MergeSort

Generated by Doxygen 1.12.0

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 merge_sort< skladowe > Class Template Reference

Define the class holding the array and the workings of the MergeSort.

```
#include <MergeSort.h>
```

Public Member Functions

- merge_sort (vector< skladowe > x)
- void mergesort ()

  Initiate the MergeSort.
- vector< skladowe > ret ()

  Return the sorted array.
- void print ()

  Print the element on the screen.

3.1.1 Detailed Description

template<typename skladowe>
class merge_sort< skladowe >

Define the class holding the array and the workings of the MergeSort.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 merge_sort()

```
template<typename skladowe >
merge_sort< skladowe >::merge_sort (
            vector< skladowe > x)  [inline]
```

3.1.3 Member Function Documentation

3.1.3.1 mergesort()

```
template<typename skladowe >
void merge_sort< skladowe >::mergesort ()
```

Initiate the MergeSort.

3.1.3.2 print()

```
template<typename skladowe >
void merge_sort< skladowe >::print ()
```

Print the element on the screen.

3.1.3.3 ret()

```
template<typename skladowe >
vector< skladowe > merge_sort< skladowe >::ret ()
```

Return the sorted array.

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

- MergeSort.h
- mergesort.cpp
- print.cpp
- ret.cpp

Chapter 4

File Documentation

4.1 main.cpp File Reference

```
#include "pch.h"
#include "mergesort.h"
```

Functions

- int main ()
  The main body of the program, removing it shows more in-depth test results.

4.1.1 Function Documentation

4.1.1.1 main()

```
int main ()
```

The main body of the program, removing it shows more in-depth test results.

4.2 mergesort.cpp File Reference

```
#include "pch.h"
#include "mergesort.h"
```

4.3 MergeSort.h File Reference

```
#include <iostream>
#include <vector>
#include <algorithm>
```

Classes

- class merge_sort< skladowe >

  Define the class holding the array and the workings of the MergeSort.

4.4 MergeSort.h

Go to the documentation of this file.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

template <typename skladowe>
class merge_sort {
private:
    vector<skladowe> arr;

    void dziel(vector<skladowe>& arr, int lewo, int prawo) {
        if (lewo >= prawo) return;
        int srodek = lewo + (prawo - lewo) / 2;
        dziel(arr, lewo, srodek);
        dziel(arr, srodek + 1, prawo);
        zwyciezaj(arr, lewo, srodek, prawo);
    }

    void zwyciezaj(vector<skladowe>& arr, int lewo, int srodek, int prawo) {
        int n1 = srodek - lewo + 1;
        int n2 = prawo - srodek;
        vector<skladowe> L(n1), R(n2);

        for (int i = 0; i < n1; i++)
            L[i] = arr[lewo + i];
        for (int j = 0; j < n2; j++)
            R[j] = arr[srodek + 1 + j];

        int i = 0, j = 0;
        int k = lewo;

        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                arr[k] = L[i];
                i++;
            }
            else {
                arr[k] = R[j];
                j++;
            }
            k++;
        }
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }
        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }
public:
    merge_sort(vector<skladowe> x) : arr(x) {}
    void mergesort();
    vector<skladowe> ret();
    void print();
};
```

4.5 pch.cpp File Reference

```
#include "pch.h"
```

4.6 pch.h File Reference

```
#include "gtest/gtest.h"
```

4.7 pch.h

Go to the documentation of this file.

```
#pragma once
#include "gtest/gtest.h"
```

4.8 print.cpp File Reference

```
#include "pch.h"
#include "mergesort.h"
```

4.9 ret.cpp File Reference

```
#include "pch.h"
#include "mergesort.h"
```

4.10 test.cpp File Reference

```
#include "pch.h"
#include "mergesort.h"
```

Functions

- TEST (TestCaseName, Test1)

  Test1- keeps the array unchanged when it is already sorted in ascending order.
- TEST (TestCaseName, Test2)

  Test2- can sort an array that is sorted in reverse order.
- TEST (TestCaseName, Test3)

  Test3- correctly sorts a random array of numbers.
- TEST (TestCaseName, Test4)

  Test4- correctly sorts arrays containing only negative numbers.
- TEST (TestCaseName, Test5)

  Test5- correctly sorts arrays containing negative and positive numbers.
- TEST (TestCaseName, Test6)

  Test6- handles empty arrays without throwing an exception.
- TEST (TestCaseName, Test7)

Test7- does not change an array that contains only one element.
- TEST (TestCaseName, Test8)

  Test8- correctly sorts an array containing duplicate numbers.
- TEST (TestCaseName, Test9)

  Test9- correctly sorts a negative array with duplicates.
- TEST (TestCaseName, Test10)

  Test10- correctly sorts arrays with negative numbers, positive numbers, and duplicates.
- TEST (TestCaseName, Test11)

  Test11- correctly sorts an array containing only two elements in ascending order.
- TEST (TestCaseName, Test12)

  Test12- correctly sorts a large array containing over 100 elements.
- TEST (TestCaseName, Test13)

  Test13- correctly sorts a large array containing over 100 elements with negative numbers, positive numbers, and duplicates.
- TEST (TestCaseName, Test14)

  Test14- correctly sorts an array containing chars.

4.10.1 Function Documentation

4.10.1.1 TEST() [1/14]

```
TEST (
            TestCaseName ,
            Test1 )
```

Test1- keeps the array unchanged when it is already sorted in ascending order.

4.10.1.2 TEST() [2/14]

```
TEST (
            TestCaseName ,
            Test10 )
```

Test10- correctly sorts arrays with negative numbers, positive numbers, and duplicates.

4.10.1.3 TEST() [3/14]

```
TEST (
            TestCaseName ,
            Test11 )
```

Test11- correctly sorts an array containing only two elements in ascending order.

4.10.1.4 TEST() [4/14]

```
TEST (
            TestCaseName ,
            Test12 )
```

Test12- correctly sorts a large array containing over 100 elements.

4.10.1.5 TEST() [5/14]

```
TEST (
            TestCaseName ,
            Test13 )
```

Test13- correctly sorts a large array containing over 100 elements with negative numbers, positive numbers, and duplicates.

4.10.1.6 TEST() [6/14]

```
TEST (
            TestCaseName ,
            Test14 )
```

Test14- correctly sorts an array containing chars.

4.10.1.7 TEST() [7/14]

```
TEST (
            TestCaseName ,
            Test2 )
```

Test2- can sort an array that is sorted in reverse order.

4.10.1.8 TEST() [8/14]

```
TEST (
            TestCaseName ,
            Test3 )
```

Test3- correctly sorts a random array of numbers.

4.10.1.9 TEST() [9/14]

```
TEST (
            TestCaseName ,
            Test4 )
```

Test4- correctly sorts arrays containing only negative numbers.

4.10.1.10 TEST() [10/14]

```
TEST (
            TestCaseName ,
            Test5 )
```

Test5- correctly sorts arrays containing negative and positive numbers.

4.10.1.11 TEST() `[11/14]`

```
TEST (
            TestCaseName ,
            Test6 )
```

Test6- handles empty arrays without throwing an exception.

4.10.1.12 TEST() `[12/14]`

```
TEST (
            TestCaseName ,
            Test7 )
```

Test7- does not change an array that contains only one element.

4.10.1.13 TEST() `[13/14]`

```
TEST (
            TestCaseName ,
            Test8 )
```

Test8- correctly sorts an array containing duplicate numbers.

4.10.1.14 TEST() `[14/14]`

```
TEST (
            TestCaseName ,
            Test9 )
```

Test9- correctly sorts a negative array with duplicates.

Index