

Median Task

Generated by Doxygen 1.8.16

1 Median	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 Median Class Reference	9
5.1.1 Detailed Description	9
5.2 MedianMultisetAdvance Class Reference	10
5.2.1 Detailed Description	10
5.3 MedianMultisetIterator Class Reference	11
5.3.1 Detailed Description	11
5.4 MedianNthElement Class Reference	12
5.4.1 Detailed Description	12
5.5 MedianRBTree Class Reference	13
5.5.1 Detailed Description	13
5.6 MedianVector Class Reference	14
5.6.1 Detailed Description	14
5.7 MedianVectorLBound Class Reference	15
5.7.1 Detailed Description	15
5.8 RBTree::Node Class Reference	15
5.9 RBTree Class Reference	16
6 File Documentation	17
6.1 Median.h File Reference	17
6.1.1 Detailed Description	17
6.2 PerformanceTest.h File Reference	18
6.2.1 Detailed Description	18
6.2.2 Function Documentation	18
6.2.2.1 performanceTestAllRandom()	18
6.2.2.2 performanceTestAllRepeating()	19
6.2.2.3 performanceTestAllWorst()	19
6.3 RBTree.h File Reference	19
6.3.1 Detailed Description	20
6.4 UnitTests.h File Reference	20
6.4.1 Detailed Description	20
Index	21

Chapter 1

Median

Example project with several implementations of median calculation.

Compiled with VS2017 or later (with loading CMakeLists.txt from File->Open->CMake..).

This will build a console application that runs Unit Tests for several containers in Debug and several Performance tests in Release.

TODO

- add more documentation
- port and build with GCC for Linux (x86 and x64)

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Median	9
MedianMultisetAdvance	10
MedianMultisetIterator	11
MedianNthElement	12
MedianRBTree	13
MedianVector	14
MedianVectorLBound	15
RBTree::Node	15
RBTree	16

Chapter 3

Class Index

3.1 Class List

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

Median	
Median class	9
MedianMultisetAdvance	
MedianMultisetAdvance finds median using std::multiset and std::advance	10
MedianMultisetIterator	
MedianMultisetIterator finds median using std::multiset and saved iterator to the middle element	11
MedianNthElement	
MedianVector finds median using std::vector > and std::nth_element	12
MedianRBTREE	
MedianRBTREE finds median using RBTREE	13
MedianVector	
MedianVector finds median using std::vector and std::sort	14
MedianVectorLBound	
MedianVectorLBound calculate median using std::vector and std::lower_bound	15
RBTREE::Node	15
RBTREE	16

Chapter 4

File Index

4.1 File List

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

Median.h	Main declaration header for Median Interface and several Median implemntations	17
PerformanceTest.h	Performance tests for the Median Class	18
RBTree.h	Multiset implementation of a Red-Black tree	19
UnitTests.h	Provides unit tests for Median Class	20

Chapter 5

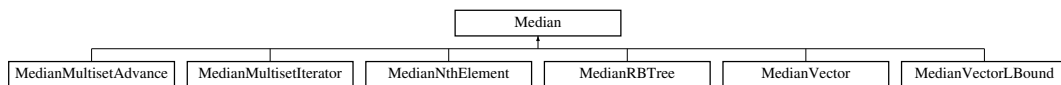
Class Documentation

5.1 Median Class Reference

[Median](#) class.

```
#include <Median.h>
```

Inheritance diagram for Median:



Public Member Functions

- virtual bool [add](#) (int a)=0
add new integer element
- virtual int [middle](#) () const =0
return value of the middle element
- virtual size_t [size](#) () const =0
return number of all elements
- virtual void [clear](#) ()=0
empty container

5.1.1 Detailed Description

[Median](#) class.

[Median](#) calculation class interface

All [Median](#) Calculations classes needs to support [Median](#) Units Tests platform and should inherit [Median](#) interface and implement its virtual methods.

Purpose of [Median](#) and the derived classes is to find the **middle** element. By **middle** it is understood the sorted in ascending order element exactly at position $N/2$, where N is the number of all added elements, if N is odd. If N is even, the **middle** element is at position $N/2-1$. This is shown below:

Odd - 1,2,3 N=3, **middle** is 2, at position 1
 Even - 1,2,3,4 N=4, **middle** is 2 also at position 1
 Odd - 1,2,3,4,5 N=5, **middle** is 3 at position 2

[Median](#) classes implementations were tested with units test (see [UnitTests.h](#)).

Performance tests comparing the algorithms are run in Release (see [PerformanceTest.h](#)).

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

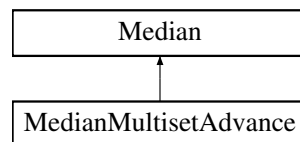
- [Median.h](#)

5.2 MedianMultisetAdvance Class Reference

[MedianMultisetAdvance](#) finds median using **std::multiset** and **std::advance**

```
#include <Median.h>
```

Inheritance diagram for MedianMultisetAdvance:



Public Member Functions

- virtual bool [add](#) (int a)
add new integer element
- virtual int [middle](#) () const
return value of the middle element
- virtual size_t [size](#) () const
return number of all elements
- virtual void [clear](#) ()
empty container

Protected Attributes

- std::multiset< int > [m_set](#)
multiset used for storage of the sorted elements

5.2.1 Detailed Description

[MedianMultisetAdvance](#) finds median using **std::multiset** and **std::advance**

[MedianMultisetAdvance](#) uses for storage container **std::multiset** all values are sorted by default in ascending order. Insert is done with the standard insert() routine of the multiset, obtaining then the middle element is done with std::advance.

add() Average Complexity $O(\log n)$

middle() Average Complexity $O(n)$

The documentation for this class was generated from the following files:

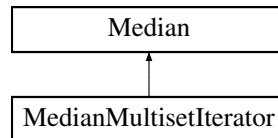
- [Median.h](#)
- [Median.cpp](#)

5.3 MedianMultisetIterator Class Reference

[MedianMultisetIterator](#) finds median using `std::multiset` and saved iterator to the middle element.

```
#include <Median.h>
```

Inheritance diagram for MedianMultisetIterator:



Public Member Functions

- virtual bool [add](#) (int a)
add new integer element
- virtual int [middle](#) () const
return value of the middle element
- virtual size_t [size](#) () const
return number of all elements
- virtual void [clear](#) ()
empty container

Protected Attributes

- std::multiset< int > [m_set](#)
vector used for storage of the sorted elements
- std::multiset< int >::iterator [m_it](#)
iterator to the middle element inside the set
- unsigned int [m_pos](#)
position of the middle element inside the set

5.3.1 Detailed Description

[MedianMultisetIterator](#) finds median using `std::multiset` and saved iterator to the middle element.

[MedianMultisetIterator](#) uses for storage container `std::multiset`, all values are sorted by default in ascending order. After insert [m_it](#) and [m_pos](#) variables are updated to point to the middle element.

[add\(\)](#) Average Complexity $O(\log n)$

[middle\(\)](#) Average Complexity $O(1)$

The documentation for this class was generated from the following files:

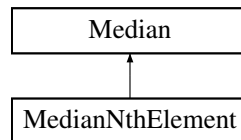
- [Median.h](#)
- Median.cpp

5.4 MedianNthElement Class Reference

[MedianVector](#) finds median using `std::vector` and `std::nth_element`

```
#include <Median.h>
```

Inheritance diagram for MedianNthElement:



Public Member Functions

- virtual bool [add](#) (int a)
add new integer element
- virtual int [middle](#) () const
return value of the middle element
- virtual size_t [size](#) () const
return number of all elements
- virtual void [clear](#) ()
empty container

Protected Attributes

- std::vector< int > [m_arr](#)
vector used for storage of the sorted elements

5.4.1 Detailed Description

[MedianVector](#) finds median using `std::vector` and `std::nth_element`

[MedianNthElement](#) class implements [Median](#) with using for storage `std::vector` and `std::nth_element` to keep sorted the middle element only.

[add\(\)](#) Average Complexity $O(n)$

[middle\(\)](#) Average Complexity $O(1)$

There is also additional cost for memory copies.

The documentation for this class was generated from the following files:

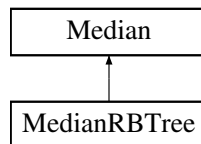
- [Median.h](#)
- [Median.cpp](#)

5.5 MedianRBTree Class Reference

[MedianRBTree](#) finds median using [RBTree](#)

```
#include <Median.h>
```

Inheritance diagram for MedianRBTree:



Public Member Functions

- virtual bool [add](#) (int a)
add new integer element
- virtual int [middle](#) () const
return value of the middle element
- virtual size_t [size](#) () const
return number of all elements
- virtual void [clear](#) ()
empty container

Protected Attributes

- [RBTree m_bt](#)
RBTree container used for storage of the sorted elements.

5.5.1 Detailed Description

[MedianRBTree](#) finds median using [RBTree](#)

[MedianRBTree](#) uses for storage container Red-Black tree, all values are sorted in ascending order. Allows obtaining of the middle value as start iterating from the root. Since Red-Black is a self balancing tree, the middle value would be very near to the root so the search has constant average complexity

[add\(\)](#) Average Complexity $O(\log n)$

[middle\(\)](#) Average Complexity $O(1)$

The documentation for this class was generated from the following files:

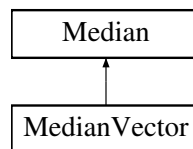
- [Median.h](#)
- [Median.cpp](#)

5.6 MedianVector Class Reference

[MedianVector](#) finds median using `std::vector` and `std::sort`

```
#include <Median.h>
```

Inheritance diagram for MedianVector:



Public Member Functions

- virtual bool [add](#) (int a)
add new integer element
- virtual int [middle](#) () const
return value of the middle element
- virtual size_t [size](#) () const
return number of all elements
- virtual void [clear](#) ()
empty container

Protected Attributes

- `std::vector< int >` [m_arr](#)
vector used for storage of the sorted elements

5.6.1 Detailed Description

[MedianVector](#) finds median using `std::vector` and `std::sort`

[MedianVector](#) class implements [Median](#) with using for storage `std::vector` and `std::sort` to keep elements sorted.

[add\(\)](#) Average Complexity $O(n \log n)$

[middle\(\)](#) Average Complexity $O(1)$

There is also additional cost for memory copies.

The documentation for this class was generated from the following files:

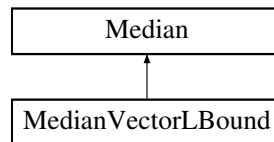
- [Median.h](#)
- [Median.cpp](#)

5.7 MedianVectorLBound Class Reference

[MedianVectorLBound](#) calculate median using `std::vector` and `std::lower_bound`

```
#include <Median.h>
```

Inheritance diagram for MedianVectorLBound:



Public Member Functions

- virtual bool [add](#) (int a)
add new integer element
- virtual int [middle](#) () const
return value of the middle element
- virtual size_t [size](#) () const
return number of all elements
- virtual void [clear](#) ()
empty container

5.7.1 Detailed Description

[MedianVectorLBound](#) calculate median using `std::vector` and `std::lower_bound`

[MedianVectorLBound](#) class implements [Median](#) with using for storage `std::vector` and `std::lower_bound` to sort only the last inserted element only. First the position of the inserted element is found with `std::lower_bound`, and then the new element is inserted there.

[add\(\)](#) Average Complexity $O(\log n)$

[middle\(\)](#) Average Complexity $O(1)$

There is also additional cost for memory copies.

The documentation for this class was generated from the following files:

- [Median.h](#)
- Median.cpp

5.8 RBTREE::Node Class Reference

Public Member Functions

- **Node** (const int &key, [Node](#) *parent=0)
- void **init** ()
- bool **isEmpty** () const
- int **numChildsLeft** () const
- int **numChildsRight** () const

Public Attributes

- [Node](#) * **left**
- [Node](#) * **right**
- [Node](#) * **parent**
- int **key**
- int **type**
- int **nChilds**

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

- [RBTREE.h](#)

5.9 RBTREE Class Reference

Classes

- class [Node](#)

Public Types

- enum { **BLACK** = 0, **RED** = 1 }

Public Member Functions

- bool [insert](#) (const int &key)
insert new element
- int [size](#) () const
return number of all inserted elements
- bool [clear](#) ()
empty tree
- int [middleValue](#) () const
get the value of the middle element

Protected Member Functions

- void [rotateRight](#) ([Node](#) *node)
- void [rotateLeft](#) ([Node](#) *node)
- void [insertFixup](#) ([Node](#) *x)
- virtual [Node](#) * [allocNode](#) (const int &key, [Node](#) *parent=0)
- virtual void [deleteNode](#) ([Node](#) *node)
- [Node](#) * [findParent](#) ([Node](#) *node, const int &key) const
- [Node](#) * [insertNode](#) ([Node](#) *node, const int &key)
- void [emptySubtree](#) ([Node](#) *node)

Protected Attributes

- [Node](#) * **root**
pointer to the root of the tree
- int **count**
integer to keep the number of the inserted elements

The documentation for this class was generated from the following files:

- [RBTREE.h](#)
- [RBTREE.cpp](#)

Chapter 6

File Documentation

6.1 Median.h File Reference

Main declaration header for [Median](#) Interface and several [Median](#) implementations.

```
#include <vector>
#include <set>
#include "RBTree.h"
```

Classes

- class [Median](#)
Median class.
- class [MedianVector](#)
MedianVector finds median using **`std::vector`** and **`std::sort`**
- class [MedianNthElement](#)
MedianVector finds median using **`std::vector`** and **`std::nth_element`**
- class [MedianVectorLBound](#)
MedianVectorLBound calculate median using **`std::vector`** and **`std::lower_bound`**
- class [MedianMultisetAdvance](#)
MedianMultisetAdvance finds median using **`std::multiset`** and **`std::advance`**
- class [MedianMultisetIterator](#)
MedianMultisetIterator finds median using **`std::multiset`** and saved iterator to the middle element.
- class [MedianRBTree](#)
MedianRBTree finds median using **`RBTree`**

6.1.1 Detailed Description

Main declaration header for [Median](#) Interface and several [Median](#) implementations.

Following [Median](#) implementations are provided: [MedianVector](#) [MedianNthElement](#) [MedianVectorLBound](#) [MedianMultisetAdvance](#) [MedianMultisetIterator](#) [MedianRBTree](#)

6.2 PerformanceTest.h File Reference

Performance tests for the [Median](#) Class.

```
#include "Median.h"
```

Functions

- bool [performanceTestAllRandom](#) (int N)
Test with randomly generated values.
- bool [performanceTestAllWorst](#) (int N)
Test with descending values.
- bool [performanceTestAllRepeating](#) (int N)
Test with repeating values.

6.2.1 Detailed Description

Performance tests for the [Median](#) Class.

Tests can be run on Windows x86/x64 platforms.

6.2.2 Function Documentation

6.2.2.1 [performanceTestAllRandom\(\)](#)

```
bool performanceTestAllRandom (  
    int N )
```

Test with randomly generated values.

Performance test with randomly generated values, also checks if the returned values are correct

Parameters

<i>N</i>	number of the test values
----------	---------------------------

Returns

True if successful

6.2.2.2 performanceTestAllRepeating()

```
bool performanceTestAllRepeating (
    int N )
```

Test with repeating values.

Performance test with repeating (0-9) values

Parameters

N	number of the test values
-----	---------------------------

Returns

True if successful

6.2.2.3 performanceTestAllWorst()

```
bool performanceTestAllWorst (
    int N )
```

Test with descending values.

Performance test with linearly descending values, also checks if the returned values are correct

Parameters

N	number of the test values
-----	---------------------------

Returns

True if successful

6.3 RBTREE.h File Reference

multiset implementation of a Red-Black tree

Classes

- class [RBTree](#)
- class [RBTree::Node](#)

6.3.1 Detailed Description

multiset implementation of a Red-Black tree

Similarly to `std::multiset` the tree allows inserting of duplicate values. It is specially designed to find the median element in efficient way.

6.4 UnitTests.h File Reference

Provides unit tests for [Median](#) Class.

```
#include "Median.h"
```

Functions

- void **printc** (int color, const char *output,...)
- void **printResult** (const char *fname, bool bRes)
- int **test_median_basic** ([Median](#) &m)
- int **test_median_iter** ([Median](#) &m)
- int **test_median_random** ([Median](#) &m)
- int **unitTest** ([Median](#) &m)
- int **unitTest_MedianVector** ()
- int **unitTest_MedianVectorLBound** ()
- int **unitTest_MedianNthElement** ()
- int **unitTest_MedianMultisetAdvance** ()
- int **unitTest_MedianMultisetIterator** ()
- int **unitTest_MedianRBTree** ()

6.4.1 Detailed Description

Provides unit tests for [Median](#) Class.

Tests can be run on Windows x86/x64 platforms.

Index

Median, [9](#)
Median.h, [17](#)
MedianMultisetAdvance, [10](#)
MedianMultisetIterator, [11](#)
MedianNthElement, [12](#)
MedianRBTree, [13](#)
MedianVector, [14](#)
MedianVectorLBound, [15](#)

PerformanceTest.h, [18](#)
 performanceTestAllRandom, [18](#)
 performanceTestAllRepeating, [18](#)
 performanceTestAllWorst, [19](#)
performanceTestAllRandom
 PerformanceTest.h, [18](#)
performanceTestAllRepeating
 PerformanceTest.h, [18](#)
performanceTestAllWorst
 PerformanceTest.h, [19](#)

RBTree, [16](#)
RBTree.h, [19](#)
RBTree::Node, [15](#)

UnitTests.h, [20](#)