

LFU cache

Generated by Doxygen 1.8.17

| | |
|---|----------|
| 1 Lfu Cache | 1 |
| 1.1 Some statistic | 1 |
| 1.1.1 Filling cache with 10^6 pages | 1 |
| 1.1.2 Filling cache with 10^7 pages | 1 |
| 1.2 Graphic of relationship between count of pages with data and time of their adding | 1 |
| 2 Class Index | 3 |
| 2.1 Class List | 3 |
| 3 File Index | 5 |
| 3.1 File List | 5 |
| 4 Class Documentation | 7 |
| 4.1 freq_node Struct Reference | 7 |
| 4.1.1 Detailed Description | 7 |
| 4.1.2 Member Data Documentation | 7 |
| 4.1.2.1 child | 7 |
| 4.1.2.2 freq_t | 8 |
| 4.1.2.3 next | 8 |
| 4.1.2.4 prev | 8 |
| 4.2 hash_cell Struct Reference | 8 |
| 4.2.1 Detailed Description | 8 |
| 4.2.2 Member Data Documentation | 8 |
| 4.2.2.1 item | 9 |
| 4.2.2.2 next | 9 |
| 4.2.2.3 prev | 9 |
| 4.3 hash_map Struct Reference | 9 |
| 4.3.1 Detailed Description | 9 |
| 4.3.2 Member Data Documentation | 9 |
| 4.3.2.1 cells | 9 |
| 4.3.2.2 size | 10 |
| 4.4 lfu_c Struct Reference | 10 |
| 4.4.1 Member Data Documentation | 10 |
| 4.4.1.1 cache_fullnes | 10 |
| 4.4.1.2 cache_size | 10 |
| 4.4.1.3 HashTable | 10 |
| 4.4.1.4 List | 11 |
| 4.5 lfu_node Struct Reference | 11 |
| 4.5.1 Detailed Description | 11 |
| 4.5.2 Member Data Documentation | 11 |
| 4.5.2.1 data_t | 11 |
| 4.5.2.2 next | 11 |
| 4.5.2.3 parent | 12 |

| | |
|--|-----------|
| 4.5.2.4 prev | 12 |
| 4.6 request_t Struct Reference | 12 |
| 4.6.1 Member Data Documentation | 12 |
| 4.6.1.1 data | 12 |
| 5 File Documentation | 13 |
| 5.1 /home/kir/and another one/lfu_cache/Hash_Map/Hash_Map.c File Reference | 13 |
| 5.2 /home/kir/and another one/lfu_cache/LFU/LFU.c File Reference | 13 |
| 5.2.1 Function Documentation | 13 |
| 5.2.1.1 FreeLFU() | 13 |
| 5.2.1.2 GetPage() | 14 |
| 5.2.1.3 InsertLFU() | 14 |
| 5.2.1.4 LfuConstruct() | 14 |
| 5.2.1.5 LFUDump() | 15 |
| 5.2.1.6 PrintPage() | 15 |
| 5.3 /home/kir/and another one/lfu_cache/LFU/LFU.h File Reference | 16 |
| 5.3.1 Macro Definition Documentation | 18 |
| 5.3.1.1 CGetPage | 18 |
| 5.3.1.2 CPrintPage | 18 |
| 5.3.1.3 NUM | 18 |
| 5.3.2 Typedef Documentation | 18 |
| 5.3.2.1 DATA | 18 |
| 5.3.2.2 LFU | 19 |
| 5.3.3 Function Documentation | 19 |
| 5.3.3.1 CreateFreq() | 19 |
| 5.3.3.2 CreateHead() | 19 |
| 5.3.3.3 CreateLfu() | 19 |
| 5.3.3.4 DelElem() | 20 |
| 5.3.3.5 DeleteList() | 20 |
| 5.3.3.6 FreeHashMap() | 20 |
| 5.3.3.7 FreeLFU() | 21 |
| 5.3.3.8 GetPage() | 21 |
| 5.3.3.9 HashofChar() | 21 |
| 5.3.3.10 HashofData() | 22 |
| 5.3.3.11 HashofInt() | 22 |
| 5.3.3.12 InitHashMap() | 23 |
| 5.3.3.13 InsertHashMap() | 23 |
| 5.3.3.14 InsertLFU() | 23 |
| 5.3.3.15 LfuConstruct() | 24 |
| 5.3.3.16 LFUDump() | 24 |
| 5.3.3.17 ListPrint() | 24 |
| 5.3.3.18 PrintHashMap() | 25 |

| | |
|--|----|
| 5.3.3.19 PrintPage() | 25 |
| 5.3.3.20 RemoveFreq() | 25 |
| 5.3.3.21 RemoveLfu() | 26 |
| 5.3.3.22 ReplaceLfu() | 26 |
| 5.3.3.23 SearchData() | 26 |
| 5.3.3.24 SearchMap() | 27 |
| 5.3.3.25 TestCreateFreq() | 27 |
| 5.3.3.26 TestCreateHead() | 27 |
| 5.3.3.27 TestCreateLfu() | 27 |
| 5.3.3.28 TestRemoveFreq() | 27 |
| 5.3.3.29 TestRemoveLfu() | 28 |
| 5.3.3.30 TestReplaceLfu() | 28 |
| 5.4 /home/kir/and another one/lfu_cache/List/List_Map.c File Reference | 28 |
| 5.4.1 Function Documentation | 28 |
| 5.4.1.1 CreateFreq() | 28 |
| 5.4.1.2 CreateHead() | 29 |
| 5.4.1.3 CreateLfu() | 29 |
| 5.4.1.4 DeleteList() | 29 |
| 5.4.1.5 ListPrint() | 30 |
| 5.4.1.6 RemoveFreq() | 30 |
| 5.4.1.7 RemoveLfu() | 30 |
| 5.4.1.8 ReplaceLfu() | 30 |
| 5.4.1.9 TestCreateFreq() | 31 |
| 5.4.1.10 TestCreateHead() | 31 |
| 5.4.1.11 TestCreateLfu() | 31 |
| 5.4.1.12 TestRemoveFreq() | 31 |
| 5.4.1.13 TestRemoveLfu() | 31 |
| 5.4.1.14 TestReplaceLfu() | 31 |
| 5.5 /home/kir/and another one/lfu_cache/main/main.c File Reference | 31 |
| 5.5.1 Function Documentation | 32 |
| 5.5.1.1 main() | 32 |
| 5.6 /home/kir/and another one/lfu_cache/main/new_main.c File Reference | 32 |
| 5.6.1 Function Documentation | 32 |
| 5.6.1.1 main() | 32 |
| 5.7 /home/kir/and another one/lfu_cache/main/test.txt File Reference | 32 |
| 5.8 /home/kir/and another one/lfu_cache/README.md File Reference | 32 |
| 5.9 /home/kir/and another one/lfu_cache/Test/HC.txt File Reference | 32 |
| 5.9.1 Variable Documentation | 33 |
| 5.9.1.1 "Hello | 33 |
| 5.9.1.2 Hello | 33 |
| 5.10 /home/kir/and another one/lfu_cache/Test/Hi.txt File Reference | 33 |
| 5.11 /home/kir/and another one/lfu_cache/Test/INIT.txt File Reference | 33 |

| | |
|--|-----------|
| 5.12 /home/kir/and another one/lfu_cache/Test/SF.txt File Reference | 33 |
| 5.13 /home/kir/and another one/lfu_cache/Test/test.c File Reference | 33 |
| 5.13.1 Function Documentation | 33 |
| 5.13.1.1 main() | 33 |
| 5.14 /home/kir/and another one/lfu_cache/Test/Test_Hash_Map.c File Reference | 34 |
| 5.14.1 Function Documentation | 34 |
| 5.14.1.1 Hash_Char_Test() | 34 |
| 5.14.1.2 Hash_Int_Test() | 34 |
| 5.14.1.3 Init_Func_Test() | 34 |
| 5.14.1.4 main() | 34 |
| 5.14.1.5 Test_SearchMap() | 34 |
| 5.15 /home/kir/and another one/lfu_cache/Test/Test_LFU.c File Reference | 35 |
| 5.15.1 Function Documentation | 35 |
| 5.15.1.1 TestLFUFunc() | 35 |
| 5.15.1.2 TestPageFunc() | 35 |
| 5.16 /home/kir/and another one/lfu_cache/Test/Test_LFU.h File Reference | 35 |
| 5.16.1 Function Documentation | 35 |
| 5.16.1.1 TestLFUFunc() | 36 |
| 5.16.1.2 TestPageFunc() | 36 |
| Index | 37 |

Chapter 1

Lfu Cache

Realisation of LFU cache algorithm ([read more about algorithm](#))

- LFU - directory with 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 relationship 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:

| | |
|---|----|
| freq_node | |
| Node which contains frequency | 7 |
| hash_cell | |
| The cell of hash table | 8 |
| hash_map | |
| The struct of Hash Map | 9 |
| lfu_c | 10 |
| lfu_node | |
| Node which contains pages with data | 11 |
| request_t | 12 |

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

| | |
|--|----|
| /home/kir/and another one/lfu_cache/Hash_Map/ Hash_Map.c | 13 |
| /home/kir/and another one/lfu_cache/LFU/ LFU.c | 13 |
| /home/kir/and another one/lfu_cache/LFU/ LFU.h | 16 |
| /home/kir/and another one/lfu_cache/List/ List_Map.c | 28 |
| /home/kir/and another one/lfu_cache/main/ main.c | 31 |
| /home/kir/and another one/lfu_cache/main/ new_main.c | 32 |
| /home/kir/and another one/lfu_cache/Test/ test.c | 33 |
| /home/kir/and another one/lfu_cache/Test/ Test_Hash_Map.c | 34 |
| /home/kir/and another one/lfu_cache/Test/ Test_LFU.c | 35 |
| /home/kir/and another one/lfu_cache/Test/ Test_LFU.h | 35 |

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 | <i>cache</i> | 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 | <i>f</i> | 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 | <i>cache</i> | Pointer to cache where we want to insert |
| in | <i>request</i> | 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 | <i>cache_size</i> | 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 | <i>cache</i> | LFU-cache which we want to print |
| in | <i>source</i> | 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 | <i>page</i> | Pointer to page with data |
| in | <i>source</i> | 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}, initializes 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 available 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 | <i>freq_dat</i> | Frequency node |
| in | <i>prev_fr</i> | 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 | <i>lfu_dat</i> | Page with data which well be added to lfu_node (p. 11) |
| in | <i>head</i> | 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 | <i>Hash_map</i> | Pointer to Hash Map |
| in | <i>request</i> | 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 | <i>Hash_map</i> | Pointer to Hash Map |
|----|-----------------|---------------------|

Returns

Integer zero

5.3.3.7 FreeLFU()

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

Destructor of LFU-cache.

Parameters

| | | |
|----|--------------|----------------------|
| in | <i>cache</i> | 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 | <i>f</i> | 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 | <i>string</i> | String |
| in | <i>len</i> | The length of string |
| in | <i>hash_size</i> | 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 | <i>request</i> | pointer to request |
| in | <i>hash_size</i> | 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 | <i>number</i> | Integer number |
| in | <i>hash_size</i> | 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 = $\text{cache_size} / 10 + 1$ }, initializes array of pointers to cells and cells.

Parameters

| | | |
|----|-------------------|-------------------|
| in | <i>cache_size</i> | 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 available place to adding new data. If this data already exists here, it does nothing.

Parameters

| | | |
|----|-----------------|---------------------|
| in | <i>Hash_Map</i> | pointer to Hash Map |
| in | <i>request</i> | pointer to request |

Returns

The pointer on an available cell in Hash Map

5.3.3.14 InsertLFU()

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

Func which insert new page to cache.

Parameters

| | | |
|----|----------------|--|
| in | <i>cache</i> | Pointer to cache where we want to insert |
| in | <i>request</i> | 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 | <i>cache_size</i> | 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 | <i>cache</i> | LFU-cache which we want to print |
| in | <i>source</i> | 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 | <i>head</i> | Pointer to head of list |
| in | <i>f</i> | 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 | <i>Hash_map</i> | Pointer to Hash Map |
| in | <i>f</i> | 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 | <i>page</i> | Pointer to page with data |
| in | <i>source</i> | 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 | <i>del</i> | 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 | <i>head</i> | 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 | <i>cur_lfu</i> | 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 | <i>cell</i> | Element of array of pointers to cells in Hash Map |
| in | <i>request</i> | 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 | <i>Hash_Map</i> | pointer to Hash Map |
| in | <i>request</i> | 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 | <i>freq_dat</i> | Frequency node |
| in | <i>prev_fr</i> | 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 | <i>lfu_dat</i> | Page with data which well be added to lfu_node (p. 11) |
| in | <i>head</i> | 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 | <i>head</i> | Pointer to head of list |
| in | <i>f</i> | Filestream pointer where we want to print list |

5.4.1.6 RemoveFreq()

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

Delete frequency node.

Parameters

| | | |
|----|------------|------------------------------|
| in | <i>del</i> | 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 | <i>head</i> | 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 replace it. If the element with current frequency + 1 doesn't exist it will be created (with function `create_freq`).

Parameters

| | | |
|-----------------|----------------------|-----------------------------------|
| <code>in</code> | <code>cur_lfu</code> | 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/HI.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

| | |
|---|---------------------|
| !Hello | data_t |
| HC.txt, 33 | lfu_node, 11 |
| /home/kir/and another one/lfu_cache/Hash_Map/Hash_MapElem | DeleteElem |
| 13 | LFU.h, 20 |
| /home/kir/and another one/lfu_cache/LFU/LFU.c, 13 | DeleteList |
| /home/kir/and another one/lfu_cache/LFU/LFU.h, 16 | LFU.h, 20 |
| /home/kir/and another one/lfu_cache/List/List_Map.c, 28 | List_Map.c, 29 |
| /home/kir/and another one/lfu_cache/README.md, 32 | FreeHashMap |
| /home/kir/and another one/lfu_cache/Test/HC.txt, 32 | LFU.h, 20 |
| /home/kir/and another one/lfu_cache/Test/Hi.txt, 33 | FreeLFU |
| /home/kir/and another one/lfu_cache/Test/INIT.txt, 33 | LFU.c, 13 |
| /home/kir/and another one/lfu_cache/Test/SF.txt, 33 | LFU.h, 21 |
| /home/kir/and another one/lfu_cache/Test/Test_Hash_Map.c, | freq_node, 7 |
| 34 | child, 7 |
| /home/kir/and another one/lfu_cache/Test/Test_LFU.c, | freq_t, 7 |
| 35 | next, 8 |
| /home/kir/and another one/lfu_cache/Test/Test_LFU.h, | prev, 8 |
| 35 | freq_t |
| /home/kir/and another one/lfu_cache/Test/test.c, 33 | freq_node, 7 |
| /home/kir/and another one/lfu_cache/main/main.c, 31 | GetPage |
| /home/kir/and another one/lfu_cache/main/new_main.c, | LFU.c, 14 |
| 32 | LFU.h, 21 |
| /home/kir/and another one/lfu_cache/main/test.txt, 32 | |
| cache_fullnes | hash_cell, 8 |
| lfu_c, 10 | item, 8 |
| cache_size | next, 9 |
| lfu_c, 10 | prev, 9 |
| cells | Hash_Char_Test |
| hash_map, 9 | Test_Hash_Map.c, 34 |
| CGetPage | Hash_Int_Test |
| LFU.h, 18 | Test_Hash_Map.c, 34 |
| child | hash_map, 9 |
| freq_node, 7 | cells, 9 |
| CPrintPage | size, 9 |
| LFU.h, 18 | HashofChar |
| CreateFreq | LFU.h, 21 |
| LFU.h, 19 | HashofData |
| List_Map.c, 28 | LFU.h, 22 |
| CreateHead | HashofInt |
| LFU.h, 19 | LFU.h, 22 |
| List_Map.c, 29 | HashTable |
| CreateLfu | lfu_c, 10 |
| LFU.h, 19 | HC.txt |
| List_Map.c, 29 | !Hello, 33 |
| | Hello, 33 |
| DATA | Hello |
| LFU.h, 18 | HC.txt, 33 |
| data | |
| request_t, 12 | Init_Func_Test |

- Test_Hash_Map.c, 34
- InitHashMap
 - LFU.h, 22
- InsertHashMap
 - LFU.h, 23
- InsertLFU
 - LFU.c, 14
 - LFU.h, 23
- item
 - hash_cell, 8
- LFU
 - LFU.h, 18
- LFU.c
 - FreeLFU, 13
 - GetPage, 14
 - InsertLFU, 14
 - LfuConstruct, 14
 - LFUDump, 15
 - PrintPage, 15
- LFU.h
 - CGetPage, 18
 - CPrintPage, 18
 - CreateFreq, 19
 - CreateHead, 19
 - CreateLfu, 19
 - DATA, 18
 - DelElem, 20
 - DeleteList, 20
 - FreeHashMap, 20
 - FreeLFU, 21
 - GetPage, 21
 - HashofChar, 21
 - HashofData, 22
 - HashofInt, 22
 - InitHashMap, 22
 - InsertHashMap, 23
 - InsertLFU, 23
 - LFU, 18
 - LfuConstruct, 24
 - LFUDump, 24
 - ListPrint, 24
 - NUM, 18
 - PrintHashMap, 25
 - PrintPage, 25
 - RemoveFreq, 25
 - RemoveLfu, 26
 - ReplaceLfu, 26
 - SearchData, 26
 - SearchMap, 27
 - TestCreateFreq, 27
 - TestCreateHead, 27
 - TestCreateLfu, 27
 - TestRemoveFreq, 27
 - TestRemoveLfu, 27
 - TestReplaceLfu, 28
- lfu_c, 10
 - cache_fullnes, 10
 - cache_size, 10
- HashTable, 10
 - List, 10
- lfu_node, 11
 - data_t, 11
 - next, 11
 - parent, 11
 - prev, 12
- LfuConstruct
 - LFU.c, 14
 - LFU.h, 24
- LFUDump
 - LFU.c, 15
 - LFU.h, 24
- List
 - lfu_c, 10
- List_Map.c
 - CreateFreq, 28
 - CreateHead, 29
 - CreateLfu, 29
 - DeleteList, 29
 - ListPrint, 29
 - RemoveFreq, 30
 - RemoveLfu, 30
 - ReplaceLfu, 30
 - TestCreateFreq, 31
 - TestCreateHead, 31
 - TestCreateLfu, 31
 - TestRemoveFreq, 31
 - TestRemoveLfu, 31
 - TestReplaceLfu, 31
- ListPrint
 - LFU.h, 24
 - List_Map.c, 29
- main
 - main.c, 32
 - new_main.c, 32
 - test.c, 33
 - Test_Hash_Map.c, 34
- main.c
 - main, 32
- new_main.c
 - main, 32
- next
 - freq_node, 8
 - hash_cell, 9
 - lfu_node, 11
- NUM
 - LFU.h, 18
- parent
 - lfu_node, 11
- prev
 - freq_node, 8
 - hash_cell, 9
 - lfu_node, 12
- PrintHashMap
 - LFU.h, 25

PrintPage
 LFU.c, 15
 LFU.h, 25

RemoveFreq
 LFU.h, 25
 List_Map.c, 30

RemoveLfu
 LFU.h, 26
 List_Map.c, 30

ReplaceLfu
 LFU.h, 26
 List_Map.c, 30

request_t, 12
 data, 12

SearchData
 LFU.h, 26

SearchMap
 LFU.h, 27

size
 hash_map, 9

test.c
 main, 33

Test_Hash_Map.c
 Hash_Char_Test, 34
 Hash_Int_Test, 34
 Init_Func_Test, 34
 main, 34
 Test_SearchMap, 34

Test_LFU.c
 TestLFUFunc, 35
 TestPageFunc, 35

Test_LFU.h
 TestLFUFunc, 35
 TestPageFunc, 36

Test_SearchMap
 Test_Hash_Map.c, 34

TestCreateFreq
 LFU.h, 27
 List_Map.c, 31

TestCreateHead
 LFU.h, 27
 List_Map.c, 31

TestCreateLfu
 LFU.h, 27
 List_Map.c, 31

TestLFUFunc
 Test_LFU.c, 35
 Test_LFU.h, 35

TestPageFunc
 Test_LFU.c, 35
 Test_LFU.h, 36

TestRemoveFreq
 LFU.h, 27
 List_Map.c, 31

TestRemoveLfu
 LFU.h, 27

 List_Map.c, 31

TestReplaceLfu
 LFU.h, 28
 List_Map.c, 31