Struktury

Wygenerowano przez Doxygen 1.8.4

Pn, 24 mar 2014 00:14:47

Spis treści

1 Struktury - podstawowe informacje na temat programu

1.1 Opis programu

Program structures wykonuje pomiaru czasu wykonania algorytmu wypelniania jednej z czterech opcji:

STOS

Sposob wywolania: `./structures stack [rozmiar problemu] [ilosc powtorzen]`

Wykonuje podstawowe operacje na stosie. W przypadku przekroczenia rozmiaru tablicy zwieksza jej rozmiar o 1.

STOS2

Sposob wywolania: `./structures stack2 [rozmiar problemu] [ilosc powtorzen]`

Wykonuje podstawowe operacje na stosie. W przypadku przekroczenia rozmiaru tablicy zwieksza jej rozmiar 2 razy.

STOS NA LISCIE

Sposob wywolania: `./structures stonlist [rozmiar problemu] [ilosc powtorzen]`

Wykonuje podstawowe operacje stosu na liscie.

KOLEJKA

Sposob wywolania: `./structures queue [rozmiar problemu] [ilosc powtorzen]`

Wykonuje podstawowe operacje na kolejce (liscie jednokierunkowej).

1.2 Autor

Program wykonala: 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:

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

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 pozwalajace na wykonywaniu operacji na tablicy

Template Parameters

type	typ danych przechowywanych w tablicy

4.1.2 Dokumentacja konstruktora i destruktora

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

Konstruktor tworzacy obiekty klasy Array.

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

Destruktor usuwajacy obiekt klasy Array.

4.1.3 Dokumentacja funkcji składowych

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

Laczy dwie tablice.

Parametry

CA	tablica, ktorej zawartosc chcemy dolaczyc

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

Dodaje nowy element na koniec tablicy.

Parametry

e	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

a	pierwszy z elementow, ktore zamieniamy miejscami
b	drugi element, ktory zamieniamy z a

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

Przeciazenie operatora dodawania.

Zwraca

tablica zawierajaca elementy z obu tablic

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

Przeciazenie operatora przypisania.

Parametry

value	tablica z ktorej wartosci zostana przypisane

Zwraca

referencje na obiekt wywolujacy

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

Przeciazenie operatora porownania.

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]

Przeciazenie operatora indeksujacego.

Parametry

nr	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

newSize	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 wczytujaca dane z wybranego pliku.
- bool check (char *fileName)

  Funkcja porownujaca dane z pliku zrodlowego po wykonaniu wybranego sortowania z danym z pliku sprawdzajacego.
- double benchmark (int noIteration, sortingType type)

  Funkcja mierzaca 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 obiektow umozliwiajaca wczytywanie danych z plikow do tablicy, wykonania wybranych obliczen oraz kontrole poprawnosci wykonywania obliczen z plikiem sprawdzajacym. Glownym zadaniem klasy jest mierzenie czasu wykonywanych algorytmow.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 double Benchmark::benchmark (int *noIteration,* sortingType *type*)

Funkcja mierzaca czas wykonywania algorytmu sortowania.

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

4.2.2.3 bool Benchmark::check (char * *fileName*)

Funkcja porownujaca dane z pliku zrodlowego po wykonaniu wybranego sortowania z danym z pliku sprawdzajacego.

4.2.2.4 bool Benchmark::load (char * *fileName*)

Funkcja wczytujaca dane z wybranego pliku.

4.2.2.5 int Benchmark::size ( ) [inline]

Zwraca

Zwraca dlugosc 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 przechowujacy 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 przechowujacy 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 glowna funkcje programu.

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

Funkcje

- int main (int argc, char **argv)

  Glowna funkcja programu.

5.5.1 Opis szczegółowy

Plik zawiera glowna funkcje programu.

5.5.2 Dokumentacja funkcji

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

Glowna funkcja programu.

Pobiera argumenty z linii polecen, przekazuje je do odpowiedniej klasy i wyswietla wyniki pomiarow 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