Lista, Stos, Kolejka

0.1

Wygenerowano przez Doxygen 1.8.6

N, 3 kwi 2016 18:39:22

Spis treści

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

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

Rozdział 2

Indeks klas

2.1 Lista klas

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

Rozdział 3

Indeks plików

3.1 Lista plików

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

Rozdział 4

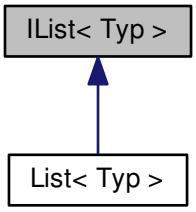
Dokumentacja klas

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 mozemy wpisywac na każde miejsce i usuwać z każdego miejsca

Definicja w linii 18 pliku IList.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

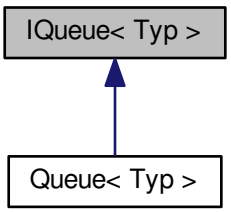
- IList.hh

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:

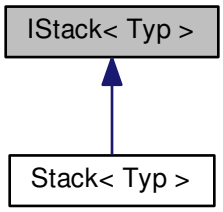
- IQueue.hh

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 >

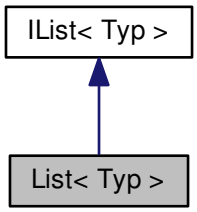
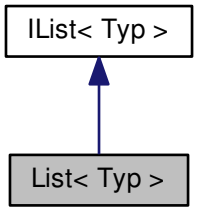


Diagram współpracy dla List< Typ >:



Metody publiczne

- void add (Typ item, int index)

  Zapisywanie na liste.
- 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 mozemy wpisywac 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 pozycje, a na jego miejsce zostaje wpisany item

Parametry

in	*item*	- element, który chcemy umieścić na liście
in	*index*	- 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

EmptyListException	wyjątek pustej listy, wyrzucany gdy chcemy odczytać element o indexie którego nie ma na liście

Parametry

in	*index*	- 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

true	- lista pusty
false	- 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

EmptyListException	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 >

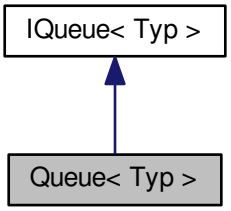
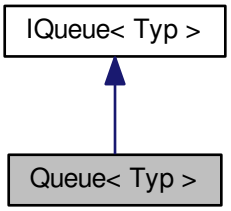


Diagram współpracy dla Queue< Typ >:



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-Fist-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 poczatku kolejki

Wyjątki

EmptyQueueException	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

EmptyQueueException	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

true	- kolejka pusty
false	- 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 >

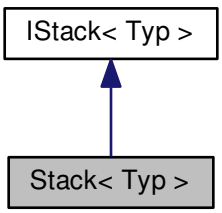
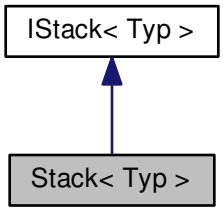


Diagram współpracy dla Stack< Typ >:



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

true	- stos pusty
false	- 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

EmptyStackException	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

in	*item*	- 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 szcycie stosu

Wyjątki

EmptyStackException	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

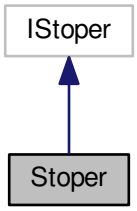
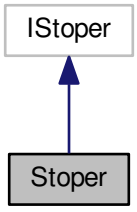


Diagram współpracy dla Stoper:



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

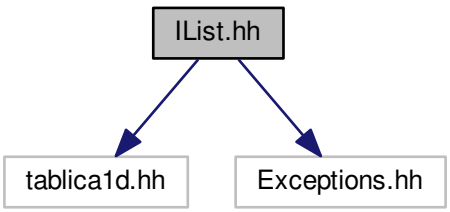
Dokumentacja plików

5.1 Dokumentacja pliku IList.hh

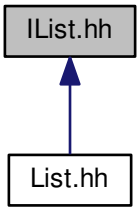
Definicja klasy IList.

```
#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

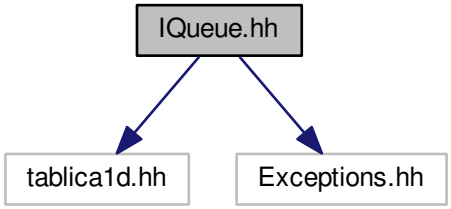
Definicja w pliku IList.hh.

5.2 Dokumentacja pliku IQueue.hh

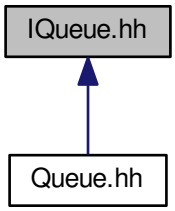
Definicja klasy IQueue.

```
#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 klase abstrakcyjną, która jest interfejsem kolejki

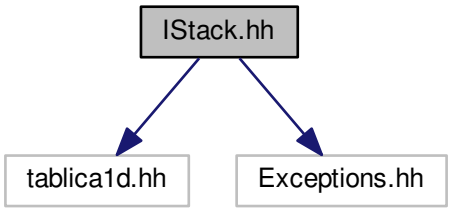
Definicja w pliku IQueue.hh.

5.3 Dokumentacja pliku IStack.hh

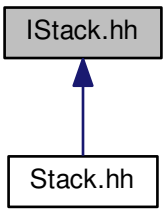
Definicja klasy IStack.

```
#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 definicje szablonu klasy IStack, który jest interfejsem stosu

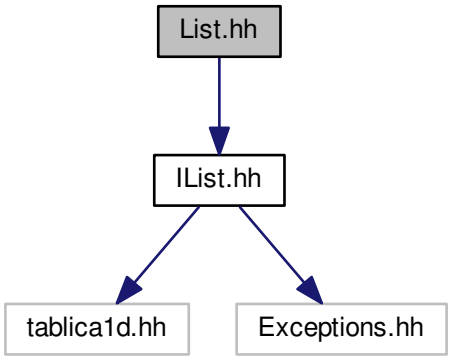
Definicja w pliku IStack.hh.

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

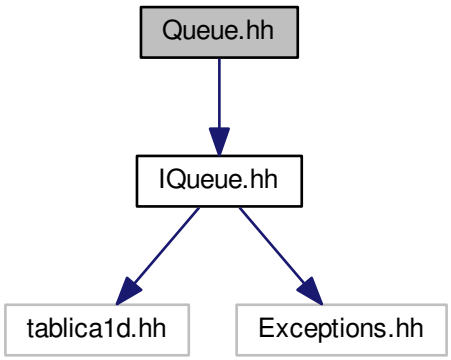
Plik zawiera definicje szablonu klasy IList, który jest interfejsem listy

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

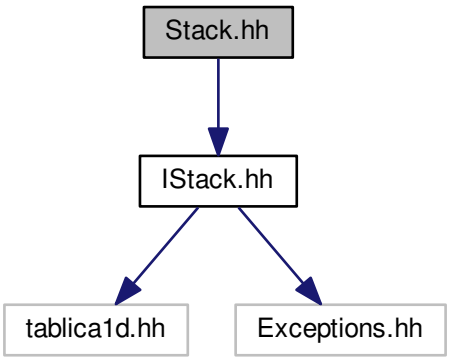
Plik zawiera definicje szablonu klasy IQueue, który jest interfejsem kolejki

Definicja w pliku Queue.hh.

5.6 Dokumentacja pliku Stack.hh

Definicja klasy Stack W pliku znajduje sie klasa Stack.

```
#include "IStack.hh"
```
Wykres zależności załączania dla Stack.hh:



Komponenty

- class Stack< Typ >

  Szablon stosu.

Skorowidz