

Algorytmy sortowania

Wygenerowano przez Doxygen 1.8.4

Pn, 24 mar 2014 02:17:16

Spis treści

1	Algorytmy sortowania	2
1.1	Opis programu	2
1.2	Autor	2
2	Indeks klas	2
2.1	Lista klas	2
3	Indeks plików	2
3.1	Lista plików	2
4	Dokumentacja klas	3
4.1	Dokumentacja szablonu klasy <code>Array< type ></code>	3
4.1.1	Opis szczegółowy	4
4.1.2	Dokumentacja konstruktora i destruktora	5
4.1.3	Dokumentacja funkcji składowych	5
4.1.4	Dokumentacja atrybutów składowych	6
4.2	Dokumentacja klasy <code>Benchmark</code>	6
4.2.1	Opis szczegółowy	7
4.2.2	Dokumentacja funkcji składowych	7
4.2.3	Dokumentacja atrybutów składowych	7
5	Dokumentacja plików	8
5.1	Dokumentacja pliku <code>array.hh</code>	8
5.1.1	Opis szczegółowy	8
5.2	Dokumentacja pliku <code>benchmark.cpp</code>	8
5.2.1	Opis szczegółowy	8
5.3	Dokumentacja pliku <code>benchmark.hh</code>	8
5.3.1	Opis szczegółowy	9
5.3.2	Dokumentacja typów wyliczanych	9
5.4	Dokumentacja pliku <code>heap.hh</code>	9
5.4.1	Dokumentacja funkcji	9
5.5	Dokumentacja pliku <code>main.cpp</code>	9
5.5.1	Opis szczegółowy	9
5.5.2	Dokumentacja funkcji	10
5.6	Dokumentacja pliku <code>mainpage.dox</code>	10
5.7	Dokumentacja pliku <code>merge.hh</code>	10
5.7.1	Dokumentacja funkcji	10
5.8	Dokumentacja pliku <code>quicksort.hh</code>	10
5.8.1	Dokumentacja funkcji	10

Indeks**11****1 Algorytmy sortowania****1.1 Opis programu**

Program structures wykonuje pomiaru czasu wykonania algorytmu sortowania w jednej z trzech opcji:

SORTOWANIE SZYBKIE

Sposob wywołania: `./structures quick [plik wejsciowy] [plik sprawdzajacy] [ilosc powtorzen]`

SORTOWANIE PRZEZ SCALANIE

Sposob wywołania: `./structures merge [plik wejsciowy] [plik sprawdzajacy] [ilosc powtorzen]`

SORTOWANIE PRZEZ KOPCOWANIE

Sposob wywołania: `./structures heap [plik wejsciowy] [plik sprawdzajacy] [ilosc powtorzen]`

1.2 Autor

Program wykonała: Agnieszka Wisniewska, nr albumu: 200 466

2 Indeks klas**2.1 Lista klas**

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Array < type > Szablon klasy Array	3
Benchmark Definicja klasy Benchmark	6

3 Indeks plików**3.1 Lista plików**

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

array.hh Plik zawiera definicje szablonu klasy Array	8
benchmark.cpp Plik zawiera implementacje klasy Benchmark	8
benchmark.hh Plik zawiera definicje klasy Benchmark oraz typu sortingType	8
heap.hh	9

main.cpp	
Plik zawiera glowna funkcje programu	9
merge.hh	10
quicksort.hh	10

4 Dokumentacja klas

4.1 Dokumentacja szablonu klasy `Array< type >`

Szablon klasy [Array](#).

```
#include <array.hh>
```

Metody publiczne

- [Array](#) ()
Konstruktor tworzacy obiekty klasy [Array](#).
- [~Array](#) ()
Destruktor usuwajacy obiekt klasy [Array](#).
- unsigned int [size](#) () const
Zwraca dlugosc tablicy.
- void [resize](#) (unsigned int newSize)
Zmienia ilosc elementow tablicy.
- void [changePlaces](#) (unsigned int a, unsigned int b)
Zamienia miejscami dwa wybrane elementy.
- void [reverse](#) ()
Odwraca kolejnosc elementow tablicy.
- void [addElem](#) (type e)
Dodaje nowy element na koniec tablicy.
- void [addArrays](#) (const [Array](#)< type > &CA)
Laczy dwie tablice.
- type & [operator](#)[] (const unsigned int nr)
Przeciazenie operatora indeksujacego.
- const type & [operator](#)[] (const unsigned int nr) const
- [Array](#)< type > & [operator](#)= (const [Array](#)< type > &value)
Przeciazenie operatora przypisania.
- [Array](#)< type > & [operator](#)+ (const [Array](#)< type > &value) const
Przeciazenie operatora dodawania.
- bool [operator](#)== (const [Array](#)< type > &value) const
Przeciazenie operatora porownania.

Atrybuty prywatne

- type * [T](#)
- unsigned int [arraySize](#)

4.1.1 Opis szczegółowy

```
template<typename type>class Array< type >
```

Szablon klasy [Array](#).

Definiuje funkcje pozwalające na wykonywaniu operacji na tablicy

Template Parameters

<i>type</i>	typ danych przechowywanych w tablicy
-------------	--------------------------------------

4.1.2 Dokumentacja konstruktora i destruktor

4.1.2.1 `template<typename type> Array< type >::Array () [inline]`

Konstruktor tworzący obiekty klasy `Array`.

4.1.2.2 `template<typename type> Array< type >::~~Array () [inline]`

Destruktor usuwający obiekt klasy `Array`.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 `template<typename type> void Array< type >::addArrays (const Array< type > & CA)`

Łączy dwie tablice.

Parametry

<i>CA</i>	tablica, której zawartość chcemy dołączyć
-----------	---

4.1.3.2 `template<typename type> void Array< type >::addElem (type e)`

Dodaje nowy element na koniec tablicy.

Parametry

<i>e</i>	element dodawany na koniec tablicy
----------	------------------------------------

4.1.3.3 `template<typename type > void Array< type >::changePlaces (unsigned int a, unsigned int b)`

Zamienia miejscami dwa wybrane elementy.

Parametry

<i>a</i>	pierwszy z elementów, które zamieniamy miejscami
<i>b</i>	drugi element, który zamieniamy z <i>a</i>

4.1.3.4 `template<typename type> Array< type > & Array< type >::operator+ (const Array< type > & value) const`

Przeciążenie operatora dodawania.

Zwraca

tablica zawierająca elementy z obu tablic

4.1.3.5 `template<typename type> Array< type > & Array< type >::operator= (const Array< type > & value)`

Przeciążenie operatora przypisania.

Parametry

<i>value</i>	tablica z której wartości zostaną przypisane
--------------	--

Zwraca

referencje na obiekt wywołujący

4.1.3.6 `template<typename type> bool Array< type >::operator==(const Array< type > & value) const`

Przeciążenie operatora porównania.

Zwraca

true - gdy tablice sa takie same
false - gdy tablice sie roznia

4.1.3.7 `template<typename type> type& Array< type >::operator[] (const unsigned int nr) [inline]`

Przeciążenie operatora indeksujacego.

Parametry

<i>nr</i>	Indeks wybranego elementu
-----------	---------------------------

Zwraca

Wartosc pod wskazanym indeksem

4.1.3.8 `template<typename type> const type& Array< type >::operator[] (const unsigned int nr) const [inline]`

4.1.3.9 `template<typename type > void Array< type >::resize (unsigned int newSize)`

Zmienia ilosc elementow tablicy.

Parametry

<i>newSize</i>	rozmiar tablicy
----------------	-----------------

4.1.3.10 `template<typename type > void Array< type >::reverse ()`

Odwraca kolejnosc elementow tablicy.

4.1.3.11 `template<typename type> unsigned int Array< type >::size () const [inline]`

Zwraca dlugosc tablicy.

Zwraca

liczbe elementow tablicy

4.1.4 Dokumentacja atrybutów składowych

4.1.4.1 `template<typename type> unsigned int Array< type >::arraySize [private]`

4.1.4.2 `template<typename type> type* Array< type >::T [private]`

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [array.hh](#)

4.2 Dokumentacja klasy Benchmark

Definicja klasy [Benchmark](#).

```
#include <benchmark.hh>
```

Metody publiczne

- bool `load` (char *fileName)
Funkcja wczytująca dane z wybranego pliku.
- bool `check` (char *fileName)
Funkcja porównująca dane z pliku źródłowego po wykonaniu wybranego sortowania z danym z pliku sprawdzającego.
- double `benchmark` (int nolation, `sortingType` type)
Funkcja mierząca czas wykonywania algorytmu sortowania.
- int `size` ()

Metody prywatne

- void `calculate` (`sortingType` type)

Atrybuty prywatne

- `Array<int>` `array1`
- `Array<int>` `array2`

4.2.1 Opis szczegółowy

Definicja klasy `Benchmark`.

Klasa obiektów umożliwiająca wczytywanie danych z plików do tablicy, wykonania wybranych obliczeń oraz kontrole poprawności wykonywania obliczeń z plikiem sprawdzającym. Głównym zadaniem klasy jest mierzenie czasu wykonywanych algorytmów.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 double Benchmark::benchmark (int nolation, `sortingType` type)

Funkcja mierząca czas wykonywania algorytmu sortowania.

4.2.2.2 void Benchmark::calculate (`sortingType` type) [private]

4.2.2.3 bool Benchmark::check (char * fileName)

Funkcja porównująca dane z pliku źródłowego po wykonaniu wybranego sortowania z danym z pliku sprawdzającego.

4.2.2.4 bool Benchmark::load (char * fileName)

Funkcja wczytująca dane z wybranego pliku.

4.2.2.5 int Benchmark::size () [inline]

Zwraca

Zwraca długość tablicy.

4.2.3 Dokumentacja atrybutów składowych

4.2.3.1 `Array<int>` Benchmark::array1 [private]

4.2.3.2 `Array<int>` Benchmark::array2 [private]

Dokumentacja dla tej klasy została wygenerowana z plików:

- [benchmark.hh](#)
- [benchmark.cpp](#)

5 Dokumentacja plików

5.1 Dokumentacja pliku array.hh

Plik zawiera definicje szablonu klasy [Array](#).

```
#include <cstdlib>
#include <iostream>
```

Komponenty

- class [Array< type >](#)
Szablon klasy [Array](#).

5.1.1 Opis szczegółowy

Plik zawiera definicje szablonu klasy [Array](#).

5.2 Dokumentacja pliku benchmark.cpp

Plik zawiera implementacje klasy [Benchmark](#).

```
#include <iostream>
#include <fstream>
#include <ctime>
#include "benchmark.hh"
#include "merge.hh"
#include "quicksort.hh"
#include "heap.hh"
```

5.2.1 Opis szczegółowy

Plik zawiera implementacje klasy [Benchmark](#).

5.3 Dokumentacja pliku benchmark.hh

Plik zawiera definicje klasy [Benchmark](#) oraz typu `sortingType`.

```
#include "array.hh"
```

Komponenty

- class [Benchmark](#)
Definicja klasy [Benchmark](#).

Wyliczenia

- enum `sortingType` { `quick`, `merge`, `heap` }
- Typ danych przechowujący rodzaj sortowania.*

5.3.1 Opis szczegółowy

Plik zawiera definicje klasy `Benchmark` oraz typu `sortingType`.

5.3.2 Dokumentacja typów wyliczanych

5.3.2.1 enum `sortingType`

Typ danych przechowujący rodzaj sortowania.

Wartości wyliczeń

quick

merge

heap

5.4 Dokumentacja pliku heap.hh

Funkcje

- template<typename type >
void `heapSort` (`Array`< type > &value)

5.4.1 Dokumentacja funkcji

5.4.1.1 template<typename type > void `heapSort` (`Array`< type > & *value*)

5.5 Dokumentacja pliku main.cpp

Plik zawiera główną funkcję programu.

```
#include <iostream>
#include <cstdlib>
#include <string>
#include "benchmark.hh"
```

Funkcje

- int `main` (int argc, char **argv)
- Główna funkcja programu.*

5.5.1 Opis szczegółowy

Plik zawiera główną funkcję programu.

5.5.2 Dokumentacja funkcji

5.5.2.1 `int main (int argc, char ** argv)`

Główna funkcja programu.

Pobiera argumenty z linii poleceń, przekazuje je do odpowiedniej klasy i wyświetla wyniki pomiarów czasu.

5.6 Dokumentacja pliku `mainpage.dox`

5.7 Dokumentacja pliku `merge.hh`

```
#include "array.hh"
```

Funkcje

- `template<typename type >`
`void mergeSort (Array< type > &arr, Array< type > &tmp, unsigned int start, unsigned int end)`
- `template<typename type >`
`void mergeSort (Array< type > &value)`

5.7.1 Dokumentacja funkcji

5.7.1.1 `template<typename type > void mergeSort (Array< type > & arr, Array< type > & tmp, unsigned int start, unsigned int end)`

5.7.1.2 `template<typename type > void mergeSort (Array< type > & value)`

5.8 Dokumentacja pliku `quicksort.hh`

```
#include <array.hh>
```

Funkcje

- `template<typename type >`
`void quicksort (Array< type > &d, int left, int right)`
- `template<typename type >`
`void quicksort (Array< type > &value)`

5.8.1 Dokumentacja funkcji

5.8.1.1 `template<typename type > void quicksort (Array< type > & d, int left, int right)`

5.8.1.2 `template<typename type > void quicksort (Array< type > & value)`

Skorowidz

- ~Array
 - Array, [5](#)
- addArrays
 - Array, [5](#)
- addElem
 - Array, [5](#)
- Array
 - ~Array, [5](#)
 - addArrays, [5](#)
 - addElem, [5](#)
 - Array, [5](#)
 - arraySize, [6](#)
 - changePlaces, [5](#)
 - operator+, [5](#)
 - operator=, [5](#)
 - operator==, [5](#)
 - resize, [6](#)
 - reverse, [6](#)
 - size, [6](#)
 - T, [6](#)
- Array< type >, [3](#)
- array.hh, [8](#)
- array1
 - Benchmark, [7](#)
- array2
 - Benchmark, [7](#)
- arraySize
 - Array, [6](#)
- Benchmark, [6](#)
 - array1, [7](#)
 - array2, [7](#)
 - benchmark, [7](#)
 - calculate, [7](#)
 - check, [7](#)
 - load, [7](#)
 - size, [7](#)
- benchmark
 - Benchmark, [7](#)
- benchmark.hh
 - heap, [9](#)
 - merge, [9](#)
 - quick, [9](#)
- benchmark.cpp, [8](#)
- benchmark.hh, [8](#)
 - sortingType, [9](#)
- calculate
 - Benchmark, [7](#)
- changePlaces
 - Array, [5](#)
- check
 - Benchmark, [7](#)
- heap
 - benchmark.hh, [9](#)
 - heap.hh, [9](#)
 - heapSort, [9](#)
 - heapSort
 - heap.hh, [9](#)
- load
 - Benchmark, [7](#)
- main
 - main.cpp, [10](#)
 - main.cpp, [9](#)
 - main, [10](#)
 - mainpage.dox, [10](#)
 - merge
 - benchmark.hh, [9](#)
 - merge.hh, [10](#)
 - mergeSort, [10](#)
 - mergeSort
 - merge.hh, [10](#)
- operator+
 - Array, [5](#)
- operator=
 - Array, [5](#)
- operator==
 - Array, [5](#)
- quick
 - benchmark.hh, [9](#)
- quicksort
 - quicksort.hh, [10](#)
- quicksort.hh, [10](#)
 - quicksort, [10](#)
- resize
 - Array, [6](#)
- reverse
 - Array, [6](#)
- size
 - Array, [6](#)
 - Benchmark, [7](#)
- sortingType
 - benchmark.hh, [9](#)
- T
 - Array, [6](#)