Bloom Filters

An implementation and interactive demo

Data Structures and Algorithms
Final Assessment

Giannis Tsagatakis

Contents

1 Τα φίλτρα **Bloom** **1**

2 Συστημα **building and Configuration** **3**

2.1 The main 'CMakeLists.txt' file . . . 3

2.2 The 'src/CMakeLists.txt' file . . . 3

3 Example interactive program running. **5**

4 Συμπαγές Ευρετήριο **7**

4.1 Λίστα Κλάσεων . . . 7

5 Ευρετήριο Αρχείων **9**

5.1 Λίστα Αρχείων . . . 9

6 Τεκμηρίωση Κλάσεων **11**

6.1 Τεκμηρίωση Κλάσης BloomFilter . . . 11

6.1.1 Λεπτομερής Περιγραφή . . . 12

6.1.2 Τεκμηρίωση Constructor & Destructor . . . 12

6.1.2.1 BloomFilter() [1/2] . . . 12

6.1.2.2 BloomFilter() [2/2] . . . 13

6.1.3 Τεκμηρίωση Συναρτήσεων Μελών . . . 13

6.1.3.1 add() . . . 13

6.1.3.2 estimateP() [1/2] . . . 13

6.1.3.3 estimateP() [2/2] . . . 14

6.1.3.4 fullness() . . . 14

6.1.3.5 getCount() . . . 14

6.1.3.6 getNumHashes() . . . 15
6.1.3.7 getSize() . . . 15
6.1.3.8 notHave() . . . 15
6.1.3.9 populate() . . . 16
6.1.3.10 setVerbose() . . . 16
6.2 Τεκμηρίωση Κλάσης Mixer . . . 16
6.2.1 Λεπτομερής Περιγραφή . . . 17
6.2.2 Τεκμηρίωση Constructor & Destructor . . . 17
6.2.2.1 Mixer() . . . 17
6.2.3 Τεκμηρίωση Συναρτήσεων Μελών . . . 18
6.2.3.1 operator()() . . . 18
6.3 Τεκμηρίωση Κλάσης pasaTempo . . . 18
6.3.1 Λεπτομερής Περιγραφή . . . 19
6.4 Τεκμηρίωση Κλάσης Repl . . . 19
6.4.1 Λεπτομερής Περιγραφή . . . 20
6.4.2 Τεκμηρίωση Constructor & Destructor . . . 21
6.4.2.1 Repl() . . . 21
6.4.3 Τεκμηρίωση Συναρτήσεων Μελών . . . 21
6.4.3.1 eval_cmd_line() . . . 21
6.4.3.2 eval_line() . . . 22
6.4.3.3 register_cmd() . . . 22

7 Τεκμηρίωση Αρχείων **23**

7.1 Τεκμηρίωση Αρχείου CMakeLists.dox . . . 23

7.1.1 Λεπτομερής Περιγραφή . . . 23

7.2 Τεκμηρίωση Αρχείου src/BloomFilter.cpp . . . 23

7.2.1 Λεπτομερής Περιγραφή . . . 24

7.3 BloomFilter.cpp . . . 24

7.4 Τεκμηρίωση Αρχείου src/BloomFilter.h . . . 24

7.4.1 Λεπτομερής Περιγραφή . . . 25

7.5 BloomFilter.h . . . 26

7.6 Τεκμηρίωση Αρχείου src/Mixer.h . . . 26

7.6.1 Λεπτομερής Περιγραφή . . . 27

7.7 Mixer.h . . . 28

7.8 Τεκμηρίωση Αρχείου src/pasaTempo.h . . . 28

7.8.1 Λεπτομερής Περιγραφή . . . 29

7.9 pasaTempo.h . . . 29

7.10 Τεκμηρίωση Αρχείου src/repl.cpp . . . 30

7.10.1 Λεπτομερής Περιγραφή . . . 30

7.11 repl.cpp . . . 30

7.12 Τεκμηρίωση Αρχείου src/repl.h . . . 35

7.12.1 Λεπτομερής Περιγραφή . . . 36

7.13 repl.h . . . 36

7.14 Τεκμηρίωση Αρχείου src/repl_hooks.cpp . . . 37

7.14.1 Λεπτομερής Περιγραφή . . . 38

7.15 repl_hooks.cpp . . . 38

7.16 Τεκμηρίωση Αρχείου src/repl_hooks.h . . . 38

7.16.1 Λεπτομερής Περιγραφή . . . 39

7.17 repl_hooks.h . . . 40

7.18 Τεκμηρίωση Αρχείου src/utils.h . . . 40

7.18.1 Λεπτομερής Περιγραφή . . . 41

7.19 utils.h . . . 41

8 Τεκμηρίωση Παραδειγμάτων **43**

8.1 examples/mixer_demo.cpp . . . 43

8.2 examples/size_demo.cpp . . . 43

Index **45**

Chapter 1

Τα φίλτρα **Bloom**

Τελική εργασία για το μάθημα Algorithms and Data Structures του Τσαγκατάκη Ιωάννη.

Τα φίλτρα **Bloom**

Υπολογισμός Μεγέθους

Στο παράδειγμα βλέπουμε δύο παραδείγματα ένα άνετο φίλτρο καθώς και ένα γεμάτο φίλτρο. Το γεμάτο φίλτρο θα δώσει μόνο *positive* και *false positive* τιμές και ουσιατικά είναι άχρηστο.

```
auto bloom = BloomFilter{100, 5};
bloom.populate("../data/cia_words.txt");
```

```
Data fullness :0.52
Testing Linux : maybe yes
Testing Amiga : NO
Testing Windows : NO
```

```
auto bloom = BloomFilter{10, 5};
bloom.populate("../data/cia_words.txt");
```

```
Data fullness : 1
Testing Linux : maybe yes
Testing Amiga : maybe yes
Testing Windows : maybe yes
```

Λεπτομέρειες Υλοποίησης

Δομή δεδομένων **bitset.**

Για την εσωτερική αναπαράσταση ενός bitset ένα std::vector<bool> μοιάζει μια κατάλληλη δομή. Αλλά το std::vector<bool> έρχεται απο την πρώτη έκδοση της STL και έχει σχεδιαστεί σαν container. Η φήμη του μεταξύ των υπολοίπων μελλών της STL είναι αρκετά κακόφημη.

Η C++ committee έχει προσθέσει το std::bitset<std::size_t N> ως μια λογικότερη αντικατάσταση του, αλλά αυτό πρέπει να είναι σταθερού μεγέθους κατά το compiling. Οπότε επιλέχθηκε η δυναμική έκδοση του απο την βιβλιοθήκη boost, to boost::dynamic_bitset<>.

Για την υλοποίηση **exec**

Για το read/eval/loop ιδανικά θα ήθελε κάποιος κάτι που να συνδιάζει το GNU readline ή libedit καθώς και το GNU historyFile, πουείναι βιβλιοθήκες C, όχι απο τιθς ποιό ευχρηστες. Βρέθηκαν στο github διάφορες υλοποιησεις και επιλέχθηκε η βιβλιοθήκη replxx του Marcin Konarski (https://github.com/Amok↩Huginnsson/replxx). Προτιμήηκε απο άλλα σχετικά projects λόγω του μοντέρνου και έγρωμου UI, αλλά και γιατί το παράδειγμα που είχε ήταν στα μέτρα του προβλήματος.

Έχωντας μια καλή τέτοια βιβλιοθήκη η χρήση της ώστε να υλοποιήθει μικρή γλώσσα scripting για την εξερεύνηση και τον έλεγχο του προγράμματος ήταν μια εύκολη υπόθεση. Αυτό θα μπορούσε να χρησιμοποιηθεί και για CI/TDD αλλά με δεδομένο τον περιορισμένο χρόνο παράδοσης, δεν επιλέχθηκε μια agile και μοντέρνα μεθοδολογία, αλλά το agile και τα test cases και τα refactoring εχουν αφήσει τα ίχνη τους στο ιστορικό του versioning system (git). Οι εξαιρετικές δυνατότητες του Clion (academic licence) για refactoring έχουν αφήσει τα ίχνη τους επίσης.

Για την τεκμηρίωση

Ο συγραφέας πρώτη φορά επιχειρεί τεκμηρίωση σε εκτυπόσιμη μορφή pdf. Σαν τέταοι απέχει απο το να είναι η καλύτερη δυνατή. Το doxygen σε συνδιασμό με το LaTeX έχουν χρησιμοποιηθεί. Προσθέτωντας ελληνικά και πειράζοντας και τα temnplates του LaTeX δεν ήταν ένα εύκολο έργο. Τελικά η λύση για σωστά ελληνικά πέρναγε με την με το χέρι μεταγλώστη του doxygen στην τελευταία του έκδοση.

Chapter 2

Συστημα building and Configuration

The *cmake* is used for configuration, *ninja* for building and *clang*++ for compiling.

```
cmake -G Ninja -DCMAKE_CXX_COMPILER=clang++ -DCMAKE_C_COMPILER=clang -DCMAKE↩
_EXPORT_COMPILE_COMMANDS=ON -DCMAKE_BUILD_TYPE=Debug
```

2.1 The main 'CMakeLists.txt' file

```
cmake_minimum_required(VERSION 3.5)
project(BloomShell)
set(CMAKE_CXX_STANDARD 14)
set(CMAKE_EXPORT_COMPILE_COMMANDS ON)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall  -pedantic")
set(Boost_USE_STATIC_LIBS OFF)
set(Boost_USE_MULTITHREADED ON)
set(Boost_USE_STATIC_RUNTIME OFF)
find_package(Boost 1.45.0 REQUIRED )
#include_directories(${CMAKE_SOURCE_DIR}/ext/dynamic_bitset/include)
include_directories(${Boost_INCLUDE_DIRS})
add_library(replxx STATIC IMPORTED)
set_target_properties(replxx PROPERTIES IMPORTED_LOCATION ${CMAKE_SOURCE_DIR}/ext/replxx/build/libreplxx.a)
include_directories(${CMAKE_SOURCE_DIR}/ext/replxx/include)
add_subdirectory(src)
```

2.2 The 'src/CMakeLists.txt' file

```
set(BSHELL_SOURCES
        utils.h utils.cpp
        repl.h repl.cpp
        repl_hooks.h repl_hooks.cpp
        BloomFilter.cpp BloomFilter.h
        pasaTempo.h
        bshell.cpp)
add_executable(bshell ${BSHELL_SOURCES})
target_link_libraries(bshell  replxx ${Boost_LIBRARIES})
#Copy Data files
add_custom_command( TARGET bshell
    POST_BUILD
    COMMAND ${CMAKE_COMMAND} -E echo "Copying unit test data.."
    COMMAND ${CMAKE_COMMAND} -E copy_directory ${CMAKE_SOURCE_DIR}/data ${CMAKE_BINARY_DIR}/data
)
```

Chapter 3

Example interactive program running.

```
Welcome to bshell, an interactive scripted bloom filter demo
Type !help to see the available connands.
$ !help
Commands:
   !help    Help about commands
   !history Show the historyFile
   !exit    Exit program
   !quit    Exit program
   !verbose Set verbose mode
   !init    (Re)Initialize Bloom filter
   !design  Design Bloom filter
   !stats   Bloom filter statistics
   !add     Add words to bloom filter
   !bits    Show bloom filter bits
   !import  Import words from files
   !check   Check words
   !nbits   Debug mixer
   !run      Run script
$ !stats
   Bits   : 60
   Memory : 60B
   Hashes : 10
Num items : 0
 Fullness : 0%
False Pos   0
$ !bits
000000000000000000000000000000000000000000000000000000000000000000000000
$ !nbits Linux
Linux   : 49 30 57 24 30 49 16 32 41 59
$ !add Linux
added 'Linux'
$ !bits
000000000000000010000000100000101000000001000000010000000101
$ !stats
   Bits   : 60
   Memory : 60B
   Hashes : 10
Num items : 1
 Fullness : 13,3333%
False Pos   7,27111e-09
$ !check Love and Linux
Positives: 'Linux'
$ Love and Linux
Positives: 'Linux'
$ !history
   0: !check the hacker and the  hasker
   1: !stats
   2: !quit
   3: !help
   4: !stats
   5: !bits
   6: !nbits Linux
   7: !add Linux
   8: !bits
   9: !stats
  10: !check Love and Linux
  11: !history
$ !import ../data/cia_words.txt
Importing words form ../data/cia_words.txt.
added 'Osama'
added '17N'
added 'hacker'
```

```
added 'linux'
added 'Kufontinas'
added 'Κουφοντίνας'
added 'Οσάμα'
added 'Αναρχία'
added 'Επανάσταση'
added 'revolution'
added 'Linux'
added 'Γιαούρτι'
   Bits   : 60
   Memory : 60B
   Hashes : 10
Num items : 24
 Fullness : 88,3333%
False Pos   0,831225
$ !check the hacker and the  hasker
Positives: 'the','hacker','the','hasker'
$ !design 1000 0.001
   Bits   : 14378
   Memory : 14KB
   Hashes : 10
Num items : 0
 Fullness : 0%
False Pos   0
$ !verbose off
Verbose mode: OFF
$ !import ../data/cia_words.txt
$ !check the hacker and the  hasker
Positives: 'hacker'
$ !quit
Total run time 36,688 ms.
```

Chapter 4

Συμπαγές Ευρετήριο

4.1 Λίστα Κλάσεων

Ακολουθούν οι κλάσεις, οι δομές, οι ενώσεις και οι διασυνδέσεις με σύντομες περιγραφές:

BloomFilter
: Υλοποιηση ενός απλού φίλτρου Bloom 11

Mixer
: Ένα συναρτησιακό για τον υπολογισμό των θέσεων σε ένα φίλτρο Bloom 16

pasaTempo
: A simple class to measure timings in ns 18

Repl 19

Chapter 5

Ευρετήριο Αρχείων

5.1 Λίστα Αρχείων

Ακολουθεί μια λίστα όλων των τεκμηριωμένων αρχείων με σύντομες περιγραφές:

src/BloomFilter.cpp
Υλοποίηση της κλάσης BloomFilter . . . 23
src/BloomFilter.h
Ορισμός της κλάσης BloomFilter . . . 24
src/**bshell.cpp** . . . **??**
src/Mixer.h
Ορισμός της κλάσης Mixer . . . 26
src/pasaTempo.h
Ορισμός της κλάσης pasaTempo . . . 28
src/repl.cpp
Κλάση Pepl υλοποίηση . . . 30
src/repl.h
Κλάση Pepl ορισμοί . . . 35
src/repl_hooks.cpp
Υποσύστημα repl . . . 37
src/repl_hooks.h
Υποσύστημα repl . . . 38
src/**utils.cpp** . . . **??**
src/utils.h
Βοηθητικές συναρτήσεις . . . 40

Chapter 6

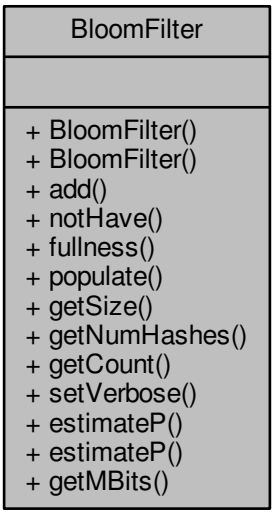
Τεκμηρίωση Κλάσεων

6.1 Τεκμηρίωση Κλάσης **BloomFilter**

Υλοποιηση ενός απλού φίλτρου Bloom.

`#include <BloomFilter.h>`

Διάγραμμα Συνεργασίας για την κλάση BloomFilter:



Δημόσιες Μέθοδοι

- BloomFilter (size_t size=80, size_t numHashes=10)

  Bloom filter by size.
- BloomFilter (size_t nwords=1000, double P=0.01)

Design a bloom filter.

- void add (const std::string &key)
- bool notHave (const std::string &key)
- float fullness () const

  Get the percentage of free bits.
- void populate (const std::string &fname)

  Populate bloom filter from filename.
- size_t getSize () const

  Get the number of bit in the hash table.
- size_t getNumHashes () const

  Get the number of hashes.
- size_t getCount () const

  Get an estimate of the number of words in the bloom filter.
- void setVerbose (bool verbose)

  Set the verbose flag.
- double estimateP ()

  Get the false possitive propability of the filter.
- double estimateP (double N)

  Get the false possitive propability of the filter.
- const dynamic_bitset & **getMBits** () const

6.1.1 Λεπτομερής Περιγραφή

Υλοποιηση ενός απλού φίλτρου Bloom.

Η κλάση Mixer είναι μια συναρτησιακή κλάση που παράγει uniformly distributed random numbers αρχικοποιημένουςμε την τιμή Hash ενός κλειδιού. Χρήσιμη για τον υπολογισμό των θέσεων σε φίλτρα Bloom.

Παραδείγματα:

examples/size_demo.cpp.

Ορισμός στη γραμμή 28 του αρχείου BloomFilter.h.

6.1.2 Τεκμηρίωση Constructor & Destructor

6.1.2.1 BloomFilter() [1/2]

```
BloomFilter::BloomFilter (
            size_t size = 80,
            size_t numHashes = 10 )  [inline], [explicit]
```

Bloom filter by size.

Παράμετροι

size	Number of bits
numHashes	Number of hashes

Ορισμός στη γραμμή 35 του αρχείου BloomFilter.h.

6.1.2.2 BloomFilter() [2/2]

```
BloomFilter::BloomFilter (
            size_t nwords = 1000,
            double P = 0.01 )  [explicit]
```

Design a bloom filter.

Παράμετροι

nwords	Capacity in words
P	Propability of false possitives

Ορισμός στη γραμμή 37 του αρχείου BloomFilter.cpp.

6.1.3 Τεκμηρίωση Συναρτήσεων Μελών

6.1.3.1 add()

```
void BloomFilter::add (
            const std::string & key )
```

Add a wprd to the floom filter

Παράμετροι

key	The word to add

Ορισμός στη γραμμή 45 του αρχείου BloomFilter.cpp.

6.1.3.2 estimateP() [1/2]

```
double BloomFilter::estimateP ( )
```

Get the false possitive propability of the filter.

Επιστρέφει

The false positives probability.

Ορισμός στη γραμμή 33 του αρχείου BloomFilter.cpp.

6.1.3.3 estimateP() [2/2]

```
double BloomFilter::estimateP (
            double N )
```

Get the false possitive propability of the filter.

Παράμετροι

N	The number of stored elements

Επιστρέφει

The false positives probability.

Ορισμός στη γραμμή 24 του αρχείου BloomFilter.cpp.

6.1.3.4 fullness()

```
float BloomFilter::fullness ( ) const  [inline]
```

Get the percentage of free bits.

If a bloom filter have no zero bits its almost useless.

Επιστρέφει

The percentage of the free bits.

Ορισμός στη γραμμή 65 του αρχείου BloomFilter.h.

6.1.3.5 getCount()

```
size_t BloomFilter::getCount ( ) const  [inline]
```

Get an estimate of the number of words in the bloom filter.

Επιστρέφει

Estimate number of words

Ορισμός στη γραμμή 95 του αρχείου BloomFilter.h.

6.1.3.6 getNumHashes()

```
size_t BloomFilter::getNumHashes ( ) const  [inline]
```

Get the number of hashes.

Επιστρέφει

The number of hashes

Ορισμός στη γραμμή 89 του αρχείου BloomFilter.h.

6.1.3.7 getSize()

```
size_t BloomFilter::getSize ( ) const  [inline]
```

Get the number of bit in the hash table.

Επιστρέφει

The number of bits

Ορισμός στη γραμμή 83 του αρχείου BloomFilter.h.

6.1.3.8 notHave()

```
bool BloomFilter::notHave (
            const std::string & key )
```

check NOT mebership

Παράμετροι

key	

Επιστρέφει

true if not have

Ορισμός στη γραμμή 62 του αρχείου BloomFilter.cpp.

6.1.3.9 populate()

```
void BloomFilter::populate (
            const std::string & fname )
```

Populate bloom filter from filename.

Smart split the finame contents to words and add each word to the bloom filter

Παράμετροι

fname	

Παραδείγματα:

examples/size_demo.cpp.

Ορισμός στη γραμμή 7 του αρχείου BloomFilter.cpp.

6.1.3.10 setVerbose()

```
void BloomFilter::setVerbose (
            bool verbose )  [inline]
```

Set the verbose flag.

Παράμετροι

verbose	

Ορισμός στη γραμμή 101 του αρχείου BloomFilter.h.

Η τεκμηρίωση για αυτή την κλάση δημιουργήθηκε από τα ακόλουθα αρχεία:

- src/BloomFilter.h
- src/BloomFilter.cpp

6.2 Τεκμηρίωση Κλάσης **Mixer**

Ένα συναρτησιακό για τον υπολογισμό των θέσεων σε ένα φίλτρο Bloom.

```
#include <Mixer.h>
```

Διάγραμμα Συνεργασίας για την κλάση Mixer:

Δημόσιες Μέθοδοι

- Mixer (const std::string &val, size_t size)

  Δημιούργησε το συναρτησιακό για μιά δεδομένη συμβολοσειρά κλειδί.
- std::size_t operator() ()

6.2.1 Λεπτομερής Περιγραφή

Ένα συναρτησιακό για τον υπολογισμό των θέσεων σε ένα φίλτρο Bloom.

Η κλάση Mixer είναι μια συναρτησιακή κλάση που παράγει uniformly distributed random numbers αρχικοποιημένους με την τιμή Hash ενός κλειδιού. Χρήσιμη για τον υπολογισμό των θέσεων σε φίλτρα Bloom.

Παράδειγμα χρήσης:
```
std::size_t index = Mixer{"Viagra",10}();
```

Η κατάσταση που αποθηκεύει είναι κατάσταση της γενήτριας των ψευδοτυχαίων αριθμών. Η γενήτρια αυτή είναι μια κλασσική linear_congruential_engine, η απολούστερη δυνατή της τυπικής βιβλιοθήκης, με εξαιρετική απόδοση ώς προς τον απαιτούμενο αποθηκευτικό χώρο για την αποθήκευση της κατάτσασης και τον υπολογιστικό χρόνο.

$$x = (\alpha \cdot x + c) \,\%\, m$$

Παραδείγματα:

examples/mixer_demo.cpp.

Ορισμός στη γραμμή 34 του αρχείου Mixer.h.

6.2.2 Τεκμηρίωση **Constructor & Destructor**

6.2.2.1 Mixer()

```
Mixer::Mixer (
            const std::string & val,
            size_t size )  [inline]
```

Δημιούργησε το συναρτησιακό για μιά δεδομένη συμβολοσειρά κλειδί.

Αλγόριθμος: Πρώτα υπολογισε την τιμή κερματισμού του κλειδίου με την std::hash<std::string> και δημιουργησε μια γενήτρια τυχαίων αριθμών αρχικοποιημένη με αυτή την τιμή.Σ

Παράμετροι

val	Το κλειδί εισόδου
size	Το μέγεθός του φίλτρου Bloom

Ορισμός στη γραμμή 46 του αρχείου Mixer.h.

6.2.3 Τεκμηρίωση Συναρτήσεων Μελών

6.2.3.1 operator()()

```
std::size_t Mixer::operator() ( )  [inline]
```

Επιστρέφει κάθε φορά που καλείτε μια θέση στον πίνακα bitvector. Συνεχείς κλησεις της θα επιστρέψουν διαφορετικές τιμές, αλλά πάντα με την ίδια σειρά για το ίδιο κείμενο.

Επιστρέφει

Μια θέση στον πίνακα bitvector.

Ορισμός στη γραμμή 61 του αρχείου Mixer.h.

Η τεκμηρίωση για αυτή την κλάση δημιουργήθηκε από το ακόλουθο αρχείο:

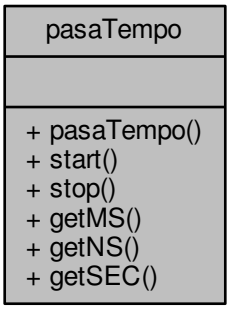
- src/Mixer.h

6.3 Τεκμηρίωση Κλάσης **pasaTempo**

A simple class to measure timings in ns.

`#include <pasaTempo.h>`

Διάγραμμα Συνεργασίας για την κλάση pasaTempo:



Δημόσιες Μέθοδοι

- void **start** ()
- long **stop** ()
- long **getMS** ()
- long **getNS** ()
- long **getSEC** ()

6.3.1 Λεπτομερής Περιγραφή

A simple class to measure timings in ns.

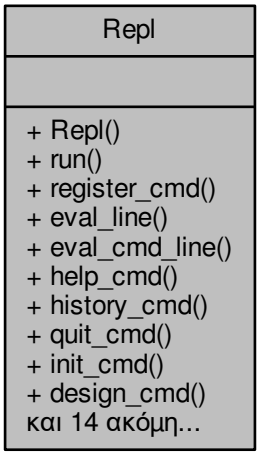
Ορισμός στη γραμμή 18 του αρχείου pasaTempo.h.

Η τεκμηρίωση για αυτή την κλάση δημιουργήθηκε από το ακόλουθο αρχείο:

- src/pasaTempo.h

6.4 Τεκμηρίωση Κλάσης **Repl**

Διάγραμμα Συνεργασίας για την κλάση Repl:



Δημόσιες Μέθοδοι

- Repl (const std::string &history, const std::string &prompt, size_t nBits=60, size_t nHashes=10)

  Construct the REPL object.
- void run ()

  Do the read, eval, print loop.
- void register_cmd (const std::string &cmd, CMD_PTR ptr, const std::string &helpText, const color &color1=color::WHITE)

  Register a command.
- bool eval_line (const std::string &line, bool doHistory=true)

  Evaluate a line.
- bool eval_cmd_line (const std::string &line, bool doHistory=true)

  Evaluate a command line.
- bool help_cmd (const std::string &input)

  Implementaion of '!help' command.
- bool history_cmd (const std::string &input)

  Implementaion of '!history' command.
- bool quit_cmd (const std::string &input)

  Implementaion of '!quit' command.
- bool init_cmd (const std::string &input)

  Implementaion of 'init' command.
- bool design_cmd (const std::string &input)

  Implementaion of 'design' command.
- bool stats_cmd (const std::string &input)

  Implementaion of 'stats' command.
- bool add_cmd (const std::string &input)

  Implementaion of 'add' command.
- bool verbose_cmd (const std::string &input)

  Implementaion of 'verbose' command.
- bool bits_cmd (const std::string &input)

  Implementaion of 'bits' command.
- bool import_cmd (const std::string &input)

  Implementaion of 'import' command.
- bool check_cmd (const std::string &input)

  Implementaion of 'check' command.
- bool nbits_cmd (const std::string &input)

  Implementaion of 'nbits' command.
- bool run_cmd (const std::string &input)

  Implementaion of 'run' command.
- bool **set_verbose_mode** ()
- const std::string & **getPrompt** () const
- void **setPrompt** (const std::string &prompt)
- bool **do_check_cmd** (const std::vector< std::string > &tokens)
- bool **do_run_cmd** (const std::vector< std::string > &tokens)
- bool **run_script** (const std::string &filename)

6.4.1 Λεπτομερής Περιγραφή

Ορισμός στη γραμμή 21 του αρχείου repl.h.

6.4.2 Τεκμηρίωση **Constructor & Destructor**

6.4.2.1 Repl()

```
Repl::Repl (
            const std::string & history,
            const std::string & prompt,
            size_t nBits = 60,
            size_t nHashes = 10 )  [explicit]
```

Construct the REPL object.

Παράμετροι

history	The history file
prompt	The prompt string
nBits	Number of bit for the initial bloom filter
nHashes	Number of hashed for the initial bloom filter

Ορισμός στη γραμμή 48 του αρχείου repl.cpp.

6.4.3 Τεκμηρίωση Συναρτήσεων Μελών

6.4.3.1 eval_cmd_line()

```
bool Repl::eval_cmd_line (
            const std::string & line,
            bool doHistory = true )
```

Evaluate a command line.

Find and execute the command. return the exit status to caller.

Παράμετροι

line	The input line

Επιστρέφει

false on quit commands

Ορισμός στη γραμμή 130 του αρχείου repl.cpp.

6.4.3.2 eval_line()

```
bool Repl::eval_line (
            const std::string & line,
            bool doHistory = true )
```

Evaluate a line.

Ignore comments. If the line contains a command, then run the command otherwise analyze text as if command 'check' is given

Παράμετροι

line	

Επιστρέφει

Ορισμός στη γραμμή 97 του αρχείου repl.cpp.

6.4.3.3 register_cmd()

```
void Repl::register_cmd (
            const std::string & cmd,
            CMD_PTR ptr,
            const std::string & helpText,
            const color & color1 = color::WHITE )
```

Register a command.

A lookup table with pointers to member functions as keys. This is tricky C++ Also register a color and a help text.

Παράμετροι

cmd	The name of the command (without '!')
ptr	The internal pointer to a mamber function of signature CMD_PTR
helpText	The help text
color1	The command color

Ορισμός στη γραμμή 83 του αρχείου repl.cpp.

Η τεκμηρίωση για αυτή την κλάση δημιουργήθηκε από τα ακόλουθα αρχεία:

- src/repl.h
- src/repl.cpp

Chapter 7

Τεκμηρίωση Αρχείων

7.1 Τεκμηρίωση Αρχείου **CMakeLists.dox**

Συστημα building and Configuration.

7.1.1 Λεπτομερής Περιγραφή

Συστημα building and Configuration.

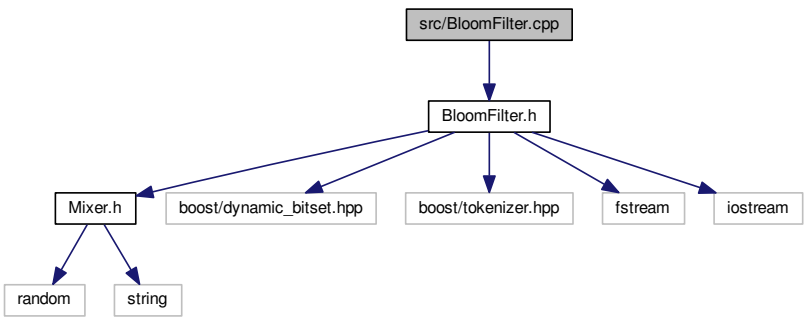
Ορισμός στο αρχείο CMakeLists.dox.

7.2 Τεκμηρίωση Αρχείου **src/BloomFilter.cpp**

Υλοποίηση της κλάσης BloomFilter.

```
#include "BloomFilter.h"
```

Διάγραμμα εξάρτησης αρχείου συμπερίληψης για το BloomFilter.cpp:



7.2.1 Λεπτομερής Περιγραφή

Υλοποίηση της κλάσης BloomFilter.

Ορισμός στο αρχείο BloomFilter.cpp.

7.3 BloomFilter.cpp

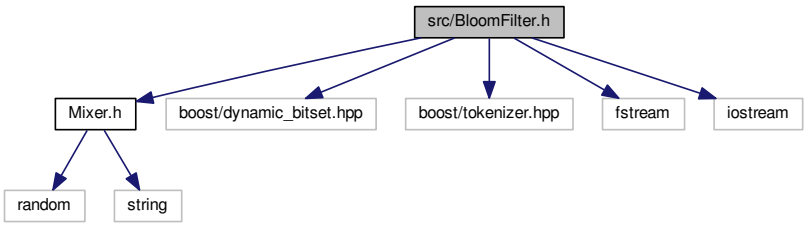
```
00001
00005 #include "BloomFilter.h"
00006
00007 void BloomFilter::populate(const std::string &fname) {
00008   std::ifstream in(fname);
00009   if (!in.is_open())
00010     throw std::invalid_argument("Can't read file contents.");
00011
00012   std::string line;
00013   while (getline(in, line)) {
00014     Tokeniizer tok(line);
00015     for (auto const &word : tok) {
00016       if (notHave(word)) {
00017         add(word);
00018         count++;
00019       }
00020     }
00021   }
00022 }
00023
00024 double BloomFilter::estimateP(double N) {
00025   double k = mNumHashes;
00026   double m = mSize;
00027   double ek = -1.0 * k * N / m;
00028   double b = (1 - exp(ek));
00029   double res = pow(b, k);
00030   return res;
00031 }
00032
00033 double BloomFilter::estimateP() { return       BloomFilter::estimateP(count); }
00034
00035 const dynamic_bitset<> &BloomFilter::getMBits() const { return mBits; }
00036
00037 BloomFilter::BloomFilter(size_t nwords, double P) : count{0} {
00038   double m = ceil(-1.0 * nwords * log(P) / pow(log(2), 2));
00039   double k = ceil((m / nwords) * log(2));
00040   mBits = boost::dynamic_bitset<>{static_cast<size_t>(m)};
00041   mSize = static_cast<size_t>(m);
00042   mNumHashes = static_cast<size_t>(k);
00043 }
00044
00045 void BloomFilter::add(const std::string &key) {
00046   auto mixer = Mixer(key, mSize);
00047   if (notHave(key)) {
00048     for (auto i = 0; i < mNumHashes; i++) {
00049       mBits.set(mixer());
00050     }
00051     count++;
00052     if (verbose) {
00053       std::cout « "added '" « key « "'\n";
00054     }
00055   } else {
00056     if (verbose) {
00057       std::cout « "skipped '" « key « "'\n";
00058     }
00059   }
00060 }
00061
00062 bool BloomFilter::notHave(const std::string &key) {
00063   auto mixer = Mixer(key, mSize);
00064   bool res = true;
00065   for (auto i = 0; i < mNumHashes; i++) {
00066     res &= mBits[mixer()];
00067   }
00068   return !res;
00069 }
```

7.4 Τεκμηρίωση Αρχείου src/BloomFilter.h

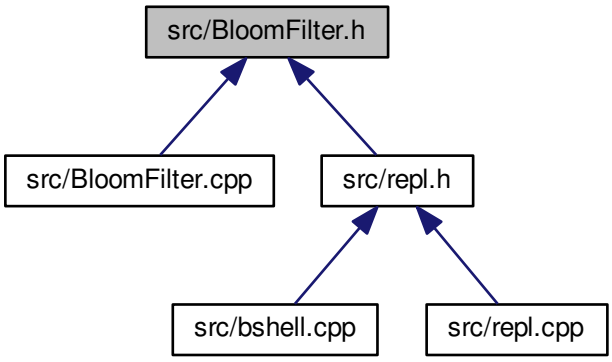
Ορισμός της κλάσης BloomFilter.

```
#include "Mixer.h"
#include <boost/dynamic_bitset.hpp>
#include <boost/tokenizer.hpp>
#include <fstream>
#include <iostream>
```

Διάγραμμα εξάρτησης αρχείου συμπερίληψης για το BloomFilter.h:



Το διάγραμμα αυτό παρουσιάζει ποιά αρχεία άμεσα ή έμμεσα περιλαμβάνουν αυτό το αρχείο:



Κλάσεις

- class BloomFilter

  Υλοποιηση ενός απλού φίλτρου *Bloom.*

Ορισμοί Τύπων

- using **Tokeniizer** = boost::tokenizer<>

7.4.1 Λεπτομερής Περιγραφή

Ορισμός της κλάσης BloomFilter.

Ορισμός στο αρχείο BloomFilter.h.

7.5 BloomFilter.h

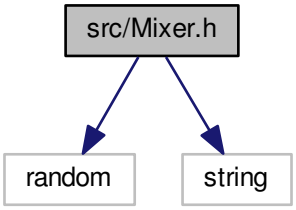
```
00001 #ifndef BLOOM_BLOOMFILTER_H
00002 #define BLOOM_BLOOMFILTER_H
00003
00008 #include "Mixer.h"
00009 #include <boost/dynamic_bitset.hpp>
00010 #include <boost/tokenizer.hpp>
00011 #include <fstream>
00012 #include <iostream>
00013
00014 using Tokeniizer = boost::tokenizer<>;
00015 using boost::dynamic_bitset;
00016
00028 class BloomFilter {
00029 public:
00035   explicit BloomFilter(size_t size = 80, size_t numHashes = 10)
00036       : mBits{size}, mSize{size}, mNumHashes{numHashes}, count{0} {}
00037
00043   explicit BloomFilter(size_t nwords = 1000, double P = 0.01);
00044
00049   void add(const std::string &key);
00050
00056   bool notHave(const std::string &key);
00057
00065   float fullness() const {
00066     return static_cast<float>(mBits.count()) / static_cast<float>(mBits.size());
00067   }
00068
00077   void populate(const std::string &fname);
00078
00083   size_t getSize() const { return mSize; }
00084
00089   size_t getNumHashes() const { return mNumHashes; }
00090
00095   size_t getCount() const { return count; }
00096
00101   void setVerbose(bool verbose) { BloomFilter::verbose = verbose; }
00102
00107   double estimateP();
00108
00114   double estimateP(double N);
00115
00116   const dynamic_bitset<> &getMBits() const;
00117
00118 private:
00119   dynamic_bitset<> mBits;
00120   size_t mSize;
00121   size_t mNumHashes;
00122   size_t count;
00123 #ifndef NDEBUG
00124   bool verbose = true;
00125 #else
00126   bool verbose = false;
00127 #endif
00128 };
00129
00130 #endif // BLOOM_BLOOMFILTER_H
```

7.6 Τεκμηρίωση Αρχείου src/Mixer.h

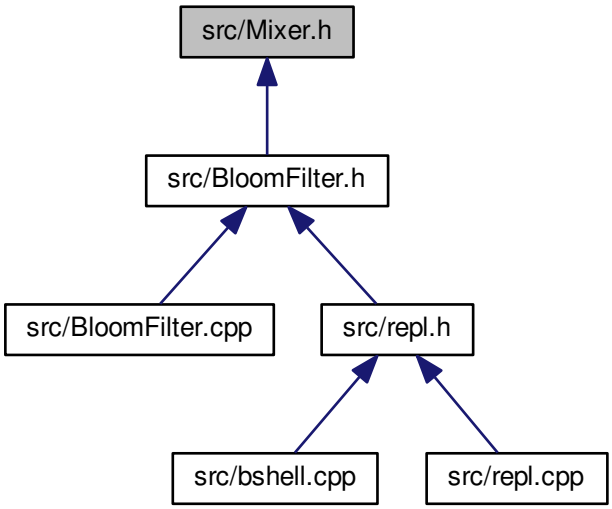
Ορισμός της κλάσης Mixer.

```
#include <random>
#include <string>
```

Διάγραμμα εξάρτησης αρχείου συμπερίληψης για το Mixer.h:



Το διάγραμμα αυτό παρουσιάζει ποιά αρχεία άμεσα ή έμμεσα περιλαμβάνουν αυτό το αρχείο:



Κλάσεις

- class Mixer

  Ένα συναρτησιακό για τον υπολογισμό των θέσεων σε ένα φίλτρο *Bloom*.

7.6.1 Λεπτομερής Περιγραφή

Ορισμός της κλάσης Mixer.

Ορισμός στο αρχείο Mixer.h.

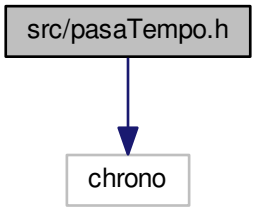
7.7 Mixer.h

```
00001 #ifndef BLOOM_MIXER_H
00002 #define BLOOM_MIXER_H
00003
00008 #include <random>
00009 #include <string>
00010
00034 class Mixer {
00035 public:
00046   Mixer(const std::string &val, size_t size) : mSize(size) {
00047     // Generate the hash value of the given key
00048     auto hashVal = mHashFn(val);
00049     // Initialize random number generator.
00050     mRandomFn = std::minstd_rand{hashVal};
00051   }
00052
00061   std::size_t operator()() { return mRandomFn() % mSize; }
00062
00063 private:
00064   size_t mSize{};                   // The max size
00065   std::minstd_rand mRandomFn{};     // The random number generator
00066   std::hash<std::string> mHashFn{}; // The hash function
00067 };
00068
00069 #endif // BLOOM_MIXER_H
```

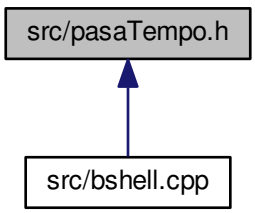
7.8 Τεκμηρίωση Αρχείου src/pasaTempo.h

Ορισμός της κλάσης pasaTempo.

`#include <chrono>`
Διάγραμμα εξάρτησης αρχείου συμπερίληψης για το pasaTempo.h:



Το διάγραμμα αυτό παρουσιάζει ποιά αρχεία άμεσα ή έμμεσα περιλαμβάνουν αυτό το αρχείο:



Κλάσεις

- class pasaTempo

  A simple class to measure timings in ns.

Ορισμοί Τύπων

- using **timePointMS** = time_point< high_resolution_clock >

7.8.1 Λεπτομερής Περιγραφή

Ορισμός της κλάσης pasaTempo.

Ορισμός στο αρχείο pasaTempo.h.

7.9 pasaTempo.h

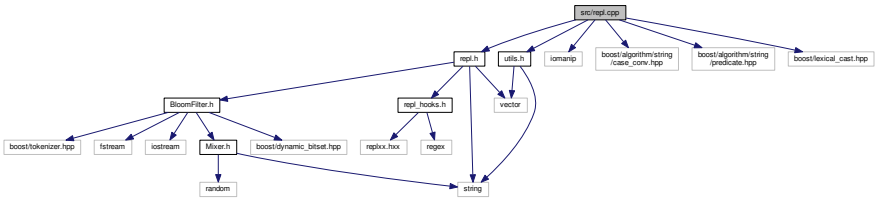
```
00001 #ifndef BLOOM_PASATEMPO_H
00002 #define BLOOM_PASATEMPO_H
00003
00008 #include <chrono>
00009 using std::chrono::duration_cast;
00010 using std::chrono::high_resolution_clock;
00011 using std::chrono::time_point;
00012 using timePointMS = time_point<high_resolution_clock>;
00013
00018 class pasaTempo {
00019 public:
00020   explicit pasaTempo() { start(); }
00021
00022   void start() {
00023     mStart = high_resolution_clock::now();
00024     mDuration = 0;
00025     isStopped = false;
00026   }
00027
00028   long stop() {
00029     if (!isStopped) {
00030       mEnd = high_resolution_clock::now();
00031       mDuration = duration_cast<std::chrono::milliseconds>(mEnd - mStart).count();
00032       isStopped = true;
00033     }
00034     return getMS();
00035   }
00036
00037   long getMS() { return duration_cast<std::chrono::milliseconds>(mEnd - mStart).count(); }
00038   long getNS() { return duration_cast<std::chrono::nanoseconds>(mEnd - mStart).count(); }
00039   long getSEC() { return duration_cast<std::chrono::seconds>(mEnd - mStart).count(); }
00040
00041 private:
00042   timePointMS mStart{};
00043   timePointMS mEnd{};
00044   long mDuration{};
00045   bool isStopped{};
00046 };
00047
00048 #endif // BLOOM_PASATEMPO_H
```

7.10 Τεκμηρίωση Αρχείου **src/repl.cpp**

Κλάση Repl υλοποίηση.

```
#include "repl.h"
#include <iomanip>
#include <boost/algorithm/string/case_conv.hpp>
#include <boost/algorithm/string/predicate.hpp>
#include <boost/lexical_cast.hpp>
#include "utils.h"
```

Διάγραμμα εξάρτησης αρχείου συμπερίληψης για το repl.cpp:



Ορισμοί Τύπων

- using **Replxx** = replxx::Replxx
- using **color** = Replxx::Color
- typedef bool(Repl::∗ **CMD_PTR**) (const string &)
- using **FcnMap** = std::map< string, CMD_PTR >
- using **FcnMap_pair** = std::pair< std::string, CMD_PTR >

7.10.1 Λεπτομερής Περιγραφή

Κλάση Repl υλοποίηση.

Ορισμός στο αρχείο repl.cpp.

7.11 repl.cpp

```
00001
00005 #include "repl.h"
00006
00007 #include <iomanip>
00008
00009 // Boost
00010 #include <boost/algorithm/string/case_conv.hpp>
00011 #include <boost/algorithm/string/predicate.hpp>
00012 #include <boost/lexical_cast.hpp>
00013
00014 #include "utils.h"
00015
00016 using boost::lexical_cast;
00017 using std::map;
00018 using std::pair;
00019 using std::size_t;
00020 using std::string;
00021 using std::vector;
00022
00023 using Replxx = replxx::Replxx;
00024 using color = Replxx::Color;
00025
```

```
00026 class Repl;
00027 typedef bool (Repl::*CMD_PTR)(const string &);
00028 using FcnMap = std::map<string, CMD_PTR>;
00029 using FcnMap_pair = std::pair<std::string, CMD_PTR>;
00030
00031 void Repl::run() {
00032   char const *cinput{nullptr};
00033   for (;;) {
00034     do {
00035       cinput = rx.input(prompt);
00036     } while ((cinput == nullptr) && (errno == EAGAIN));
00037
00038     if (cinput == nullptr)
00039       break;
00040
00041     // Exec the line
00042     if (!eval_line(cinput))
00043       break;
00044   } // forever
00045   rx.history_save(historyFile);
00046 }
00047
00048 Repl::Repl(const string &history, const string &prompt, size_t nBits, size_t nHashes)
00049     : prompt(prompt), historyFile{history}, bloom{nBits, nHashes} {
00050
00051   // Configure historyFile
00052   rx.history_load(history);
00053   rx.set_max_history_size(12);
00054   rx.set_max_line_size(128);
00055   rx.set_max_hint_rows(8);
00056
00057   // Configure commands
00058   register_cmd("!help", &Repl::help_cmd, "Help about commands", color::BRIGHTGREEN);
00059   register_cmd("!history", &Repl::history_cmd, "Show the historyFile", color::BRIGHTGREEN);
00060   register_cmd("!exit", &Repl::quit_cmd, "Exit program", color::BRIGHTGREEN);
00061   register_cmd("!quit", &Repl::quit_cmd, "Exit program", color::BRIGHTGREEN);
00062   register_cmd("!verbose", &Repl::verbose_cmd, "Set verbose mode", color::BRIGHTGREEN);
00063   register_cmd("!init", &Repl::init_cmd, "(Re)Initialize Bloom filter", color::BRIGHTGREEN);
00064   register_cmd("!design", &Repl::design_cmd, "Design Bloom filter", color::BRIGHTBLUE);
00065   register_cmd("!stats", &Repl::stats_cmd, "Bloom filter statistics", color::BRIGHTGREEN);
00066   register_cmd("!add", &Repl::add_cmd, "\tAdd words to bloom filter", color::BRIGHTGREEN);
00067   register_cmd("!bits", &Repl::bits_cmd, "Show bloom filter bits", color::BRIGHTGREEN);
00068   register_cmd("!import", &Repl::import_cmd, "Import words from files", color::BRIGHTGREEN);
00069   register_cmd("!check", &Repl::check_cmd, "Check words", color::BRIGHTGREEN);
00070   register_cmd("!nbits", &Repl::nbits_cmd, "Debug mixer", color::BRIGHTGREEN);
00071   register_cmd("!run", &Repl::run_cmd, "\t Run script", color::BRIGHTGREEN);
00072
00073   rx.install_window_change_handler();
00074   rx.set_completion_callback(hook_completion, static_cast<void *>(&commands));
00075   rx.set_hint_callback(hook_hint, static_cast<void *>(&commands));
00076   rx.set_highlighter_callback(hook_color, static_cast<void *>(&regex_color));
00077
00078   // Verbose mode
00079   verboseMode = set_verbose_mode();
00080   bloom.setVerbose(verboseMode);
00081 }
00082
00083 void Repl::register_cmd(const string &cmd, CMD_PTR ptr, const string &helpText,
00084                         const color &color1) {
00085   // Insert command to function table.
00086   map_fcn.insert(FcnMap_pair{cmd, ptr});
00087
00088   commands.emplace_back(cmd);
00089
00090   // Register help text
00091   help_text.emplace_back(cmd, helpText);
00092
00093   // Register regexp color
00094   regex_color.emplace_back(string("\\") + cmd, color1);
00095 }
00096
00097 bool Repl::eval_line(const string &line, bool doHistory) {
00098   if (boost::starts_with(line, "#"))
00099     return true;
00100
00101   if (boost::starts_with(line, "@")) {
00102     std::vector<std::string> words = getTokens(line);
00103     if (words.empty())
00104       return true;
00105     auto histLine = words[0];
00106     histLine.replace(0, 1, "");
00107     int n = 0;
00108     try {
00109       n = lexical_cast<int>(histLine);
00110     } catch (...) {
00111       return true;
00112     }
```

```
00113     if (n - 1 < rx.history_size()) {
00114       const auto cmdHist = rx.history_line(n - 1);
00115       std::cout « cmdHist « "\n";
00116       return eval_line(cmdHist);
00117     }
00118   }
00119
00120   if (boost::starts_with(line, "!"))
00121     return eval_cmd_line(line, true);
00122
00123   // fallback to check command
00124   std::vector<std::string> tokens = getTokens(line, fullTokens, true);
00125   do_check_cmd(tokens);
00126
00127   return true;
00128 }
00129
00130 bool Repl::eval_cmd_line(const string &line, bool doHistory) {
00131   std::vector<std::string> words = getTokens(line, fullTokens);
00132   if (words.empty())
00133     return true;
00134
00135   auto cmd = words[0];
00136   auto it = map_fcn.find(cmd);
00137   if (it != map_fcn.end()) {
00138     if (doHistory) {
00139       rx.history_add(line);
00140     }
00141     CMD_PTR commandFn = it->second;
00142     return (this->*commandFn)(line);
00143   } else {
00144     reportError(string{"Bad Command : "} + cmd);
00145   }
00146   // keep going
00147   return true;
00148 }
00149
00150 bool Repl::help_cmd(const std::string &input) {
00151   std::cout « "Commands:\n";
00152   for (const auto &it : help_text) {
00153     std::cout « "    " « it.first « "\t" « it.second « "\n";
00154   }
00155   return true;
00156 }
00157
00158 bool Repl::history_cmd(const std::string &input) {
00159   for (int i = 0, sz = rx.history_size(); i < sz; ++i) {
00160     std::cout « std::setw(4) « i « ": " « rx.history_line(i) « "\n";
00161   }
00162   return true;
00163 }
00164
00165 bool Repl::quit_cmd(const std::string &input) { return false; }
00166
00167 bool Repl::init_cmd(const std::string &input) {
00168   std::vector<std::string> tokens = getTokens(input, fullTokens);
00169   bool proceed = true;
00170   if (tokens.size() < 3)
00171     proceed = false;
00172   size_t n = 0, m = 0;
00173   try {
00174     n = lexical_cast<size_t>(tokens[1]);
00175     m = lexical_cast<size_t>(tokens[2]);
00176   } catch (...) {
00177     proceed = false;
00178   }
00179   if (!proceed) {
00180     reportUsage("!init <number of bits> <number of hashes>");
00181     return true;
00182   }
00183
00184   bloom = BloomFilter(n, m);
00185   if (verboseMode) {
00186     eval_cmd_line("!stats", false);
00187   }
00188   return true;
00189 }
00190
00191 bool Repl::design_cmd(const std::string &input) {
00192   std::vector<std::string> tokens = getTokens(input, simpleTokens);
00193   size_t n = 0;
00194   double P = 0.001;
00195
00196   bool proceed = true;
00197   if (tokens.size() <= 1)
00198     proceed = false;
00199
```

```
00200   if (tokens.size() >= 2) {
00201     try {
00202       n = lexical_cast<size_t>(tokens[1]);
00203     } catch (...) {
00204       proceed = false;
00205     }
00206   }
00207
00208   if (tokens.size() >= 3) {
00209     try {
00210       auto s = tokens[2];
00211       P = stod(s);
00212     } catch (...) {
00213       proceed = false;
00214     }
00215   }
00216
00217   if (!proceed) {
00218     reportUsage("!design <word capacity> [Propability]");
00219     return true;
00220   }
00221
00222   if (P <= 0)
00223     P = 0.001;
00224
00225   bloom = BloomFilter(n, P);
00226   if (verboseMode) {
00227     eval_cmd_line("!stats", false);
00228   }
00229   return true;
00230 }
00231
00232 bool Repl::stats_cmd(const std::string &input) {
00233   std::cout « "   Bits   : " « bloom.getSize() « std::endl;
00234   std::cout « "   Memory : " « convertSize(bloom.getSize()) « std::endl;
00235   std::cout « "   Hashes : " « bloom.getNumHashes() « std::endl;
00236   std::cout « "Num items : " « bloom.getCount() « std::endl;
00237   std::cout « " Fullness : " « bloom.fullness() * 100 « "%" « std::endl;
00238   std::cout « "False Pos   " « bloom.estimateP() « std::endl;
00239   return true;
00240 }
00241
00242 bool Repl::add_cmd(const std::string &input) {
00243   std::vector<std::string> tokens = getTokens(input, fullTokens, true);
00244
00245   for (auto &token : tokens) {
00246     bloom.add(token);
00247   }
00248   return true;
00249 }
00250
00251 bool Repl::verbose_cmd(const std::string &input) {
00252   std::vector<std::string> tokens = getTokens(input, fullTokens);
00253   if (tokens.size() > 1) {
00254     auto v = tokens[1];
00255     boost::to_upper(v);
00256     verboseMode = (v == "1") || (v == "ON") || (v == "TRUE");
00257     bloom.setVerbose(verboseMode);
00258   }
00259   std::cout « "Verbose mode: " « (verboseMode ? "ON" : "OFF") « "\n";
00260
00261   return true;
00262 }
00263
00264 bool Repl::bits_cmd(const std::string &input) {
00265   const auto &bits = bloom.getMBits();
00266   std::ostringstream ss;
00267   for (auto i = 0; i < bits.size(); i++) {
00268     ss « (bits[i] ? "1" : "0");
00269   }
00270   std::cout « ss.str() « "\n";
00271   return true;
00272 }
00273
00274 bool Repl::import_cmd(const std::string &input) {
00275   std::vector<std::string> tokens = getTokens(input, simpleTokens, true);
00276
00277   for (auto &token : tokens) {
00278     if (fnameIsReadable(token)) {
00279       if (verboseMode)
00280         std::cout « (string{"Importing words form "} + token + ".\n");
00281       bloom.populate(token);
00282     } else {
00283       if (verboseMode) {
00284         reportError(string{"Filename :"} + token + " is not readable");
00285       }
00286     }
```

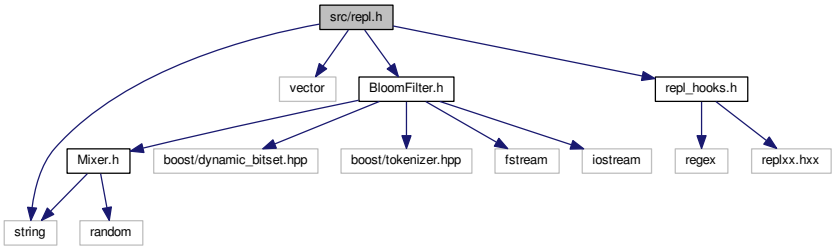
```
00287   }
00288   if (verboseMode) {
00289     eval_cmd_line("!stats", false);
00290   }
00291   return true;
00292 }
00293
00294 bool Repl::check_cmd(const std::string &input) {
00295   std::vector<std::string> tokens = getTokens(input, fullTokens, true);
00296   return do_check_cmd(tokens);
00297 }
00298
00299 bool Repl::do_check_cmd(const vector<string> &tokens) {
00300   vector<string> bad_words;
00301
00302   for (auto &word : tokens) {
00303     if (!bloom.notHave(word)) {
00304       bad_words.emplace_back(word);
00305     }
00306   }
00307
00308   if (!bad_words.empty()) {
00309     auto txt = toStringList(bad_words, ",", "'", "\n");
00310     std::cout « "Positives: ";
00311     std::cout « txt;
00312   } else {
00313     if (verboseMode) {
00314       std::cout « "OK\n";
00315     }
00316   }
00317   return true;
00318 }
00319
00320 bool Repl::nbits_cmd(const std::string &input) {
00321   std::vector<std::string> tokens = getTokens(input, fullTokens, true);
00322
00323   for (const auto &token : tokens) {
00324     auto m = Mixer{token, bloom.getSize()};
00325     std::ostringstream ss;
00326     ss « token « "\t: ";
00327     for (auto i = 0; i < bloom.getNumHashes(); i++) {
00328       ss « m() « " ";
00329     }
00330     std::cout « ss.str() « "\n";
00331   }
00332
00333   return true;
00334 }
00335
00336 bool Repl::run_cmd(const std::string &input) {
00337   std::vector<std::string> tokens = getTokens(input, simpleTokens, true);
00338   return do_run_cmd(tokens);
00339 }
00340
00341 bool Repl::do_run_cmd(const std::vector<std::string> &tokens) {
00342   for (const auto &token : tokens) {
00343     auto res = run_script(token);
00344     if (res == false)
00345       return false;
00346   }
00347   return true;
00348 }
00349
00350 bool Repl::run_script(const string &filename) {
00351   if (fnameIsReadable(filename)) {
00352     std::ifstream in(filename);
00353     if (verboseMode) {
00354       std::cout « (string{"Execute script from :"} + filename + ".\n");
00355     }
00356     string line;
00357     while (getline(in, line)) {
00358       if (verboseMode) {
00359         // std::cout « prompt « " ";
00360         std::cout « line « "\n";
00361       }
00362       auto res = eval_line(line, false);
00363       // handle quit
00364       if (res == false)
00365         return false;
00366     }
00367   } else {
00368     if (verboseMode) {
00369       reportError(string{"Filename "} + filename + " is not readable");
00370     }
00371   }
00372   return true;
00373 }
```

7.12 Τεκμηρίωση Αρχείου **src/repl.h**

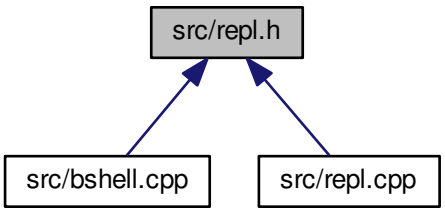
Κλάση Pepl ορισμοί.

```
#include <string>
#include <vector>
#include "BloomFilter.h"
#include "repl_hooks.h"
```

Διάγραμμα εξάρτησης αρχείου συμπερίληψης για το repl.h:



Το διάγραμμα αυτό παρουσιάζει ποιά αρχεία άμεσα ή έμμεσα περιλαμβάνουν αυτό το αρχείο:



Κλάσεις

- class Repl

Ορισμοί Τύπων

- using **Replxx** = replxx::Replxx
- using **color** = Replxx::Color
- typedef bool(Repl::∗ **CMD_PTR**) (const std::string &)
- using **FcnMap** = std::map< std::string, CMD_PTR >
- using **FcnMap_pair** = std::pair< std::string, CMD_PTR >

7.12.1 Λεπτομερής Περιγραφή

Κλάση Repl ορισμοί.

Ορισμός στο αρχείο repl.h.

7.13 repl.h

```
00001
00004 #ifndef BLOOM_REPL_H
00005 #define BLOOM_REPL_H
00006
00007 #include <string>
00008 #include <vector>
00009
00010 #include "BloomFilter.h"
00011 #include "repl_hooks.h"
00012
00013 using Replxx = replxx::Replxx;
00014 using color = Replxx::Color;
00015
00016 class Repl;
00017 typedef bool (Repl::*CMD_PTR)(const std::string &);
00018 using FcnMap = std::map<std::string, CMD_PTR>;
00019 using FcnMap_pair = std::pair<std::string, CMD_PTR>;
00020
00021 class Repl {
00022 private:
00023   std::string prompt;
00024   std::string historyFile;
00025   bool verboseMode;
00026
00027   BloomFilter bloom;
00028
00029   // Repl UI
00030   Replxx rx;
00031   FcnMap map_fcn;
00032
00033   std::vector<std::string> commands;
00034   std::vector<std::pair<std::string, color» regex_color;
00035   std::vector<std::pair<std::string, std::string» help_text;
00036
00037 public:
00045   explicit Repl(const std::string &history, const std::string &prompt, size_t nBits = 60,
00046                 size_t nHashes = 10);
00047
00051   void run();
00052
00065   void register_cmd(const std::string &cmd, CMD_PTR ptr, const std::string &helpText,
00066                     const color &color1 = color::WHITE);
00067
00077   bool eval_line(const std::string &line, bool doHistory = true);
00078
00087   bool eval_cmd_line(const std::string &line, bool doHistory = true);
00088
00092   bool help_cmd(const std::string &input);
00093
00097   bool history_cmd(const std::string &input);
00098
00102   bool quit_cmd(const std::string &input);
00103
00107   bool init_cmd(const std::string &input);
00108
00112   bool design_cmd(const std::string &input);
00113
00117   bool stats_cmd(const std::string &input);
00118
00122   bool add_cmd(const std::string &input);
00123
00127   bool verbose_cmd(const std::string &input);
00128
00132   bool bits_cmd(const std::string &input);
00133
00137   bool import_cmd(const std::string &input);
00138
00142   bool check_cmd(const std::string &input);
00143
00147   bool nbits_cmd(const std::string &input);
00148
00152   bool run_cmd(const std::string &input);
```

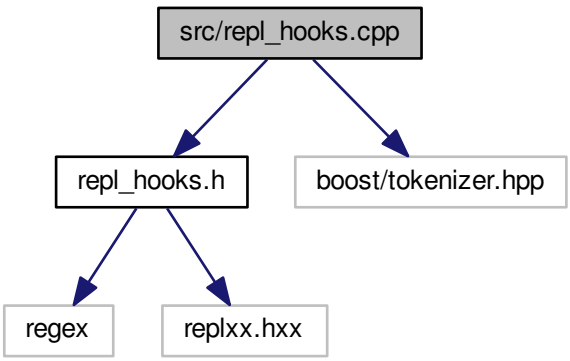
```
00153
00154   bool set_verbose_mode() {
00155 #ifndef NDEBUG
00156     return true;
00157 #else
00158     return false;
00159 #endif
00160   }
00161
00162   const std::string &getPrompt() const { return prompt; }
00163   void setPrompt(const std::string &prompt) { Repl::prompt = prompt; }
00164
00165   bool do_check_cmd(const std::vector<std::string> &tokens);
00166   bool do_run_cmd(const std::vector<std::string> &tokens);
00167
00168   bool run_script(const std::string &filename);
00169 };
00170
00171 #endif // BLOOM_REPL_H
```

7.14 Τεκμηρίωση Αρχείου src/repl_hooks.cpp

Υποσύστημα repl.

```
#include "repl_hooks.h"
#include <boost/tokenizer.hpp>
```

Διάγραμμα εξάρτησης αρχείου συμπερίληψης για το repl_hooks.cpp:



Ορισμοί Τύπων

- using **Tokeniizer** = boost::tokenizer<>

Συναρτήσεις

- Replxx::completions_t **hook_completion** (std::string const &context, int index, void *user_data)
- Replxx::hints_t **hook_hint** (std::string const &context, int index, Replxx::Color &color, void *user_data)
- void **hook_color** (std::string const &context, Replxx::colors_t &colors, void *user_data)

7.14.1 Λεπτομερής Περιγραφή

Υποσύστημα repl.

Ορισμός στο αρχείο repl_hooks.cpp.

7.15 repl_hooks.cpp

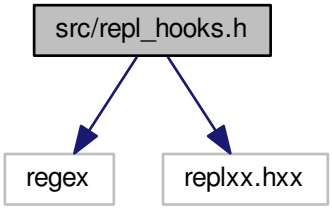
```
00001
00004 #include "repl_hooks.h"
00005 #include <boost/tokenizer.hpp>
00006
00007 using Tokeniizer = boost::tokenizer<>;
00008
00009 Replxx::completions_t hook_completion(std::string const &context, int index, void *user_data) {
00010   auto *examples = static_cast<std::vector<std::string> *>(user_data);
00011   Replxx::completions_t completions;
00012
00013   std::string prefix{context.substr(index)};
00014   for (auto const &e : *examples) {
00015     if (e.compare(0, prefix.size(), prefix) == 0) {
00016       completions.emplace_back(e.c_str());
00017     }
00018   }
00019
00020   return completions;
00021 }
00022
00023 Replxx::hints_t hook_hint(std::string const &context, int index, Replxx::Color &color,
00024                           void *user_data) {
00025   auto *examples = static_cast<std::vector<std::string> *>(user_data);
00026   Replxx::hints_t hints;
00027
00028   // only show hint if prefix is at least 'n' chars long
00029   // or if prefix begins with a specific character
00030   std::string prefix{context.substr(static_cast<unsigned long>(index))};
00031   if (prefix.size() >= 2 || (!prefix.empty() && prefix.at(0) == '.')) {
00032     for (auto const &e : *examples) {
00033       if (e.compare(0, prefix.size(), prefix) == 0) {
00034         hints.emplace_back(e.substr(prefix.size()).c_str());
00035       }
00036     }
00037   }
00038
00039   // set hint color to green if single match found
00040   if (hints.size() == 1) {
00041     color = Replxx::Color::GREEN;
00042   }
00043
00044   return hints;
00045 }
00046
00047 void hook_color(std::string const &context, Replxx::colors_t &colors, void *user_data) {
00048   auto *regex_color = static_cast<std::vector<std::pair<std::string, Replxx::Color>> *>(user_data);
00049
00050   // highlight matching regex sequences
00051   for (auto const &e : *regex_color) {
00052     size_t pos{0};
00053     std::string str = context;
00054     std::smatch match;
00055
00056     while (std::regex_search(str, match, std::regex(e.first))) {
00057       std::string c{match[0]};
00058       pos += std::string(match.prefix()).size();
00059
00060       for (size_t i = 0; i < c.size(); ++i) {
00061         colors.at(pos + i) = e.second;
00062       }
00063
00064       pos += c.size();
00065       str = match.suffix();
00066     }
00067   }
00068 }
```

7.16 Τεκμηρίωση Αρχείου src/repl_hooks.h

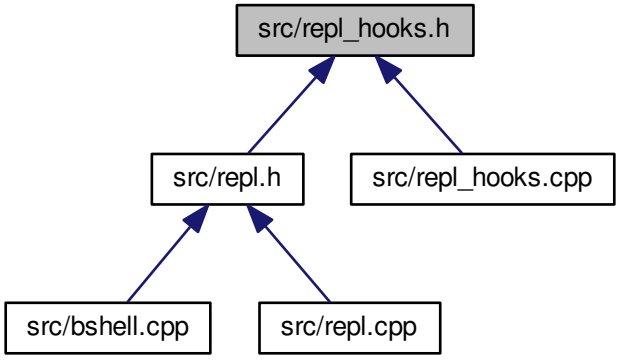
Υποσύστημα repl.

```
#include <regex>
#include <replxx.hxx>
```

Διάγραμμα εξάρτησης αρχείου συμπερίληψης για το repl_hooks.h:



Το διάγραμμα αυτό παρουσιάζει ποιά αρχεία άμεσα ή έμμεσα περιλαμβάνουν αυτό το αρχείο:



Ορισμοί Τύπων

- using **Replxx** = replxx::Replxx

Συναρτήσεις

- Replxx::completions_t **hook_completion** (std::string const &context, int index, void ∗user_data)
- Replxx::hints_t **hook_hint** (std::string const &context, int index, Replxx::Color &color, void ∗user_data)
- void **hook_color** (std::string const &str, Replxx::colors_t &colors, void ∗user_data)

7.16.1 Λεπτομερής Περιγραφή

Υποσύστημα repl.

Ορισμός στο αρχείο repl_hooks.h.

7.17 repl_hooks.h

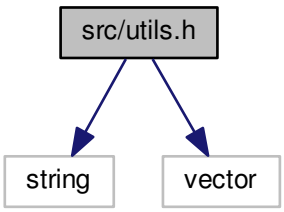
```
00001
00005 #ifndef BLOOM_REPL_HOOKS_H
00006 #define BLOOM_REPL_HOOKS_H
00007
00008 #include <regex>
00009 #include <replxx.hxx>
00010 using Replxx = replxx::Replxx;
00011
00012 // prototypes
00013 Replxx::completions_t hook_completion(std::string const &context, int index, void *user_data);
00014 Replxx::hints_t hook_hint(std::string const &context, int index, Replxx::Color &color,
00015                           void *user_data);
00016 void hook_color(std::string const &str, Replxx::colors_t &colors, void *user_data);
00017
00018 #endif // BLOOM_REPL_HOOKS_H
```

7.18 Τεκμηρίωση Αρχείου src/utils.h

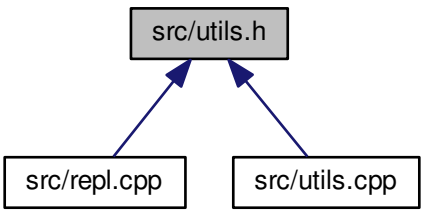
Βοηθητικές συναρτήσεις.

```
#include <string>
#include <vector>
```

Διάγραμμα εξάρτησης αρχείου συμπερίληψης για το utils.h:



Το διάγραμμα αυτό παρουσιάζει ποιά αρχεία άμεσα ή έμμεσα περιλαμβάνουν αυτό το αρχείο:



Συναρτήσεις

- std::vector< std::string > **getTokens** (const std::string &input, const std::string &splitChars=fullTokens, bool skip_first=false)
- std::string **toStringList** (const std::vector< std::string > &data, const std::string &separator=", ", const std::string &quoter="'", const std::string &concluder="")
- bool **fnameIsReadable** (const std::string &filename)
- bool **fnameIsWriteable** (const std::string &filename)
- void **reportError** (const std::string &error)
- void **reportUsage** (const std::string &text)
- std::string **convertSize** (size_t size)

Μεταβλητές

- const char * **fullTokens**
- const char * **simpleTokens**

7.18.1 Λεπτομερής Περιγραφή

Βοηθητικές συναρτήσεις.

Ορισμός στο αρχείο utils.h.

7.19 utils.h

```
00001 #ifndef BLOOM_UTILS_H
00002 #define BLOOM_UTILS_H
00003
00007 #include <string>
00008 #include <vector>
00009
00010 extern const char *fullTokens;
00011 extern const char *simpleTokens;
00012
00013 std::vector<std::string> getTokens(const std::string &input,
00014                                    const std::string &splitChars = fullTokens,
00015                                    bool skip_first = false);
00016
00017 std::string toStringList(const std::vector<std::string> &data, const std::string &separator = ", ",
00018                          const std::string &quoter = "'", const std::string &concluder = "");
00019
00020 bool fnameIsReadable(const std::string &filename);
00021 bool fnameIsWriteable(const std::string &filename);
00022
00023 void reportError(const std::string &error);
00024 void reportUsage(const std::string &text);
00025
00026 std::string convertSize(size_t size);
00027 #endif // BLOOM_UTILS_H
```

Chapter 8

Τεκμηρίωση Παραδειγμάτων

8.1 examples/mixer_demo.cpp

```
#include "Mixer.h"
#include <iostream>
#include <string>
int main() {
  auto text1 = std::string{"Viagra"};
  auto mixer1 = Mixer{text1, 10};
  std::cout « text1 « ": " « mixer1() « " " « mixer1() « " " « mixer1()
            « std::endl;
  auto text2 = std::string{"Viagra"};
  auto mixer2 = Mixer{text1, 10};
  std::cout « text2 « ": " « mixer2() « " " « mixer2() « " " « mixer2()
            « std::endl;
  auto text3 = std::string{"Anarchy"};
  auto mixer3 = Mixer{text3, 10};
  std::cout « text3 « ": " « mixer3() « " " « mixer3() « " " « mixer3()
            « std::endl;
}
```

8.2 examples/size_demo.cpp

```
#include "BloomFilter.h"
#include <iostream>
void test_word(const BloomFilter &bloom, const std::string &key) {
  std::cout « "Testing " « key « " : ";
  auto res = (bloom.maybeHave(key) ? "maybe yes" : "NO");
  std::cout « res « std::endl;
}
int main() {
  auto bloom1 = BloomFilter{100, 5};
  bloom1.populate("../data/cia_words.txt");
  test_word(bloom1, "Linux");
  test_word(bloom1, "Amiga");
  test_word(bloom1, "Windows");
  auto bloom2 = BloomFilter{100, 5};
  bloom1.populate("../data/cia_words.txt");
  test_word(bloom2, "Linux");
  test_word(bloom2, "Amiga");
  test_word(bloom2, "Windows");
}
```

Index

add
    BloomFilter, 13

BloomFilter, 11
    add, 13
    BloomFilter, 12, 13
    estimateP, 13
    fullness, 14
    getCount, 14
    getNumHashes, 14
    getSize, 15
    notHave, 15
    populate, 15
    setVerbose, 16

CMakeLists.dox, 23

estimateP
    BloomFilter, 13
eval_cmd_line
    Repl, 21
eval_line
    Repl, 21

fullness
    BloomFilter, 14

getCount
    BloomFilter, 14
getNumHashes
    BloomFilter, 14
getSize
    BloomFilter, 15

Mixer, 16
    Mixer, 17
    operator(), 18

notHave
    BloomFilter, 15

operator()
    Mixer, 18

pasaTempo, 18
populate
    BloomFilter, 15

register_cmd
    Repl, 22
Repl, 19
    eval_cmd_line, 21
    eval_line, 21
    register_cmd, 22
    Repl, 21

setVerbose
    BloomFilter, 16
src/BloomFilter.cpp, 23, 24
src/BloomFilter.h, 24, 26
src/Mixer.h, 26, 28
src/pasaTempo.h, 28, 29
src/repl.cpp, 30
src/repl.h, 35, 36
src/repl_hooks.cpp, 37, 38
src/repl_hooks.h, 38, 40
src/utils.h, 40, 41