

pamsi
0.4

Wygenerowano przez Doxygen 1.8.9.1

N, 20 mar 2016 01:23:23

Spis treści

1	Indeks hierarchiczny	1
1.1	Hierarchia klas	1
2	Indeks klas	3
2.1	Lista klas	3
3	Indeks plików	5
3.1	Lista plików	5
4	Dokumentacja klas	7
4.1	Dokumentacja szablonu klasy <code>IKolejka< T ></code>	7
4.1.1	Opis szczegółowy	7
4.2	Dokumentacja szablonu klasy <code>ILista< T ></code>	7
4.2.1	Opis szczegółowy	7
4.2.2	Dokumentacja konstruktora i destruktora	7
4.2.2.1	<code>~ILista</code>	8
4.2.3	Dokumentacja funkcji składowych	8
4.2.3.1	<code>add</code>	8
4.2.3.2	<code>get</code>	8
4.2.3.3	<code>isEmpty</code>	8
4.2.3.4	<code>remove</code>	8
4.2.3.5	<code>size</code>	8
4.3	Dokumentacja klasy <code>IRunnable</code>	8
4.3.1	Opis szczegółowy	9
4.3.2	Dokumentacja konstruktora i destruktora	9
4.3.2.1	<code>~IRunnable</code>	9
4.3.3	Dokumentacja funkcji składowych	9
4.3.3.1	<code>prepare</code>	9
4.3.3.2	<code>run</code>	9
4.4	Dokumentacja klasy <code>IStoper</code>	10
4.4.1	Opis szczegółowy	10
4.4.2	Dokumentacja konstruktora i destruktora	10

4.4.2.1	~IStoper	10
4.4.3	Dokumentacja funkcji składowych	10
4.4.3.1	getElapsedTimeMs	10
4.4.3.2	start	11
4.4.3.3	stop	11
4.5	Dokumentacja szablonu klasy IStos< T >	11
4.5.1	Opis szczegółowy	12
4.6	Dokumentacja szablonu klasy Itabn< T >	12
4.6.1	Opis szczegółowy	13
4.6.2	Dokumentacja konstruktora i destruktor	13
4.6.2.1	~Itabn	13
4.6.3	Dokumentacja funkcji składowych	13
4.6.3.1	add	13
4.6.3.2	add	13
4.6.3.3	aSize	14
4.6.3.4	isEmpty	14
4.6.3.5	nOE	14
4.6.3.6	operator[]	14
4.6.3.7	operator[]	15
4.6.3.8	remove	15
4.6.3.9	remove	15
4.6.3.10	showElems	15
4.7	Dokumentacja klasy Kolejka	16
4.7.1	Opis szczegółowy	16
4.8	Dokumentacja klasy Lista	16
4.8.1	Opis szczegółowy	16
4.9	Dokumentacja klasy Starter	16
4.9.1	Opis szczegółowy	16
4.9.2	Dokumentacja konstruktora i destruktor	17
4.9.2.1	Starter	17
4.9.2.2	~Starter	17
4.9.3	Dokumentacja funkcji składowych	17
4.9.3.1	dumpToFile	17
4.9.3.2	printResults	17
4.9.3.3	setTestSize	18
4.9.3.4	test	19
4.10	Dokumentacja klasy Stoper	19
4.10.1	Opis szczegółowy	20
4.10.2	Dokumentacja funkcji składowych	20
4.10.2.1	getElapsedTimeMs	21

4.10.2.2	start	22
4.10.2.3	stop	22
4.11	Dokumentacja klasy Stos	22
4.11.1	Opis szczegółowy	22
4.12	Dokumentacja szablonu klasy tabn< T >	22
4.12.1	Opis szczegółowy	24
4.12.2	Dokumentacja konstruktora i destruktora	24
4.12.2.1	tabn	24
4.12.2.2	~tabn	24
4.12.3	Dokumentacja funkcji składowych	24
4.12.3.1	add	24
4.12.3.2	add	24
4.12.3.3	aSize	25
4.12.3.4	isEmpty	26
4.12.3.5	nOE	26
4.12.3.6	operator[]	26
4.12.3.7	operator[]	26
4.12.3.8	remove	27
4.12.3.9	remove	27
4.12.3.10	showElems	27
4.13	Dokumentacja klasy tabn_test	27
4.13.1	Opis szczegółowy	28
4.13.2	Dokumentacja konstruktora i destruktora	28
4.13.2.1	tabn_test	28
4.13.2.2	~tabn_test	28
4.13.3	Dokumentacja funkcji składowych	28
4.13.3.1	prepare	28
4.13.3.2	run	29
5	Dokumentacja plików	31
5.1	Dokumentacja pliku kolejka.cpp	31
5.2	Dokumentacja pliku kolejka.hh	31
5.3	Dokumentacja pliku lista.cpp	32
5.4	Dokumentacja pliku lista.hh	32
5.5	Dokumentacja pliku main.cpp	32
5.5.1	Opis szczegółowy	33
5.5.2	Dokumentacja funkcji	33
5.5.2.1	main	33
5.6	Dokumentacja pliku main.hh	34
5.6.1	Opis szczegółowy	34

5.7	Dokumentacja pliku run.cpp	35
5.8	Dokumentacja pliku run.hh	35
5.8.1	Opis szczegółowy	36
5.9	Dokumentacja pliku starter.cpp	37
5.10	Dokumentacja pliku starter.hh	37
5.10.1	Opis szczegółowy	38
5.11	Dokumentacja pliku stoper.cpp	39
5.12	Dokumentacja pliku stoper.hh	39
5.13	Dokumentacja pliku stos.cpp	40
5.14	Dokumentacja pliku stos.hh	41
5.15	Dokumentacja pliku tabl.cpp	41
5.16	Dokumentacja pliku tabl.hh	42
5.16.1	Opis szczegółowy	43
5.16.2	Dokumentacja definicji	43
5.16.2.1	SIZE	43
Indeks		45

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

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

IKolejka< T >	7
ILista< T >	7
IRunnable	8
tabn_test	27
IStoper	10
Stoper	19
IStos< T >	11
Itabn< T >	12
tabn< T >	22
Itabn< int >	12
Kolejka	16
Lista	16
Starter	16
Stos	22

Rozdział 2

Indeks klas

2.1 Lista klas

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

IKolejka< T >	7
ILista< T >	7
IRunnable	
Interfejs ujednolicający sposób uruchamiania klasy badającej algorytm	8
IStoper	
Interfejs IStoper	10
IStos< T >	11
Itabn< T >	
Interfejs klasy tabn	12
Kolejka	16
Lista	16
Starter	
Klasa pozwala na przeprowadzenie testów	16
Stoper	
Klasa stoper implementująca interfejs IStoper	19
Stos	22
tabn< T >	
Modeluje tablicę dynamicznie rozszerzalną	22
tabn_test	
Definiuje sposób testowania wypełniania tablicy tabn	27

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

kolejka.cpp	31
kolejka.hh	31
lista.cpp	32
lista.hh	32
main.cpp	
Główny plik programu	32
main.hh	
Plik posiada wspólne definicje	34
run.cpp	35
run.hh	
Plik definiuje interfejs IRunnable , ujednolicający klasy umożliwiające badanie algorytmów	35
starter.cpp	37
starter.hh	
Plik definiuje klasę Starter	37
stoper.cpp	39
stoper.hh	39
stos.cpp	40
stos.hh	41
tabl.cpp	41
tabl.hh	
Definicja interfejsu Itabn , klasy tabn oraz klasy tabn_test	42

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 >

```
#include <lista.hh>
```

Metody publiczne

- virtual void [add](#) (T, int)=0
- virtual void [remove](#) (int)=0
- virtual bool [isEmpty](#) (void)=0
- virtual T [get](#) (int)=0
- virtual int [size](#) (void)=0
- virtual [~ILista](#) ()

4.2.1 Opis szczegółowy

```
template<class T>class ILista< T >
```

Definicja w linii 9 pliku lista.hh.

4.2.2 Dokumentacja konstruktora i destruktor

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

Definicja w linii 16 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]`

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

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

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

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

Dokumentacja dla tej klasy została wygenerowana z pliku:

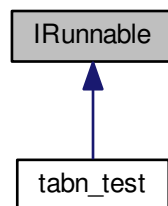
- [lista.hh](#)

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

<i>bool</i>	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

<i>bool</i>	zawsze true
-------------	-------------

Implementowany w [tabn_test](#).

Oto graf wywoływań tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z pliku:

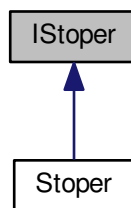
- [run.hh](#)

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 destruktor

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

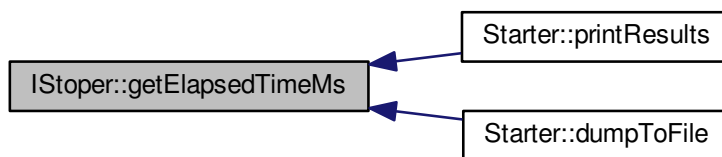
Definicja w linii 27 pliku `stoper.hh`.

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 >

```
#include <stos.hh>
```

4.5.1 Opis szczegółowy

```
template<class T>class IStos< T >
```

Definicja w linii 10 pliku stos.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

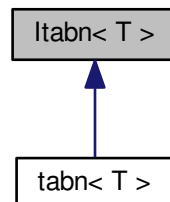
- [stos.hh](#)

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 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 wirtualny 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 wirtualny interfejsu.

Definicja w linii 81 pliku tabl.hh.

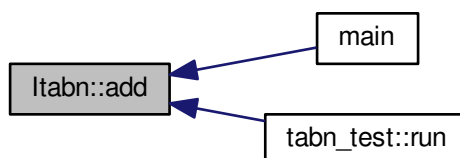
4.6.3 Dokumentacja funkcji składowych

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

<i>element</i>	- wstawiany element
----------------	---------------------

<i>positionShifted</i>	- indeks pola, w które ma być wstawiony element.
------------------------	--

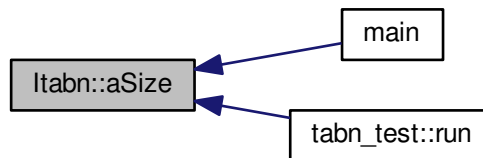
Implementowany w [tabn< T >](#).

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.

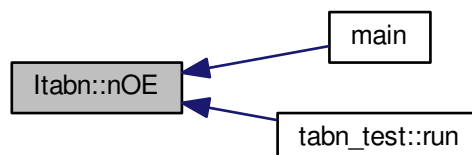
Implementowany w [tabn< T >](#).

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 >](#).

Oto graf wywoływań tej funkcji:



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

Usuwa wybrany element z listy.

Parametry

<i>positionShifted</i>	- indeks pola, z którego ma być usunięty element.
------------------------	---

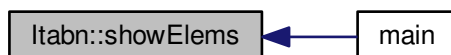
Implementowany w [tabn< T >](#).

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

Wyświetla elementy listy.

Implementowany w [tabn< T >](#).

Oto graf wywoływań tej funkcji:



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 klasy Lista

```
#include <lista.hh>
```

4.8.1 Opis szczegółowy

Definicja w linii 19 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

<i>nameOfFile</i>	- 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

<i>testsize</i>	- wielkość testu
-----------------	------------------

Definicja w linii 3 pliku starter.cpp.

Oto graf wywołań dla tej funkcji:

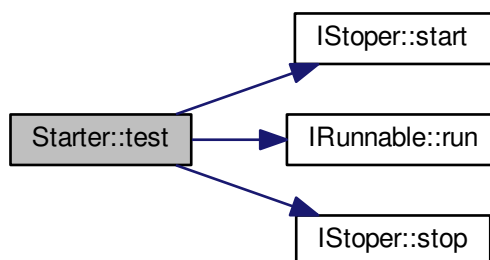


4.9.3.4 void Starter::test (void)

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

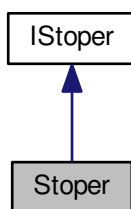
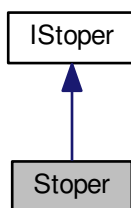


Diagram współpracy dla Stoper:



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

<i>long_double</i>	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 18 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 >`

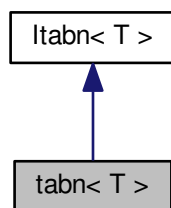
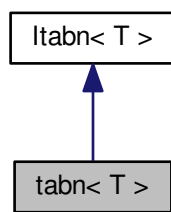


Diagram współpracy dla `tabn< T >`:



Metody publiczne

- `tabn()`
Konstruktor klasy `tabn`.
- `virtual ~tabn()`
Destruktor klasy `tabn`.
- `virtual bool isEmpty()`
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 void showElems()`
Wyświetla listę elementów.
- `virtual int nOE()`
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 91 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 102 pliku `tabl.hh`.

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

Destruktor klasy `tabn`.

Definicja w linii 111 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

<i>element</i>	- element do dodania
----------------	----------------------

Implementuje `ltabn< T >`.

Definicja w linii 229 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

<i>element</i>	- wstawiany element
<i>positionShifted</i>	- indeks pola, w które ma być wstawiony element.

Implementuje `ltabn< T >`.

Definicja w linii 237 pliku `tabl.hh`.

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

zwraca wielkość zaalokowanej przestrzeni dla tablicy

Zwracane wartości

<i>int</i>	Ilość zaalokowanych pól
------------	-------------------------

Implementuje `ltabn< T >`.

Definicja w linii 404 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 `ltabn< T >`.

Definicja w linii 364 pliku `tabl.hh`.

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

zwraca liczbę elementów w tablicy

Zwracane wartości

<i>int</i>	Liczba elementów w tablicy
------------	----------------------------

Implementuje `ltabn< T >`.

Definicja w linii 399 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

<i>index</i>	- numer elementu tablicy
--------------	--------------------------

Zwracane wartości

<i>T*</i>	Wskaźnik na wybrany element tablicy
-----------	-------------------------------------

Implementuje `ltabn< T >`.

Definicja w linii 381 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

<i>index</i>	- numer elementu tablicy
--------------	--------------------------

Zwracane wartości

<code>T</code>	Element tablicy
----------------	-----------------

Implementuje `Itabn< T >`.

Definicja w linii 386 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 270 pliku `tabl.hh`.

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

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

Parametry

<code><i>positionShifted</i></code>	- indeks pola, z którego ma być usunięty element.
-------------------------------------	---

Implementuje `Itabn< T >`.

Definicja w linii 291 pliku `tabl.hh`.

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

Wyświetla listę elementów.

Implementuje `Itabn< T >`.

Definicja w linii 391 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`

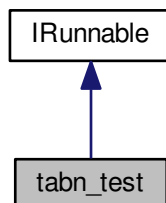
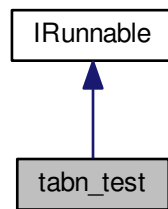


Diagram współpracy dla `tabn_test`:



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 413 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 421 pliku `tabl.hh`.

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

Destruktor klasy `tabn_test`.

Definicja w linii 427 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

<i>sizeofTest</i>	- rozmiar testu
-------------------	-----------------

Zwracane wartości

<i>bool</i>	zawsze true
-------------	-------------

Implementuje [IRunnable](#).

Definicja w linii 458 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()`

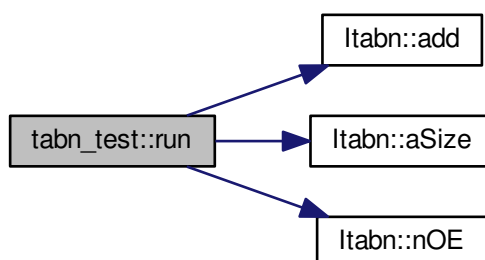
Zwracane wartości

<i>bool</i>	zawsze true
-------------	-------------

Implementuje [IRunnable](#).

Definicja w linii 473 pliku `tabl.hh`.

Oto graf wywołań dla tej funkcji:



Dokumentacja dla tej klasy została wygenerowana z pliku:

- [tabl.hh](#)

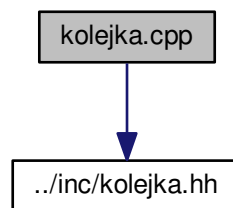
Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku kolejka.cpp

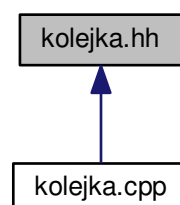
```
#include "../inc/kolejka.hh"
```

Wykres zależności załączania dla kolejka.cpp:



5.2 Dokumentacja pliku kolejka.hh

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



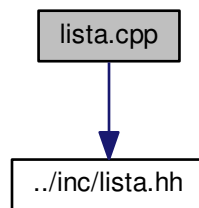
Komponenty

- class `IKolejka< T >`
- class `Kolejka`

5.3 Dokumentacja pliku lista.cpp

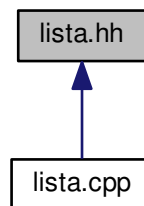
```
#include "../inc/lista.hh"
```

Wykres zależności załączania dla lista.cpp:



5.4 Dokumentacja pliku lista.hh

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



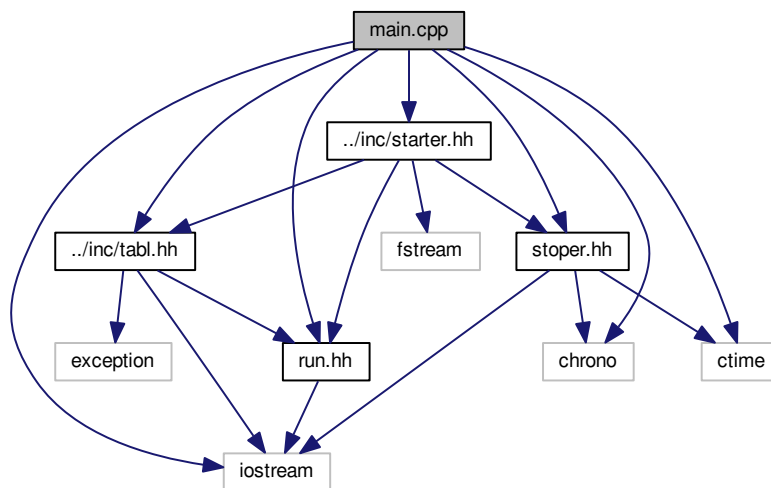
Komponenty

- class `ILista< T >`
- class `Lista`

5.5 Dokumentacja pliku main.cpp

Główny plik programu.

```
#include <iostream>
#include <chrono>
#include <ctime>
#include "../inc/tabl.hh"
#include "../inc/run.hh"
#include "../inc/starter.hh"
#include "../inc/stoper.hh"
Wykres zależności załączania dla main.cpp:
```



Funkcje

- int [main](#) (void)

5.5.1 Opis szczegółowy

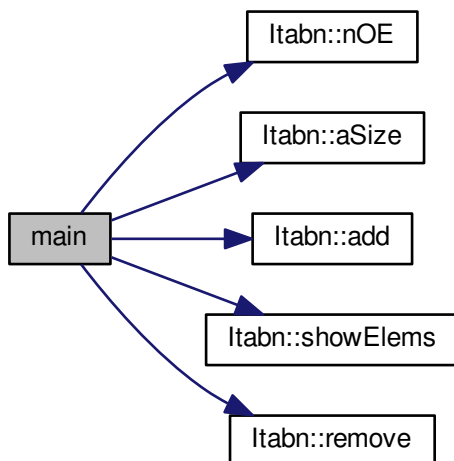
Główny plik programu.

5.5.2 Dokumentacja funkcji

5.5.2.1 int main (void)

Definicja w linii 18 pliku main.cpp.

Oto graf wywołań dla tej funkcji:



5.6 Dokumentacja pliku main.hh

Plik posiada wspólne definicje.

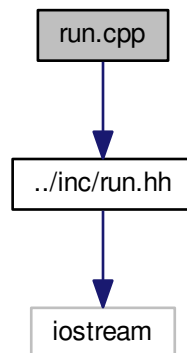
5.6.1 Opis szczegółowy

Plik posiada wspólne definicje.

5.7 Dokumentacja pliku run.cpp

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

Wykres zależności załączania dla run.cpp:

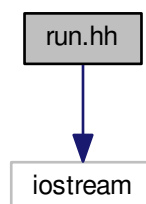


5.8 Dokumentacja pliku run.hh

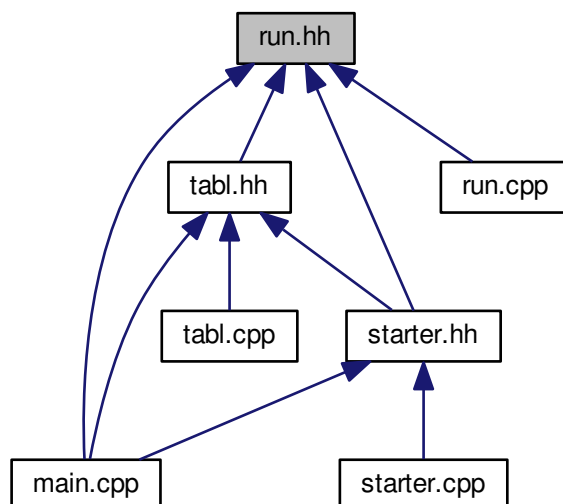
Plik definiuje interfejs [IRunnable](#), ujednolicający klasy umożliwiające badanie algorytmów.

```
#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.

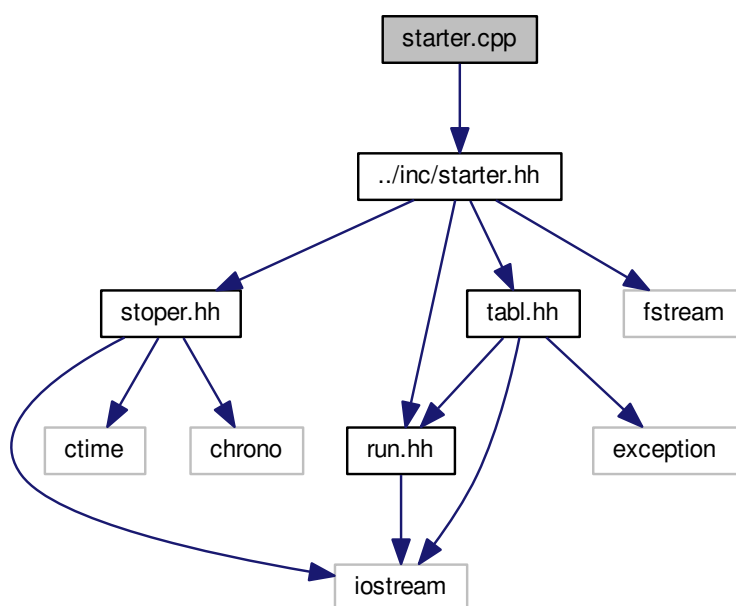
5.8.1 Opis szczegółowy

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:

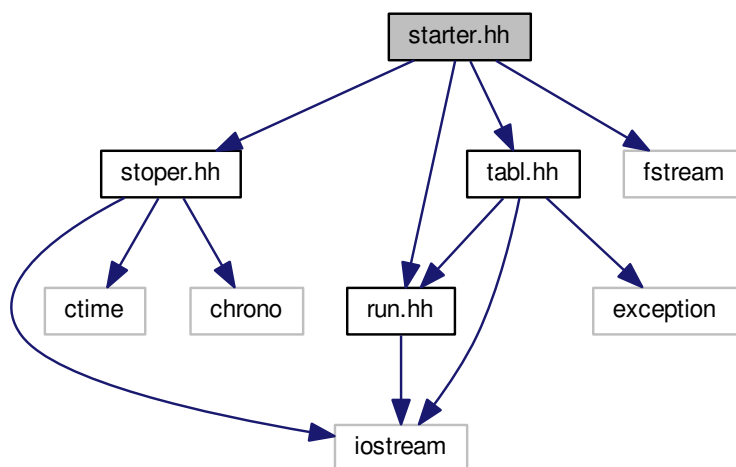


5.10 Dokumentacja pliku starter.hh

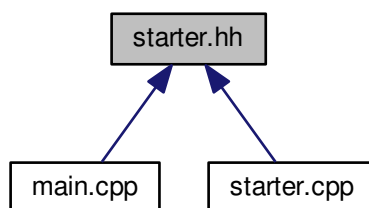
Plik definiuje klasę [Starter](#).

```
#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.

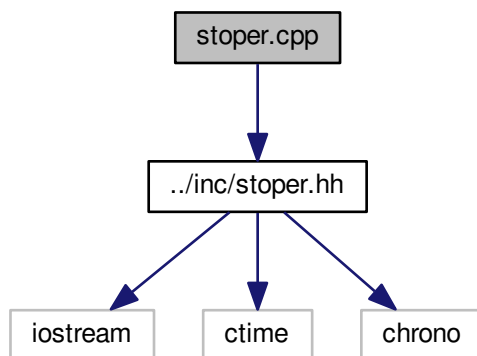
5.10.1 Opis szczegółowy

Plik definiuje klasę [Starter](#).

5.11 Dokumentacja pliku stoper.cpp

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

Wykres zależności załączania dla stoper.cpp:



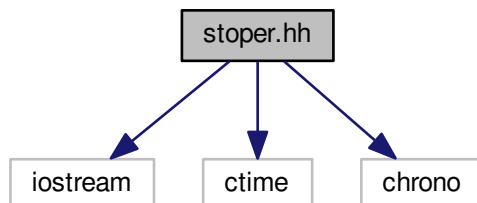
5.12 Dokumentacja pliku stoper.hh

```
#include <iostream>
```

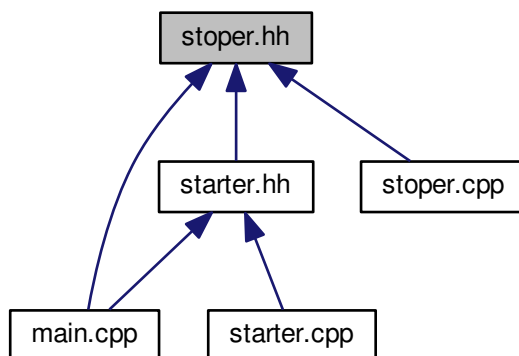
```
#include <ctime>
```

```
#include <chrono>
```

Wykres zależności załączania dla stoper.hh:



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



Komponenty

- class [IStoper](#)

Interfejs [IStoper](#).

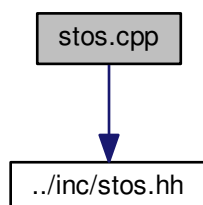
- class [Stoper](#)

Klasa [stoper](#) implementująca interfejs [IStoper](#).

5.13 Dokumentacja pliku stos.cpp

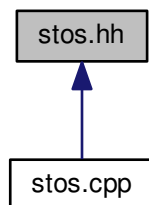
```
#include "../inc/stos.hh"
```

Wykres zależności załączania dla stos.cpp:



5.14 Dokumentacja pliku stos.hh

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



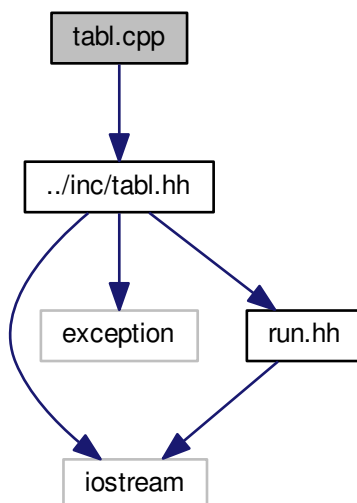
Komponenty

- class `IStos< T >`
- class `Stos`

5.15 Dokumentacja pliku tabl.cpp

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

Wykres zależności załączania dla tabl.cpp:

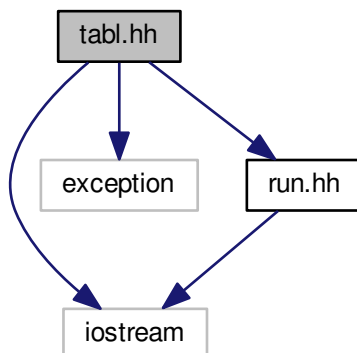


5.16 Dokumentacja pliku tabl.hh

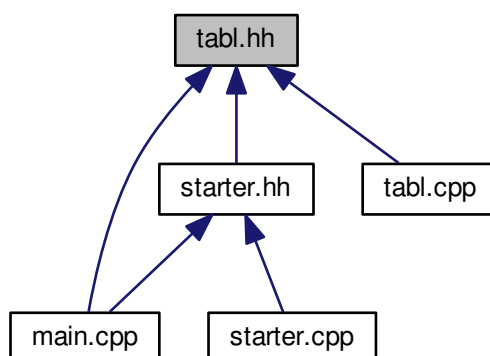
Definicja interfejsu `ltabn`, klasy `tabn` oraz klasy `tabn_test`.

```
#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 `ltabn`, 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

- ~ILista
 - ILista, 7
- ~IRunnable
 - IRunnable, 9
- ~IStoper
 - IStoper, 10
- ~Itabn
 - Itabn, 13
- ~Starter
 - Starter, 17
- ~tabn
 - tabn, 24
- ~tabn_test
 - tabn_test, 28
- aSize
 - Itabn, 14
 - tabn, 24
- add
 - ILista, 8
 - Itabn, 13
 - tabn, 24
- dumpToFile
 - Starter, 17
- get
 - ILista, 8
- getElapsedTimeMs
 - IStoper, 10
 - Stoper, 20
- IKolejka< T >, 7
- ILista
 - ~ILista, 7
 - add, 8
 - get, 8
 - isEmpty, 8
 - remove, 8
 - size, 8
- ILista< T >, 7
- IRunnable, 8
 - ~IRunnable, 9
 - prepare, 9
 - run, 9
- IStoper, 10
 - ~IStoper, 10
 - getElapsedTimeMs, 10
 - start, 11
 - stop, 11
- IStos< T >, 11
- isEmpty
 - ILista, 8
 - Itabn, 14
 - tabn, 26
- Itabn
 - ~Itabn, 13
 - aSize, 14
 - add, 13
 - isEmpty, 14
 - nOE, 14
 - operator[], 14, 15
 - remove, 15
 - showElements, 15
- Itabn< T >, 12
- Kolejka, 16
- kolejka.cpp, 31
- kolejka.hh, 31
- Lista, 16
- lista.cpp, 32
- lista.hh, 32
- main
 - main.cpp, 33
- main.cpp, 32
 - main, 33
- main.hh, 34
- nOE
 - Itabn, 14
 - tabn, 26
- operator[]
 - Itabn, 14, 15
 - tabn, 26
- prepare
 - IRunnable, 9
 - tabn_test, 28
- printResults
 - Starter, 17
- remove
 - ILista, 8
 - Itabn, 15
 - tabn, 27
- run
 - IRunnable, 9
 - tabn_test, 29

- run.cpp, [35](#)
- run.hh, [35](#)
- SIZE
 - tabl.hh, [43](#)
- setTestSize
 - Starter, [17](#)
- showElems
 - ltabn, [15](#)
 - tabn, [27](#)
- size
 - lLista, [8](#)
- start
 - IStoper, [11](#)
 - Stoper, [22](#)
- Starter, [16](#)
 - ~Starter, [17](#)
 - dumpToFile, [17](#)
 - printResults, [17](#)
 - setTestSize, [17](#)
 - Starter, [17](#)
 - test, [19](#)
- starter.cpp, [37](#)
- starter.hh, [37](#)
- stop
 - IStoper, [11](#)
 - Stoper, [22](#)
- Stoper, [19](#)
 - getElapsedTimeMs, [20](#)
 - start, [22](#)
 - stop, [22](#)
- stoper.cpp, [39](#)
- stoper.hh, [39](#)
- Stos, [22](#)
- stos.cpp, [40](#)
- stos.hh, [41](#)
- tabl.cpp, [41](#)
- tabl.hh, [42](#)
 - SIZE, [43](#)
- tabn
 - ~tabn, [24](#)
 - aSize, [24](#)
 - add, [24](#)
 - isEmpty, [26](#)
 - nOE, [26](#)
 - operator[], [26](#)
 - remove, [27](#)
 - showElems, [27](#)
 - tabn, [24](#)
- tabn< T >, [22](#)
- tabn_test, [27](#)
 - ~tabn_test, [28](#)
 - prepare, [28](#)
 - run, [29](#)
 - tabn_test, [28](#)
- test
 - Starter, [19](#)