

## Struktury

Wygenerowano przez Doxygen 1.8.4

N, 16 mar 2014 23:22:47

## Spis treści

<b>1</b>	<b>Struktury - podstawowe informacje na temat programu</b>	<b>1</b>
1.1	Opis programu . . . . .	1
1.2	Autor . . . . .	2
<b>2</b>	<b>Indeks klas</b>	<b>2</b>
2.1	Lista klas . . . . .	2
<b>3</b>	<b>Indeks plików</b>	<b>2</b>
3.1	Lista plików . . . . .	2
<b>4</b>	<b>Dokumentacja klas</b>	<b>3</b>
4.1	Dokumentacja szablonu klasy <code>Array&lt; type &gt;</code> . . . . .	3
4.1.1	Opis szczegółowy . . . . .	3
4.1.2	Dokumentacja konstruktora i destruktora . . . . .	4
4.1.3	Dokumentacja funkcji składowych . . . . .	4
4.1.4	Dokumentacja atrybutów składowych . . . . .	5
4.2	Dokumentacja szablonu klasy <code>BenchQueue&lt; queueClass &gt;</code> . . . . .	5
4.2.1	Opis szczegółowy . . . . .	6
4.2.2	Dokumentacja funkcji składowych . . . . .	6
4.3	Dokumentacja szablonu klasy <code>BenchStack&lt; stackClass &gt;</code> . . . . .	6
4.3.1	Opis szczegółowy . . . . .	6
4.3.2	Dokumentacja funkcji składowych . . . . .	7
4.4	Dokumentacja szablonu klasy <code>Queue&lt; type &gt;</code> . . . . .	7
4.4.1	Opis szczegółowy . . . . .	7
4.4.2	Dokumentacja funkcji składowych . . . . .	8
4.4.3	Dokumentacja atrybutów składowych . . . . .	8
4.5	Dokumentacja szablonu klasy <code>Stack&lt; type &gt;</code> . . . . .	8
4.5.1	Opis szczegółowy . . . . .	9
4.5.2	Dokumentacja funkcji składowych . . . . .	9
4.5.3	Dokumentacja atrybutów składowych . . . . .	10
4.6	Dokumentacja szablonu klasy <code>Stack2&lt; type &gt;</code> . . . . .	10
4.6.1	Opis szczegółowy . . . . .	11
4.6.2	Dokumentacja konstruktora i destruktora . . . . .	11
4.6.3	Dokumentacja funkcji składowych . . . . .	11
4.6.4	Dokumentacja atrybutów składowych . . . . .	12
4.7	Dokumentacja szablonu klasy <code>stOnList&lt; type &gt;</code> . . . . .	12
4.7.1	Opis szczegółowy . . . . .	12
4.7.2	Dokumentacja funkcji składowych . . . . .	13
4.7.3	Dokumentacja atrybutów składowych . . . . .	13

<b>5 Dokumentacja plików</b>	<b>14</b>
5.1 Dokumentacja pliku array.hh	14
5.1.1 Opis szczegółowy	14
5.2 Dokumentacja pliku benchmark.hh	14
5.2.1 Opis szczegółowy	14
5.3 Dokumentacja pliku main.cpp	14
5.3.1 Opis szczegółowy	15
5.3.2 Dokumentacja funkcji	15
5.4 Dokumentacja pliku mainpage.dox	15
5.5 Dokumentacja pliku queue.hh	15
5.5.1 Opis szczegółowy	15
5.6 Dokumentacja pliku stack.hh	15
5.6.1 Opis szczegółowy	15
5.7 Dokumentacja pliku stack2.hh	16
5.7.1 Opis szczegółowy	16
5.8 Dokumentacja pliku stOnList.hh	16
5.8.1 Opis szczegółowy	16
5.8.2 Dokumentacja definicji	16
<b>Indeks</b>	<b>17</b>

## 1 Struktury - podstawowe informacje na temat programu

### 1.1 Opis programu

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

#### STOS

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

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

#### STOS2

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

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

#### STOS NA LISCIE

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

Wykonuje podstawowe operacje stosu na liscie.

#### KOLEJKA

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

Wykonuje podstawowe operacje na kolejce (liscie jednokierunkowej).

## 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:

<a href="#">Array&lt; type &gt;</a> Szablon klasy <a href="#">Array</a>	3
<a href="#">BenchQueue&lt; queueClass &gt;</a> Szablon klasy <a href="#">BenchQueue</a>	5
<a href="#">BenchStack&lt; stackClass &gt;</a> Szablon klasy <a href="#">BenchStack</a>	6
<a href="#">Queue&lt; type &gt;</a> Szablon klasy <a href="#">Queue</a>	7
<a href="#">Stack&lt; type &gt;</a> Szablon klasy <a href="#">Stack</a>	8
<a href="#">Stack2&lt; type &gt;</a> Szablon klasy <a href="#">Stack2</a>	10
<a href="#">stOnList&lt; type &gt;</a> Szablon klasy <a href="#">stOnList</a>	12

## 3 Indeks plików

### 3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

<a href="#">array.hh</a> Plik zawiera definicje szablonu klasy <a href="#">Array</a>	14
<a href="#">benchmark.hh</a> Plik zawiera deklaracje zwiazane z szablonem klas <a href="#">BenchStack</a> i <a href="#">BenchQueue</a>	14
<a href="#">main.cpp</a> Plik zawiera glowna funkcje programu	14
<a href="#">queue.hh</a> Plik zawiera deklaracje zwiazane z szablonem klasy <a href="#">Queue</a>	15
<a href="#">stack.hh</a> Plik zawiera deklaracje zwiazane z szablonem klasy <a href="#">Stack</a>	15
<a href="#">stack2.hh</a> Plik zawiera deklaracje zwiazane z szablonem klasy <a href="#">Stack2</a>	16
<a href="#">stOnList.hh</a> Plik zawiera deklaracje zwiazane z szablonem klasy <a href="#">stOnList</a>	16

## 4 Dokumentacja klas

### 4.1 Dokumentacja szablonu klasy `Array< type >`

Szablon klasy `Array`.

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

#### Metody publiczne

- `Array ()`  
*Konstruktor tworzący obiekty klasy `Array`.*
- `~Array ()`  
*Destruktor usuwający obiekt klasy `Array`.*
- `unsigned int size () const`  
*Zwraca długość tablicy.*
- `void resize (unsigned int newSize)`  
*Zmienia ilość elementów tablicy.*
- `void changePlaces (unsigned int a, unsigned int b)`  
*Zamienia miejscami dwa wybrane elementy.*
- `void reverse ()`  
*Odwraca kolejność elementów 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)`  
*Przeciążenie operatora indeksującego.*
- `const type & operator[] (const unsigned int nr) const`
- `Array< type > & operator= (const Array< type > &value)`  
*Przeciążenie operatora przypisania.*
- `Array< type > & operator+ (const Array< type > &value) const`  
*Przeciążenie operatora dodawania.*
- `bool operator== (const Array< type > &value) const`  
*Przeciążenie operatora porównania.*

#### Atrybuty prywatne

- `int * 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 destruktora

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> 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 *nolteration*, unsigned int *problemSize*)  
*Mierzy czas trwania algorytmu wykonywanego na kolejce dla zadanej ilości 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

<i>queueClass</i>	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 *nolteration*, unsigned int *problemSize* )

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

#### Parametry

<i>nolteration</i>	liczba powtorzen wykonania algorytmu
<i>problemSize</i>	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 *nolteration*, unsigned int *problemSize*)  
*Mierzy czas trwania algorytmu wykonywanego na stosie dla zadanej ilości 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

<i>stackClass</i>	klasa stosu dla której przeprowadzamy benchmark
-------------------	---

## 4.3.2 Dokumentacja funkcji składowych

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

Mierzy czas trwania algorytmu wykonywanego na stosie dla zadanej ilości powtorzeń oraz rozmiaru problemu.

## Parametry

<i>nolteration</i>	liczba powtórzeń wykonania algorytmu
<i>problemSize</i>	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&lt; type &gt;

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  
*Udostępnia pierwszy element listy(kolejki)*
- void [enqueue](#) (const type elem)  
*Dodaje element na koniec kolejki.*
- void [dequeue](#) ()  
*Zabiera element z początku listy.*

## Atrybuty prywatne

- std::list< type > [flist](#)

## 4.4.1 Opis szczegółowy

```
template<typename type>class Queue< type >
```

Szablon klasy [Queue](#).

Definiuje funkcje pozwalające na wykonywaniu operacji na kolejce

## Template Parameters

<i>type</i>	typ danych przechowywanych na liście
-------------	--------------------------------------

## 4.4.2 Dokumentacja funkcji składowych

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

Zabiera element z początku listy.

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

Dodaje element na koniec kolejki.

## Parametry

<i>elem</i>	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 - jeśli lista jest pusta  
false - jeśli na liście jest co najmniej 1 element

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

Pobiera rozmiar kolejki.

## Zwraca

ilość elementów kolejki

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

Udostępnia pierwszy element listy(kolejki)

## Zwraca

zawartość 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&lt; type &gt;

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  
*Udostępnia 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 pozwalające na wykonywanie operacji na stosie

## Template Parameters

<code>type</code>	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 - jeśli stos jest pusty  
false - jeśli 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 może być pusty

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

Dodaje element na szczyt stosu.

**Parametry**

<i>elem</i>	element, który 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]`

Udostępnia element ze szczytu stosu.

**Warunek wstępny**

stos nie może być pusty

**Zwraca**

zawartość 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 tworzący 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  
*Udostępnia 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 pozwalające na wykonywanie operacji na stosie

## Template Parameters

<i>type</i>	typ danych przechowywanych na stosie
-------------	--------------------------------------

## 4.6.2 Dokumentacja konstruktora i destruktoru

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

Konstruktor tworzący 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 - jeśli stos jest pusty  
false - jeśli 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 może być pusty

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

Dodaje element na stos.

## Parametry

<i>elem</i>	element, który dodajemy na szczyt stosu
-------------	---

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

Pobiera rozmiar stosu.

## Zwraca

ilość elementów na stosie

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

Udostępnia element ze szczytu stosu.

Warunek wstępny

stos nie może być pusty

Zwraca

zawartość 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 końca listy.*
- bool [isEmpty](#) () const  
*Sprawdza, czy lista jest pusta.*
- const type & [top](#) () const  
*Udostępnia 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 pozwalające na wykonywanie operacji na liście

## Template Parameters

<i>type</i>	typ danych przechowywanych na liście
-------------	--------------------------------------

## 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 - jeśli lista jest pusta  
false - jeśli na liście jest co najmniej 1 element

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

Zabiera element z końca listy.

## Warunek wstępny

lista nie może być pusta

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

Dodaje element na koniec listy.

## Parametry

<i>elem</i>	element dodawany na koniec listy
-------------	----------------------------------

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

Pobiera rozmiar listy.

## Zwraca

liczba elementów listy

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

Udostępnia ostatni element listy.

## Zwraca

zawartość 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 związane 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 związane z szablonem klas [BenchStack](#) i [BenchQueue](#).

### 5.3 Dokumentacja pliku main.cpp

Plik zawiera główną funkcję programu.

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



## Funkcje

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

### 5.3.1 Opis szczegółowy

Plik zawiera główną funkcję programu.

### 5.3.2 Dokumentacja funkcji

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

Główna funkcja programu.

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

## 5.4 Dokumentacja pliku mainpage.dox

## 5.5 Dokumentacja pliku queue.hh

Plik zawiera deklaracje związane z szablonem klasy `Queue`.

```
#include <list>
```

## Komponenty

- `class Queue< type >`  
*Szablon klasy `Queue`.*

### 5.5.1 Opis szczegółowy

Plik zawiera deklaracje związane z szablonem klasy `Queue`.

## 5.6 Dokumentacja pliku stack.hh

Plik zawiera deklaracje związane z szablonem klasy `Stack`.

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

## Komponenty

- `class Stack< type >`  
*Szablon klasy `Stack`.*

### 5.6.1 Opis szczegółowy

Plik zawiera deklaracje związane z szablonem klasy `Stack`.

## 5.7 Dokumentacja pliku stack2.hh

Plik zawiera deklaracje związane z szablonem klasy [Stack2](#).

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

### Komponenty

- class [Stack2](#)< type >  
*Szablon klasy [Stack2](#).*

### 5.7.1 Opis szczegółowy

Plik zawiera deklaracje związane z szablonem klasy [Stack2](#).

## 5.8 Dokumentacja pliku stOnList.hh

Plik zawiera deklaracje związane 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 związane z szablonem klasy [stOnList](#).

### 5.8.2 Dokumentacja definicji

#### 5.8.2.1 #define STOLIST\_HH\_

## Skorowidz

- ~Array
  - Array, [4](#)
- addArrays
  - Array, [4](#)
- addElem
  - Array, [4](#)
- arrSize
  - Stack2, [12](#)
- Array
  - ~Array, [4](#)
  - addArrays, [4](#)
  - addElem, [4](#)
  - Array, [4](#)
  - arraySize, [5](#)
  - changePlaces, [4](#)
  - operator+, [4](#)
  - operator=, [4](#)
  - operator==, [4](#)
  - resize, [5](#)
  - reverse, [5](#)
  - size, [5](#)
  - T, [5](#)
- array
  - Stack, [10](#)
  - Stack2, [12](#)
- Array< type >, [3](#)
- array.hh, [14](#)
- arraySize
  - Array, [5](#)
- BenchQueue
  - benchmark, [6](#)
- BenchQueue< queueClass >, [5](#)
- BenchStack
  - benchmark, [7](#)
- BenchStack< stackClass >, [6](#)
- benchmark
  - BenchQueue, [6](#)
  - BenchStack, [7](#)
- benchmark.hh, [14](#)
- changePlaces
  - Array, [4](#)
- dequeue
  - Queue, [8](#)
- enqueue
  - Queue, [8](#)
- flist
  - Queue, [8](#)
  - stOnList, [13](#)
- isEmpty
  - Queue, [8](#)
- Stack, [9](#)
- Stack2, [11](#)
- stOnList, [13](#)
- main
  - main.cpp, [15](#)
  - main.cpp, [14](#)
  - main, [15](#)
  - mainpage.dox, [15](#)
- operator+
  - Array, [4](#)
- operator=
  - Array, [4](#)
- operator==
  - Array, [4](#)
- pop
  - Stack, [9](#)
  - Stack2, [11](#)
  - stOnList, [13](#)
- push
  - Stack, [9](#)
  - Stack2, [11](#)
  - stOnList, [13](#)
- Queue
  - dequeue, [8](#)
  - enqueue, [8](#)
  - flist, [8](#)
  - isEmpty, [8](#)
  - size, [8](#)
  - top, [8](#)
- Queue< type >, [7](#)
- queue.hh, [15](#)
- resize
  - Array, [5](#)
- reverse
  - Array, [5](#)
- STOLIST\_HH\_
  - stOnList.hh, [16](#)
- size
  - Array, [5](#)
  - Queue, [8](#)
  - Stack, [10](#)
  - Stack2, [11](#)
  - stOnList, [13](#)
- stOnList
  - flist, [13](#)
  - isEmpty, [13](#)
  - pop, [13](#)
  - push, [13](#)
  - size, [13](#)
  - top, [13](#)
- stOnList< type >, [12](#)

- stOnList.hh, [16](#)
  - STOLIST\_HH\_, [16](#)
- Stack
  - array, [10](#)
  - isEmpty, [9](#)
  - pop, [9](#)
  - push, [9](#)
  - size, [10](#)
  - top, [10](#)
- Stack< type >, [8](#)
- stack.hh, [15](#)
- Stack2
  - arrSize, [12](#)
  - array, [12](#)
  - isEmpty, [11](#)
  - pop, [11](#)
  - push, [11](#)
  - size, [11](#)
  - Stack2, [11](#)
  - top, [11](#)
- Stack2< type >, [10](#)
- stack2.hh, [16](#)
- T
  - Array, [5](#)
- top
  - Queue, [8](#)
  - Stack, [10](#)
  - Stack2, [11](#)
  - stOnList, [13](#)