LFU cache

Generated by Doxygen 1.8.17

Chapter 1

Lfu Cache

Realisation of LFU cache algorithm (`read more about algorithm`)

- LFU - directory wirh main header **LFU.h** (p. 16) + main lib **LFU.c** (p. 13)
- Hash_Map - directory with lib of hash-table (**Hash_Map.c** (p. 13))
- List - directory with lib of frequency list (**List_Map.c** (p. 28))
- Test - directory with some tests
- main - directory with **new_main.c** (p. 32) file which is used for "Problem LC" contest

1.1 Some statistic

1.1.1 Filling cache with 10^6 pages

1.1.2 Filling cache with 10^7 pages

1.2 Graphic of relatiomship between count of pages with data and time of their adding

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 freq_node Struct Reference

Node which contains frequency.

`#include <LFU.h>`

Collaboration diagram for freq_node:

Public Attributes

- int **freq_t**
- struct **freq_node** * **next**
- struct **freq_node** * **prev**
- struct **lfu_node** * **child**

4.1.1 Detailed Description

Node which contains frequency.

4.1.2 Member Data Documentation

4.1.2.1 child

`struct lfu_node* freq_node::child`

4.1.2.2 freq_t

```
int freq_node::freq_t
```

4.1.2.3 next

```
struct  freq_node* freq_node::next
```

4.1.2.4 prev

```
struct  freq_node* freq_node::prev
```

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

- /home/kir/and another one/lfu_cache/LFU/ **LFU.h**

4.2 hash_cell Struct Reference

The cell of hash table.

```
#include <LFU.h>
```

Collaboration diagram for hash_cell:

Public Attributes

- struct **hash_cell** * **next**
- struct **hash_cell** * **prev**
- struct **lfu_node** * **item**

4.2.1 Detailed Description

The cell of hash table.

4.2.2 Member Data Documentation

4.2.2.1 item

```
struct lfu_node* hash_cell::item
```

4.2.2.2 next

```
struct hash_cell* hash_cell::next
```

4.2.2.3 prev

```
struct hash_cell* hash_cell::prev
```

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

- /home/kir/and another one/lfu_cache/LFU/ **LFU.h**

4.3 hash_map Struct Reference

The struct of Hash Map.

```
#include <LFU.h>
```

Collaboration diagram for hash_map:

Public Attributes

- struct **hash_cell** ** **cells**
- int **size**

4.3.1 Detailed Description

The struct of Hash Map.

4.3.2 Member Data Documentation

4.3.2.1 cells

```
struct hash_cell** hash_map::cells
```

4.3.2.2 size

```
int hash_map::size
```

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

- /home/kir/and another one/lfu_cache/LFU/ **LFU.h**

4.4 lfu_c Struct Reference

```
#include <LFU.h>
```

Collaboration diagram for lfu_c:

Public Attributes

- struct **hash_map** * **HashTable**
- struct **freq_node** * **List**
- int **cache_size**
- int **cache_fullnes**

4.4.1 Member Data Documentation

4.4.1.1 cache_fullnes

```
int lfu_c::cache_fullnes
```

4.4.1.2 cache_size

```
int lfu_c::cache_size
```

4.4.1.3 HashTable

```
struct  hash_map* lfu_c::HashTable
```

4.4.1.4 List

```
struct freq_node* lfu_c::List
```

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

- /home/kir/and another one/lfu_cache/LFU/ **LFU.h**

4.5 lfu_node Struct Reference

Node which contains pages with data.

```
#include <LFU.h>
```

Collaboration diagram for lfu_node:

Public Attributes

- struct **request_t data_t**
- struct **lfu_node** * **next**
- struct **lfu_node** * **prev**
- struct **freq_node** * **parent**

4.5.1 Detailed Description

Node which contains pages with data.

4.5.2 Member Data Documentation

4.5.2.1 data_t

```
struct request_t lfu_node::data_t
```

4.5.2.2 next

```
struct lfu_node* lfu_node::next
```

4.5.2.3 parent

```
struct freq_node* lfu_node::parent
```

4.5.2.4 prev

```
struct lfu_node* lfu_node::prev
```

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

- /home/kir/and another one/lfu_cache/LFU/ **LFU.h**

4.6 request_t Struct Reference

```
#include <LFU.h>
```

Public Attributes

- int **data**

4.6.1 Member Data Documentation

4.6.1.1 data

```
int request_t::data
```

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

- /home/kir/and another one/lfu_cache/LFU/ **LFU.h**

Chapter 5

File Documentation

5.1 /home/kir/and another one/lfu_cache/Hash_Map/Hash_Map.c File Reference

`#include "../LFU/LFU.h"`
Include dependency graph for Hash_Map.c:

5.2 /home/kir/and another one/lfu_cache/LFU/LFU.c File Reference

`#include "LFU.h"`
Include dependency graph for LFU.c:

Functions

- **DATA GetPage** (FILE *f)
  Function which get stream-variable and read page with data from it.
- void **PrintPage** (**DATA** *page, char *source)
  Function which print page with data to file with name "source".
- **LFU** * **LfuConstruct** (int cache_size)
  Constructor of LFU - cache.
- int **InsertLFU** (**LFU** *cache, **DATA** *request)
  Func which insert new page to cache.
- void **FreeLFU** (**LFU** *cache)
  Destructor of LFU-cache.
- void **LFUDump** (**LFU** *cache, char *source)
  Func which print LFU-cache to file with name "source".

5.2.1 Function Documentation

5.2.1.1 FreeLFU()

```
void FreeLFU (
              LFU * cache )
```

Destructor of LFU-cache.

Parameters

in	*cache*	Pointer to LFU-cache

5.2.1.2 GetPage()

```
DATA GetPage (
            FILE * f )
```

Function which get stream-variable and read page with data from it.

Parameters

in	*f*	Filestream which you want to take a page from

Returns

Page of data (type DATA)

5.2.1.3 InsertLFU()

```
int InsertLFU (
             LFU * cache,
             DATA * request )
```

Func which insert new page to cache.

Parameters

in	*cache*	Pointer to cache where we want to insert
in	*request*	Page of data which we want to insert

Returns

1, if there is this page in cache< 0 if it isn't

5.2.1.4 LfuConstruct()

```
LFU* LfuConstruct (
            int cache_size )
```

Constructor of LFU - cache.

Parameters

in	*cache_size*	Size of cache

Returns

Pointer to initialized cache

5.2.1.5 LFUDump()

```
void LFUDump (
              LFU * cache,
             char * source )
```

Func which print LFU-cache to file with name "source".

Parameters

in	*cache*	LFU-cache which we want to print
in	*source*	Name of file where we want to print

Note

If source is "stdout" data will be printed to console

5.2.1.6 PrintPage()

```
void PrintPage (
              DATA * page,
             char * source )
```

Function which print page with data to file with name "source".

Parameters

in	*page*	Pointer to page with data
in	*source*	String with name of file

Note

If source is "stdout", data will be printed to console

5.3 /home/kir/and another one/lfu_cache/LFU/LFU.h File Reference

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

Include dependency graph for LFU.h: This graph shows which files directly or indirectly include this file:

Classes

- struct **request_t**
- struct **lfu_node**

  Node which contains pages with data.
- struct **freq_node**

  Node which contains frequency.
- struct **hash_cell**

  The cell of hash table.
- struct **hash_map**

  The struct of Hash Map.
- struct **lfu_c**

Macros

- #define **NUM** 100
- #define **CGetPage**() **GetPage**(stdin);

  Macros which call GetPage func with f = stdin.
- #define **CPrintPage**(page) **PrintPage**(page, "stdout")

  Macros which call PrintPage func with source = "stdout".

Typedefs

- typedef struct **request_t** **DATA**
- typedef struct **lfu_c** **LFU**

  Struct of LFU - cache.

Functions

- struct **freq_node** * **CreateFreq** (int freq_dat, struct **freq_node** *prev_fr)

  Creates an element in the place the user wants. Function need data(frequency) and previous item (place where we should create)
- struct **lfu_node** * **CreateLfu** (**DATA** lfu_dat, struct **freq_node** *head)

  *Creates last **lfu_node*** (p. *11) at frequency 1. If frequency 1 is not exist, it will be created (with function create_freq).*
- void **RemoveFreq** (struct **freq_node** *del)

  Delete frequency node.
- void **RemoveLfu** (struct **freq_node** *head)

  Delete first lfu node with frequency 1.
- void **ReplaceLfu** (struct **lfu_node** *cur_lfu)

  When we already have the resulting item in the list, we need to replaceit. If the element with current frequency + 1 doesn`t exist it will be created (with function create_freq).
- struct **freq_node** * **CreateHead** ()

  Initialization of head of list.
- void **DeleteList** (struct **freq_node** *head)

  Function which free's list.
- void **ListPrint** (struct **freq_node** *head, FILE *f)

  Function which print List to filestream.
- void **TestCreateHead** ()
- void **TestCreateFreq** ()
- void **TestCreateLfu** ()
- void **TestRemoveFreq** ()
- void **TestRemoveLfu** ()
- void **TestReplaceLfu** ()
- struct **hash_map** * **InitHashMap** (int cache_size)

  This function is constructor of hash table; it initializes Hash Map, calculates the size of hash using the formula {Size = cache_size / 10 + 1}, initialzes array of pointers to cells and cells.
- struct **hash_cell** * **SearchData** (struct **hash_cell** *cell, **DATA** *request)

  This function searches data in list of collisions.
- struct **hash_cell** * **InsertHashMap** (struct **hash_map** *Hash_Map, **DATA** *request)

  This function finds an avaliable place to adding new data If this data already exists here, it does nothing.
- struct **hash_cell** * **SearchMap** (struct **hash_map** *Hash_Map, **DATA** *request)

  This function searches data in Hash Map.
- int **HashofData** (**DATA** *request, int hash_size)

  This function transforms struct to string for calculating hash later.
- int **HashofChar** (char *string, int len, int hash_size)

  This function calculates hash of string.
- int **HashofInt** (int number, int hash_size)

  This function calculates hash of integer number.
- int **DelElem** (struct **hash_map** *Hash_Map, **DATA** *request)

  This function deletes cell with node with data from Hash Map and do nothing if data isn't here.
- int **FreeHashMap** (struct **hash_map** *Hash_Map)

  This function clear memory allocated for Hash_Map, array of pointers to cells and cells.
- int **PrintHashMap** (struct **hash_map** *Hash_Map, FILE *f)

  This function print Hash Map.
- **DATA** **GetPage** (FILE *f)

  Function which get stream-variable and read page with data from it.
- void **PrintPage** (**DATA** *page, char *source)

  Function which print page with data to file with name "source".
- **LFU** * **LfuConstruct** (int cache_size)

Constructor of LFU - cache.

- int **InsertLFU** (**LFU** ∗cache, **DATA** ∗request)

  Func which insert new page to cache.

- void **FreeLFU** (**LFU** ∗cache)

  Destructor of LFU-cache.

- void **LFUDump** (**LFU** ∗cache, char ∗source)

  Func which print LFU-cache to file with name "source".

5.3.1 Macro Definition Documentation

5.3.1.1 CGetPage

```
#define CGetPage( )  GetPage(stdin);
```

Macros which call GetPage func with f = stdin.

5.3.1.2 CPrintPage

```
#define CPrintPage(
              page )  PrintPage(page, "stdout")
```

Macros which call PrintPage func with source = "stdout".

5.3.1.3 NUM

```
#define NUM 100
```

5.3.2 Typedef Documentation

5.3.2.1 DATA

```
typedef struct  request_t  DATA
```

5.3.2.2 LFU

```
typedef struct  lfu_c  LFU
```

Struct of LFU - cache.

5.3.3 Function Documentation

5.3.3.1 CreateFreq()

```
struct  freq_node* CreateFreq (
            int freq_dat,
            struct  freq_node * prev_fr )
```

Creates an element in the place the user wants. Function need data(frequency) and previous item (place where we should create)

Parameters

in	*freq_dat*	Frequency node
in	*prev_fr*	Previous item (lfu node)

Returns

Pointer to element which was created

5.3.3.2 CreateHead()

```
struct  freq_node* CreateHead ( )
```

Initialization of head of list.

5.3.3.3 CreateLfu()

```
struct  lfu_node* CreateLfu (
             DATA lfu_dat,
            struct  freq_node * head )
```

Creates last **lfu_node** (p. 11) at frequency 1. If frequency 1 is not exist, it will be created (with function create_freq).

Parameters

in	*lfu_dat*	Page with data which well be added to **lfu_node** (p. 11)
in	*head*	Head of list

Returns

Pointer to node which was created

5.3.3.4 DelElem()

```
int DelElem (
            struct  hash_map * Hash_Map,
             DATA * request )
```

This function deletes cell with node with data from Hash Map and do nothing if data isn't here.

Parameters

in	*Hash_map*	Pointer to Hash Map
in	*request*	Request

Returns

Integer zero

5.3.3.5 DeleteList()

```
void DeleteList (
            struct  freq_node * head )
```

Function which free's list.

5.3.3.6 FreeHashMap()

```
int FreeHashMap (
            struct  hash_map * Hash_Map )
```

This function clear memory allocated for Hash_Map, array of pointers to cells and cells.

Parameters

in	*Hash_map*	Pointer to Hash Map

Returns

Integer zero

5.3.3.7 FreeLFU()

```
void FreeLFU (
              LFU * cache )
```

Destructor of LFU-cache.

Parameters

in	*cache*	Pointer to LFU-cache

5.3.3.8 GetPage()

```
DATA GetPage (
             FILE * f )
```

Function which get stream-variable and read page with data from it.

Parameters

in	*f*	Filestream which you want to take a page from

Returns

Page of data (type DATA)

5.3.3.9 HashofChar()

```
int HashofChar (
             char * string,
             int len,
             int hash_size )
```

This function calculates hash of string.

Parameters

in	*string*	String
in	*len*	The length of string
in	*hash_size*	The size of Hash

Returns

Hash of string

5.3.3.10 HashofData()

```
int HashofData (
              DATA * request,
             int hash_size )
```

This function transforms struct to string for calculating hash later.

Parameters

in	*request*	pointer to request
in	*hash_size*	The size of Hash

Returns

Key of Hash Table

5.3.3.11 HashofInt()

```
int HashofInt (
             int number,
             int hash_size )
```

This function calculates hash of integer number.

Parameters

in	*number*	Integer number
in	*hash_size*	The size of Hash

Returns

Hash of integer number

5.3.3.12 InitHashMap()

```
struct  hash_map* InitHashMap (
            int cache_size )
```

This function is constructor of hash table; it initializes Hash Map, calculates the size of hash using the formula {Size = cache_size / 10 + 1}, initialzes array of pointers to cells and cells.

Parameters

in	*cache_size*	The size of cache

Returns

Pointer to allocated memory to Hash Map

5.3.3.13 InsertHashMap()

```
struct  hash_cell* InsertHashMap (
            struct  hash_map * Hash_Map,
             DATA * request )
```

This function finds an avaliable place to adding new data If this data already exists here, it does nothing.

Parameters

in	*Hash_Map*	pointer to Hash Map
in	*request*	pointer to request

Returns

The pointer on an avaliable cell in Hash Map

5.3.3.14 InsertLFU()

```
int InsertLFU (
             LFU * cache,
             DATA * request )
```

Func which insert new page to cache.

Parameters

in	*cache*	Pointer to cache where we want to insert
in	*request*	Page of data which we want to insert

Returns

1, if there is this page in cache< 0 if it isn't

5.3.3.15 LfuConstruct()

```
LFU* LfuConstruct (
            int cache_size )
```

Constructor of LFU - cache.

Parameters

in	*cache_size*	Size of cache

Returns

Pointer to initialized cache

5.3.3.16 LFUDump()

```
void LFUDump (
             LFU * cache,
            char * source )
```

Func which print LFU-cache to file with name "source".

Parameters

in	*cache*	LFU-cache which we want to print
in	*source*	Name of file where we want to print

Note

If source is "stdout" data will be printed to console

5.3.3.17 ListPrint()

```
void ListPrint (
            struct  freq_node * head,
            FILE * f )
```

Function which print List to filestream.

Parameters

in	*head*	Pointer to head of list
in	*f*	Filestream pointer where we want to print list

5.3.3.18 PrintHashMap()

```
int PrintHashMap (
            struct  hash_map * Hash_Map,
            FILE * f )
```

This function print Hash Map.

Parameters

in	*Hash_map*	Pointer to Hash Map
in	*f*	Pointer to file stream

Returns

Integer zero

5.3.3.19 PrintPage()

```
void PrintPage (
             DATA * page,
            char * source )
```

Function which print page with data to file with name "source".

Parameters

in	*page*	Pointer to page with data
in	*source*	String with name of file

Note

If source is "stdout", data will be printed to console

5.3.3.20 RemoveFreq()

```
void RemoveFreq (
            struct  freq_node * del )
```

Delete frequency node.

Parameters

in	*del*	Node which we want to delete

5.3.3.21 RemoveLfu()

```
void RemoveLfu (
            struct  freq_node * head )
```

Delete first lfu node with frequency 1.

Parameters

in	*head*	Pointer to List

5.3.3.22 ReplaceLfu()

```
void ReplaceLfu (
            struct  lfu_node * cur_lfu )
```

When we already have the resulting item in the list, we need to replaceit. If the element with current frequency + 1 doesn`t exist it will be created (with function create_freq).

Parameters

in	*cur_lfu*	Lfu node which we want to replace

5.3.3.23 SearchData()

```
struct  hash_cell* SearchData (
            struct  hash_cell * cell,
             DATA * request )
```

This function searches data in list of collisions.

Parameters

in	*cell*	Element of array of pointers to cells in Hash Map
in	*request*	Request

Returns

Pointer to cell and NULL pointer if data hasn't been found

5.3.3.24 SearchMap()

```
struct  hash_cell* SearchMap (
             struct  hash_map * Hash_Map,
              DATA * request )
```

This function searches data in Hash Map.

Parameters

in	*Hash_Map*	pointer to Hash Map
in	*request*	pointer to request

Returns

Pointer to cell and NULL pointer if data hasn't been found

5.3.3.25 TestCreateFreq()

```
void TestCreateFreq ( )
```

5.3.3.26 TestCreateHead()

```
void TestCreateHead ( )
```

5.3.3.27 TestCreateLfu()

```
void TestCreateLfu ( )
```

5.3.3.28 TestRemoveFreq()

```
void TestRemoveFreq ( )
```

5.3.3.29 TestRemoveLfu()

```
void TestRemoveLfu ( )
```

5.3.3.30 TestReplaceLfu()

```
void TestReplaceLfu ( )
```

5.4 /home/kir/and another one/lfu_cache/List/List_Map.c File Reference

```
#include "../LFU/LFU.h"
```
Include dependency graph for List_Map.c:

Functions

- struct **freq_node** * **CreateFreq** (int freq_dat, struct **freq_node** *prev_fr)

  Creates an element in the place the user wants. Function need data(frequency) and previous item (place where we should create)
- struct **lfu_node** * **CreateLfu** (**DATA** lfu_dat, struct **freq_node** *head)

  *Creates last **lfu_node** (p. 11) at frequency 1. If frequency 1 is not exist, it will be created (with function create_freq).*
- void **RemoveFreq** (struct **freq_node** *del)

  Delete frequency node.
- void **RemoveLfu** (struct **freq_node** *head)

  Delete first lfu node with frequency 1.
- void **ReplaceLfu** (struct **lfu_node** *cur_lfu)

  When we already have the resulting item in the list, we need to replaceit. If the element with current frequency + 1 doesn`t exist it will be created (with function create_freq).
- struct **freq_node** * **CreateHead** ()

  Initialization of head of list.
- void **DeleteList** (struct **freq_node** *head)

  Function which free's list.
- void **ListPrint** (struct **freq_node** *head, FILE *f)

  Function which print List to filestream.
- void **TestCreateHead** ()
- void **TestCreateFreq** ()
- void **TestCreateLfu** ()
- void **TestRemoveFreq** ()
- void **TestRemoveLfu** ()
- void **TestReplaceLfu** ()

5.4.1 Function Documentation

5.4.1.1 CreateFreq()

```
struct  freq_node* CreateFreq (
            int freq_dat,
            struct  freq_node * prev_fr )
```

Creates an element in the place the user wants. Function need data(frequency) and previous item (place where we should create)

Parameters

in	*freq_dat*	Frequency node
in	*prev_fr*	Previous item (lfu node)

Returns

Pointer to element which was created

5.4.1.2 CreateHead()

```
struct  freq_node* CreateHead ( )
```

Initialization of head of list.

5.4.1.3 CreateLfu()

```
struct  lfu_node* CreateLfu (
              DATA lfu_dat,
             struct  freq_node * head )
```

Creates last **lfu_node** (p. 11) at frequency 1. If frequency 1 is not exist, it will be created (with function create_freq).

Parameters

in	*lfu_dat*	Page with data which well be added to **lfu_node** (p. 11)
in	*head*	Head of list

Returns

Pointer to node which was created

5.4.1.4 DeleteList()

```
void DeleteList (
             struct  freq_node * head )
```

Function which free's list.

5.4.1.5 ListPrint()

```
void ListPrint (
            struct  freq_node * head,
            FILE * f )
```

Function which print List to filestream.

Parameters

in	*head*	Pointer to head of list
in	*f*	Filestream pointer where we want to print list

5.4.1.6 RemoveFreq()

```
void RemoveFreq (
            struct  freq_node * del )
```

Delete frequency node.

Parameters

in	*del*	Node which we want to delete

5.4.1.7 RemoveLfu()

```
void RemoveLfu (
            struct  freq_node * head )
```

Delete first lfu node with frequency 1.

Parameters

in	*head*	Pointer to List

5.4.1.8 ReplaceLfu()

```
void ReplaceLfu (
            struct  lfu_node * cur_lfu )
```

When we already have the resulting item in the list, we need to replaceit. If the element with current frequency + 1 doesn`t exist it will be created (with function create_freq).

Parameters

in	*cur_lfu*	Lfu node which we want to replace

5.4.1.9 TestCreateFreq()

```
void TestCreateFreq ( )
```

5.4.1.10 TestCreateHead()

```
void TestCreateHead ( )
```

5.4.1.11 TestCreateLfu()

```
void TestCreateLfu ( )
```

5.4.1.12 TestRemoveFreq()

```
void TestRemoveFreq ( )
```

5.4.1.13 TestRemoveLfu()

```
void TestRemoveLfu ( )
```

5.4.1.14 TestReplaceLfu()

```
void TestReplaceLfu ( )
```

5.5 /home/kir/and another one/lfu_cache/main/main.c File Reference

```
#include "../LFU/LFU.h"
```

Include dependency graph for main.c:

Functions

- int **main** ()

5.5.1 Function Documentation

5.5.1.1 main()

```
int main ( )
```

5.6 /home/kir/and another one/lfu_cache/main/new_main.c File Reference

```
#include "../LFU/LFU.h"
```
Include dependency graph for new_main.c:

Functions

- int **main** (int argc, char ∗argv[ ])

5.6.1 Function Documentation

5.6.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

5.7 /home/kir/and another one/lfu_cache/main/test.txt File Reference

5.8 /home/kir/and another one/lfu_cache/README.md File Reference

5.9 /home/kir/and another one/lfu_cache/Test/HC.txt File Reference

Variables

- **Hello**
- World World **!Hello**

5.9.1 Variable Documentation

5.9.1.1 "!Hello

```
World World !  Hello
```

5.9.1.2 Hello

```
World Hello
```

5.10 /home/kir/and another one/lfu_cache/Test/Hl.txt File Reference

5.11 /home/kir/and another one/lfu_cache/Test/INIT.txt File Reference

5.12 /home/kir/and another one/lfu_cache/Test/SF.txt File Reference

5.13 /home/kir/and another one/lfu_cache/Test/test.c File Reference

```
#include "Test_LFU.h"
```
Include dependency graph for test.c:

Functions

- int **main** ()

5.13.1 Function Documentation

5.13.1.1 main()

```
int main ( )
```

5.14 /home/kir/and another one/lfu_cache/Test/Test_Hash_Map.c File Reference

```
#include "../LFU/LFU.h"
#include <time.h>
```

Include dependency graph for Test_Hash_Map.c:

Functions

- int **Hash_Int_Test** ()
- int **Hash_Char_Test** ()
- int **Init_Func_Test** ()
- int **Test_SearchMap** ()
- int **main** ()

5.14.1 Function Documentation

5.14.1.1 Hash_Char_Test()

```
int Hash_Char_Test ( )
```

5.14.1.2 Hash_Int_Test()

```
int Hash_Int_Test ( )
```

5.14.1.3 Init_Func_Test()

```
int Init_Func_Test ( )
```

5.14.1.4 main()

```
int main ( )
```

5.14.1.5 Test_SearchMap()

```
int Test_SearchMap ( )
```

5.15 /home/kir/and another one/lfu_cache/Test/Test_LFU.c File Reference

```
#include "../LFU/LFU.h"
#include <time.h>
```

Include dependency graph for Test_LFU.c:

Functions

- void **TestPageFunc** (char *file)
- void **TestLFUFunc** (char *file)

5.15.1 Function Documentation

5.15.1.1 TestLFUFunc()

```
void TestLFUFunc (
            char * file )
```

5.15.1.2 TestPageFunc()

```
void TestPageFunc (
            char * file )
```

5.16 /home/kir/and another one/lfu_cache/Test/Test_LFU.h File Reference

```
#include "../LFU/LFU.h"
```

Include dependency graph for Test_LFU.h: This graph shows which files directly or indirectly include this file:

Functions

- void **TestPageFunc** (char *file)
- void **TestLFUFunc** (char *file)

5.16.1 Function Documentation

5.16.1.1 TestLFUFunc()

```
void TestLFUFunc (
            char * file )
```

5.16.1.2 TestPageFunc()

```
void TestPageFunc (
            char * file )
```

Index