

Lista, Stos, Kolejka

0.1

Wygenerowano przez Doxygen 1.8.6

N, 3 kwi 2016 18:39:22



# Spis treści

<b>1</b>	<b>Indeks hierarchiczny</b>	<b>1</b>
1.1	Hierarchia klas . . . . .	1
<b>2</b>	<b>Indeks klas</b>	<b>3</b>
2.1	Lista klas . . . . .	3
<b>3</b>	<b>Indeks plików</b>	<b>5</b>
3.1	Lista plików . . . . .	5
<b>4</b>	<b>Dokumentacja klas</b>	<b>7</b>
4.1	Dokumentacja szablonu klasy <code>IList&lt; Typ &gt;</code> . . . . .	7
4.1.1	Opis szczegółowy . . . . .	7
4.2	Dokumentacja szablonu klasy <code>IQueue&lt; Typ &gt;</code> . . . . .	7
4.2.1	Opis szczegółowy . . . . .	8
4.3	Dokumentacja szablonu klasy <code>IStack&lt; Typ &gt;</code> . . . . .	8
4.3.1	Opis szczegółowy . . . . .	8
4.4	Dokumentacja szablonu klasy <code>List&lt; Typ &gt;</code> . . . . .	9
4.4.1	Opis szczegółowy . . . . .	10
4.4.2	Dokumentacja funkcji składowych . . . . .	10
4.4.2.1	<code>add</code> . . . . .	10
4.4.2.2	<code>get</code> . . . . .	10
4.4.2.3	<code>isEmpty</code> . . . . .	10
4.4.2.4	<code>remove</code> . . . . .	11
4.4.2.5	<code>size</code> . . . . .	11
4.5	Dokumentacja szablonu klasy <code>Queue&lt; Typ &gt;</code> . . . . .	11
4.5.1	Opis szczegółowy . . . . .	13
4.5.2	Dokumentacja funkcji składowych . . . . .	13
4.5.2.1	<code>dequeue</code> . . . . .	13
4.5.2.2	<code>enqueue</code> . . . . .	13
4.5.2.3	<code>front</code> . . . . .	13
4.5.2.4	<code>isEmpty</code> . . . . .	13
4.5.2.5	<code>size</code> . . . . .	14

4.6	Dokumentacja szablonu klasy Stack< Typ > . . . . .	14
4.6.1	Opis szczegółowy . . . . .	15
4.6.2	Dokumentacja funkcji składowych . . . . .	15
4.6.2.1	isEmpty . . . . .	15
4.6.2.2	pop . . . . .	16
4.6.2.3	push . . . . .	16
4.6.2.4	size . . . . .	16
4.6.2.5	top . . . . .	16
4.7	Dokumentacja klasy Stoper . . . . .	17
4.7.1	Opis szczegółowy . . . . .	18
<b>5</b>	<b>Dokumentacja plików</b>	<b>19</b>
5.1	Dokumentacja pliku IList.hh . . . . .	19
5.1.1	Opis szczegółowy . . . . .	20
5.2	Dokumentacja pliku IQueue.hh . . . . .	20
5.2.1	Opis szczegółowy . . . . .	21
5.3	Dokumentacja pliku IStack.hh . . . . .	21
5.3.1	Opis szczegółowy . . . . .	21
5.4	Dokumentacja pliku List.hh . . . . .	22
5.4.1	Opis szczegółowy . . . . .	22
5.5	Dokumentacja pliku Queue.hh . . . . .	22
5.5.1	Opis szczegółowy . . . . .	23
5.6	Dokumentacja pliku Stack.hh . . . . .	23
<b>Indeks</b>		<b>25</b>

# Rozdział 1

## Indeks hierarchiczny

### 1.1 Hierarchia klas

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

ICollection< Typ > . . . . .	7
List< Typ > . . . . .	9
ICollection< Typ > . . . . .	7
Queue< Typ > . . . . .	11
ICollection< Typ > . . . . .	8
Stack< Typ > . . . . .	14
ICollection	
Stack . . . . .	17



## Rozdział 2

# Indeks klas

### 2.1 Lista klas

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

<a href="#">IList&lt; Typ &gt;</a>		
Interfejs listy	.....	7
<a href="#">IQueue&lt; Typ &gt;</a>		
Interfejs kolejki	.....	7
<a href="#">IStack&lt; Typ &gt;</a>		
Interfejs stosu	.....	8
<a href="#">List&lt; Typ &gt;</a>		
Szablon listy	.....	9
<a href="#">Queue&lt; Typ &gt;</a>		
Szablon kolejki	.....	11
<a href="#">Stack&lt; Typ &gt;</a>		
Szablon stosu	.....	14
<a href="#">Stoper</a>	.....	17





## Rozdział 3

# Indeks plików

### 3.1 Lista plików

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

<a href="#">IList.hh</a>	Definicja klasy <a href="#">IList</a> . . . . .	19
<a href="#">IQueue.hh</a>	Definicja klasy <a href="#">IQueue</a> . . . . .	20
<a href="#">IStack.hh</a>	Definicja klasy <a href="#">IStack</a> . . . . .	21
<a href="#">List.hh</a>	Definicja klasy <a href="#">List</a> . . . . .	22
<a href="#">main.cpp</a>	. . . . .	??
<a href="#">Queue.hh</a>	Definicja klasy <a href="#">Queue</a> . . . . .	22
<a href="#">Stack.hh</a>	Definicja klasy <a href="#">Stack</a> W pliku znajduje się klasa <a href="#">Stack</a> . . . . .	23
<a href="#">Stoper.cpp</a>	. . . . .	??
<a href="#">Stoper.hh</a>	. . . . .	??



## Rozdział 4

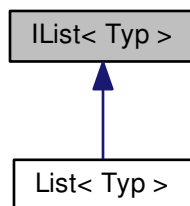
# Dokumentacja klas

### 4.1 Dokumentacja szablonu klasy IList< Typ >

Interfejs listy.

```
#include <IList.hh>
```

Diagram dziedziczenia dla IList< Typ >



#### 4.1.1 Opis szczegółowy

```
template<typename Typ>class IList< Typ >
```

Na liste możemy wpisywać na każde miejsce i usuwać z każdego miejsca

Definicja w linii 18 pliku IList.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

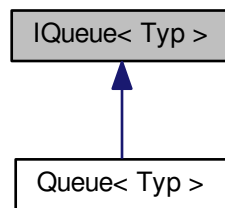
- [IList.hh](#)

### 4.2 Dokumentacja szablonu klasy IQueue< Typ >

Interfejs kolejki.

```
#include <IQueue.hh>
```

Diagram dziedziczenia dla `IQueue< Typ >`



#### 4.2.1 Opis szczegółowy

```
template<typename Typ>class IQueue< Typ >
```

Kolejka jest strukturą typu FIFO (First-In-Fist-Out).

Definicja w linii 17 pliku `IQueue.hh`.

Dokumentacja dla tej klasy została wygenerowana z pliku:

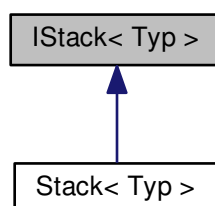
- [IQueue.hh](#)

### 4.3 Dokumentacja szablonu klasy `IStack< Typ >`

Interfejs stosu.

```
#include <IStack.hh>
```

Diagram dziedziczenia dla `IStack< Typ >`



#### 4.3.1 Opis szczegółowy

```
template<typename Typ>class IStack< Typ >
```

Stos jest strukturą typu LIFO (Last-In-Fist-Out).

Definicja w linii 19 pliku IStack.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [IStack.hh](#)

## 4.4 Dokumentacja szablonu klasy List< Typ >

Szablon listy.

```
#include <List.hh>
```

Diagram dziedziczenia dla List< Typ >

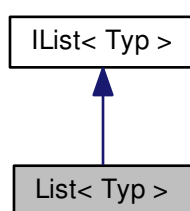
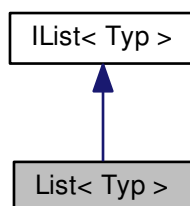


Diagram współpracy dla List< Typ >:



### Metody publiczne

- void [add](#) (Typ item, int index)  
*Zapisywanie na liście.*
- Typ [remove](#) (int index) throw (EmptyListException)  
*Ściąganie z listy.*
- int [size](#) ()  
*Rozmiar listy.*
- bool [isEmpty](#) ()  
*Czy pusty?*

- Typ `get` (int index)  
*Element listy.*

## Atrybuty chronione

- `tablica1D< Typ >` **Tablica**

### 4.4.1 Opis szczegółowy

`template<typename Typ>class List< Typ >`

Na liste możemy wpisywać na każde miejsce i usuwać z każdego miejsca

Definicja w linii 18 pliku List.hh.

### 4.4.2 Dokumentacja funkcji składowych

4.4.2.1 `template<typename Typ > void List< Typ >::add ( Typ item, int index )` `[inline],[virtual]`

Wkłada element na dowolne miejsce na liście, jeżeli int index jest większy minimum o 2 od indexu ostatniego elementu to przestrzeń między elementami zostaje uzupełniona przez wartości 0. Jeżeli int index wskazuje na miejsce, gdzie znajduje się element to zostaje on przepisany na kolejną pozycję, a na jego miejsce zostaje wpisany item

Parametry

<code>in</code>	<code>item</code>	- element, który chcemy umieścić na liście
<code>in</code>	<code>index</code>	- miejsce, gdzie chcemy zapisać element (liczony od 0)

Implementuje `IList< Typ >`.

Definicja w linii 34 pliku List.hh.

4.4.2.2 `template<typename Typ > Typ List< Typ >::get ( int index )` `[inline],[virtual]`

Dostęp do dowolnego elementu listy

Wyjątki

<code>EmptyListException</code>	wyjątek pustej listy, wyrzucany gdy chcemy odczytać element o indexie którego nie ma na liście
---------------------------------	--

Parametry

<code>in</code>	<code>index</code>	- Numer elementu, który chcemy odczytać, gdzie index 0 to pierwszy element
-----------------	--------------------	--

Zwraca

Zwraca element, bez jego usuwania

Implementuje `IList< Typ >`.

Definicja w linii 84 pliku List.hh.

4.4.2.3 `template<typename Typ > bool List< Typ >::isEmpty ( )` `[inline],[virtual]`

Sprawdza czy na liście znajdują się elementy

Zwracane wartości

<i>true</i>	- lista pusty
<i>false</i>	- na liście są elementy

Implementuje [IList< Typ >](#).

Definicja w linii 70 pliku List.hh.

```
4.4.2.4 template<typename Typ > Typ List< Typ >::remove ( int index ) throw EmptyListException) [inline],  
[virtual]
```

Usuwa element z listy

Wyjątki

<i>EmptyListException</i>	wyjątek pustej listy, wyrzucany gdy chcemy usunąć element o indexie którego nie ma na liście
---------------------------	--

Zwraca

Zwraca usunięty element

Implementuje [IList< Typ >](#).

Definicja w linii 46 pliku List.hh.

```
4.4.2.5 template<typename Typ > int List< Typ >::size ( ) [inline],[virtual]
```

Rozmiar listy jest liczbą całkowitą liczoną od 0, gdy lista pusty,

Zwraca

Zwraca liczbę elementów zapisanych na liście

Implementuje [IList< Typ >](#).

Definicja w linii 61 pliku List.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [List.hh](#)

## 4.5 Dokumentacja szablonu klasy Queue< Typ >

Szablon kolejki.

```
#include <Queue.hh>
```

Diagram dziedziczenia dla Queue< Typ >

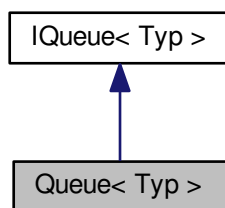
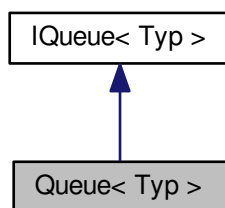


Diagram współpracy dla Queue< Typ >:



## Metody publiczne

- int `size` ()  
*Rozmiar kolejki.*
- bool `isEmpty` ()  
*Czy pusty?*
- void `enqueue` (Typ item)  
*Wkładanie do kolejki.*
- Typ `dequeue` () throw (EmptyQueueException)  
*Wychodzenie z kolejki.*
- Typ `front` () throw (EmptyQueueException)  
*Front kolejki.*

## Atrybuty chronione

- `tablica1D< Typ >` **Tablica**



### 4.5.1 Opis szczegółowy

template<class Typ>class Queue< Typ >

Kolejka jest strukturą typu FIFO (First-In-First-Out), zaimplementowana na tablicy dynamicznej

Definicja w linii 19 pliku Queue.hh.

### 4.5.2 Dokumentacja funkcji składowych

4.5.2.1 template<class Typ > Typ Queue< Typ >::dequeue ( ) throw EmptyQueueException) [inline],  
[virtual]

Usuwa element z początku kolejki

Wyjątki

<i>EmptyQueueException</i>	wyjątek pustej kolejki
----------------------------	------------------------

Zwraca

Zwraca usunięty element

Implementuje [IQueue< Typ >](#).

Definicja w linii 63 pliku Queue.hh.

4.5.2.2 template<class Typ > void Queue< Typ >::enqueue ( Typ item ) [inline], [virtual]

Umieszcza element na końcu kolejki

Parametry

in	item	- element, który chcemy umieścić w kolejce
----	------	--

Implementuje [IQueue< Typ >](#).

Definicja w linii 52 pliku Queue.hh.

4.5.2.3 template<class Typ > Typ Queue< Typ >::front ( ) throw EmptyQueueException) [inline], [virtual]

Sprawdza co znajduje się na przodzie kolejki

Wyjątki

<i>EmptyQueueException</i>	wyjątek pustej kolejki
----------------------------	------------------------

Zwraca

Zwraca pierwszy element w kolejce, bez jego usuwania

Implementuje [IQueue< Typ >](#).

Definicja w linii 78 pliku Queue.hh.

4.5.2.4 template<class Typ > bool Queue< Typ >::isEmpty ( ) [inline], [virtual]

Sprawdza czy w kolejce znajdują się elementy

## Zwracane wartości

<i>true</i>	- kolejka pusty
<i>false</i>	- w kolejce są elementy

Implementuje [IQueue< Typ >](#).

Definicja w linii 40 pliku Queue.hh.

4.5.2.5 `template<class Typ> int Queue< Typ >::size ( ) [inline],[virtual]`

Rozmiar kolejki jest liczbą całkowitą liczoną od 0, gdy stos pusty,

## Zwraca

Zwraca liczbę elementów zapisanych w kolejce

Implementuje [IQueue< Typ >](#).

Definicja w linii 31 pliku Queue.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Queue.hh](#)

## 4.6 Dokumentacja szablonu klasy Stack< Typ >

Szablon stosu.

```
#include <Stack.hh>
```

Diagram dziedziczenia dla Stack< Typ >

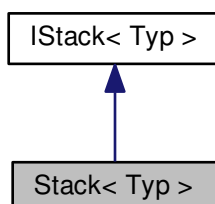
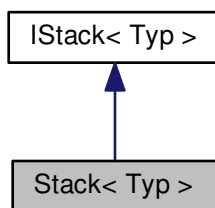


Diagram współpracy dla Stack< Typ >:



### Metody publiczne

- int `size` ()  
*Rozmiar stosu.*
- bool `isEmpty` ()  
*Czy pusty?*
- void `push` (Typ item)  
*Wkładanie na stos.*
- Typ `pop` () throw (EmptyStackException)  
*Ściąganie ze stosu.*
- Typ `top` () throw (EmptyStackException)  
*Szczyt stosu.*

### Atrybuty chronione

- `tablica1D< Typ >` **Tablica**

#### 4.6.1 Opis szczegółowy

```
template<class Typ>class Stack< Typ >
```

Stos jest strukturą typu LIFO (Last-In-Fist-Out), zaimplementowany na tablicy dynamicznej  
Definicja w linii 16 pliku Stack.hh.

#### 4.6.2 Dokumentacja funkcji składowych

4.6.2.1 `template<class Typ> bool Stack< Typ >::isEmpty ( )` `[inline], [virtual]`

Sprawdza czy na stosie znajdują się elementy

Zwracane wartości

---

<i>true</i>	- stos pusty
<i>false</i>	- na stosie są elementy

Implementuje [IStack< Typ >](#).

Definicja w linii 38 pliku Stack.hh.

**4.6.2.2** `template<class Typ> Typ Stack< Typ >::pop ( ) throw EmptyStackException) [inline],[virtual]`

Usuwa element ze szczytu stosu

Wyjątki

<i>EmptyStackException</i>	wyjątek pustego stosu
----------------------------	-----------------------

Zwraca

Zwraca usunięty element

Implementuje [IStack< Typ >](#).

Definicja w linii 62 pliku Stack.hh.

**4.6.2.3** `template<class Typ> void Stack< Typ >::push ( Typ item ) [inline],[virtual]`

Wkłada element na szczyt stosu

Parametry

<i>in</i>	<i>item</i>	- element, który chcemy umieścić na stosie
-----------	-------------	--

Implementuje [IStack< Typ >](#).

Definicja w linii 50 pliku Stack.hh.

**4.6.2.4** `template<class Typ> int Stack< Typ >::size ( ) [inline],[virtual]`

Rozmiar stosu jest liczbą całkowitą liczoną od 0, gdy stos pusty,

Zwraca

Zwraca liczbę elementów zapisanych na stosie

Implementuje [IStack< Typ >](#).

Definicja w linii 28 pliku Stack.hh.

**4.6.2.5** `template<class Typ> Typ Stack< Typ >::top ( ) throw EmptyStackException) [inline],[virtual]`

Element na szczycie stosu

Wyjątki

<i>EmptyStackException</i>	wyjątek pustego stosu
----------------------------	-----------------------

Zwraca

Zwraca element ze szczytu, bez jego usuwania

Implementuje `IStack< Typ >`.

Definicja w linii 77 pliku Stack.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Stack.hh](#)

## 4.7 Dokumentacja klasy Stoper

Diagram dziedziczenia dla Stoper

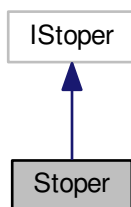
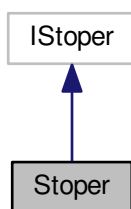


Diagram współpracy dla Stoper:



### Metody publiczne

- void **start** ()
- void **stop** ()
- double **getElapsedTime** ()
- bool **dumpToFile** (ofstream &Plik)

#### 4.7.1 Opis szczegółowy

Definicja w linii 6 pliku Stoper.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- Stoper.hh
- Stoper.cpp

## Rozdział 5

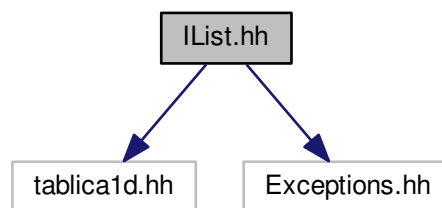
# Dokumentacja plików

### 5.1 Dokumentacja pliku IList.hh

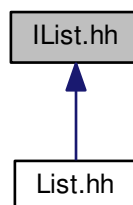
Definicja klasy [IList](#).

```
#include "tablica1d.hh"  
#include "Exceptions.hh"
```

Wykres zależności załączania dla IList.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



## Komponenty

- class `IList< Typ >`  
*Interfejs listy.*

### 5.1.1 Opis szczegółowy

Plik zawiera definicje interfejsu listy

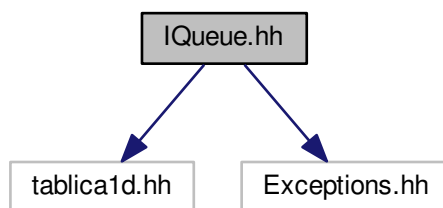
Definicja w pliku [IList.hh](#).

## 5.2 Dokumentacja pliku IQueue.hh

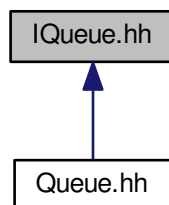
Definicja klasy [IQueue](#).

```
#include "tablica1d.hh"  
#include "Exceptions.hh"
```

Wykres zależności załączania dla IQueue.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



## Komponenty

- class `IQueue< Typ >`  
*Interfejs kolejki.*



### 5.2.1 Opis szczegółowy

Plik zawiera klasę abstrakcyjną, która jest interfejsem kolejki

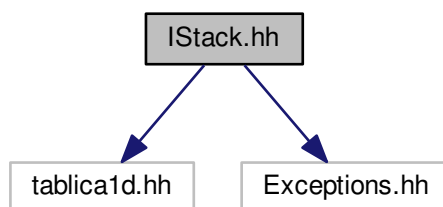
Definicja w pliku [IQueue.hh](#).

## 5.3 Dokumentacja pliku IStack.hh

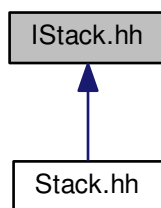
Definicja klasy [IStack](#).

```
#include "tablica1d.hh"  
#include "Exceptions.hh"
```

Wykres zależności załączania dla IStack.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



### Komponenty

- class [IStack< Typ >](#)  
*Interfejs stosu.*

### 5.3.1 Opis szczegółowy

Plik zawiera definicję szablonu klasy [IStack](#), który jest interfejsem stosu

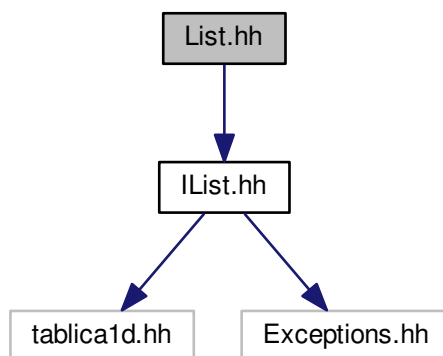
Definicja w pliku [IStack.hh](#).

## 5.4 Dokumentacja pliku List.hh

Definicja klasy [List](#).

```
#include "IList.hh"
```

Wykres zależności załączania dla List.hh:



### Komponenty

- class [List](#)< [Typ](#) >

*Szablon listy.*

#### 5.4.1 Opis szczegółowy

Plik zawiera definicje szablonu klasy [IList](#), który jest interfejsem listy

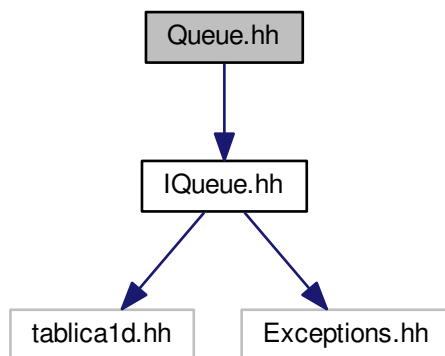
Definicja w pliku [List.hh](#).

## 5.5 Dokumentacja pliku Queue.hh

Definicja klasy [Queue](#).

```
#include "IQueue.hh"
```

Wykres zależności załączania dla Queue.hh:



## Komponenty

- class [Queue](#)< [Typ](#) >

*Szablon kolejki.*

### 5.5.1 Opis szczegółowy

Plik zawiera definicje szablону klasy [IQueue](#), który jest interfejsem kolejki

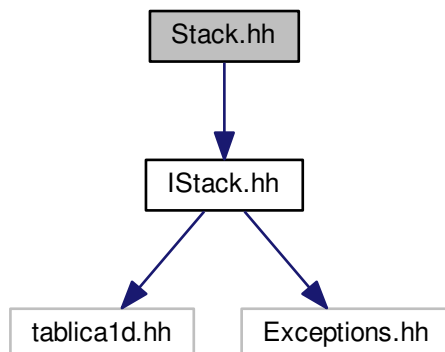
Definicja w pliku [Queue.hh](#).

## 5.6 Dokumentacja pliku Stack.hh

Definicja klasy [Stack](#) W pliku znajduje się klasa [Stack](#).

```
#include "IStack.hh"
```

Wykres zależności załączania dla Stack.hh:



## Komponenty

- class `Stack< Typ >`  
*Szablon stosu.*

# Skorowidz

add  
    List, [10](#)

dequeue  
    Queue, [13](#)

enqueue  
    Queue, [13](#)

front  
    Queue, [13](#)

get  
    List, [10](#)

IList< Typ >, [7](#)  
IList.hh, [19](#)  
IQueue< Typ >, [7](#)  
IQueue.hh, [20](#)  
IStack< Typ >, [8](#)  
IStack.hh, [21](#)  
isEmpty  
    List, [10](#)  
    Queue, [13](#)  
    Stack, [15](#)

List  
    add, [10](#)  
    get, [10](#)  
    isEmpty, [10](#)  
    remove, [11](#)  
    size, [11](#)  
List< Typ >, [9](#)  
List.hh, [22](#)

pop  
    Stack, [16](#)

push  
    Stack, [16](#)

Queue  
    dequeue, [13](#)  
    enqueue, [13](#)  
    front, [13](#)  
    isEmpty, [13](#)  
    size, [14](#)  
Queue< Typ >, [11](#)  
Queue.hh, [22](#)

remove  
    List, [11](#)

size  
    List, [11](#)  
    Queue, [14](#)  
    Stack, [16](#)

Stack  
    isEmpty, [15](#)  
    pop, [16](#)  
    push, [16](#)  
    size, [16](#)  
    top, [16](#)  
Stack< Typ >, [14](#)  
Stack.hh, [23](#)  
Stoper, [17](#)

top  
    Stack, [16](#)