- void `delete_where_t1` (FILE *file, `index_st1` index, `Field` *f, int n)
Delete registries with specified fields.
- void `update_where_t1` (FILE *binFile, `index_st1` index, `Field` *search, int x, `Field` *update, int y)
Function 8-1. Update fields of specified registries.
- int `binary_search_t1` (`index_st1` index, int id)
Execute binary search for id at index_st.

4.11.1 Detailed Description

Functions to deal with type 1 index files.

4.11.2 Function Documentation

4.11.2.1 `binary_search_t1()`

```
int binary_search_t1 (  
    index_st1 index,  
    int id )
```

Execute binary search for id at index_st.

Parameters

<i>index</i>	Struct to be searched
<i>id</i>	ID to be found

Returns

int Found registry index

4.11.2.2 `create_index_t1()`

```
void create_index_t1 (  
    FILE * binFile,  
    FILE * indexFile )
```

Create a index file based on binFile.

Parameters

<i>binFile</i>	binFile
<i>indexFile</i>	indexFile

4.11.2.3 delete_where_t1()

```
void delete_where_t1 (
    FILE * file,
    index_st1 index,
    Field * f,
    int n )
```

Delete registries with specified fields.

Parameters

<i>file</i>	binFile
<i>f</i>	Array of structs to check
<i>n</i>	Size of array

4.11.2.4 get_index_t1()

```
index_st1 get_index_t1 (
    FILE * indexFile )
```

Read all index data in index file.

Parameters

<i>indexFile</i>	Index file
------------------	------------

Returns

index_st1 Returns a array of index data that was read

4.11.2.5 update_where_t1()

```
void update_where_t1 (
    FILE * binFile,
    index_st1 index,
```



```

    Field * search,
    int x,
    Field * update,
    int y )

```

Function 8-1. Update fields of specified registries.

Parameters

<i>binFile</i>	binFile
<i>indexFile</i>	indexFile
<i>search</i>	Array of fields present in registries that should be updated
<i>x</i>	Size of search
<i>update</i>	Array of fields to update
<i>y</i>	Size of update

4.11.2.6 write_index_file_t1()

```

void write_index_file_t1 (
    FILE * indexFile,
    index_st1 index )

```

Write index_st1 values at index file.

Parameters

<i>indexFile</i>	index file
<i>index</i>	Array of index data

4.12 index_t1.h

[Go to the documentation of this file.](#)

```

1
2 #ifndef __INDEX_T1_H__
3 #define __INDEX_T1_H__
4
5 #include <stdlib.h>
6 #include "utils.h"
7 #include <stdio.h>
8
9 #define DATA_HEADER_SIZE 182
10
11 typedef struct index_data_s1{
12     int id;
13     int rrn;
14 }index_data_st1;
15
16 typedef struct index_s1{
17     index_data_st1 *data;
18     int size;
19 }index_st1;
20
21 void create_index_t1(FILE *binFile, FILE *indexFile);
22
23 void write_index_file_t1(FILE *indexFile, index_st1 index);
24
25

```

```

52 index_st1 get_index_t1(FILE *indexFile);
53
61 void delete_where_t1(FILE *file, index_st1 index, Field *f, int n);
62
73 void update_where_t1(FILE *binFile, index_st1 index, Field *search, int x, Field *update, int y);
74
82 int binary_search_t1(index_st1 index, int id);
83
84 #endif

```

4.13 includes/index_t2.h File Reference

Functions to deal with type 1 index files.

```

#include <stdio.h>
#include "../includes/utils.h"

```

Classes

- struct [index_data_s2](#)
Struct that represents a single item data in the index.
- struct [index_s2](#)
Struct that stores a array of index_data_st1 and it size.

Typedefs

- typedef long long int [lli](#)
- typedef struct [index_data_s2](#) [index_data_st2](#)
Struct that represents a single item data in the index.
- typedef struct [index_s2](#) [index_st2](#)
Struct that stores a array of index_data_st1 and it size.

Functions

- void [create_index_t2](#) (FILE *dataFile, FILE *indexFile)
Create a index file based on binFile.
- void [delete_where_t2](#) (FILE *binFile, [index_st2](#) index, [Field](#) *search, int n)
Delete registries with specified fields.
- [registry_data_st](#) [update_registry_t2](#) ([registry_data_st](#) newReg, [Field](#) *update, int n)
Updates a single registry.
- int [binary_search_t2](#) ([index_st2](#) index, int id)
Execute binary search for id at index_st.
- [index_st2](#) [get_index_t2](#) (FILE *indexFile)
Read all index data in index file.
- void [write_index_file_t2](#) (FILE *indexFile, [index_st2](#) index)
Write index_st1 values at index file.
- void [update_where_t2](#) (FILE *binFile, [index_st2](#) index, [Field](#) *search, int x, [Field](#) *update, int y)
Function 8-1. Update fields of specified registries.

4.13.1 Detailed Description

Functions to deal with type 1 index files.

4.13.2 Function Documentation

4.13.2.1 `binary_search_t2()`

```
int binary_search_t2 (
    index_st2 index,
    int id )
```

Execute binary search for id at index_st.

Parameters

<i>index</i>	Struct to be searched
<i>id</i>	ID to be found

Returns

int Found registry index

4.13.2.2 `create_index_t2()`

```
void create_index_t2 (
    FILE * dataFile,
    FILE * indexFile )
```

Create a index file based on binFile.

Parameters

<i>binFile</i>	binFile
<i>indexFile</i>	indexFile

4.13.2.3 `delete_where_t2()`

```
void delete_where_t2 (
    FILE * binFile,
```

```

    index_st2 index,
    Field * search,
    int n )

```

Delete registries with specified fields.

Parameters

<i>file</i>	binFile
<i>f</i>	Array of structs to check
<i>n</i>	Size of array

4.13.2.4 get_index_t2()

```

index_st2 get_index_t2 (
    FILE * indexFile )

```

Read all index data in index file.

Parameters

<i>indexFile</i>	Index file
------------------	------------

Returns

index_st1 Returns a array of index data that was read

4.13.2.5 update_registry_t2()

```

registry_data_st update_registry_t2(
    registry_data_st newReg,
    Field * update,
    int n )

```

Updates a single registry.

Parameters

<i>newReg</i>	Registry to be updates
<i>update</i>	Field of new values
<i>n</i>	Size of update

Returns

registry_data_st Updated registry

4.13.2.6 update_where_t2()

```
void update_where_t2 (
    FILE * binFile,
    index_st2 index,
    Field * search,
    int x,
    Field * update,
    int y )
```

Function 8-1. Update fields of specified registries.

Parameters

<i>binFile</i>	binFile
<i>indexFile</i>	indexFile
<i>search</i>	Array of fields present in registries that should be updated
<i>x</i>	Size of search
<i>update</i>	Array of fields to update
<i>y</i>	Size of update

4.13.2.7 write_index_file_t2()

```
void write_index_file_t2 (
    FILE * indexFile,
    index_st2 index )
```

Write index_st1 values at index file.

Parameters

<i>indexFile</i>	index file
<i>index</i>	Array of index data

4.14 index_t2.h

[Go to the documentation of this file.](#)

```
1
5 #ifndef __INDEX_T2_H__
6 #define __INDEX_T2_H__
7
8 #include <stdio.h>
9 #include "../includes/utlis.h"
10
11 typedef long long int lli;
12
16 typedef struct index_data_s2{
```

```

17     int id;
18     long long int byteOffset;
19 }index_data_st2;
20
24 typedef struct index_s2{
25     index_data_st2 *data;
26     int size;
27 }index_st2;
28
35 void create_index_t2(FILE *dataFile, FILE *indexFile);
36
44 void delete_where_t2(FILE *binFile, index_st2 index, Field *search, int n);
45
54 registry_data_st update_registry_t2(registry_data_st newReg, Field *update, int n);
55
63 int binary_search_t2(index_st2 index, int id);
64
71 index_st2 get_index_t2(FILE *indexFile);
72
79 void write_index_file_t2(FILE *indexFile, index_st2 index);
80
91 void update_where_t2(FILE *binFile, index_st2 index, Field *search, int x, Field *update, int y);
92 #endif

```

4.15 includes/myString.h File Reference

Functions related to reading of strings.

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

Functions

- int [stringEnd](#) (int numSeparators, char separatorArray[], char c)
Retorna se a leitura da string deve ou não ser finalizada.
- char * [readFixedSizeString](#) (FILE *fp, int stringSize)
- char * [readString](#) ()
Função para leitura de strings.
- char * [readString2](#) (int num,...)
Função mais completa para leitura de strings.

4.15.1 Detailed Description

Functions related to reading of strings.

4.15.2 Function Documentation

4.15.2.1 [readString\(\)](#)

```
char * readString ( )
```

Função para leitura de strings.

Returns

char* string

4.15.2.2 readString2()

```
char * readString2 (
    int num,
    ... )
```

Função mais completa para leitura de strings.

Parameters

<i>num</i>	número de separadores que serão passados
...	separadores: EOF, quebra de linha etc...

Returns

char* string

4.15.2.3 stringEnd()

```
int stringEnd (
    int numSeparators,
    char separatorArray[],
    char c )
```

Retorna se a leitura da string deve ou não ser finalizada.

Parameters

<i>numSeparators</i>	número de separadores
<i>separatorArray</i>	array com todos finalizadores
<i>c</i>	caractere lido

Returns

int 1 para leitura finalizada, caso contrário, 0

4.16 myString.h

[Go to the documentation of this file.](#)

```
1
2 #ifndef __MY_STRING_H__
3 #define __MY_STRING_H__
4 #include <stdio.h>
5
6 int stringEnd(int numSeparators, char separatorArray[], char c);
7
8 char *readFixedSizeString(FILE *fp, int stringSize);
9
10 char* readString();
11
12 char* readString2(int num, ...);
13
14 #endif
```

4.17 includes/utils.h File Reference

Common functions across files in project.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "../includes/myString.h"
```

Classes

- struct [registry_data_s](#)
Struct that stores registry data.
- struct [WField](#)
Struct that stores searching data.

Macros

- #define **ARR_SIZE**(x) (sizeof(x) / sizeof((x)[0]))

Typedefs

- typedef struct [registry_data_s](#) **registry_data_st**
Struct that stores registry data.
- typedef struct [WField](#) **Field**
Struct that stores searching data.

Functions

- int [compare_registry](#) ([registry_data_st](#) registry, [Field](#) *f, int n)
Check if the fields of struct f exists in struct registry.
- void [print_registry](#) ([registry_data_st](#) registry)
Prints a registry.
- FILE * [open_file_rb](#) (const char *filename)
Open a binary file in "rb" mode This functino prints a error message when the file is not properly opened.
- FILE * [open_file_rplusb](#) (const char *filename)
Open a binary file in "r+b" mode This functino prints a error message when the file is not properly opened.
- FILE * [open_file_wb](#) (const char *filename)
Open a binary file in "wb" mode This functino prints a error message when the file is not properly opened.
- void **readline** (char *string)
- void **binarioNaTela** (const char *nomeArquivoBinario)
- char * **scan_quote_string** ()
- int [check_status](#) (FILE *)
Check file status.
- void [logical_deletion](#) (FILE *, int)
Logically deletes a registry The pointer should be at the beggining of the registry.
- int [search_field_id](#) ([Field](#) *f, int n)
Search for "id" in "Name" field at struct array "f" Procura se existe um campo de Name == id no array de structs "f".
- [Field](#) * [get_field](#) (int x)
Read a array of Field from stdin.
- void [free_field](#) ([Field](#) *f, int x)
Free an array of Field.
- [registry_data_st](#) **read_registry_line** ()

4.17.1 Detailed Description

Common functions across files in project.

4.17.2 Typedef Documentation

4.17.2.1 Field

```
typedef struct WField Field
```

Struct that stores searching data.

The fields of this struct are strings. Name stores name of the field and Value its value.

4.17.3 Function Documentation

4.17.3.1 check_status()

```
int check_status (
    FILE * )
```

Check file status.

Returns

If the file is corrupted, returns 0. Else, 1

4.17.3.2 compare_registry()

```
int compare_registry (
    registry_data_st registry,
    Field * f,
    int n )
```

Check if the fields of struct f exists in struct registry.

Parameters

<i>registry</i>	Struct that stores registry data
<i>f</i>	Array of searching values
<i>n</i>	Size of array f

Returns

int 1 if all fields exists, 0 else.

4.17.3.3 free_field()

```
void free_field (
    Field * f,
    int x )
```

Free an array of Field.

Parameters

<i>f</i>	Array of Field
<i>x</i>	Size of array

4.17.3.4 get_field()

```
Field * get_field (
    int x )
```

Read a array of Field from stdin.

Parameters

<i>x</i>	Size of the array
----------	-------------------

Returns

Field* Array of Field

4.17.3.5 logical_deletion()

```
void logical_deletion (
    FILE * ,
    int )
```

Logically deletes a registry The pointer should be at the beggining of the registry.

Parameters

<i>FILE*</i>	Binary file with the registry to be deleted
<i>int</i>	Size of the registry

4.17.3.6 open_file_rb()

```
FILE * open_file_rb (
    const char * filename )
```

Open a binary file in "rb" mode This functino prints a error message when the file is not properly opened.

Parameters

<i>filename</i>	File to be opened
-----------------	-------------------

Returns

FILE* Pointer to the file

4.17.3.7 open_file_rplusb()

```
FILE * open_file_rplusb (
    const char * filename )
```

Open a binary file in "r+b" mode This functino prints a error message when the file is not properly opened.

Parameters

<i>filename</i>	File to be opened
-----------------	-------------------

Returns

FILE* Pointer to the file

4.17.3.8 open_file_wb()

```
FILE * open_file_wb (
    const char * filename )
```

Open a binary file in "wb" mode This functino prints a error message when the file is not properly opened.

Parameters

<i>filename</i>	File to be opened
-----------------	-------------------

Returns

FILE* Pointer to the file

4.17.3.9 print_registry()

```
void print_registry (
    registry_data_st registry )
```

Prints a registry.

Parameters

<i>registry</i>	Registry to be printed
-----------------	------------------------

4.17.3.10 search_field_id()

```
int search_field_id (
    Field * f,
    int n )
```

Search for "id" in "Name" field at struct array "f" Procura se existe um campo de Name == id no array de structs "f".

Parameters

<i>f</i>	Struct array
<i>n</i>	Size of "f"

Returns

int Returns index of found field. If the "id" field does not exists, returns -1

4.18 utils.h

[Go to the documentation of this file.](#)

```
1
2
3
4
5 #ifndef __UTILS_H__
6 #define __UTILS_H__
7
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <string.h>
11 #include <ctype.h>
12 #include "../includes/myString.h"
13
14 #define ARR_SIZE(x) (sizeof(x) / sizeof((x)[0]))
15
16
17
18
19 typedef struct registry_data_s{
20     int id;
21     int year;
```

```

22     char city[50];
23     int amount;
24     char initials[5];
25     char brand[50];
26     char model[50];
27 }registry_data_st;
28
29 typedef struct WField{
30     char *name;
31     char *value;
32 }Field;
33
34 int compare_registry(registry_data_st registry, Field *f, int n);
35
36 void print_registry(registry_data_st registry);
37
38 FILE *open_file_rb(const char *filename);
39
40 FILE *open_file_rplusb(const char *filename);
41
42 FILE *open_file_wb(const char *filename);
43
44 void readline(char* string);
45 void binarioNaTela(const char *nomeArquivoBinario);
46 char *scan_quote_string();
47
48 int check_status(FILE*);
49
50 void logical_deletion(FILE*, int);
51
52 int search_field_id(Field* f, int n);
53
54 Field *get_field(int x);
55
56 void free_field(Field *f, int x);
57
58 registry_data_st read_registry_line();
59
60 #endif

```

4.19 main.c File Reference

File Organization final assignment.

```

#include "includes/file_t1.h"
#include "includes/file_t2.h"
#include "includes/myString.h"
#include "includes/database.h"
#include "includes/utils.h"
#include "includes/btree.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

Macros

- #define **CREATE_TABLE** 1
- #define **SELECT** 2
- #define **SELECT_WHERE** 3
- #define **SEARCH_RRN** 4
- #define **CREATE_INDEX** 5
- #define **DELETE_WHERE** 6
- #define **INSERT_INTO** 7
- #define **UPDATE_WHERE** 8
- #define **CREATE_INDEX_B** 9
- #define **SEARCH_WHERE_B** 10
- #define **INSERT_INTO_B** 11

Functions

- `int main ()`

4.19.1 Detailed Description

File Organization final assignment.

Author

Murillo Moraes Martins
Vítor Amorim Fróis

Version

4.0

Date

2022-07

Index

- add_registry_t1
 - file_t1.h, [23](#)
- add_registry_t2
 - file_t2.h, [28](#)
- binary_search_t1
 - index_t1.h, [33](#)
- binary_search_t2
 - index_t2.h, [37](#)
- bNode, [5](#)
- bTree, [5](#)
- btree.h
 - compareKeys, [12](#)
 - create_btree_t1, [12](#)
 - create_btree_t2, [13](#)
 - destroyTree, [13](#)
 - insertKey, [13](#)
 - read_tree_header, [14](#)
 - readNode, [14](#)
 - searchID, [15](#)
- calculate_registry_size
 - file_t2.h, [29](#)
- check_status
 - utils.h, [43](#)
- close_file_t1
 - file_t1.h, [24](#)
- close_file_t2
 - file_t2.h, [29](#)
- compare_registry
 - utils.h, [43](#)
- compareKeys
 - btree.h, [12](#)
- create_btree_t1
 - btree.h, [12](#)
- create_btree_t2
 - btree.h, [13](#)
- create_file_t1
 - file_t1.h, [24](#)
- create_file_t2
 - file_t2.h, [29](#)
- create_index
 - database.h, [18](#)
- create_index_b
 - database.h, [18](#)
- create_index_t1
 - index_t1.h, [33](#)
- create_index_t2
 - index_t2.h, [37](#)
- create_table
 - database.h, [18](#)
- csv.h
 - read_from_csv, [16](#)
- database.h
 - create_index, [18](#)
 - create_index_b, [18](#)
 - create_table, [18](#)
 - delete_from_where, [19](#)
 - insert_into, [19](#)
 - insert_into_b, [19](#)
 - search_where_b, [20](#)
 - select_from_RRN, [20](#)
 - select_from_where, [20](#)
 - update_where, [22](#)
- delete_from_where
 - database.h, [19](#)
- delete_where_t1
 - index_t1.h, [34](#)
- delete_where_t2
 - index_t2.h, [37](#)
- destroyTree
 - btree.h, [13](#)
- Field
 - utils.h, [43](#)
- file_t1.h
 - add_registry_t1, [23](#)
 - close_file_t1, [24](#)
 - create_file_t1, [24](#)
 - insert_into_t1, [24](#)
 - push_deleted_stack, [25](#)
 - read_file_t1, [25](#)
 - read_registry_t1, [25](#)
 - search_rrn, [25](#)
 - search_where_t1, [26](#)
 - update_registry_t1, [26](#)
 - update_RRN, [27](#)
- file_t2.h
 - add_registry_t2, [28](#)
 - calculate_registry_size, [29](#)
 - close_file_t2, [29](#)
 - create_file_t2, [29](#)
 - insert_into_t2, [29](#)
 - read_file_t2, [30](#)
 - read_registry_t2, [30](#)
 - search_byteOffset, [30](#)
 - search_where_t2, [31](#)
 - update_byte_offset, [31](#)
- free_field

- utils.h, 44
- get_field
 - utils.h, 44
- get_index_t1
 - index_t1.h, 34
- get_index_t2
 - index_t2.h, 38
- includes/btree.h, 11, 15
- includes/csv.h, 16, 17
- includes/database.h, 17, 22
- includes/file_t1.h, 22, 27
- includes/file_t2.h, 27, 32
- includes/index_t1.h, 32, 35
- includes/index_t2.h, 36, 39
- includes/myString.h, 40, 41
- includes/utils.h, 42, 46
- index_data_s1, 6
- index_data_s2, 6
- index_s1, 7
- index_s2, 7
- index_t1.h
 - binary_search_t1, 33
 - create_index_t1, 33
 - delete_where_t1, 34
 - get_index_t1, 34
 - update_where_t1, 34
 - write_index_file_t1, 35
- index_t2.h
 - binary_search_t2, 37
 - create_index_t2, 37
 - delete_where_t2, 37
 - get_index_t2, 38
 - update_registry_t2, 38
 - update_where_t2, 39
 - write_index_file_t2, 39
- insert_into
 - database.h, 19
- insert_into_b
 - database.h, 19
- insert_into_t1
 - file_t1.h, 24
- insert_into_t2
 - file_t2.h, 29
- insertKey
 - btree.h, 13
- key, 8
- logical_deletion
 - utils.h, 44
- main.c, 47
- myString.h
 - readString, 40
 - readString2, 40
 - stringEnd, 41
- open_file_rb
 - utils.h, 45
- open_file_rplusb
 - utils.h, 45
- open_file_wb
 - utils.h, 45
- print_registry
 - utils.h, 46
- push_deleted_stack
 - file_t1.h, 25
- read_file_t1
 - file_t1.h, 25
- read_file_t2
 - file_t2.h, 30
- read_from_csv
 - csv.h, 16
- read_registry_t1
 - file_t1.h, 25
- read_registry_t2
 - file_t2.h, 30
- read_tree_header
 - btree.h, 14
- readNode
 - btree.h, 14
- readString
 - myString.h, 40
- readString2
 - myString.h, 40
- registry_data_s, 8
- search_byteOffset
 - file_t2.h, 30
- search_field_id
 - utils.h, 46
- search_rrn
 - file_t1.h, 25
- search_where_b
 - database.h, 20
- search_where_t1
 - file_t1.h, 26
- search_where_t2
 - file_t2.h, 31
- searchID
 - btree.h, 15
- select_from_RRN
 - database.h, 20
- select_from_where
 - database.h, 20
- stringEnd
 - myString.h, 41
- update_byte_offset
 - file_t2.h, 31
- update_registry_t1
 - file_t1.h, 26
- update_registry_t2
 - index_t2.h, 38
- update_RRN

- file_t1.h, [27](#)
- update_where
 - database.h, [22](#)
- update_where_t1
 - index_t1.h, [34](#)
- update_where_t2
 - index_t2.h, [39](#)
- utils.h
 - check_status, [43](#)
 - compare_registry, [43](#)
 - Field, [43](#)
 - free_field, [44](#)
 - get_field, [44](#)
 - logical_deletion, [44](#)
 - open_file_rb, [45](#)
 - open_file_rplusb, [45](#)
 - open_file_wb, [45](#)
 - print_registry, [46](#)
 - search_field_id, [46](#)
- WField, [9](#)
- write_index_file_t1
 - index_t1.h, [35](#)
- write_index_file_t2
 - index_t2.h, [39](#)