Struktury

Wygenerowano przez Doxygen 1.8.4

N, 16 mar 2014 23:22:47

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

- int * 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> int* Array< type >::T [private]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- array.hh

4.2 Dokumentacja szablonu klasy BenchQueue< queueClass >

Szablon klasy BenchQueue.

`#include <benchmark.hh>`

Metody publiczne

- double benchmark (unsigned int noIteration, unsigned int problemSize)

  Mierzy czas trwania algorytmu wykonywanego na kolejce dla zadanej ilosci powtorzen oraz rozmiaru problemu.

4.2.1 Opis szczegółowy

template<class queueClass>class BenchQueue< queueClass >

Szablon klasy BenchQueue.

Definiuje funkcje sluzace do pomiaru czasu wykonywania algorytmu na kolejce.

Template Parameters

queueClass	klasa kolejki dla ktorej przeprowadzamy benchmark

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 template<class queueClass > double BenchQueue< queueClass >::benchmark (unsigned int *noIteration,* unsigned int *problemSize*)

Mierzy czas trwania algorytmu wykonywanego na kolejce dla zadanej ilosci powtorzen oraz rozmiaru problemu.

Parametry

noIteration	liczba powtorzen wykonania algorytmu
problemSize	liczba danych dodawanych do kolejki podczas jednej iteracji algorytmu

Zwraca

czas wykonywania algorytmu

Dokumentacja dla tej klasy została wygenerowana z pliku:

- benchmark.hh

4.3 Dokumentacja szablonu klasy BenchStack< stackClass >

Szablon klasy BenchStack.

`#include <benchmark.hh>`

Metody publiczne

- double benchmark (unsigned int noIteration, unsigned int problemSize)

  Mierzy czas trwania algorytmu wykonywanego na stosie dla zadanej ilosci powtorzen oraz rozmiaru problemu.

4.3.1 Opis szczegółowy

template<class stackClass>class BenchStack< stackClass >

Szablon klasy BenchStack.

Definiuje funkcje sluzace do pomiaru czasu wykonywania algorytmu na stosie.

Template Parameters

stackClass	klasa stosu dla ktorej przeprowadzamy benchmark

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 template<class stackClass > double BenchStack< stackClass >::benchmark (unsigned int *noIteration,* unsigned int *problemSize*)

Mierzy czas trwania algorytmu wykonywanego na stosie dla zadanej ilosci powtorzen oraz rozmiaru problemu.

Parametry

noIteration	liczba powtorzen wykonania algorytmu
problemSize	liczba danych wrzucanych na stos podczas jednej iteracji algorytmu

Zwraca

czas wykonywania algorytmu

Dokumentacja dla tej klasy została wygenerowana z pliku:

- benchmark.hh

4.4 Dokumentacja szablonu klasy Queue< type >

Szablon klasy Queue.

`#include <queue.hh>`

Metody publiczne

- unsigned int size () const
  Pobiera rozmiar kolejki.
- bool isEmpty () const
  Sprawdza, czy kolejka jest pusta.
- const type & top () const
  Udostepnia pierwszy element listy(kolejki)
- void enqueue (const type elem)
  Dodaje element na koniec kolejki.
- void dequeue ()
  Zabiera element z poczatku listy.

Atrybuty prywatne

- std::list< type > flist

4.4.1 Opis szczegółowy

template<typename type>class Queue< type >

Szablon klasy Queue.

Definiuje funkcje pozwalajace na wykonywaniu operacji na kolejce

Template Parameters

type	typ danych przechowywanych na liscie

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 template<typename type > void Queue< type >::dequeue ( ) [inline]

Zabiera element z poczatku listy.

4.4.2.2 template<typename type > void Queue< type >::enqueue (const type *elem*) [inline]

Dodaje element na koniec kolejki.

Parametry

elem	element dodawany na koniec kolejki

4.4.2.3 template<typename type > bool Queue< type >::isEmpty ( ) const [inline]

Sprawdza, czy kolejka jest pusta.

Zwraca

true - jesli lista jest pusta
false - jesli na liscie jest co najmniej 1 element

4.4.2.4 template<typename type > unsigned int Queue< type >::size ( ) const [inline]

Pobiera rozmiar kolejki.

Zwraca

ilosc elementow kolejki

4.4.2.5 template<typename type > const type& Queue< type >::top ( ) const [inline]

Udostepnia pierwszy element listy(kolejki)

Zwraca

zawartosc pierwszego elementu kolejki

4.4.3 Dokumentacja atrybutów składowych

4.4.3.1 template<typename type > std::list<type> Queue< type >::flist [private]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- queue.hh

4.5 Dokumentacja szablonu klasy Stack< type >

Szablon klasy Stack.

`#include <stack.hh>`

Metody publiczne

- unsigned int size () const

  Pobiera rozmiar stosu.
- void push (type elem)

  Dodaje element na szczyt stosu.
- void pop ()

  Zabiera element ze szczytu stosu.
- bool isEmpty () const

  Sprawdza, czy stos nie jest pusty.
- const type & top () const

  Udostepnia element ze szczytu stosu.

Atrybuty prywatne

- Array< type > array

4.5.1 Opis szczegółowy

template<typename type>class Stack< type >

Szablon klasy Stack.

Definiuje funkcje pozwalajace na wykonywanie operacji na stosie

Template Parameters

type	typ danych przechowywanych na stosie

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 template<typename type > bool Stack< type >::isEmpty ( ) const [inline]

Sprawdza, czy stos nie jest pusty.

Zwraca

true - jesli stos jest pusty
false - jesli stos zawiera co najmniej 1 element

4.5.2.2 template<typename type > void Stack< type >::pop ( )

Zabiera element ze szczytu stosu.

Warunek wstępny

stos nie moze byc pusty

4.5.2.3 template<typename type > void Stack< type >::push (type *elem*)

Dodaje element na szczyt stosu.

Parametry

elem	element, ktory dodajemy na szczyt stosu

4.5.2.4 template<typename type > unsigned int Stack< type >::size ( ) const [inline]

Pobiera rozmiar stosu.

Zwraca

ilosc elementow na stosie

4.5.2.5 template<typename type > const type& Stack< type >::top ( ) const [inline]

Udostepnia element ze szczytu stosu.

Warunek wstępny

stos nie moze byc pusty

Zwraca

zawartosc ostatniego elementu stosu

4.5.3 Dokumentacja atrybutów składowych

4.5.3.1 template<typename type > Array<type> Stack< type >::array [private]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- stack.hh

4.6 Dokumentacja szablonu klasy Stack2< type >

Szablon klasy Stack2.

`#include <stack2.hh>`

Metody publiczne

- Stack2 ()

  Konstruktor tworzacy obiekt klasy Stack2.
- unsigned int size () const

  Pobiera rozmiar stosu.
- void push (type elem)

  Dodaje element na stos.
- void pop ()

  Zabiera element ze szczytu stosu.
- bool isEmpty () const

  Sprawdza, czy stos jest pusty.
- const type & top () const

  Udostepnia element ze szczytu stosu.

Atrybuty prywatne

- Array< type > array
- unsigned int arrSize

4.6.1 Opis szczegółowy

template<typename type>class Stack2< type >

Szablon klasy Stack2.

Definiuje funkcje pozwalajace na wykonywanie operacji na stosie

Template Parameters

type	typ danych przechowywanych na stosie

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 template<typename type > Stack2< type >::Stack2 ( ) [inline]

Konstruktor tworzacy obiekt klasy Stack2.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 template<typename type > bool Stack2< type >::isEmpty ( ) const [inline]

Sprawdza, czy stos jest pusty.

Zwraca

true - jesli stos jest pusty
false - jesli stos zawiera co najmniej 1 element

4.6.3.2 template<typename type > void Stack2< type >::pop ( )

Zabiera element ze szczytu stosu.

Warunek wstępny

stos nie moze byc pusty

4.6.3.3 template<typename type > void Stack2< type >::push (type *elem*)

Dodaje element na stos.

Parametry

elem	element, ktory dodajemy na szczyt stosu

4.6.3.4 template<typename type > unsigned int Stack2< type >::size ( ) const [inline]

Pobiera rozmiar stosu.

Zwraca

ilosc elementow na stosie

4.6.3.5 template<typename type > const type& Stack2< type >::top ( ) const [inline]

Udostepnia element ze szczytu stosu.

Warunek wstępny

stos nie moze byc pusty

Zwraca

zawartosc ostatniego elementu stosu

4.6.4 Dokumentacja atrybutów składowych

4.6.4.1 template<typename type > Array<type> Stack2< type >::array [private]

4.6.4.2 template<typename type > unsigned int Stack2< type >::arrSize [private]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- stack2.hh

4.7 Dokumentacja szablonu klasy stOnList< type >

Szablon klasy stOnList.

`#include <stOnList.hh>`

Metody publiczne

- unsigned int size () const
  Pobiera rozmiar listy.
- void push (type elem)
  Dodaje element na koniec listy.
- void pop ()
  Zabiera element z konca listy.
- bool isEmpty () const
  Sprawdza, czy lista jest pusta.
- const type & top () const
  Udostepnia ostatni element listy.

Atrybuty prywatne

- std::list< type > flist

4.7.1 Opis szczegółowy

template<typename type>class stOnList< type >

Szablon klasy stOnList.

Definiuje funkcje pozwalajace na wykonywanie operacji na liscie

Template Parameters

type	typ danych przechowywanych na liscie

4.7.2 Dokumentacja funkcji składowych

4.7.2.1 template<typename type > bool stOnList< type >::isEmpty ( ) const [inline]

Sprawdza, czy lista jest pusta.

Zwraca

true - jesli lista jest pusta
false - jesli na liscie jest co najmniej 1 element

4.7.2.2 template<typename type > void stOnList< type >::pop ( ) [inline]

Zabiera element z konca listy.

Warunek wstępny

lista nie moze byc pusta

4.7.2.3 template<typename type > void stOnList< type >::push (type *elem*) [inline]

Dodaje element na koniec listy.

Parametry

elem	element dodawany na koniec listy

4.7.2.4 template<typename type > unsigned int stOnList< type >::size ( ) const [inline]

Pobiera rozmiar listy.

Zwraca

liczba elementow listy

4.7.2.5 template<typename type > const type& stOnList< type >::top ( ) const [inline]

Udostepnia ostatni element listy.

Zwraca

zawartosc ostatniego elementu listy

4.7.3 Dokumentacja atrybutów składowych

4.7.3.1 template<typename type > std::list<type> stOnList< type >::flist [private]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- stOnList.hh

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.hh

Plik zawiera deklaracje zwiazane z szablonem klas BenchStack i BenchQueue.

```
#include <ctime>
#include "stack.hh"
#include "stack2.hh"
#include "stOnList.hh"
#include "queue.hh"
```

Komponenty

- class BenchStack< stackClass >

  Szablon klasy BenchStack.
- class BenchQueue< queueClass >

  Szablon klasy BenchQueue.

5.2.1 Opis szczegółowy

Plik zawiera deklaracje zwiazane z szablonem klas BenchStack i BenchQueue.

5.3 Dokumentacja pliku main.cpp

Plik zawiera glowna funkcje programu.

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

Funkcje

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

  Glowna funkcja programu.

5.3.1 Opis szczegółowy

Plik zawiera glowna funkcje programu.

5.3.2 Dokumentacja funkcji

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

Glowna funkcja programu.

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

5.4 Dokumentacja pliku mainpage.dox

5.5 Dokumentacja pliku queue.hh

Plik zawiera deklaracje zwiazane z szablonem klasy Queue.

```
#include <list>
```

Komponenty

- class Queue< type >

  Szablon klasy Queue.

5.5.1 Opis szczegółowy

Plik zawiera deklaracje zwiazane z szablonem klasy Queue.

5.6 Dokumentacja pliku stack.hh

Plik zawiera deklaracje zwiazane z szablonem klasy Stack.

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

Komponenty

- class Stack< type >

  Szablon klasy Stack.

5.6.1 Opis szczegółowy

Plik zawiera deklaracje zwiazane z szablonem klasy Stack.

5.7 Dokumentacja pliku stack2.hh

Plik zawiera deklaracje zwiazane z szablonem klasy Stack2.

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

Komponenty

- class Stack2< type >
  Szablon klasy Stack2.

5.7.1 Opis szczegółowy

Plik zawiera deklaracje zwiazane z szablonem klasy Stack2.

5.8 Dokumentacja pliku stOnList.hh

Plik zawiera deklaracje zwiazane z szablonem klasy stOnList.

```
#include <list>
```

Komponenty

- class stOnList< type >
  Szablon klasy stOnList.

Definicje

- #define STOLIST_HH_

5.8.1 Opis szczegółowy

Plik zawiera deklaracje zwiazane z szablonem klasy stOnList.

5.8.2 Dokumentacja definicji

5.8.2.1 #define STOLIST_HH_

Skorowidz