File Organization Final Work

Generated by Doxygen 1.9.3

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

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

x	id 1
y	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

binFile	binFile
indexFile	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

binFile	binFile
indexFile	B-Tree index file

4.1.2.4 destroyTree()

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

Free all allocated memory for B-Tree.

Parameters

tree	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

indexFile	B-Tree index file
node	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

indexFile	B-Tree index file
type	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

indexFile	B-Tree index file
nodeRRN	Node RRN
type	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

node	B-Tree root
id	ID to be searched

Returns

Key Struct

4.2 btree.h

Go to the documentation of this file.

```

#ifndef BTREE_H
#define BTREE_H

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

#define INF 10e8

#define MAX_KEYS 3
#define MAX_DESC 4

//cada nó possui um tipo
#define ROOT '0'
#define MID '1'
#define LEAF '2'

typedef struct key{
    int id;
    char fileType;
    union{
        long int byteOffset;
        int rrn;
    } value;
}Key;

struct bNode{
    char nodeType;
    int numKeys;
    Key key[MAX_KEYS+1];
    int desc[MAX_DESC+1];
    int rrn;
};

struct bTree{
    char status;
    struct bNode *root;
    int rootRRN;
    int nextRRN;
    int numNodes;
};


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

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

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

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

struct bNode *insertKey(FILE *indexFile, struct bNode *node, Key);

void destroyTree(struct bTree *tree);

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

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

#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

file	File that will be read
data	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.

```

#ifndef __CSV_H__
#define __CSV_H__
#include <stdio.h>
#include "utils.h"

int read_from_csv(FILE *file, registry_data_st *data);

#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

type	Type of file. 1 for fixed size and 2 for variable size
binFile	File with registries
indexFile	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

binFile	binFile
indexFile	indexFile
type	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

type	Type of file. 1 for fixed size and 2 for variable size
CSVFilename	Name of CSV file
binFile	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

binFile	binFile
indexFilename	indexFile
type	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

binFile	binFile
indexFilename	indexFilename
type	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

binFile	binFile
indexFile	indexFile
type	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

binFile	binFile
indexFile	indexFile
type	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

binFile	

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

binFile	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

binFile	binFile
indexFilename	indexFilename
type	Type of file. 1 for fixed size and 2 for variable size

4.6 database.h

Go to the documentation of this file.

```

#ifndef __DATABASE_H__
#define __DATABASE_H__

#include <stdio.h>

void create_table(char type, const char *CSVFileName, const char *binFileName);

void select_from(FILE *binFile, char type);

void select_from_where(FILE *binFile, char type);

void select_from_RRN(FILE *binFile);

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

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

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

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

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

void insert_into_b(FILE *binFile, FILE *indexFile, char type);

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

#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

file	File of type 1
data	Registry data

4.7.2.2 close_file_t1()

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

Adequately closes a type 1 file.

Parameters

file	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

filename	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

binFile	Type 1 file
data	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

binFile	Stack file
rrn	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

file	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

file	File that contains the registry
registrySize	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

file	File to be searched
rrn	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

file	File with registries
f	Array of structs to check
n	Size of array f

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

newReg	Registry to be updated
update	Array of new values
n	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

file	File to be updated
byteOffset	New RRN

4.8 file_t1.h

Go to the documentation of this file.

```
#ifndef __FILE_T1_H__
#define __FILE_T1_H__
#include <stdio.h>
#include "csv.h"
#include "index_t1.h"

FILE *create_file_t1(const char *filename);

void add_registry_t1(FILE *file, registry_data_st data);

registry_data_st read_registry_t1(FILE *file);

void read_file_t1(FILE *file);

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

void close_file_t1(FILE *file);

void search_rrn(FILE *file, int rrn);

void update_RRN(FILE *file, int finalRRN);

void push_deleted_stack(FILE *binFile, int rrn);

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

int insert_into_t1(FILE *binFile, registry_data_st data);

#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

file	File of type 2
data	Registry data

4.9.2.2 calculate_registry_size()

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

Calculate size of registry.

Parameters

data	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

file	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

filename	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

binFile	Type 2 file
data	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

file	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

file	File that contains the registry
registrySize	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

file	File to be searched
byteOffset	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

file	File with registries
f	Array of structs to check
n	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

file	File to be updated
byteOffset	New byte offset

4.10 file_t2.h

Go to the documentation of this file.

```
#ifndef __FILE_T2_H__
#define __FILE_T2_H__

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

typedef long long int lli;

FILE *create_file_t2(const char *filename);

int calculate_registry_size(registry_data_st data);

void add_registry_t2(FILE *file, registry_data_st data, int additionalSize);

void update_byte_offset(FILE *file, lli byteOffset);

registry_data_st read_registry_t2(FILE* file);

void read_file_t2(FILE *file);

void close_file_t2(FILE *file);

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

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

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

#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

index	Struct to be searched
id	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

binFile	binFile
indexFile	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

file	binFile
f	Array of structs to check
n	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

indexFile	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

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

indexFile	index file
index	Array of index data

4.12 index_t1.h

Go to the documentation of this file.

```
#ifndef __INDEX_T1_H__
#define __INDEX_T1_H__

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

#define DATA_HEADER_SIZE 182

typedef struct index_data_s1{
    int id;
    int rrn;
}index_data_st1;

typedef struct index_s1{
    index_data_st1 *data;
    int size;
}index_st1;

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

void write_index_file_t1(FILE *indexFile, index_st1 index);

```

```
index_st1 get_index_t1(FILE *indexFile);

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

void update_where_t1(FILE *binFile, index_st1 index, Field *search, int x, Field *update, int y);

int binary_search_t1(index_st1 index, int id);

#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

index	Struct to be searched
id	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

binFile	binFile
indexFile	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

file	binFile
f	Array of structs to check
n	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

indexFile	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

newReg	Registry to be updates
update	Field of new values
n	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

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

indexFile	index file
index	Array of index data

4.14 index_t2.h

Go to the documentation of this file.

```
#ifndef __INDEX_T2_H__
#define __INDEX_T2_H__

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

typedef long long int lli;

typedef struct index_data_s2{
    int id;
    long long int byteOffset;
}index_data_st2;

typedef struct index_s2{
    index_data_st2 *data;
    int size;
}index_st2;

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

void delete_where_t2(FILE *binFile, index_st2 index, Field *search, int n);

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

int binary_search_t2(index_st2 index, int id);

index_st2 get_index_t2(FILE *indexFile);

void write_index_file_t2(FILE *indexFile, index_st2 index);

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

num	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

numSeparators	número de separadores
separatorArray	array com todos finalizadores
c	caractere lido

Returns

int 1 para leitura finalizada, caso contrário, 0

4.16 myString.h

Go to the documentation of this file.

```c

#ifndef __MY_STRING_H__
#define __MY_STRING_H__
#include <stdio.h>

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

char *readFixedSizeString(FILE *fp, int stringSize);

char* readString();


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

#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

registry	Struct that stores registry data
f	Array of searching values
n	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

f	Array of Field
x	Size of array

4.17.3.4 get_field()

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

Read a array of Field from stdin.

Parameters

x	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

FILE∗	Binary file with the registry to be deleted
int	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

filename	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

filename	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

filename	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

registry	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

f	Struct array
n	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.

```
#ifndef __UTILS_H__
#define __UTILS_H__

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

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

typedef struct registry_data_s{
    int id;
    int year;
    char city[50];
    int amount;
    char initials[5];
    char brand[50];
    char model[50];
}registry_data_st;

typedef struct WField{
    char *name;
    char *value;
}Field;

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

void print_registry(registry_data_st registry);

FILE *open_file_rb(const char *filename);

FILE *open_file_rplusb(const char *filename);

FILE *open_file_wb(const char *filename);

void readline(char* string);
void binarioNaTela(const char *nomeArquivoBinario);
char *scan_quote_string();

int check_status(FILE*);

void logical_deletion(FILE*, int);

int search_field_id(Field* f, int n);

Field *get_field(int x);

void free_field(Field *f, int x);

registry_data_st read_registry_line();

#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