pamsi

0.4

Wygenerowano przez Doxygen 1.8.9.1

N, 20 mar 2016 14:59:56

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 plików z ich krótkimi opisami:

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja szablonu klasy IKolejka< T >

`#include <kolejka.hh>`

4.1.1 Opis szczegółowy

template<class T>class IKolejka< T >

Definicja w linii 10 pliku kolejka.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

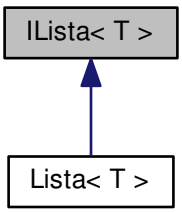
- kolejka.hh

4.2 Dokumentacja szablonu klasy ILista< T >

Interfejs listy.

`#include <lista.hh>`

Diagram dziedziczenia dla ILista< T >



Metody publiczne

- virtual void add (T, int)=0

Dodaje element do zadanego miejsca listy.

- virtual T remove (int)=0

  Usuwa element z zadanego miejsca listy.

- virtual bool isEmpty (void)=0

  Sprawdza, czy lista jest pusta.

- virtual T get (int)=0

  Zwraca element z zadanego miejsca bez usunięcia.

- virtual int size (void)=0

  Zwraca ilość elementów w liście.

- virtual ~ILista ()

  Destruktor wirtualny interfejsu ILista.

4.2.1 Opis szczegółowy

template<class T>class ILista< T >

Interfejs listy.

Definiuje dostępne operacje na klasie Lista

Definicja w linii 17 pliku lista.hh.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 template<class T > virtual ILista< T >::~ILista ( ) [inline], [virtual]

Destruktor wirtualny interfejsu ILista.

Definicja w linii 58 pliku lista.hh.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 template<class T > virtual void ILista< T >::add (T , int) [pure virtual]

Dodaje element do zadanego miejsca listy.

Jeśli następuje próba dodania elementu w miejscu istniejącego, następuje przesunięcie następujących po nim elementów na następne pozycje

Implementowany w Lista< T >.

4.2.3.2 template<class T > virtual T ILista< T >::get (int) [pure virtual]

Zwraca element z zadanego miejsca bez usunięcia.

Zwracane wartości

T	element w zadanym miejscu

Implementowany w Lista< T >.

4.2.3.3 template<class T > virtual bool ILista< T >::isEmpty (void) [pure virtual]

Sprawdza, czy lista jest pusta.

Zwracane wartości

0	gdy niepusta
1	gdy pusta

Implementowany w Lista< T >.

4.2.3.4 template<class T > virtual T ILista< T >::remove (int) [pure virtual]

Usuwa element z zadanego miejsca listy.

Jeśli usunięcie następuje w środku listy, następujące po usuwanym elementy są przesuwane o jedną pozycję wcześniej.

Zwracane wartości

T	Usunięty element

Implementowany w Lista< T >.

4.2.3.5 template<class T > virtual int ILista< T >::size (void) [pure virtual]

Zwraca ilość elementów w liście.

Zwracane wartości

int	ilość elementów

Implementowany w Lista< T >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

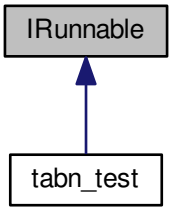
- lista.hh

4.3 Dokumentacja klasy IRunnable

Interfejs ujednolicający sposób uruchamiania klasy badającej algorytm.

`#include <run.hh>`

Diagram dziedziczenia dla IRunnable



Metody publiczne

- virtual bool prepare (int)=0

Przygotowanie badań

- virtual bool run ()=0

  Przeprowadzanie badań

- virtual ~IRunnable ()

  Destruktor wirtualny IRunnable.

4.3.1 Opis szczegółowy

Interfejs ujednolicający sposób uruchamiania klasy badającej algorytm.

Definicja w linii 18 pliku run.hh.

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 virtual IRunnable::~IRunnable ( ) [inline], [virtual]

Destruktor wirtualny IRunnable.

Definicja w linii 37 pliku run.hh.

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 virtual bool IRunnable::prepare (int) [pure virtual]

Przygotowanie badań

Zwracane wartości

bool	zawsze true

Implementowany w tabn_test.

Oto graf wywoływań tej funkcji:

4.3.3.2 virtual bool IRunnable::run ( ) [pure virtual]

Przeprowadzanie badań

Zwracane wartości

bool	zawsze true

Implementowany w tabn_test.

Oto graf wywoływań tej funkcji:

Dokumentacja dla tej klasy została wygenerowana z pliku:

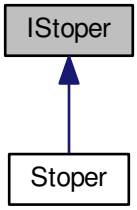
- run.hh

4.4 Dokumentacja klasy IStoper

Interfejs IStoper.

`#include <stoper.hh>`

Diagram dziedziczenia dla IStoper



Metody publiczne

- virtual void start (void)=0
- virtual void stop (void)=0
- virtual long double getElapsedTimeMs (void)=0
- virtual ~IStoper ()

4.4.1 Opis szczegółowy

Interfejs IStoper.

Definicja w linii 21 pliku stoper.hh.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 virtual IStoper::~IStoper ( ) [inline], [virtual]

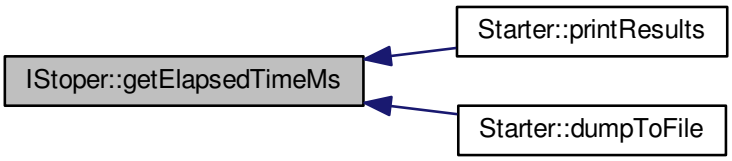
Definicja w linii 27 pliku stoper.hh.

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 virtual long double IStoper::getElapsedTimeMs (void) [pure virtual]

Implementowany w Stoper.

Oto graf wywoływań tej funkcji:



4.4.3.2 virtual void IStoper::start (void) [pure virtual]

Implementowany w Stoper.

Oto graf wywoływań tej funkcji:

4.4.3.3 virtual void IStoper::stop (void) [pure virtual]

Implementowany w Stoper.

Oto graf wywoływań tej funkcji:

Dokumentacja dla tej klasy została wygenerowana z pliku:

- stoper.hh

4.5 Dokumentacja szablonu klasy IStos< T >

Interfejs stosu.

`#include <stos.hh>`

4.5.1 Opis szczegółowy

template<class T>class IStos< T >

Interfejs stosu.

Definiuje dostępne operacje na klasie Stos

Definicja w linii 15 pliku stos.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

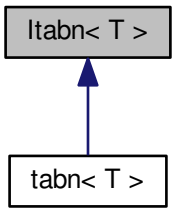
- stos.hh

4.6 Dokumentacja szablonu klasy Itabn< T >

Interfejs klasy tabn.

`#include <tabl.hh>`

Diagram dziedziczenia dla Itabn< T >



Metody publiczne

- virtual bool isEmpty (void)=0

  Sprawdza, czy tablica jest pusta.
- virtual void add (T)=0

  Dodaje element na koniec tablicy.
- virtual void add (T, int)=0

  Dodaje element w dane miejsce do tablicy, przesuwając wszystkie następne elementy o miejsce w prawo.
- virtual void remove ()=0

  Usuwa element z końca tablicy.
- virtual void remove (int)=0

  Usuwa wybrany element z listy.
- virtual T show (int)=0

  Zwraca żadany element, o ile istnieje.
- virtual void showElems (void)=0

  Wyświetla elementy listy.
- virtual int nOE (void)=0

  Zwraca liczbę elementów w tablicy.
- virtual int aSize (void)=0

  Zwraca ilość miejsca w tablicy.
- virtual T & operator[ ] (int)=0

  Pozwala na dostęp do dowolnego elementu.
- virtual T operator[ ] (int) const =0

  Pozwala na dostęp do dowolnego elementu.
- virtual ~Itabn ()

  Destruktor witrualny interfejsu.

4.6.1 Opis szczegółowy

template<class T>class Itabn< T >

Interfejs klasy tabn.

Definiuje jednolity sposób dostępu do tablicy rozszerzalnej.

Definicja w linii 22 pliku tabl.hh.

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 template<class T> virtual Itabn< T >::~Itabn ( ) [inline], [virtual]

Destruktor witrualny interfejsu.

Definicja w linii 85 pliku tabl.hh.

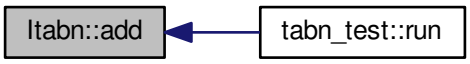
4.6.3 Dokumentacja funkcji składowych

4.6.3.1 template<class T> virtual void Itabn< T >::add (T) [pure virtual]

Dodaje element na koniec tablicy.

Implementowany w tabn< T >.

Oto graf wywoływań tej funkcji:



4.6.3.2 template<class T> virtual void Itabn< T >::add (T , int) [pure virtual]

Dodaje element w dane miejsce do tablicy, przesuwając wszystkie następne elementy o miejsce w prawo.

Parametry

element	- wstawiany element
positionShifted	- indeks pola, w które ma być wstawiony element.

Implementowany w tabn< T >.

4.6.3.3 template<class T> virtual int Itabn< T >::aSize (void) [pure virtual]

Zwraca ilość miejsca w tablicy.

Implementowany w tabn< T >.

Oto graf wywoływań tej funkcji:

4.6.3.4 template<class T> virtual bool Itabn< T >::isEmpty (void) [pure virtual]

Sprawdza, czy tablica jest pusta.

Implementowany w tabn< T >.

4.6.3.5 template<class T> virtual int Itabn< T >::nOE (void) [pure virtual]

Zwraca liczbę elementów w tablicy.

Implementowany w tabn< T >.

Oto graf wywoływań tej funkcji:

4.6.3.6 template<class T> virtual T& Itabn< T >::operator[ ] (int) [pure virtual]

Pozwala na dostęp do dowolnego elementu.

Implementowany w tabn< T >.

4.6.3.7 template<class T> virtual T Itabn< T >::operator[ ] (int) const [pure virtual]

Pozwala na dostęp do dowolnego elementu.

Implementowany w tabn< T >.

4.6.3.8 template<class T> virtual void Itabn< T >::remove ( ) [pure virtual]

Usuwa element z końca tablicy.

Implementowany w tabn< T >.

4.6.3.9 template<class T> virtual void Itabn< T >::remove (int) [pure virtual]

Usuwa wybrany element z listy.

Parametry

positionShifted	- indeks pola, z którego ma być usunięty element.

Implementowany w tabn< T >.

4.6.3.10 template<class T> virtual T Itabn< T >::show (int) [pure virtual]

Zwraca żadany element, o ile istnieje.

Implementowany w tabn< T >.

4.6.3.11 template<class T> virtual void Itabn< T >::showElems (void) [pure virtual]

Wyświetla elementy listy.

Implementowany w tabn< T >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- tabl.hh

4.7 Dokumentacja klasy Kolejka

`#include <kolejka.hh>`

4.7.1 Opis szczegółowy

Definicja w linii 18 pliku kolejka.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- kolejka.hh

4.8 Dokumentacja szablonu klasy Lista< T >

Klasa lista.

`#include <lista.hh>`

Diagram dziedziczenia dla Lista< T >

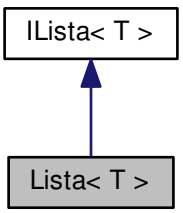
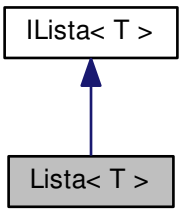


Diagram współpracy dla Lista< T >:



Metody publiczne

- Lista ()

  Konstruktor tablicy obsługującej listę
- virtual void add (T, int)

  Dodaje element do zadanego miejsca listy.
- virtual T remove (int position)

  Usuwa element z zadanego miejsca listy.
- virtual bool isEmpty (void)

  Sprawdza, czy lista jest pusta.
- virtual T get (int position)

  Zwraca element z zadanego miejsca bez usunięcia.
- virtual int size (void)

  Zwraca ilość elementów w liście.
- virtual ~Lista ()

  Destruktor Listy.

4.8.1 Opis szczegółowy

template<class T>class Lista< T >

Klasa lista.

Modeluje pojęcie listy

Definicja w linii 67 pliku lista.hh.

4.8.2 Dokumentacja konstruktora i destruktora

4.8.2.1 template<class T > Lista< T >::Lista ( ) [inline]

Konstruktor tablicy obsługującej listę

Definicja w linii 74 pliku lista.hh.

4.8.2.2 template<class T > virtual Lista< T >::~Lista ( ) [inline], [virtual]

Destruktor Listy.

Definicja w linii 115 pliku lista.hh.

4.8.3 Dokumentacja funkcji składowych

4.8.3.1 template<class T > void Lista< T >::add (T *element*, int *position*) [virtual]

Dodaje element do zadanego miejsca listy.

Jeśli następuje próba dodania elementu w miejscu istniejącego, następuje przesunięcie następujących po nim elementów na następne pozycje

Implementuje ILista< T >.

Definicja w linii 121 pliku lista.hh.

4.8.3.2 template<class T > T Lista< T >::get (int *position*) [virtual]

Zwraca element z zadanego miejsca bez usunięcia.

Zwracane wartości

T	element w zadanym miejscu

Implementuje ILista< T >.

Definicja w linii 138 pliku lista.hh.

4.8.3.3 template<class T > bool Lista< T >::isEmpty (void) [virtual]

Sprawdza, czy lista jest pusta.

Zwracane wartości

0	gdy niepusta
1	gdy pusta

Implementuje ILista< T >.

Definicja w linii 133 pliku lista.hh.

4.8.3.4 template<class T > T Lista< T >::remove (int *position*) [virtual]

Usuwa element z zadanego miejsca listy.

Jeśli usunięcie następuje w środku listy, następujące po usuwanym elementy są przesuwane o jedną pozycję wcześniej.

Zwracane wartości

T	Usunięty element

Implementuje ILista< T >.

Definicja w linii 126 pliku lista.hh.

4.8.3.5 template<class T > int Lista< T >::size (void) [virtual]

Zwraca ilość elementów w liście.

Zwracane wartości

int	ilość elementów

Implementuje ILista< T >.

Definicja w linii 155 pliku lista.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- lista.hh

4.9 Dokumentacja klasy Starter

Klasa pozwala na przeprowadzenie testów.

`#include <starter.hh>`

Metody publiczne

- Starter ()

  Konstruktor klasy tabn.
- virtual ~Starter ()

  Destruktor klasy tabn.
- void setTestSize (unsigned int)

  Metoda ustawia wielkość testu.
- void printResults (void)

  Metoda wyświetla czas trwania testu na standardowym wyjściu.
- void test (void)

  Metoda przeprowadza test.
- void dumpToFile (string)

  Metoda dopisuje dane do pliku.

4.9.1 Opis szczegółowy

Klasa pozwala na przeprowadzenie testów.

Definicja w linii 17 pliku starter.hh.

4.9.2 Dokumentacja konstruktora i destruktora

4.9.2.1 Starter::Starter ( ) [inline]

Konstruktor klasy tabn.

Definicja w linii 27 pliku starter.hh.

4.9.2.2 virtual Starter::~Starter ( ) [inline], [virtual]

Destruktor klasy tabn.

Definicja w linii 33 pliku starter.hh.

4.9.3 Dokumentacja funkcji składowych

4.9.3.1 void Starter::dumpToFile (string *nameOfFile*)

Metoda dopisuje dane do pliku.

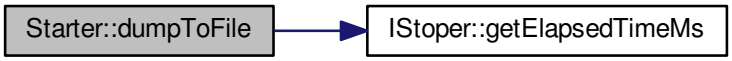
Format zapisu: wielkość_testu czas_trwania_ms

Parametry

nameOfFile	- nazwa pliku wyjściowego

Definicja w linii 20 pliku starter.cpp.

Oto graf wywołań dla tej funkcji:

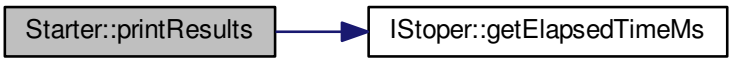


4.9.3.2 void Starter::printResults (void)

Metoda wyświetla czas trwania testu na standardowym wyjściu.

Definicja w linii 8 pliku starter.cpp.

Oto graf wywołań dla tej funkcji:



4.9.3.3 void Starter::setTestSize (unsigned int *testsize*)

Metoda ustawia wielkość testu.

Parametry

testsize	- wielkość testu

Definicja w linii 3 pliku starter.cpp.

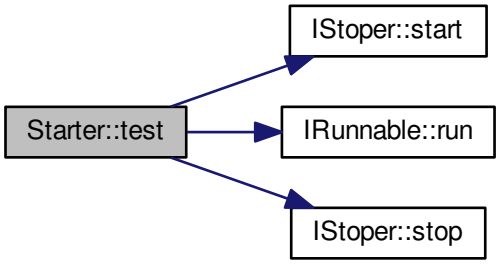
Oto graf wywołań dla tej funkcji:

4.9.3.4 void Starter::test (void)

Metoda przeprowadza test.

Definicja w linii 14 pliku starter.cpp.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z plików:

- starter.hh
- starter.cpp

4.10 Dokumentacja klasy Stoper

Klasa stoper implementująca interfejs IStoper.

`#include <stoper.hh>`

Diagram dziedziczenia dla Stoper

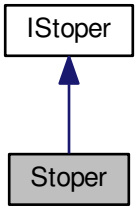
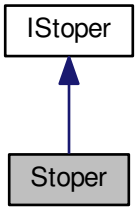


Diagram współpracy dla Stoper:



Metody publiczne

- virtual void start (void)

  Uruchamia zegar.
- virtual void stop (void)

  Zatrzymuje zegar.
- virtual long double getElapsedTimeMs (void)

  Oblicza i zwraca czas pomiędzy uruchomieniem zegara a jego zatrzymaniem.

4.10.1 Opis szczegółowy

Klasa stoper implementująca interfejs IStoper.

Klasa symuluje działanie stopera - zapisuje początkowy i końcowy moment działania (użycie start i stop), oraz odejmuje obie te wartości od siebie, by uzyskać czas działania.

Definicja w linii 37 pliku stoper.hh.

4.10.2 Dokumentacja funkcji składowych

4.10.2.1 long double Stoper::getElapsedTimeMs (void) [virtual]

Oblicza i zwraca czas pomiędzy uruchomieniem zegara a jego zatrzymaniem.

Zwracane wartości

long_double	Czas pomiędzy startem a zatrzymaniem zegara

Implementuje IStoper.

Definicja w linii 12 pliku stoper.cpp.

4.10.2.2 void Stoper::start (void) [virtual]

Uruchamia zegar.

Implementuje IStoper.

Definicja w linii 4 pliku stoper.cpp.

4.10.2.3 void Stoper::stop (void) [virtual]

Zatrzymuje zegar.

Implementuje IStoper.

Definicja w linii 8 pliku stoper.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

- stoper.hh
- stoper.cpp

4.11 Dokumentacja klasy Stos

`#include <stos.hh>`

4.11.1 Opis szczegółowy

Definicja w linii 25 pliku stos.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- stos.hh

4.12 Dokumentacja szablonu klasy tabn< T >

Modeluje tablicę dynamicznie rozszerzalną

`#include <tabl.hh>`

Diagram dziedziczenia dla tabn< T >

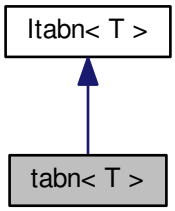
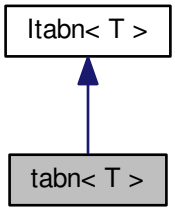


Diagram współpracy dla tabn< T >:



Metody publiczne

- tabn ()

  Konstruktor klasy tabn.
- virtual ~tabn ()

  Destruktor klasy tabn.
- virtual bool isEmpty (void)

  Sprawdza, czy tablica jest pusta.
- virtual void add (T)

  Dodaje element Dodaje element do tablicy dynamicznej, odpowiednio ją rozszerzając.
- virtual void add (T, int)

  Dodaje element w dane miejsce do tablicy, przesuwając wszystkie następne elementy o miejsce w prawo.
- virtual void remove ()

  Usuwa ostatni element z listy.
- virtual void remove (int)

  Usuwa wybrany element z listy, przesuwając wszystkie następne elementy o miejsce w lewo.
- virtual T show (int)

  Zwraca żadany element, o ile istnieje.
- virtual void showElems (void)

  Wyświetla listę elementów.

- virtual int nOE (void)

  zwraca liczbę elementów w tablicy
- virtual int aSize (void)

  zwraca wielkość zaalokowanej przestrzeni dla tablicy
- virtual T & operator[ ] (int)

  Umożliwia dostęp do dowolnego elementu tablicy bez sprawdzania zakresu (debug)
- virtual T operator[ ] (int) const

  Umożliwia odczyt dowolnego elementu tablicy bez sprawdzania zakresu (debug)

4.12.1 Opis szczegółowy

template<class T>class tabn< T >

Modeluje tablicę dynamicznie rozszerzalną

Przechowuje elementy w rozszerzalnej tablicy o rozmiarze początkowym SIZE

Definicja w linii 95 pliku tabl.hh.

4.12.2 Dokumentacja konstruktora i destruktora

4.12.2.1 template<class T > tabn< T >::tabn ( ) [inline]

Konstruktor klasy tabn.

Definicja w linii 106 pliku tabl.hh.

4.12.2.2 template<class T > virtual tabn< T >::~tabn ( ) [inline], [virtual]

Destruktor klasy tabn.

Definicja w linii 115 pliku tabl.hh.

4.12.3 Dokumentacja funkcji składowych

4.12.3.1 template<class T > void tabn< T >::add (T *element*) [virtual]

Dodaje element Dodaje element do tablicy dynamicznej, odpowiednio ją rozszerzając.

Parametry

element	- element do dodania

Implementuje Itabn< T >.

Definicja w linii 240 pliku tabl.hh.

4.12.3.2 template<class T > void tabn< T >::add (T *element,* int *position*) [virtual]

Dodaje element w dane miejsce do tablicy, przesuwając wszystkie następne elementy o miejsce w prawo.

Parametry

element	- wstawiany element
positionShifted	- indeks pola, w które ma być wstawiony element.

Implementuje Itabn< T >.

Definicja w linii 248 pliku tabl.hh.

4.12.3.3 template<class T > int tabn< T >::aSize (void) [virtual]

zwraca wielkość zaalokowanej przestrzeni dla tablicy

Zwracane wartości

int	Ilość zaalokowanych pól

Implementuje Itabn< T >.

Definicja w linii 423 pliku tabl.hh.

4.12.3.4 template<class T > bool tabn< T >::isEmpty (void) [virtual]

Sprawdza, czy tablica jest pusta.

Zwracane wartości

0	gdy tablica nie jest pusta
1	gdy tablica jest pusta

Implementuje Itabn< T >.

Definicja w linii 370 pliku tabl.hh.

4.12.3.5 template<class T > int tabn< T >::nOE (void) [virtual]

zwraca liczbę elementów w tablicy

Zwracane wartości

int	Liczba elementów w tablicy

Implementuje Itabn< T >.

Definicja w linii 418 pliku tabl.hh.

4.12.3.6 template<class T > T & tabn< T >::operator[ ] (int *index*) [virtual]

Umożliwia dostęp do dowolnego elementu tablicy bez sprawdzania zakresu (debug)

Parametry

index	- numer elementu tablicy

Zwracane wartości

*T**	Wskaźnik na wybrany element tablicy

Implementuje Itabn< T >.

Definicja w linii 387 pliku tabl.hh.

4.12.3.7 template<class T > T tabn< T >::operator[ ] (int *index*) const [virtual]

Umożliwia odczyt dowolnego elementu tablicy bez sprawdzania zakresu (debug)

Parametry

index	- numer elementu tablicy

Zwracane wartości

T	Element tablicy

Implementuje Itabn< T >.

Definicja w linii 392 pliku tabl.hh.

4.12.3.8 template<class T > void tabn< T >::remove ( ) [virtual]

Usuwa ostatni element z listy.

Implementuje Itabn< T >.

Definicja w linii 280 pliku tabl.hh.

4.12.3.9 template<class T > void tabn< T >::remove (int *position*) [virtual]

Usuwa wybrany element z listy, przesuwając wszystkie następne elementy o miejsce w lewo.

Parametry

positionShifted	- indeks pola, z którego ma być usunięty element.

Implementuje Itabn< T >.

Definicja w linii 299 pliku tabl.hh.

4.12.3.10 template<class T > T tabn< T >::show (int *position*) [virtual]

Zwraca żadany element, o ile istnieje.

Implementuje Itabn< T >.

Definicja w linii 397 pliku tabl.hh.

4.12.3.11 template<class T > void tabn< T >::showElems (void) [virtual]

Wyświetla listę elementów.

Implementuje Itabn< T >.

Definicja w linii 408 pliku tabl.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- tabl.hh

4.13 Dokumentacja klasy tabn_test

Definiuje sposób testowania wypełniania tablicy tabn.

`#include <tabl.hh>`

Diagram dziedziczenia dla tabn_test

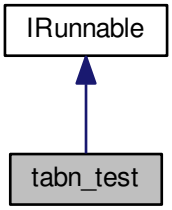
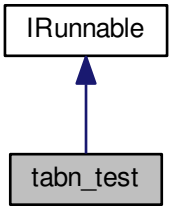


Diagram współpracy dla tabn_test:



Metody publiczne

- tabn_test ()

  Konstruktor klasy tabn_test.
- virtual ~tabn_test ()

  Destruktor klasy tabn_test.
- virtual bool prepare (int sizeOfTest)

  Przygotowuje rozmiar testu.
- virtual bool run ()

  Wykonuje test.

4.13.1 Opis szczegółowy

Definiuje sposób testowania wypełniania tablicy tabn.

Definicja w linii 432 pliku tabl.hh.

4.13.2 Dokumentacja konstruktora i destruktora

4.13.2.1 tabn_test::tabn_test ( ) [inline]

Konstruktor klasy tabn_test.

Definicja w linii 440 pliku tabl.hh.

4.13.2.2 virtual tabn_test::~tabn_test ( ) [inline], [virtual]

Destruktor klasy tabn_test.

Definicja w linii 446 pliku tabl.hh.

4.13.3 Dokumentacja funkcji składowych

4.13.3.1 virtual bool tabn_test::prepare (int *sizeOfTest*) [inline], [virtual]

Przygotowuje rozmiar testu.

Parametry

sizeOfTest	- rozmiar testu

Zwracane wartości

bool	zawsze true

Implementuje IRunnable.

Definicja w linii 477 pliku tabl.hh.

4.13.3.2 virtual bool tabn_test::run ( ) [inline], [virtual]

Wykonuje test.

Pozwala na wykonanie testu w pętli for iterującej counter razy. Zasila funkcję dodawania generując losowe cyfry w funkcji generateRandomDgt()

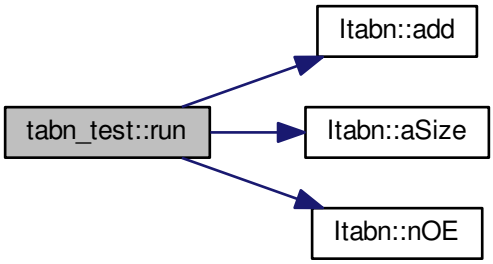
Zwracane wartości

bool	zawsze true

Implementuje IRunnable.

Definicja w linii 492 pliku tabl.hh.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z pliku:

- tabl.hh

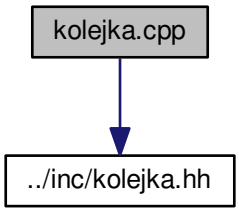
Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku kolejka.cpp

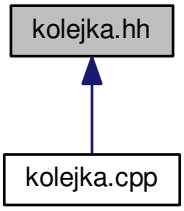
```
#include "../inc/kolejka.hh"
```

Wykres zależności załączania dla kolejka.cpp:



5.2 Dokumentacja pliku kolejka.hh

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



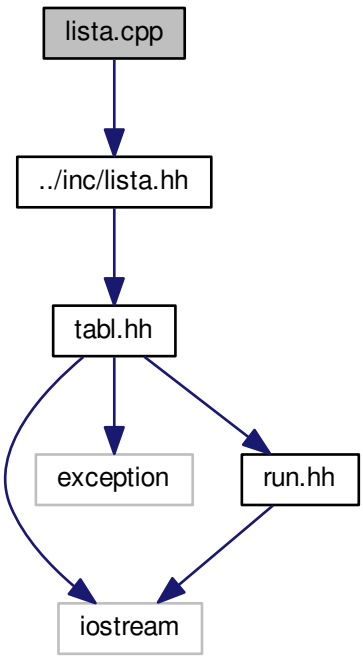
Komponenty

- class IKolejka< T >
- class Kolejka

5.3 Dokumentacja pliku lista.cpp

```
#include "../inc/lista.hh"
```

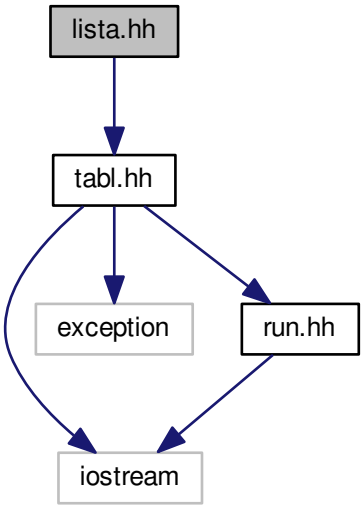
Wykres zależności załączania dla lista.cpp:



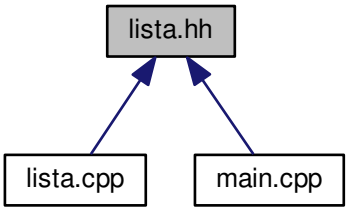
5.4 Dokumentacja pliku lista.hh

```
#include "tabl.hh"
```

Wykres zależności załączania dla lista.hh:



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



Komponenty

- class ILista< T >

  Interfejs listy.
- class Lista< T >

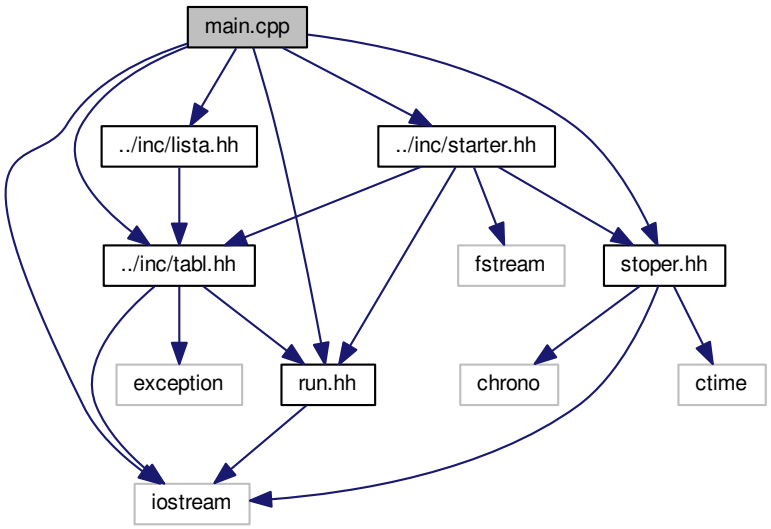
  Klasa lista.

5.5 Dokumentacja pliku main.cpp

Główny plik programu.

```
#include <iostream>
#include "../inc/tabl.hh"
#include "../inc/run.hh"
#include "../inc/starter.hh"
#include "../inc/stoper.hh"
#include "../inc/lista.hh"
```

Wykres zależności załączania dla main.cpp:



Funkcje

- int main (void)

5.5.1 Opis szczegółowy

Główny plik programu.

5.5.2 Dokumentacja funkcji

5.5.2.1 int main (void)

Definicja w linii 19 pliku main.cpp.

5.6 Dokumentacja pliku main.hh

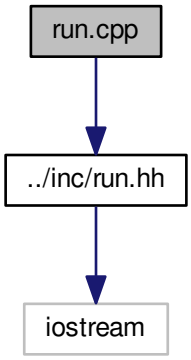
Plik posiada wspólne definicje.

5.6.1 Opis szczegółowy

Plik posiada wspólne definicje.

5.7 Dokumentacja pliku run.cpp

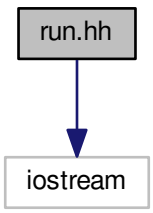
```
#include "../inc/run.hh"
```
Wykres zależności załączania dla run.cpp:



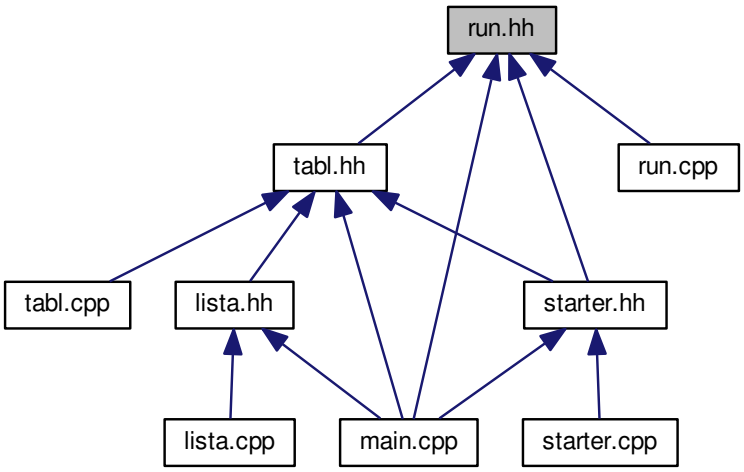
5.8 Dokumentacja pliku run.hh

Plik definiuje interfejs IRunnable, ujednolicający klasy umożliwiające badanie algorytmów.

```
#include <iostream>
```
Wykres zależności załączania dla run.hh:



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



Komponenty

- class IRunnable

  Interfejs ujednolicający sposób uruchamiania klasy badającej algorytm.

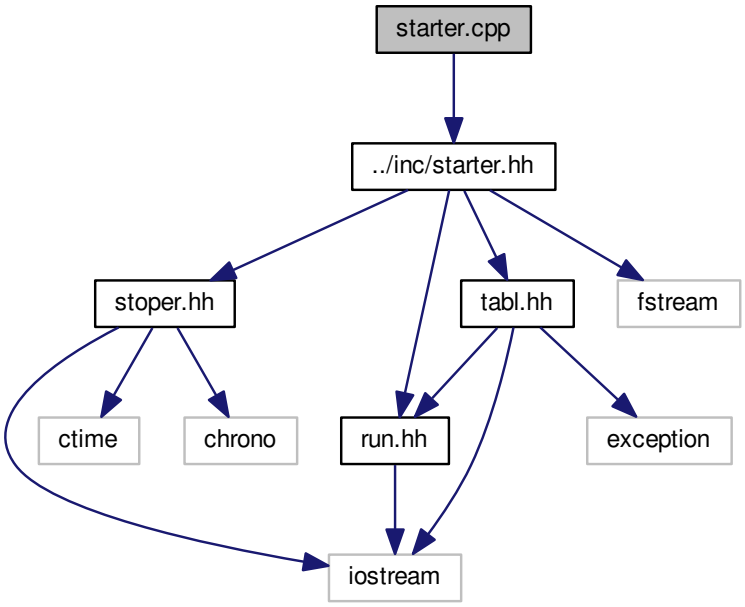
5.8.1 Opis szczegółowy

Plik definiuje interfejs IRunnable, ujednolicający klasy umożliwiające badanie algorytmów.

5.9 Dokumentacja pliku starter.cpp

```
#include "../inc/starter.hh"
```

Wykres zależności załączania dla starter.cpp:

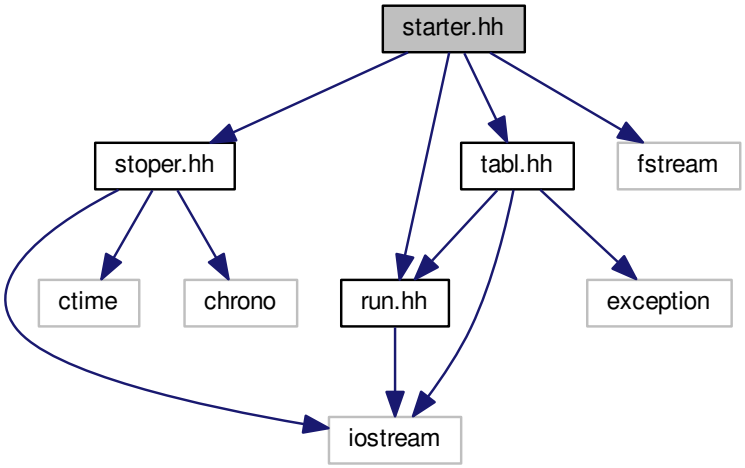


5.10 Dokumentacja pliku starter.hh

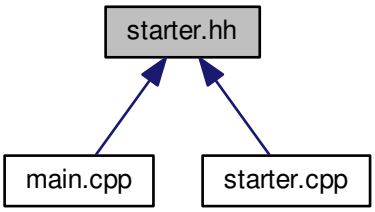
Plik definiuje klasę Starter.

```
#include "stoper.hh"
#include "run.hh"
#include "tabl.hh"
#include <fstream>
```

Wykres zależności załączania dla starter.hh:



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



Komponenty

- class Starter

  Klasa pozwala na przeprowadzenie testów.

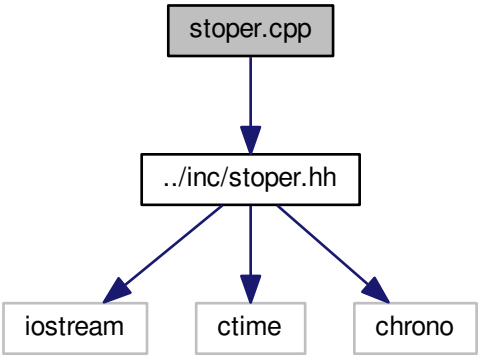
5.10.1 Opis szczegółowy

Plik definiuje klasę Starter.

5.11 Dokumentacja pliku stoper.cpp

```
#include "../inc/stoper.hh"
```

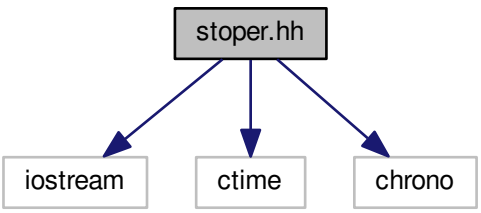
Wykres zależności załączania dla stoper.cpp:



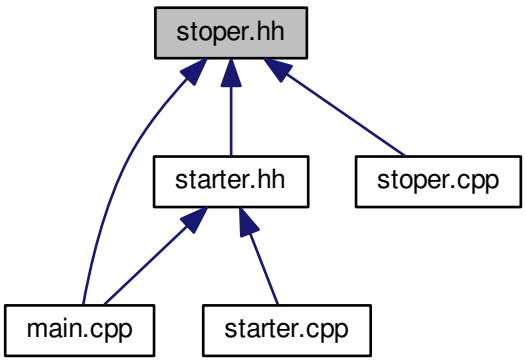
5.12 Dokumentacja pliku stoper.hh

```
#include <iostream>
#include <ctime>
#include <chrono>
```

Wykres zależności załączania dla stoper.hh:



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



Komponenty

- class IStoper

  Interfejs IStoper.

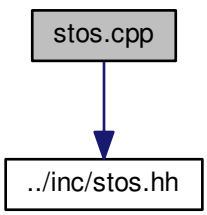
- class Stoper

  Klasa stoper implementująca interfejs IStoper.

5.13 Dokumentacja pliku stos.cpp

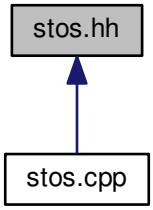
```
#include "../inc/stos.hh"
```

Wykres zależności załączania dla stos.cpp:



5.14 Dokumentacja pliku stos.hh

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



Komponenty

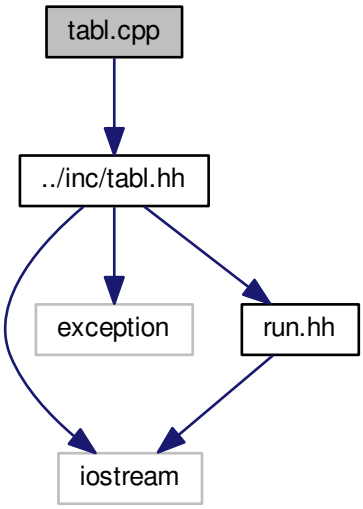
- class IStos< T >

  Interfejs stosu.
- class Stos

5.15 Dokumentacja pliku tabl.cpp

```
#include "../inc/tabl.hh"
```

Wykres zależności załączania dla tabl.cpp:

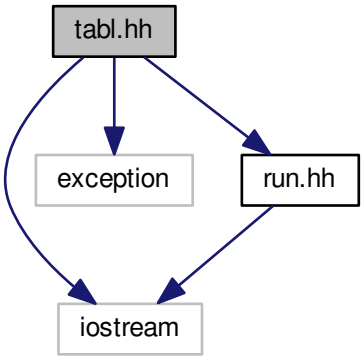


5.16 Dokumentacja pliku tabl.hh

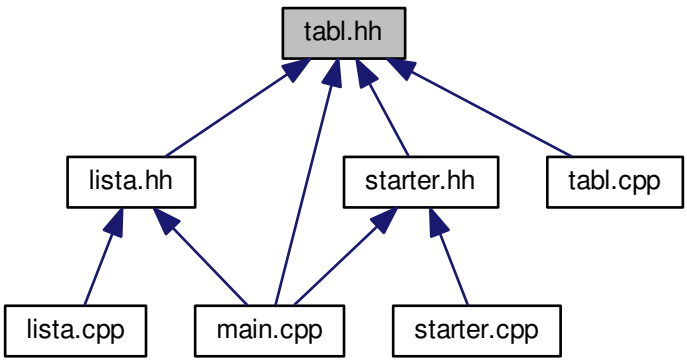
Definicja interfejsu ltabn, klasy tabn oraz klasy tabn_test.

```
#include <iostream>
#include <exception>
#include "run.hh"
```

Wykres zależności załączania dla tabl.hh:



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



Komponenty

- class ltabn< T >

  Interfejs klasy tabn.
- class tabn< T >

  Modeluje tablicę dynamicznie rozszerzalną

- class tabn_test

  Definiuje sposób testowania wypełniania tablicy tabn.

Definicje

- #define SIZE 10

5.16.1 Opis szczegółowy

Definicja interfejsu Itabn, klasy tabn oraz klasy tabn_test.

5.16.2 Dokumentacja definicji

5.16.2.1 #define SIZE 10

Definicja w linii 12 pliku tabl.hh.

Skorowidz