

## Test struktur danych

Wygenerowano przez Doxygen 1.8.6

Cz, 26 mar 2015 11:49:23



# Spis treści

<b>1</b>	<b>Strona główna</b>	<b>1</b>
<b>2</b>	<b>Indeks hierarchiczny</b>	<b>3</b>
2.1	Hierarchia klas	3
<b>3</b>	<b>Indeks klas</b>	<b>5</b>
3.1	Lista klas	5
<b>4</b>	<b>Indeks plików</b>	<b>7</b>
4.1	Lista plików	7
<b>5</b>	<b>Dokumentacja klas</b>	<b>9</b>
5.1	Dokumentacja szablonu klasy <code>Benchmark&lt; T &gt;</code>	9
5.2	Dokumentacja szablonu klasy <code>Element&lt; T &gt;</code>	9
5.2.1	Opis szczegółowy	10
5.2.2	Dokumentacja konstruktora i destruktora	10
5.2.2.1	<code>Element</code>	10
5.2.2.2	<code>~Element</code>	10
5.2.3	Dokumentacja funkcji składowych	10
5.2.3.1	<code>getData</code>	10
5.2.3.2	<code>next</code>	10
5.2.3.3	<code>prev</code>	10
5.2.3.4	<code>setData</code>	11
5.2.3.5	<code>setNext</code>	11
5.2.3.6	<code>setPrev</code>	11
5.3	Dokumentacja szablonu klasy <code>ArrayImplementation::List&lt; T &gt;</code>	11
5.3.1	Opis szczegółowy	12
5.3.2	Dokumentacja konstruktora i destruktora	12
5.3.2.1	<code>List</code>	12
5.3.2.2	<code>~List</code>	12
5.3.3	Dokumentacja funkcji składowych	12
5.3.3.1	<code>isEmpty</code>	12
5.3.3.2	<code>pop</code>	12

5.3.3.3	push	12
5.3.3.4	size	13
5.4	Dokumentacja szablonu klasy <code>ListImplementation::List&lt; T &gt;</code>	13
5.4.1	Opis szczegółowy	13
5.4.2	Dokumentacja konstruktora i destruktoru	14
5.4.2.1	<code>~List</code>	14
5.4.3	Dokumentacja funkcji składowych	14
5.4.3.1	<code>isEmpty</code>	14
5.4.3.2	<code>pop</code>	14
5.4.3.3	<code>push</code>	14
5.4.3.4	<code>size</code>	14
5.5	Dokumentacja szablonu klasy <code>ArrayImplementation::Queue&lt; T &gt;</code>	14
5.5.1	Opis szczegółowy	15
5.5.2	Dokumentacja konstruktora i destruktoru	15
5.5.2.1	<code>Queue</code>	15
5.5.3	Dokumentacja funkcji składowych	15
5.5.3.1	<code>pop</code>	15
5.5.3.2	<code>push</code>	15
5.6	Dokumentacja szablonu klasy <code>ListImplementation::Queue&lt; T &gt;</code>	16
5.6.1	Opis szczegółowy	16
5.6.2	Dokumentacja funkcji składowych	16
5.6.2.1	<code>pop</code>	16
5.6.2.2	<code>push</code>	16
5.7	Dokumentacja szablonu klasy <code>ListImplementation::Stack&lt; T &gt;</code>	17
5.7.1	Opis szczegółowy	17
5.7.2	Dokumentacja funkcji składowych	17
5.7.2.1	<code>pop</code>	17
5.7.2.2	<code>push</code>	17
5.8	Dokumentacja szablonu klasy <code>ArrayImplementation::Stack&lt; T &gt;</code>	18
5.8.1	Opis szczegółowy	18
5.8.2	Dokumentacja konstruktora i destruktoru	18
5.8.2.1	<code>Stack</code>	18
5.8.3	Dokumentacja funkcji składowych	18
5.8.3.1	<code>isEmpty</code>	18
5.8.3.2	<code>pop</code>	19
5.8.3.3	<code>push</code>	19
5.8.3.4	<code>size</code>	19
5.9	Dokumentacja klasy <code>Timer</code>	19
5.9.1	Opis szczegółowy	19
5.9.2	Dokumentacja konstruktora i destruktoru	20

5.9.2.1	Timer	20
5.9.3	Dokumentacja funkcji składowych	20
5.9.3.1	diffTimeMs	20
5.9.3.2	startTimer	20
5.9.3.3	stopTimer	20
<b>6</b>	<b>Dokumentacja plików</b>	<b>21</b>
6.1	Dokumentacja pliku inc/ArrayImplementation/List.h	21
6.1.1	Opis szczegółowy	21
6.2	Dokumentacja pliku inc/LinkedListImplementation/List.h	21
6.2.1	Opis szczegółowy	22
6.3	Dokumentacja pliku inc/ArrayImplementation/Queue.h	22
6.3.1	Opis szczegółowy	22
6.4	Dokumentacja pliku inc/LinkedListImplementation/Queue.h	22
6.4.1	Opis szczegółowy	22
6.5	Dokumentacja pliku inc/ArrayImplementation/Stack.h	23
6.5.1	Opis szczegółowy	23
6.6	Dokumentacja pliku inc/LinkedListImplementation/Stack.h	23
6.6.1	Opis szczegółowy	23
6.7	Dokumentacja pliku inc/Element.h	23
6.7.1	Opis szczegółowy	24
6.8	Dokumentacja pliku inc/Timer.h	24
6.8.1	Opis szczegółowy	24
6.9	Dokumentacja pliku src/main.cpp	24
6.9.1	Opis szczegółowy	24
6.10	Dokumentacja pliku src/Timer.cpp	24
6.10.1	Opis szczegółowy	25
<b>Indeks</b>		<b>26</b>



# Rozdział 1

## Strona główna

Czas wykonywania algorytmu wykonującego dodawanie elementów do podatawowych struktur danych

Program realizuje operacje dodawania n liczby elementów do listy, kolejki i stosu i mierzy czas tych operacji

### Autor

Mateusz Bencer

### Data

2015.03.19

### Wersja

1.0

### Mail:

[209360@pwr.wroc.edu.pl](mailto:209360@pwr.wroc.edu.pl)

test



## Rozdział 2

# Indeks hierarchiczny

### 2.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Benchmark< T > . . . . .	9
Element< T > . . . . .	9
ArrayImplementation::List< T > . . . . .	11
ListImplementation::List< T > . . . . .	13
ListImplementation::Queue< T > . . . . .	16
ListImplementation::Stack< T > . . . . .	17
ArrayImplementation::Queue< T > . . . . .	14
ArrayImplementation::Stack< T > . . . . .	18
Timer . . . . .	19



## Rozdział 3

# Indeks klas

### 3.1 Lista klas

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

<a href="#">Benchmark&lt; T &gt;</a>	9
<a href="#">Element&lt; T &gt;</a>	
Klasa reprezentująca abstrakcyjny "pojemnik na dane"	9
<a href="#">ArrayImplementation::List&lt; T &gt;</a>	
Klasa reprezentująca podstawy konterner danych - Listę zaimplementowaną na tablicy	11
<a href="#">ListImplementation::List&lt; T &gt;</a>	
Klasa reprezentująca podstawy konterner danych z którego korzystają inne - Listę	13
<a href="#">ArrayImplementation::Queue&lt; T &gt;</a>	
Klasa reprezentująca podstawy konterner danych kolejke zaimplementowaną na tablicy	14
<a href="#">ListImplementation::Queue&lt; T &gt;</a>	
Klasa reprezentująca podstawy konterner danych kolejke	16
<a href="#">ListImplementation::Stack&lt; T &gt;</a>	
Klasa reprezentująca podstawy konterner danych - stos	17
<a href="#">ArrayImplementation::Stack&lt; T &gt;</a>	
Klasa reprezentująca podstawy konterner danych - stos w implementacji tablicowej	18
<a href="#">Timer</a>	
Klasa do pomiaru różnicy czasów	19



## Rozdział 4

# Indeks plików

### 4.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

inc/ <b>Benchmark.h</b> . . . . .	??
inc/ <b>Element.h</b>	
Deklaracja i definicja klasy <b>Element</b> . . . . .	23
inc/ <b>Timer.h</b>	
Plik zawierający deklaracje klasy <b>Timer</b> służącej do pomiaru różnicy czasów . . . . .	24
inc/ArrayImplementation/ <b>Increase.h</b> . . . . .	??
inc/ArrayImplementation/ <b>List.h</b>	
Deklaracja klasy Lista (w implementacji opartej na tablicy) . . . . .	21
inc/ArrayImplementation/ <b>Queue.h</b>	
Deklaracja i definicja klasy Queue . . . . .	22
inc/ArrayImplementation/ <b>Stack.h</b>	
Deklaracja i definicja klasy Stack w wersji tablicowej . . . . .	23
inc/LinkedListImplementation/ <b>List.h</b>	
Deklaracja klasy List . . . . .	21
inc/LinkedListImplementation/ <b>Queue.h</b>	
Deklaracja i definicja klasy Queue . . . . .	22
inc/LinkedListImplementation/ <b>Stack.h</b>	
Deklaracja i definicja klasy Stack . . . . .	23
src/ <b>main.cpp</b>	
Plik zawierający sekwencje operacji do mierzenia czasu operacji mnożenia elementów tablicy przez 2 . . . . .	24
src/ <b>Timer.cpp</b>	
Plik zawierający definicje funkcji klasy <b>Timer</b> służącej do pomiaru różnicy czasów . . . . .	24



## Rozdział 5

# Dokumentacja klas

### 5.1 Dokumentacja szablonu klasy `Benchmark< T >`

#### Metody publiczne

- **Benchmark** (unsigned int testPower)
- void **testList** (`ArrayImplementation::List< T > *list`, `ArrayImplementation::Increase inc`)
- void **testQueue** (`ArrayImplementation::Queue< T > *queue`, `ArrayImplementation::Increase inc`)
- void **testStack** (`ArrayImplementation::Stack< T > *stack`, `ArrayImplementation::Increase inc`)
- void **testList** (`ListImplementation::List< T > *list`)
- void **testQueue** (`ListImplementation::Queue< T > *queue`)
- void **testStack** (`ListImplementation::Stack< T > *stack`)

Dokumentacja dla tej klasy została wygenerowana z pliku:

- inc/Benchmark.h

### 5.2 Dokumentacja szablonu klasy `Element< T >`

Klasa reprezentująca abstrakcyjny "pojemnik na dane".

```
#include <Element.h>
```

#### Metody publiczne

- `Element` (T \*data)  
*Konstruktor zapamiętujący adres przechowywanego obiektu.*
- void `setData` (const T \*data)  
*setter do przechowywanej danej*
- T \* `getData` () const  
*getter do przechowywanej danej*
- `Element< T > * next` () const  
*zwraca wskaźnik do kolejnego elementu na liście*
- `Element< T > * prev` () const  
*zwraca wskaźnik do poprzedniego elementu na liście*
- void `setNext` (`Element< T > *next`)  
*ustawia kolejny element listy*
- void `setPrev` (`Element< T > *prev`)

- ustawia poprzedni element listy
- [~Element](#) ()  
zwalnianie elementu przechowywanego przez klasę [Element](#)

### 5.2.1 Opis szczegółowy

```
template<class T>class Element< T >
```

Klasa reprezentująca abstrakcyjny "pojemnik na dane".

Klasa może przechowywać dane zdeterminowane przez typ szablonu, zawiera wskaźnik do kolejnego i poprzedniego elementu (implementacja elementu listy).

### 5.2.2 Dokumentacja konstruktora i destruktora

5.2.2.1 `template<typename T > Element< T >::Element ( T * data )`

Konstruktor zapamiętujący adres przechowywanego obiektu.

Parametry

<code>data</code>	obiekt/zmienna do przechowania
-------------------	--------------------------------

5.2.2.2 `template<typename T > Element< T >::~~Element ( )`

zwalnianie elementu przechowywanego przez klasę [Element](#)

Destruktor zapewniający zwalnianie elementu przechowywanego przez klasę [Element](#)

### 5.2.3 Dokumentacja funkcji składowych

5.2.3.1 `template<typename T > T * Element< T >::getData ( ) const`

getter do przechowywanej danej

Metoda do pobrania wskaźnika przechowywanego danej

Zwraca

wskaźnik do zmiennej przechowywanej przez klasę

5.2.3.2 `template<typename T > Element< T > * Element< T >::next ( ) const`

zwraca wskaźnik do kolejnego elementu na liście

Zwraca

wskaźnik do kolejnego elementu na liście

5.2.3.3 `template<typename T > Element< T > * Element< T >::prev ( ) const`

zwraca wskaźnik do poprzedniego elementu na liście



**Zwraca**

wskaźnik do poprzedniego elementu na liście

**5.2.3.4** `template<typename T > void Element< T >::setData ( const T * data )`

setter do przechowywanej danej

Metoda do ustawiania danej przechowywanej przez klasę

obiekt/zmienna, która będzie przechowywana

**5.2.3.5** `template<typename T > void Element< T >::setNext ( Element< T > * next )`

ustawia kolejny element listy

**Parametry**

<code>next</code>	kolejny element listy
-------------------	-----------------------

**5.2.3.6** `template<typename T > void Element< T >::setPrev ( Element< T > * prev )`

ustawia poprzedni element listy

**Parametry**

<code>prev</code>	poprzedni element listy
-------------------	-------------------------

Dokumentacja dla tej klasy została wygenerowana z pliku:

- inc/[Element.h](#)

**5.3 Dokumentacja szablonu klasy `ArrayImplementation::List< T >`**

Klasa reprezentująca podstawy konterner danych - Listę zaimplementowaną na tablicy.

```
#include <List.h>
```

**Metody publiczne**

- [List](#) ()  
*Konstruktor zerujący aktualny rozmiar i przydzielający pamięć tablicy przechowującej dane w ilości `_MAX_SIZE`.*
- [List](#) (unsigned int max\_size)  
*Konstruktor zerujący rozmiar i przydzielający pamięć tablicy przechowującej dane w ilości określonej przez parametr `max_size`.*
- T \* [pop](#) (unsigned int index)  
*zwraca element o indeksie określonej przez parametr `index` (równocześnie usuwając go z listy)*
- void [push](#) (T \*elem, unsigned int index, Increase inc)  
*dodaje element o indeksie `index` do listy*
- unsigned int [size](#) ()  
*zwraca aktualny rozmiar listy*
- unsigned short [isEmpty](#) ()  
*zwraca 1, gdy lista jest pusta. 0, gdy są jakieś elementy*
- [~List](#) ()  
*destruktor czyszczący listę*

### 5.3.1 Opis szczegółowy

```
template<class T>class ArrayImplementation::List< T >
```

Klasa reprezentująca podstawy konterner danych - Listę zaimplementowaną na tablicy.

Klasa reprezentująca podstawy kontener - listę. Jest to implemetacja listy oparta na tablicy.

### 5.3.2 Dokumentacja konstruktora i destruktora

5.3.2.1 `template<typename T > ArrayImplementation::List< T >::List ( unsigned int max_size )`

Konstruktor zerujący rozmiar i przydzielający pamięć tablicy przechowującej dane w ilości określonej przez parametr *max\_size*.

Parametry

<i>max_size</i>	maksymalna liczba elementów w liście
-----------------	--------------------------------------

5.3.2.2 `template<typename T > ArrayImplementation::List< T >::~~List ( )`

destruktor czyszczący listę

Destruktor usuwawa wszystkie elementy z listy

### 5.3.3 Dokumentacja funkcji składowych

5.3.3.1 `template<typename T > unsigned short ArrayImplementation::List< T >::isEmpty ( )`

zwraca 1, gdy lista jest pusta. 0, gdy są jakieś elementy

Zwraca

zwraca informacje, czy w liście są jakieś elementy

5.3.3.2 `template<typename T > T * ArrayImplementation::List< T >::pop ( unsigned int index )`

zwraca element o indeksie określonej przez parametr *index* (równocześnie usuwając go z listy)

Parametry

<i>index</i>	określa indeks elementu znajdującego się na liście, który zostanie zwrócony
--------------	---

Zwraca

element listy o pozycji *index*

5.3.3.3 `template<typename T > void ArrayImplementation::List< T >::push ( T * elem, unsigned int index, Increase inc )`

dodaje element o indkesie *index* do listy

Warunek wstępny

Indeksowanie zaczyna się od 1

## Parametry

<i>elem</i>	element umieszczany do listy
<i>index</i>	określa pozycję na liście dodawanego elementu (numeracja od 1!)
<i>inc</i>	określa sposób powiększania się listy w razie braku miejsca

## 5.3.3.4 template&lt;typename T &gt; unsigned int ArrayImplementation::List&lt; T &gt;::size ( )

zwraca aktualny rozmiar listy

## Zwraca

rozmiar listy

Dokumentacja dla tej klasy została wygenerowana z pliku:

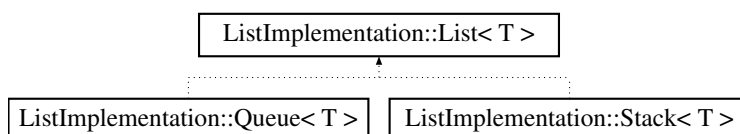
- inc/ArrayImplementation/List.h

## 5.4 Dokumentacja szablonu klasy ListImplementation::List&lt; T &gt;

Klasa reprezentująca podstawy kontener danych z którego korzystają inne - Listę

```
#include <List.h>
```

Diagram dziedziczenia dla ListImplementation::List< T >



## Metody publiczne

- [List](#) ()  
*Konstruktor zerujący pola klasy i przydzielający pamięć na `_head` i `_tail`.*
- [Element](#)< T > \* [pop](#) (Direction dir)  
*zwraca element z początku(zależy od użytej struktury danych) listy*
- void [push](#) ([Element](#)< T > \*elem, Direction dir)  
*dodaje element na początek/koniec(zależy od implementacji) listy*
- unsigned int [size](#) ()  
*zwraca rozmiar użytej struktury danych*
- unsigned short [isEmpty](#) ()  
*zwraca 1, gdy kontener jest pusty, 0 - gdy jest już jakiś element*
- virtual [~List](#) ()  
*wirtualny destruktor czyszczący listę*

## 5.4.1 Opis szczegółowy

```
template<class T>class ListImplementation::List< T >
```

Klasa reprezentująca podstawy kontener danych z którego korzystają inne - Listę

Klasa reprezentująca podstawy kontener - listę. Jest to podstawowa implementacja listy stanowiąca klasę bazową dla listy, stosu i kolejki

## 5.4.2 Dokumentacja konstruktora i destruktora

5.4.2.1 `template<typename T> ListImplementation::List< T>::~~List( ) [virtual]`

wirtualny destruktor czyszczący listę

Destruktor usuwa wszystkie elementy z listy

## 5.4.3 Dokumentacja funkcji składowych

5.4.3.1 `template<typename T> unsigned short ListImplementation::List< T>::isEmpty( )`

zwraca 1, gdy kontener jest pusty, 0 - gdy jest już jakiś element

Zwraca

zwraca informacje, czy w kontenerze są już jakieś elementy

5.4.3.2 `template<typename T> Element< T> * ListImplementation::List< T>::pop( Direction dir )`

zwraca element z początku(zależy od użytej struktury danych) listy

Parametry

<i>dir</i>	określa czy zdjąć element z początku (Front), czy z końca (Back) listy
------------	--

Zwraca

element będący na początku/końcu listy

5.4.3.3 `template<typename T> void ListImplementation::List< T>::push( Element< T> * elem, Direction dir )`

dodaje element na początek/koniec(zależy od implementacji) listy

Parametry

<i>elem</i>	element umieszczany na początku/końcu listy
<i>dir</i>	określa czy włożyć element na początek(Front), na koniec (Back) listy

5.4.3.4 `template<typename T> unsigned int ListImplementation::List< T>::size( )`

zwraca rozmiar użytej struktury danych

Zwraca

rozmiar użytej struktury danych

Dokumentacja dla tej klasy została wygenerowana z pliku:

- inc/LinkedListImplementation/[List.h](#)

## 5.5 Dokumentacja szablonu klasy ArrayImplementation::Queue< T>

Klasa reprezentująca podstawy konterner danych kolejke zaimplementowaną na tablicy.

```
#include <Queue.h>
```

## Metody publiczne

- `Queue ()`  
*Konstruktor alokujący pamięć na kolejkę oraz zerujący indeks elementu na końcu kolejki.*
- `Queue (unsigned int max_size)`  
*Konstruktor alokujący pamięć na kolejkę w ilości max\_size oraz zerujący indeks elementu na końcu kolejki.*
- `T * pop ()`  
*zwraca element z początku kolejki*
- `void push (T *elem, Increase inc)`  
*dodaje element na koniec kolejki*
- `~Queue ()`  
*Destruktor usuwa tablicę przechowującą elementy kolejki.*

### 5.5.1 Opis szczegółowy

```
template<class T>class ArrayImplementation::Queue< T >
```

Klasa reprezentująca podstawy konterner danych kolejke zaimplementowaną na tablicy.

Kolejka jest strukturą danych typu FIFO, First In, First Out; pierwszy na wejściu, pierwszy na wyjściu

### 5.5.2 Dokumentacja konstruktora i destruktora

5.5.2.1 `template<typename T > ArrayImplementation::Queue< T >::Queue ( unsigned int max_size )`

Konstruktor alokujący pamięć na kolejkę w ilości max\_size oraz zerujący indeks elementu na końcu kolejki.

Parametry

<code>max_size</code>	maksymalna liczba elementów w kolejce
-----------------------	---------------------------------------

### 5.5.3 Dokumentacja funkcji składowych

5.5.3.1 `template<typename T > T * ArrayImplementation::Queue< T >::pop ( )`

zwraca element z początku kolejki

Zwraca

element będący na początku kolejki

5.5.3.2 `template<typename T > void ArrayImplementation::Queue< T >::push ( T * elem, Increase inc )`

dodaje element na koniec kolejki

Parametry

<code>elem</code>	element umieszczany na końcu kolejki
<code>inc</code>	określa sposób powiększania się kolejki w razie braku miejsca

Dokumentacja dla tej klasy została wygenerowana z pliku:

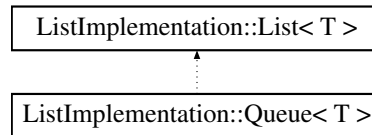
- `inc/ArrayImplementation/Queue.h`

## 5.6 Dokumentacja szablonu klasy ListImplementation::Queue< T >

Klasa reprezentująca podstawy konterner danych kolejke.

```
#include <Queue.h>
```

Diagram dziedziczenia dla ListImplementation::Queue< T >



### Metody publiczne

- [Queue](#) ()  
*Konstruktor wywołujący konstruktor klasy bazowej [List](#).*
- [Element](#)< T > \* [pop](#) ()  
*zwraca element z początku kolejki*
- void [push](#) ([Element](#)< T > \*elem)  
*dodaje element na koniec kolejki*
- virtual [~Queue](#) ()  
*Wywołuje destruktorki klasy bazowej.*

### 5.6.1 Opis szczegółowy

```
template<class T>class ListImplementation::Queue< T >
```

Klasa reprezentująca podstawy konterner danych kolejke.

Kolejka jest strukturą danych typu FIFO, First In, First Out; pierwszy na wejściu, pierwszy na wyjściu

### 5.6.2 Dokumentacja funkcji składowych

5.6.2.1 `template<typename T> Element< T > * ListImplementation::Queue< T >::pop ( )`

zwraca element z początku kolejki

Zwraca

element będący na początku kolejki

5.6.2.2 `template<typename T> void ListImplementation::Queue< T >::push ( Element< T > * elem )`

dodaje element na koniec kolejki

Parametry

<i>elem</i>	element umieszczany na końcu
-------------	------------------------------

Dokumentacja dla tej klasy została wygenerowana z pliku:

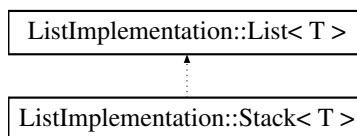
- inc/LinkedListImplementation/[Queue.h](#)

## 5.7 Dokumentacja szablonu klasy ListImplementation::Stack< T >

Klasa reprezentująca podstawy konterner danych - stos.

```
#include <Stack.h>
```

Diagram dziedziczenia dla ListImplementation::Stack< T >



### Metody publiczne

- [Stack](#) ()  
*Konstruktor wywołujący konstruktor klasy bazowej [List](#).*
- [Element](#)< T > \* [pop](#) ()  
*zwraca element z wierzchu stosu*
- void [push](#) ([Element](#)< T > \*elem)  
*dodaje element na wierzch*
- virtual [~Stack](#) ()  
*Wywołuje destruktorki klasy bazowej.*

### 5.7.1 Opis szczegółowy

```
template<class T>class ListImplementation::Stack< T >
```

Klasa reprezentująca podstawy konterner danych - stos.

Kolejka jest strukturą danych typu LIFO, Last In, First Out; ostatni na wejściu, pierwszy na wyjściu

### 5.7.2 Dokumentacja funkcji składowych

5.7.2.1 `template<typename T > Element< T > * ListImplementation::Stack< T >::pop ( )`

zwraca element z wierzchu stosu

Zwraca

element będący na wierzchu stosu

5.7.2.2 `template<typename T > void ListImplementation::Stack< T >::push ( Element< T > * elem )`

dodaje element na wierzch

Parametry

<i>elem</i>	element, który zostanie umieszczony na wierzchu stosu
-------------	---

Dokumentacja dla tej klasy została wygenerowana z pliku:

- inc/LinkedListImplementation/[Stack.h](#)

## 5.8 Dokumentacja szablonu klasy `ArrayImplementation::Stack< T >`

Klasa reprezentująca podstawy konterner danych - stos w implementacji tablicowej.

```
#include <Stack.h>
```

### Metody publiczne

- `Stack ()`  
*Konstruktor przydzielający pamięć na stos w ilości `_MAX_SIZE` oraz zerujący szczyt stosu.*
- `Stack (int max_size)`  
*Konstruktor przydzielający pamięć na stos w ilości `max_size` oraz zerujący szczyt stosu.*
- `Element< T > * pop ()`  
*zwraca element z wierzchu stosu*
- `void push (T *elem, Increase inc)`  
*dodaje element na wierzch*
- `int size ()`  
*zwraca aktualny rozmiar stosu*
- `unsigned short isEmpty ()`  
*zwraca 1, gdy stos jest pusty. 0, gdy są jakieś elementy*
- `~Stack ()`  
*Destruktor usuwa pamięć po tablicy `_elements`.*

### 5.8.1 Opis szczegółowy

```
template<class T>class ArrayImplementation::Stack< T >
```

Klasa reprezentująca podstawy konterner danych - stos w implementacji tablicowej.

Stos jest strukturą danych typu FIFO, First In, First Out; pierwszy na wejściu, pierwszy na wyjściu

### 5.8.2 Dokumentacja konstruktora i destruktora

5.8.2.1 `template<typename T > ArrayImplementation::Stack< T >::Stack ( int max_size )`

Konstruktor przydzielający pamięć na stos w ilości `max_size` oraz zerujący szczyt stosu.

Parametry

<code>max_size</code>	maksymalna liczba elementów na stosie
-----------------------	---------------------------------------

### 5.8.3 Dokumentacja funkcji składowych

5.8.3.1 `template<typename T > unsigned short ArrayImplementation::Stack< T >::isEmpty ( )`

zwraca 1, gdy stos jest pusty. 0, gdy są jakieś elementy

Zwraca

zwraca informacje, czy na stosie są jakieś elementy



5.8.3.2 `template<typename T> Element< T> * ArrayImplementation::Stack< T>::pop ( )`

zwraca element z wierzchu stosu

Zwraca

element będący na wierzchu stosu

5.8.3.3 `template<typename T> void ArrayImplementation::Stack< T>::push ( T * elem, Increase inc )`

dodaje element na wierzch

Parametry

<i>elem</i>	element, który zostanie umieszczony na wierzchu stosu
<i>inc</i>	określa sposób powiększania się stosu w razie braku miejsca

5.8.3.4 `template<typename T> int ArrayImplementation::Stack< T>::size ( )`

zwraca aktualny rozmiar stosu

Zwraca

rozmiar stosu

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [inc/ArrayImplementation/Stack.h](#)

## 5.9 Dokumentacja klasy Timer

Klasa do pomiaru różnicy czasów.

```
#include <Timer.h>
```

### Metody publiczne

- [Timer](#) ()  
*Konstruktor zerujący parametry.*
- void [startTimer](#) ()  
*Zmierzenie czasu rozpoczęcia pomiaru.*
- void [stopTimer](#) ()  
*Zmierzenie czasu zakończenia pomiaru.*
- double [diffTimeMs](#) ()  
*Funkcja zwracająca różnicę czasu pomiędzy czasem rozpoczęcia i zakończenia pomiaru.*

### 5.9.1 Opis szczegółowy

Klasa do pomiaru różnicy czasów.

Klasa pozwala na pomiar czasów w danych momentach oraz na zwrócenie czasu, który upłynął pomiędzy tymi momentami

## 5.9.2 Dokumentacja konstruktora i destruktora

### 5.9.2.1 `Timer::Timer ( )`

Konstruktor zerujący parametry.

Konstruktor ten odpowiada za zerowania zmiennych startu i stopu w celu możliwości późniejszego sprawdzenia, czy pomiary czasu konieczne do wyznaczenia różnicy zostały zrealizowane.

## 5.9.3 Dokumentacja funkcji składowych

### 5.9.3.1 `double Timer::diffTimeMs ( )`

Funkcja zwracająca różnicę czasu pomiędzy czasem rozpoczęcia i zakończenia pomiaru.

Różnica czasu zwracana jest w milisekundach.

#### Warunek wstępny

Czas zakończenia pomiaru musi być większy (późniejszy) od czasu jego rozpoczęcia

#### Zwraca

Zwracana jest różnica czasu rzutowana do typu `double`

### 5.9.3.2 `void Timer::startTimer ( )`

Zmierzenie czasu rozpoczęcia pomiaru.

Funkcja zapamiętuje bieżący czas, jako czas rozpoczęcia pomiaru.

### 5.9.3.3 `void Timer::stopTimer ( )`

Zmierzenie czasu zakończenia pomiaru.

Funkcja zapamiętuje bieżący czas, jako czas zakończenia pomiaru.

Dokumentacja dla tej klasy została wygenerowana z plików:

- [inc/Timer.h](#)
- [src/Timer.cpp](#)

## Rozdział 6

# Dokumentacja plików

### 6.1 Dokumentacja pliku inc/ArrayImplementation/List.h

Deklaracja klasy Lista (w implementacji opartej na tablicy)

```
#include "../Element.h"
#include "Increase.h"
#include <iostream>
```

#### Komponenty

- class `ArrayImplementation::List< T >`

*Klasa reprezentująca podstawy konterner danych - Listę zaimplementowaną na tablicy.*

#### Definicje

- `#define DEFAULT_MAX_SIZE 1`

#### 6.1.1 Opis szczegółowy

Deklaracja klasy Lista (w implementacji opartej na tablicy) List.h

### 6.2 Dokumentacja pliku inc/LinkedListImplementation/List.h

Deklaracja klasy List.

```
#include "../Element.h"
#include <iostream>
```

#### Komponenty

- class `LinkedListImplementation::List< T >`

*Klasa reprezentująca podstawy konterner danych z którego korzystają inne - Listę*

## Wyliczenia

- enum **Direction** { **Front**, **Back** }

*Enumerator przekazywany do funkcji push w celu określenia, czy umieszczamy element na początku lub na końcu listy [na początku (Front), czy na końcu (Back), None - domyślne dla kontenera].*

### 6.2.1 Opis szczegółowy

Deklaracja klasy List. List.h

## 6.3 Dokumentacja pliku inc/ArrayImplementation/Queue.h

Deklaracja i definicja klasy Queue.

```
#include "../Element.h"
#include "Increase.h"
```

## Komponenty

- class [ArrayImplementation::Queue< T >](#)

*Klasa reprezentująca podstawy konterner danych kolejke zaimplementowaną na tablicy.*

## Definicje

- #define **DEFAULT\_MAX\_SIZE** 1

### 6.3.1 Opis szczegółowy

Deklaracja i definicja klasy Queue. Queue.h

## 6.4 Dokumentacja pliku inc/LinkedListImplementation/Queue.h

Deklaracja i definicja klasy Queue.

```
#include "List.h"
```

## Komponenty

- class [ListImplementation::Queue< T >](#)

*Klasa reprezentująca podstawy konterner danych kolejke.*

### 6.4.1 Opis szczegółowy

Deklaracja i definicja klasy Queue. Queue.h

## 6.5 Dokumentacja pliku inc/ArrayImplementation/Stack.h

Deklaracja i definicja klasy Stack w wersji tablicowej.

```
#include "List.h"
#include "../Element.h"
#include "Increase.h"
```

### Komponenty

- class [ArrayImplementation::Stack< T >](#)

*Klasa reprezentująca podstawy konterner danych - stos w implementacji tablicowej.*

### Definicje

- #define **DEFAULT\_MAX\_SIZE** 1

#### 6.5.1 Opis szczegółowy

Deklaracja i definicja klasy Stack w wersji tablicowej. Stack.h

## 6.6 Dokumentacja pliku inc/LinkedListImplementation/Stack.h

Deklaracja i definicja klasy Stack.

```
#include "List.h"
```

### Komponenty

- class [ListImplementation::Stack< T >](#)

*Klasa reprezentująca podstawy konterner danych - stos.*

#### 6.6.1 Opis szczegółowy

Deklaracja i definicja klasy Stack. Stack.h

## 6.7 Dokumentacja pliku inc/Element.h

Deklaracja i definicja klasy [Element](#).

```
#include <stddef.h>
```

### Komponenty

- class [Element< T >](#)

*Klasa reprezentująca abstrakcyjny "pojemnik na dane".*

### 6.7.1 Opis szczegółowy

Deklaracja i definicja klasy [Element](#). [Element.h](#)

## 6.8 Dokumentacja pliku inc/Timer.h

Plik zawierający deklaracje klasy [Timer](#) służącej do pomiaru różnicy czasów.

```
#include <ctime>
```

### Komponenty

- class [Timer](#)

*Klasa do pomiaru różnicy czasów.*

### 6.8.1 Opis szczegółowy

Plik zawierający deklaracje klasy [Timer](#) służącej do pomiaru różnicy czasów. [Timer.h](#)

## 6.9 Dokumentacja pliku src/main.cpp

Plik zawierający sekwencje operacji do mierzenia czasu operacji mnożenia elementów tablicy przez 2.

```
#include <iostream>
#include "../inc/LinkedListImplementation/List.h"
#include "../inc/LinkedListImplementation/Queue.h"
#include "../inc/LinkedListImplementation/Stack.h"
#include "../inc/ArrayImplementation/List.h"
#include "../inc/ArrayImplementation/Queue.h"
#include "../inc/ArrayImplementation/Stack.h"
#include "../inc/Benchmark.h"
```

### Funkcje

- int **main** ()

### 6.9.1 Opis szczegółowy

Plik zawierający sekwencje operacji do mierzenia czasu operacji mnożenia elementów tablicy przez 2. [main.cpp](#)

## 6.10 Dokumentacja pliku src/Timer.cpp

Plik zawierający definicje funkcji klasy [Timer](#) służącej do pomiaru różnicy czasów.

```
#include "../inc/Timer.h"
#include <iostream>
```

### 6.10.1 Opis szczegółowy

Plik zawierający definicje funkcji klasy [Timer](#) służącej do pomiaru różnicy czasów. [Timer.cpp](#)

# Skorowidz

- ~Element
  - Element, [10](#)
- ~List
  - ArrayImplementation::List, [12](#)
  - ListImplementation::List, [14](#)
- ArrayImplementation::List
  - ~List, [12](#)
  - isEmpty, [12](#)
  - List, [12](#)
  - pop, [12](#)
  - push, [12](#)
  - size, [13](#)
- ArrayImplementation::List< T >, [11](#)
- ArrayImplementation::Queue
  - pop, [15](#)
  - push, [15](#)
  - Queue, [15](#)
- ArrayImplementation::Queue< T >, [14](#)
- ArrayImplementation::Stack
  - isEmpty, [18](#)
  - pop, [18](#)
  - push, [19](#)
  - size, [19](#)
  - Stack, [18](#)
- ArrayImplementation::Stack< T >, [18](#)
- Benchmark< T >, [9](#)
- diffTimeMs
  - Timer, [20](#)
- Element
  - ~Element, [10](#)
  - Element, [10](#)
  - getData, [10](#)
  - next, [10](#)
  - prev, [10](#)
  - setData, [11](#)
  - setNext, [11](#)
  - setPrev, [11](#)
- Element< T >, [9](#)
- getData
  - Element, [10](#)
- inc/ArrayImplementation/List.h, [21](#)
- inc/ArrayImplementation/Queue.h, [22](#)
- inc/ArrayImplementation/Stack.h, [23](#)
- inc/Element.h, [23](#)
- inc/LinkedListImplementation/List.h, [21](#)
- inc/LinkedListImplementation/Queue.h, [22](#)
- inc/LinkedListImplementation/Stack.h, [23](#)
- inc/Timer.h, [24](#)
- isEmpty
  - ArrayImplementation::List, [12](#)
  - ArrayImplementation::Stack, [18](#)
  - ListImplementation::List, [14](#)
- List
  - ArrayImplementation::List, [12](#)
- ListImplementation::List
  - ~List, [14](#)
  - isEmpty, [14](#)
  - pop, [14](#)
  - push, [14](#)
  - size, [14](#)
- ListImplementation::List< T >, [13](#)
- ListImplementation::Queue
  - pop, [16](#)
  - push, [16](#)
- ListImplementation::Queue< T >, [16](#)
- ListImplementation::Stack
  - pop, [17](#)
  - push, [17](#)
- ListImplementation::Stack< T >, [17](#)
- next
  - Element, [10](#)
- pop
  - ArrayImplementation::List, [12](#)
  - ArrayImplementation::Queue, [15](#)
  - ArrayImplementation::Stack, [18](#)
  - ListImplementation::List, [14](#)
  - ListImplementation::Queue, [16](#)
  - ListImplementation::Stack, [17](#)
- prev
  - Element, [10](#)
- push
  - ArrayImplementation::List, [12](#)
  - ArrayImplementation::Queue, [15](#)
  - ArrayImplementation::Stack, [19](#)
  - ListImplementation::List, [14](#)
  - ListImplementation::Queue, [16](#)
  - ListImplementation::Stack, [17](#)
- Queue
  - ArrayImplementation::Queue, [15](#)
- setData



- Element, [11](#)
- setNext
  - Element, [11](#)
- setPrev
  - Element, [11](#)
- size
  - ArrayImplementation::List, [13](#)
  - ArrayImplementation::Stack, [19](#)
  - ListImplementation::List, [14](#)
- src/Timer.cpp, [24](#)
- src/main.cpp, [24](#)
- Stack
  - ArrayImplementation::Stack, [18](#)
- startTimer
  - Timer, [20](#)
- stopTimer
  - Timer, [20](#)
- Timer, [19](#)
  - diffTimeMs, [20](#)
  - startTimer, [20](#)
  - stopTimer, [20](#)
  - Timer, [20](#)