FirstProjectOMP

Generated by Doxygen 1.9.1

1 File Index 1

1.1 File List . . . 1

2 File Documentation 3

2.1 include/countingsort.h File Reference . . . 3

2.1.1 Macro Definition Documentation . . . 4

2.1.1.1 ENDTIME . . . 4

2.1.1.2 STARTTIME . . . 4

2.1.2 Function Documentation . . . 4

2.1.2.1 countSortOn() . . . 5

2.1.2.2 countSortOn2() . . . 6

2.1.2.3 generateArray() . . . 6

2.2 src/countingsort.c File Reference . . . 6

2.2.1 Macro Definition Documentation . . . 7

2.2.1.1 omp_get_thread_num . . . 7

2.2.2 Function Documentation . . . 7

2.2.2.1 countSortOn() . . . 8

2.2.2.2 countSortOn2() . . . 8

2.2.2.3 generateArray() . . . 8

2.3 src/main.c File Reference . . . 9

2.3.1 Function Documentation . . . 9

2.3.1.1 main() . . . 9

2.4 src/mainOn.c File Reference . . . 10

2.4.1 Function Documentation . . . 10

2.4.1.1 main() . . . 10

Index 11

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

include/countingsort.h 3
src/countingsort.c 6
src/main.c 9
src/mainOn.c 10

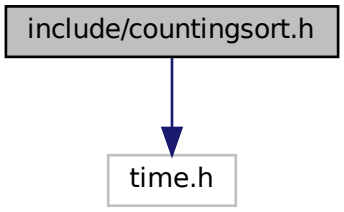
Chapter 2

File Documentation

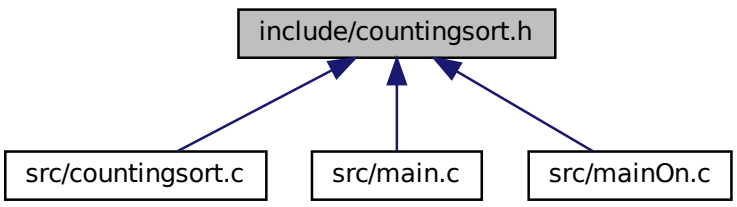
2.1 include/countingsort.h File Reference

```
#include <time.h>
```

Include dependency graph for countingsort.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define STARTTIME(id)
- #define ENDTIME(id, x)

Functions

- void generateArray (ELEMENT_TYPE *, int, int)

  The function is used to initiaslize with pseudorandomic numbers the array. This is a thread safe version as rand_r function is used instead of rand().
- void countSortOn2 (ELEMENT_TYPE *, int, int)

  The function is used to sort the array my_array according to the algorithm counting sort. This function provides an implementation with a time complexity of O(n^2), being so less convenient than another version of this algorithm in the sequencial performances.
- void countSortOn (ELEMENT_TYPE *, ELEMENT_TYPE *, int, int)

  This function sorts an array according to the counting sort algorithm using optimized loops with optimized for loop. This function has a time complexity O(n) being in the sequential version more convenien than the previous version.

2.1.1 Macro Definition Documentation

2.1.1.1 ENDTIME

```
#define ENDTIME(
               id,
               x )
```

Value:

```
  end_time_42_##id = clock(); \
  x = ((double)(end_time_42_##id - start_time_42_##id)) / CLOCKS_PER_SEC
```

2.1.1.2 STARTTIME

```
#define STARTTIME(
               id )
```

Value:

```
  clock_t start_time_42_##id, end_time_42_##id; \
  start_time_42_##id = clock()
```

macros to get execution time: both macros have to be in the same scope define a double variable to use in ENDTIME before STARTTIME: double x; the variable will hold the execution time in seconds.

2.1.2 Function Documentation

2.1.2.1 countSortOn()

```
void countSortOn (
            ELEMENT_TYPE * my_array,
            ELEMENT_TYPE * temp,
            int length,
            int threads )
```

This function sorts an array according to the counting sort algorithm using optimized loops with optimized for loop. This function has a time complexity O(n) being in the sequential version more convenien than the previous version.

Parameters

my_array	a pointer to an array which must be sorted.
temp	the pointer to the array which must be sorted.
length	size of my_array.
threads	number of threads.

2.1.2.2 countSortOn2()

```
void countSortOn2 (
             ELEMENT_TYPE * my_array,
             int length,
             int threads )
```

The function is used to sort the array my_array according to the algorithm counting sort. This function provides an implementation with a time complexity of O(n^2), being so less convenient than another version of this algorithm in the sequencial performances.

Parameters

my_array	the pointer to the array to be sorted.
length	size of my_array.
threads	number of threads.

2.1.2.3 generateArray()

```
void generateArray (
             ELEMENT_TYPE * my_array,
             int length,
             int threads )
```

The function is used to initiaslize with pseudorandomic numbers the array. This is a thread safe version as rand_r function is used instead of rand().

Parameters

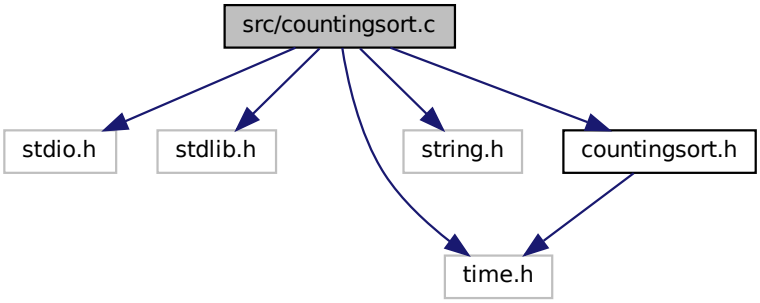
my_array	the pointer to the memory area of the array
length	size of my_array.
threads	the number of threads.

2.2 src/countingsort.c File Reference

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include "countingsort.h"
```

Include dependency graph for countingsort.c:



Macros

- #define omp_get_thread_num() 0

Functions

- void generateArray (ELEMENT_TYPE *my_array, int length, int threads)

  The function is used to initiaslize with pseudorandomic numbers the array. This is a thread safe version as rand_r function is used instead of rand().
- void countSortOn2 (ELEMENT_TYPE *my_array, int length, int threads)

  The function is used to sort the array my_array according to the algorithm counting sort. This function provides an implementation with a time complexity of O(n^2), being so less convenient than another version of this algorithm in the sequencial performances.
- void countSortOn (ELEMENT_TYPE *my_array, ELEMENT_TYPE *temp, int length, int threads)

  This function sorts an array according to the counting sort algorithm using optimized loops with optimized for loop. This function has a time complexity O(n) being in the sequential version more convenien than the previous version.

2.2.1 Macro Definition Documentation

2.2.1.1 omp_get_thread_num

```
#define omp_get_thread_num( ) 0
```

2.2.2 Function Documentation

2.2.2.1 countSortOn()

```
void countSortOn (
            ELEMENT_TYPE * my_array,
            ELEMENT_TYPE * temp,
            int length,
            int threads )
```

This function sorts an array according to the counting sort algorithm using optimized loops with optimized for loop. This function has a time complexity O(n) being in the sequential version more convenien than the previous version.

Parameters

my_array	a pointer to an array which must be sorted.
temp	the pointer to the array which must be sorted.
length	size of my_array.
threads	number of threads.

2.2.2.2 countSortOn2()

```
void countSortOn2 (
            ELEMENT_TYPE * my_array,
            int length,
            int threads )
```

The function is used to sort the array my_array according to the algorithm counting sort. This function provides an implementation with a time complexity of O(n^2), being so less convenient than another version of this algorithm in the sequencial performances.

Parameters

my_array	the pointer to the array to be sorted.
length	size of my_array.
threads	number of threads.

2.2.2.3 generateArray()

```
void generateArray (
            ELEMENT_TYPE * my_array,
            int length,
            int threads )
```

The function is used to initiaslize with pseudorandomic numbers the array. This is a thread safe version as rand_r function is used instead of rand().

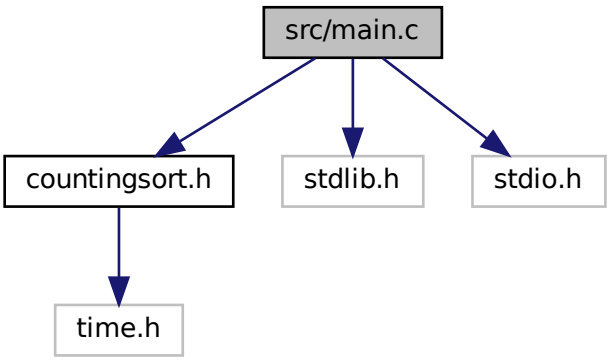
Parameters

my_array	the pointer to the memory area of the array
length	size of my_array.
threads	the number of threads.

2.3 src/main.c File Reference

```
#include "countingsort.h"
#include <stdlib.h>
#include <stdio.h>
```

Include dependency graph for main.c:



Functions

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

2.3.1 Function Documentation

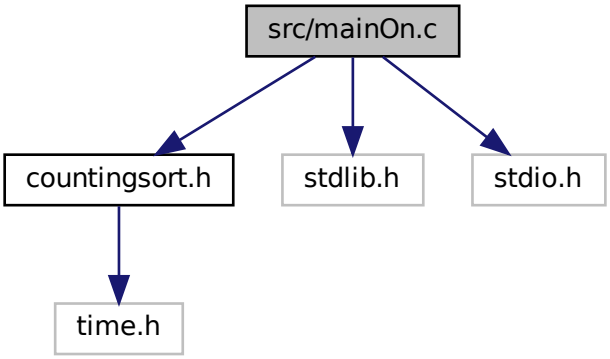
2.3.1.1 main()

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

2.4 src/mainOn.c File Reference

```
#include "countingsort.h"
#include <stdlib.h>
#include <stdio.h>
```

Include dependency graph for mainOn.c:



Functions

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

2.4.1 Function Documentation

2.4.1.1 main()

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

Index

countingsort.c
countSortOn, 7
countSortOn2, 8
generateArray, 8
omp_get_thread_num, 7
countingsort.h
countSortOn, 4
countSortOn2, 6
ENDTIME, 4
generateArray, 6
STARTTIME, 4
countSortOn
countingsort.c, 7
countingsort.h, 4
countSortOn2
countingsort.c, 8
countingsort.h, 6

ENDTIME
countingsort.h, 4

generateArray
countingsort.c, 8
countingsort.h, 6

include/countingsort.h, 3

main
main.c, 9
mainOn.c, 10
main.c
main, 9
mainOn.c
main, 10

omp_get_thread_num
countingsort.c, 7

src/countingsort.c, 6
src/main.c, 9
src/mainOn.c, 10
STARTTIME
countingsort.h, 4