

Pomiar czasu przeszukania listy jednokierunkowej

Generated by Doxygen 1.8.6

Wed Mar 30 2016 15:49:03



# Contents



# Chapter 1

## Opis programu

### Author

Kamil Kuczaj [218478@student.pwr.edu.pl](mailto:218478@student.pwr.edu.pl)

### 1.1 Wstęp

Program został zbudowany modułowo. W folderze `inc/` znajdują się pliki nagłówkowe. Folder `src/` zawiera pliki źródłowe. W głównym folderze zbudowany został Makefile. Pliki obiektowe są budowane w folderze `obj/` a następnie linkowane do głównego folderu (`prj/`). Testowano przy wykorzystaniu kompilatora `g++` w wersji 4.8.4 na systemie Linux Ubuntu 14.04.04 opartego o jądro 4.2.0-30-generic.

### 1.2 Licencja

Program udostępniam na licencji GPLv3.

### 1.3 Instalacja

Aby zbudować i jednocześnie odpalić program: `$ make`

Aby pozbyć się plików z końcówką `*~` lub zaczynających się na `#*`: `$ make order`

Aby pozbyć się programu wykonywalnego oraz plików obiektowych: `$ make clean`

Aby wyświetlić pomoc do pliku Makefile: `$ make help`



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

IKolejka . . . . .	??
Kolejka . . . . .	??
ILista . . . . .	??
Lista . . . . .	??
IRunnable . . . . .	??
Kolejka_test . . . . .	??
Lista_test . . . . .	??
Stos_test . . . . .	??
Tablica_test . . . . .	??
IStoper . . . . .	??
Stoper . . . . .	??
IStos . . . . .	??
Stos . . . . .	??
ITablica . . . . .	??
Tablica . . . . .	??
Lista::Node . . . . .	??
Sedzia . . . . .	??





## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">IKolejka</a>	Interfejs dla kolejki . . . . .	??
<a href="#">ILista</a>	Interfejs dla pojemnika <a href="#">Lista</a> . . . . .	??
<a href="#">IRunnable</a>	Interfejs dla biegacza . . . . .	??
<a href="#">IStoper</a>	Interfejs dla stopera . . . . .	??
<a href="#">IStos</a>	Interfejs dla każdego pojemnika . . . . .	??
<a href="#">ITablica</a>	Interfejs tablicy . . . . .	??
<a href="#">Kolejka</a>	Implementacja interfejsu <a href="#">IKolejka</a> w postaci klasy <a href="#">Kolejka</a> . . . . .	??
<a href="#">Kolejka_test</a>	. . . . .	??
<a href="#">Lista</a>	Klasa <a href="#">Lista</a> , która symuluje zachowanie klasy list z biblioteki STL . . . . .	??
<a href="#">Lista_test</a>	. . . . .	??
<a href="#">Lista::Node</a>	Implementacja wezlow dla listy . . . . .	??
<a href="#">Sedzia</a>	Implementacja klasy <a href="#">Sedzia</a> . . . . .	??
<a href="#">Stoper</a>	Implementacja klasy <a href="#">Stoper</a> . . . . .	??
<a href="#">Stos</a>	Implementacja klasy <a href="#">Stos</a> , złożonej z intow . . . . .	??
<a href="#">Stos_test</a>	. . . . .	??
<a href="#">Tablica</a>	Klasa <a href="#">Tablica</a> , w której odbywa się zapis dynamiczny elementow . . . . .	??
<a href="#">Tablica_test</a>	. . . . .	??



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/IKolejka.h	
Plik zawiera interfejs dla pojemnika Kolejka	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/ILista.h	
Plik zawiera interfejs dla pojemnika Lista oraz dla klasy Wezel	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/IRunnable.h	
Naglowek zawierajacy interfejs dla biegacza	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/IStoper.h	
Naglowek zawierajacy interfejs dla stopera	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/IStos.h	
Plik zawiera interfejs dla pojemnika Stos	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/ITablica.h	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Kolejka.h	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Kolejka_test.h	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Lista.h	
Implementacja jednokierunkowej listy	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Lista_test.h	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Sedzia.h	
Naglowek opisujacy implementacje Sedziego	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Stoper.h	
Implementacja interfejsu IStoper w klasie Stoper	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Stos.h	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Stos_test.h	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Tablica.h	
Implementacja interfejsu ITablica. Po konsultacji z prowadzacym zdecydowalem sie nie wyko- rzystowywac szablonow	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Tablica_test.h	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Kolejka.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Kolejka_test.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Lista.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Lista_test.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/main.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Sedzia.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Stoper.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Stos.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Stos_test.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Tablica.cpp	
	??
/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Tablica_test.cpp	
	??



## Chapter 5

# Class Documentation

### 5.1 IKolejka Class Reference

Interfejs dla kolejki.

```
#include <IKolejka.h>
```

Inheritance diagram for IKolejka:



#### Protected Member Functions

- virtual void **push** (std::string element)=0  
*Dodaje element na poczatek.*
- virtual std::string **pop** ()=0  
*Usuwa element z pojemnika.*
- virtual bool **empty** ()=0  
*Sprawdza czy pojemnika jest pusty.*
- virtual int **size** ()=0  
*Zwraca aktualny rozmiar pojemnika.*

#### 5.1.1 Detailed Description

Interfejs dla kolejki.

Abstrakcyjna klasa, która została utworzona na potrzeby ADT Abstract Data Types.

## 5.1.2 Member Function Documentation

### 5.1.2.1 `virtual bool IKolejka::empty ( )` `[protected]`, `[pure virtual]`

Sprawdza czy pojemnika jest pusty.

Sprawdza czy znajdują się jakieś elementy w pojemniku. Metoda czysto wirtualna.

Return values

<i>true</i>	Pojemnik pusty.
<i>false</i>	Pojemnik nie jest pusty.

Implemented in [Kolejka](#).

### 5.1.2.2 `virtual std::string IKolejka::pop ( )` `[protected]`, `[pure virtual]`

Usuwa element z pojemnika.

Usuwa element z pojemnika i zwraca go użytkownikowi. Metoda czysto wirtualna.

Returns

Usunięty element.

Implemented in [Kolejka](#).

### 5.1.2.3 `virtual void IKolejka::push ( std::string element )` `[protected]`, `[pure virtual]`

Dodaje element na początek.

Dodaje element na początek pojemnika.

Parameters

<i>in</i>	<i>element</i>	"Wpychany" element typu string.
-----------	----------------	---------------------------------

Implemented in [Kolejka](#).

### 5.1.2.4 `virtual int IKolejka::size ( )` `[protected]`, `[pure virtual]`

Zwraca aktualny rozmiar pojemnika.

Zwraca wartość, która reprezentuje obecną ilość elementów w pojemniku. Metoda czysto wirtualna.

Returns

Ilość elementów w pojemniku.

Implemented in [Kolejka](#).

The documentation for this class was generated from the following file:

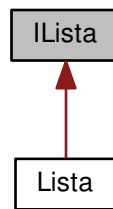
- `/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/IKolejka.h`

## 5.2 ILista Class Reference

Interfejs dla pojemnika [Lista](#).

```
#include <ILista.h>
```

Inheritance diagram for ILista:



### Protected Member Functions

- virtual void `add` (std::string item, int index)=0  
*Wstawia element w dowolnym miejscu listy.*
- virtual std::string `remove` (int index)=0  
*Usuwa element z dowolnego miejsca listy.*
- virtual bool `isEmpty` ()=0  
*Sprawdza czy lista jest pusta.*
- virtual std::string `get` (int index)=0  
*Zwraca element z dowolnego miejsca listy.*
- virtual int `size` ()=0  
*Zwraca rozmiar listy.*

#### 5.2.1 Detailed Description

Interfejs dla pojemnika [Lista](#).

Abstrakcyjna klasa, która została utworzona na potrzeby ADT Abstract Data Types.

#### 5.2.2 Member Function Documentation

**5.2.2.1** virtual void ILista::add ( std::string item, int index ) [protected],[pure virtual]

Wstawia element w dowolnym miejscu listy.

Wstawia element typu std::string w miejsce wskazywane przez zmienną index.

##### Parameters

in	item	Element wstawiany. Słowo.
in	index	Miejsce, w które ma być wstawiony element item.

Implemented in [Lista](#).

**5.2.2.2** virtual std::string ILista::get ( int index ) [protected],[pure virtual]

Zwraca element z dowolnego miejsca listy.

Zwraca element z miejsca wskazywanego przez zmienną index.

**Returns**

Zwraca element typu Type.

Implemented in [Lista](#).

**5.2.2.3 virtual bool ILista::isEmpty ( ) [protected],[pure virtual]**

Sprawdza czy lista jest pusta.

Sprawdza czy w liscie sa jakies elementy.

**Return values**

<i>true</i>	<a href="#">Lista</a> jest pusta.
<i>false</i>	<a href="#">Lista</a> nie jest pusta.

Implemented in [Lista](#).

**5.2.2.4 virtual std::string ILista::remove ( int index ) [protected],[pure virtual]**

Usuwa element z dowolnego miejsca listy.

Usuwa element z miejsca wskazywanego przez zmienna index.

**Returns**

Zwraca zawartosc komorki o tej indeksie.

Implemented in [Lista](#).

**5.2.2.5 virtual int ILista::size ( ) [protected],[pure virtual]**

Zwraca rozmiar listy.

Zwraca ilosc elementow w liscie.

**Returns**

Rozmiar listy.

Implemented in [Lista](#).

The documentation for this class was generated from the following file:

- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/inc/ILista.h](#)

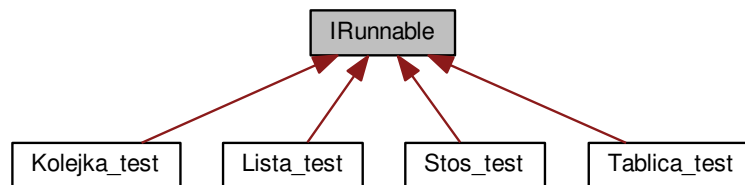
## 5.3 IRunnable Class Reference

Interfejs dla biegacza.

```
#include <IRunnable.h>
```



Inheritance diagram for IRunnable:



### Protected Member Functions

- virtual void `prepare` (int size)=0  
*Przygotowuje pojemnik przed wykonaniem czynnosci.*
- virtual void `run` ()=0  
*Odpalenie badanej czynnosci.*

#### 5.3.1 Detailed Description

Interfejs dla biegacza.

Klasa abstrakcyjna z metodami czysto wirtualnymi.

#### 5.3.2 Member Function Documentation

##### 5.3.2.1 virtual void IRunnable::prepare ( int size ) [protected],[pure virtual]

Przygotowuje pojemnik przed wykonaniem czynnosci.

Funkcja, która ma wykonać wszystkie dodatkowe czynności, których czasu nie będziemy mierzyć. Polega ona na wczytaniu konkretnej ilości elementów.

##### Parameters

<code>in</code>	<code>size</code>	Ilość elementów.
-----------------	-------------------	------------------

Implemented in [Lista\\_test](#), [Kolejka\\_test](#), [Stos\\_test](#), and [Tablica\\_test](#).

##### 5.3.2.2 virtual void IRunnable::run ( ) [protected],[pure virtual]

Odpalenie badanej czynności.

Funkcja, której ciałem mają być instrukcje, których czas chcemy zmierzyć.

Implemented in [Lista\\_test](#), [Tablica\\_test](#), [Kolejka\\_test](#), and [Stos\\_test](#).

The documentation for this class was generated from the following file:

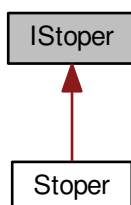
- `/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/IRunnable.h`

## 5.4 IStoper Class Reference

Interfejs dla stopera.

```
#include <IStoper.h>
```

Inheritance diagram for IStoper:



### Protected Member Functions

- virtual void `start` ()=0  
*Ma symulowac moment startu stopera.*
- virtual void `stop` ()=0
- virtual double `getElapsedTime` ()=0  
*Ma symulowac rezultat pokazania wyniku pomiaru czasu na stoperze.*
- virtual void `dumpToFile` (std::string file\_name)=0  
*Ma symulowac moment zapisu zmierzonego czasu na kartke papieru.*

### 5.4.1 Detailed Description

Interfejs dla stopera.

Klasa abstrakcyjna z metodami czysto wirtualnymi.

### 5.4.2 Member Function Documentation

**5.4.2.1** virtual void IStoper::dumpToFile ( std::string file\_name ) [protected],[pure virtual]

Ma symulowac moment zapisu zmierzonego czasu na kartke papieru.

Metoda czysto wirtualna.

Parameters

<code>file_name</code>	Nazwa pliku. Obiekt klasy string.
------------------------	-----------------------------------

Implemented in [Stoper](#).

**5.4.2.2** virtual double IStoper::getElapsedTime ( ) [protected],[pure virtual]

Ma symulowac rezultat pokazania wyniku pomiaru czasu na stoperze.

Metoda czysto wirtualna.

Implemented in [Stoper](#).

**5.4.2.3** `virtual void IStoper::start ( ) [protected],[pure virtual]`

Ma symulowac moment startu stopera.

Metoda czysto wirtualna.

Implemented in [Stoper](#).

**5.4.2.4** `virtual void IStoper::stop ( ) [protected],[pure virtual]`

Implemented in [Stoper](#).

The documentation for this class was generated from the following file:

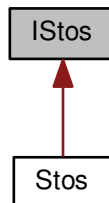
- `/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/IStoper.h`

## 5.5 IStos Class Reference

Interfejs dla każdego pojemnika.

```
#include <IStos.h>
```

Inheritance diagram for IStos:



### Protected Member Functions

- `virtual void push (std::string element)=0`  
*Dodaje element na poczatek.*
- `virtual std::string pop ()=0`  
*Usuwa element z pojemnika.*
- `virtual bool empty ()=0`  
*Sprawdza czy pojemnika jest pusty.*
- `virtual int size ()=0`  
*Zwraca aktualny rozmiar pojemnika.*

### 5.5.1 Detailed Description

Interfejs dla każdego pojemnika.

Abstrakcyjna klasa, która została utworzona na potrzeby ADT Abstract Data Types.

### 5.5.2 Member Function Documentation

#### 5.5.2.1 `virtual bool IStos::empty ( )` `[protected]`, `[pure virtual]`

Sprawdza czy pojemnika jest pusty.

Sprawdza czy znajdują się jakieś elementy w pojemniku. Metoda czysto wirtualna.

Return values

<i>true</i>	Pojemnik pusty.
<i>false</i>	Pojemnik nie jest pusty.

Implemented in [Stos](#).

#### 5.5.2.2 `virtual std::string IStos::pop ( )` `[protected]`, `[pure virtual]`

Usuwa element z pojemnika.

Usuwa element z pojemnika i zwraca go użytkownikowi. Metoda czysto wirtualna.

Returns

Usunięty element.

Implemented in [Stos](#).

#### 5.5.2.3 `virtual void IStos::push ( std::string element )` `[protected]`, `[pure virtual]`

Dodaje element na początek.

Dodaje element na początek pojemnika.

Parameters

<i>in</i>	<i>element</i>	"Wpychany" element typu <code>std::string</code> .
-----------	----------------	--

Implemented in [Stos](#).

#### 5.5.2.4 `virtual int IStos::size ( )` `[protected]`, `[pure virtual]`

Zwraca aktualny rozmiar pojemnika.

Zwraca wartość, która reprezentuje obecną ilość elementów w pojemniku. Metoda czysto wirtualna.

Returns

Ilość elementów w pojemniku.

Implemented in [Stos](#).

The documentation for this class was generated from the following file:

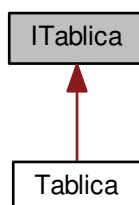
- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/inc/IStos.h](#)

## 5.6 ITablica Class Reference

Interfejs tablicy.

```
#include <ITablica.h>
```

Inheritance diagram for ITablica:



### Protected Member Functions

- virtual bool `isFull` ()=0  
*Sprawdza czy tablica jest pelna.*
- virtual void `increaseSize` ()=0  
*Zwieksza rozmiar tablicy.*
- virtual int `getSize` ()=0  
*Zwraca ilosc zapisanych elementow.*
- virtual int `getDesiredSize` () const =0  
*Zwraca maksymalny, pozadany rozmiar tablicy.*
- virtual void `setDesiredSize` (int t)=0  
*Ustawia pole desired\_size na wartosc, jaka potrzebujemy.*
- virtual int `operator[]` (int i) const =0  
*Akcesor to i-tego elementu tablicy.*
- virtual int & `operator[]` (int i)=0  
*Modyfikator do i-tego elementu tablicy.*

### 5.6.1 Detailed Description

Interfejs tablicy.

Wymuszony poprzez ISP (programowanie obiektowe SOLID).

### 5.6.2 Member Function Documentation

#### 5.6.2.1 virtual int ITablica::getDesiredSize ( ) const [protected],[pure virtual]

Zwraca maksymalny, pozadany rozmiar tablicy.

Zwraca ilosc elementow, ktore chcemy zapisac do tablicy. Nie reprezentuje ilosci zaalokowanej obecnie pamieci dla komorek. Jedynie idealny stan. Wymagany do testow. Pamietaj, ze indeksujemy od zera, wiec maksymalnie mozna zapisac do tablicy (`getDesiredSize()` - 1) elementow.

**Returns**

Maksymalna, satysfakcjonująca ilość elementów.

Implemented in [Tablica](#).

**5.6.2.2** `virtual int ITablica::getSize ( ) [protected],[pure virtual]`

Zwraca ilość zapisanych elementów.

Zwraca ilość elementów, które są w tablicy. Nie uwzględnia pustych komórek. Pamiętaj, że indeksujemy od zera, więc ostatni element ma indeks (`getSize() - 1`)

**Returns**

Ilość zapisanych elementów.

Implemented in [Tablica](#).

**5.6.2.3** `virtual void ITablica::increaseSize ( ) [protected],[pure virtual]`

Zwiększa rozmiar tablicy.

Alokuje pamięć dla nowej tablicy dynamicznej oraz kopiuje elementy starej tablicy do nowej. Następnie usuwa pamięć dla starej tablicy.

Implemented in [Tablica](#).

**5.6.2.4** `virtual bool ITablica::isFull ( ) [protected],[pure virtual]`

Sprawdza czy tablica jest pełna.

Sprawdza czy są jeszcze wolne komórki pamięci przydzielone tablicy.

**Return values**

<i>true</i>	<a href="#">Tablica</a> pełna. Należy zaalokować nową pamięć.
<i>false</i>	Jest jeszcze miejsce.

Implemented in [Tablica](#).

**5.6.2.5** `virtual int ITablica::operator[] ( int i ) const [protected],[pure virtual]`

Akcesor do i-tego elementu tablicy.

Umożliwia dostęp do i-tego elementu. Nie możemy tą metodą zmieniać wartości tego elementu, lecz możemy go odczytać.

**Returns**

i-ty element

Implemented in [Tablica](#).

**5.6.2.6** `virtual int& ITablica::operator[] ( int i ) [protected],[pure virtual]`

Modyfikator do i-tego elementu tablicy.

Umożliwia dostęp do i-tego elementu. Możemy tą metodą jedynie zmieniać wartość i-tego elementu, gdyż odwołujemy się do niego poprzez referencję.

**Returns**

Referencja do i-tego elementu

Implemented in [Tablica](#).

**5.6.2.7** `virtual void ITablica::setDesiredSize ( int t )` `[protected], [pure virtual]`

Ustawia pole `desired_size` na wartosc, jaka potrzebujemy.

Ustawia pole. Jest potrzebne gdyz w mechanizmach kontroli alokacji pamieci i zwiekszania rozmiaru tablicy zwracamy uwage na to, czy trzeba zwiekszyc rozmiar, czy nasza tablica jest juz wieksza.

Implemented in [Tablica](#).

The documentation for this class was generated from the following file:

- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/inc/ITablica.h](#)

## 5.7 Kolejka Class Reference

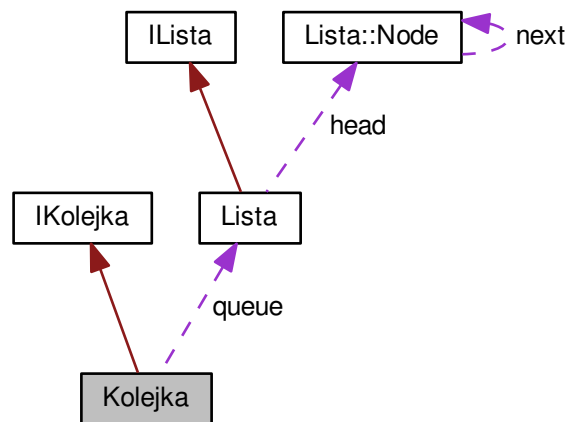
Implementacja interfejsu [IKolejka](#) w postaci klasy [Kolejka](#).

```
#include <Kolejka.h>
```

Inheritance diagram for Kolejka:



Collaboration diagram for Kolejka:



## Public Member Functions

- virtual void [push](#) (std::string element)  
*Dodaje element na poczatek.*
- virtual std::string [pop](#) ()  
*Usuwa element z pojemnika.*
- virtual bool [empty](#) ()  
*Sprawdza czy pojemnika jest pusty.*
- virtual int [size](#) ()  
*Zwraca aktualny rozmiar pojemnika.*
- void [print](#) ()  
*Wyswietla zawartosc kolejki.*

## Private Attributes

- [Lista queue](#)  
*Zawartosc kolejki.*

## Additional Inherited Members

### 5.7.1 Detailed Description

Implementacja interfejsu [IKolejka](#) w postaci klasy [Kolejka](#).

Korzysta z klasy [Lista](#), jako jej prywatne pole oraz calej jej funkcjonalnosc. W celu zrozumienia pelnej funkcjonalnosc klasy [Kolejka](#), prosze odwolac sie do dokumentacji klasy [Lista](#).



## 5.7.2 Member Function Documentation

### 5.7.2.1 `bool Kolejka::empty( )` [virtual]

Sprawdza czy pojemnika jest pusty.

Sprawdza czy znajdują się jakieś elementy w pojemniku. Metoda czysto wirtualna.

Return values

<i>true</i>	Pojemnik pusty.
<i>false</i>	Pojemnik nie jest pusty.

Implements [IKolejka](#).

Here is the call graph for this function:



### 5.7.2.2 `std::string Kolejka::pop( )` [virtual]

Usuwa element z pojemnika.

Usuwa element z pojemnika i zwraca go użytkownikowi. Metoda czysto wirtualna.

Returns

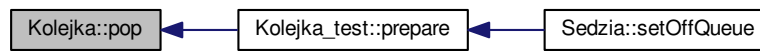
Usunięty element.

Implements [IKolejka](#).

Here is the call graph for this function:



Here is the caller graph for this function:

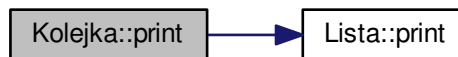


#### 5.7.2.3 void Kolejka::print ( )

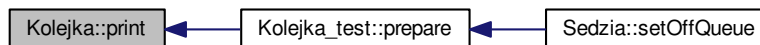
Wyswietla zawartosc kolejki.

Uzyteczna przy debugowaniu programu. Wyswietla kazde slowo w osobnej linii, zaczynajac od najstarszego.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.2.4 void Kolejka::push ( std::string *element* ) [virtual]

Dodaje element na poczatek.

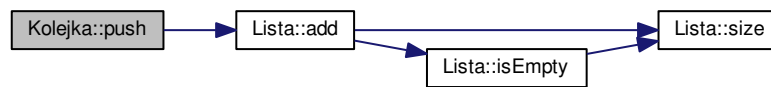
Dodaje element na poczatek pojemnika.

**Parameters**

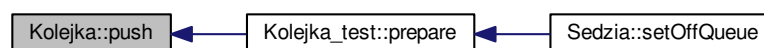
<code>in</code>	<code>element</code>	"Wpychany" element typu string.
-----------------	----------------------	---------------------------------

Implements [IKolejka](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.2.5 int Kolejka::size ( ) [virtual]

Zwraca aktualny rozmiar pojemnika.

Zwraca wartosc, ktora reprezentuje obecna ilosc elementow w pojemniku. Metoda czysto wirtualna.

##### Returns

Ilosc elementow w pojemniku.

Implements [IKolejka](#).

Here is the call graph for this function:



### 5.7.3 Member Data Documentation

#### 5.7.3.1 Lista Kolejka::queue [private]

Zawartosc kolejki.

Symuluje kolejke, poniewaz jest to bardzo prosta implementacja.

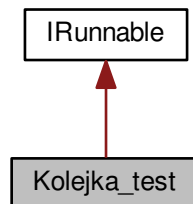
The documentation for this class was generated from the following files:

- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/inc/Kolejka.h](/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Kolejka.h)
- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/src/Kolejka.cpp](/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Kolejka.cpp)

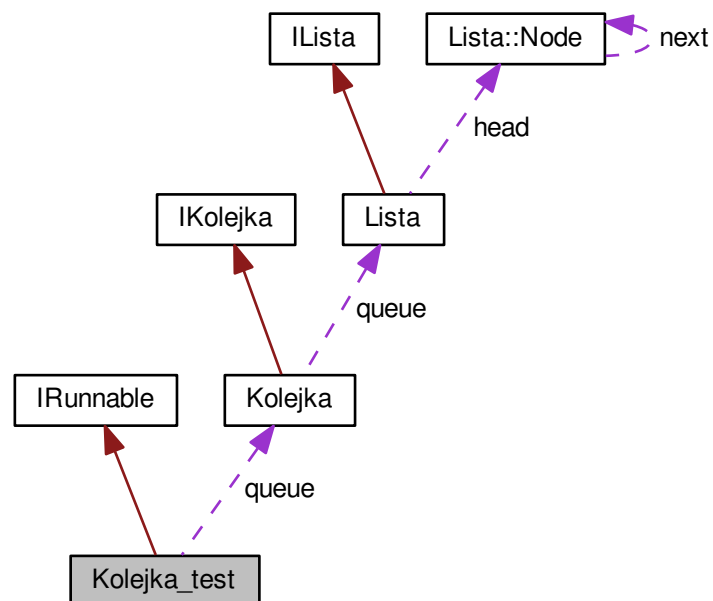
## 5.8 Kolejka\_test Class Reference

```
#include <Kolejka_test.h>
```

Inheritance diagram for Kolejka\_test:



Collaboration diagram for Kolejka\_test:



### Public Member Functions

- virtual void `prepare` (int size)  
*Przygotowuje pojemnik przed wykonaniem czynności.*
- virtual void `run` ()  
*Odpalenie badanej czynności.*

## Private Attributes

- [Kolejka queue](#)

## Additional Inherited Members

### 5.8.1 Member Function Documentation

#### 5.8.1.1 void Kolejka\_test::prepare ( int *size* ) [virtual]

Przygotowuje pojemnik przed wykonaniem czynnosci.

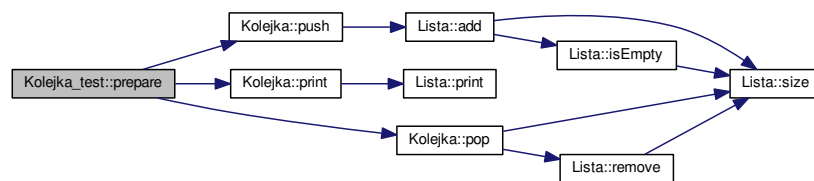
Funkcja, która ma wykonać wszystkie dodatkowe czynności, których czasu nie będziemy mierzyć. Polega ona na wczytaniu konkretnej ilości elementów.

#### Parameters

in	<i>size</i>	Ilość elementów.
----	-------------	------------------

Implements [IRunnable](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.8.1.2 void Kolejka\_test::run ( ) [virtual]

Odpalenie badanej czynności.

Funkcja, której ciałem mają być instrukcje, których czas chcemy zmierzyć.

Implements [IRunnable](#).

### 5.8.2 Member Data Documentation

### 5.8.2.1 Kolejka Kolejka\_test::queue [private]

The documentation for this class was generated from the following files:

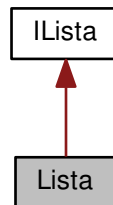
- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/inc/Kolejka\\_test.h](/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Kolejka_test.h)
- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/src/Kolejka\\_test.cpp](/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Kolejka_test.cpp)

## 5.9 Lista Class Reference

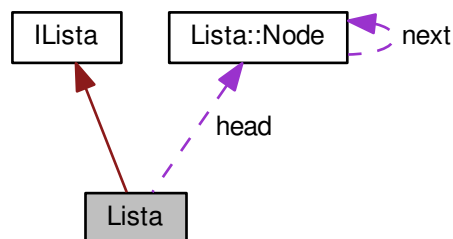
Klasa [Lista](#), która symuluje zachowanie klasy list z biblioteki STL.

```
#include <Lista.h>
```

Inheritance diagram for Lista:



Collaboration diagram for Lista:



### Classes

- struct [Node](#)  
*Implementacja węzłów dla listy.*

### Public Member Functions

- [Lista](#) ()

*Konstruktor.*

- `~Lista ()`

*Destruktor.*

- virtual void `add` (std::string item, int index)

*Wstawia element w dowolnym miejscu listy.*

- virtual std::string `remove` (int index)

*Usuwa element z dowolnego miejsca listy.*

- virtual bool `isEmpty` ()

*Sprawdza czy lista jest pusta.*

- virtual std::string `get` (int index)

*Zwraca element z dowolnego miejsca listy.*

- virtual int `size` ()

*Zwraca rozmiar listy.*

- void `print` ()

*Wypisuje zawartosc listy.*

- int `search` (std::string searched\_word)

*Wyszukuje podane slowo i zwraca jego indeks.*

## Private Attributes

- `Node * head`

*Pierwszy element listy.*

- int `size_of_list`

*Przechowuje rozmiar listy.*

## Additional Inherited Members

### 5.9.1 Detailed Description

Klasa `Lista`, która symuluje zachowanie klasy list z biblioteki STL.

Zajmuje sie dynamiczna alokacja pamieci. `Lista` jest jednokierunkowa. Mamy dostep do pierwszego elementu w liscie

### 5.9.2 Constructor & Destructor Documentation

#### 5.9.2.1 `Lista::Lista ( )`

Konstruktor.

Tworzy poczatek listy. Alokuje dla niego pamiec.

#### 5.9.2.2 `Lista::~~Lista ( )`

Destruktor.

Usuwa cala pamiec listy "skaczac" po jej elementach.

### 5.9.3 Member Function Documentation

#### 5.9.3.1 void Lista::add ( std::string *item*, int *index* ) [virtual]

Wstawia element w dowolnym miejscu listy.

Wstawia element typu Type w miejsce wskazywane przez zmienna index.

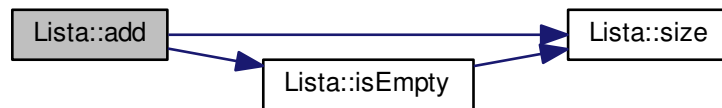


## Parameters

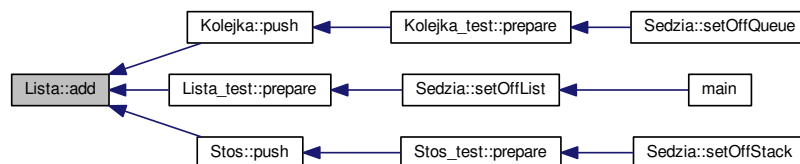
in	<i>item</i>	Element wstawiany. Słowo typu string.
in	<i>index</i>	Miejsce, w które ma być wstawiony element <i>item</i> .

Implements [ILista](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.9.3.2 `std::string Lista::get ( int index ) [virtual]`

Zwraca element z dowolnego miejsca listy.

Zwraca element z miejsca wskazywanego przez zmienną `index`. Wyjątki są typu: `const char * "Empty list"` - pusta lista `"Index out of bounds"` - przekroczono zakres, nie ma tylu elementów

## Returns

Zwraca element typu `std::string`.

Implements [ILista](#).

Here is the call graph for this function:



### 5.9.3.3 bool Lista::isEmpty ( ) [virtual]

Sprawdza czy lista jest pusta.

Sprawdza czy w liscie sa jakies elementy.

Return values

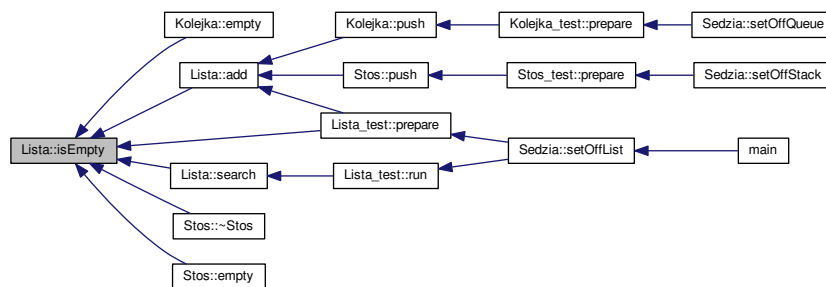
<i>true</i>	<a href="#">Lista</a> jest pusta.
<i>false</i>	<a href="#">Lista</a> nie jest pusta.

Implements [ILista](#).

Here is the call graph for this function:



Here is the caller graph for this function:

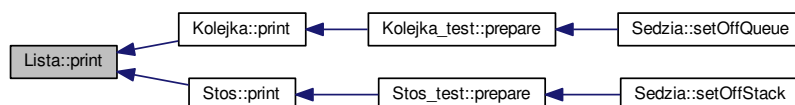


### 5.9.3.4 void Lista::print ( )

Wypisuje zawartosc listy.

Wypisuje kazdy element listy w osobnej linii. Na gorze znajduje sie poczatek listy.

Here is the caller graph for this function:



### 5.9.3.5 `std::string Lista::remove ( int index ) [virtual]`

Usuwa element z dowolnego miejsca listy.

Usuwa element z miejsca wskazywanego przez zmienna `index`.

#### Returns

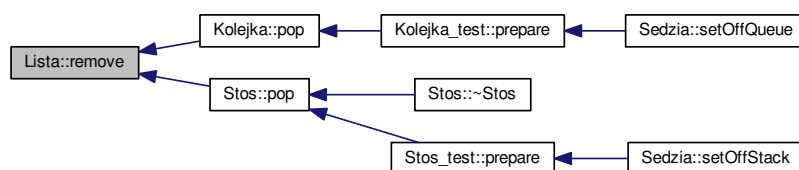
Zwraca słowo, które znajdowało się na tym indeksie.

Implements [ILista](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.9.3.6 `int Lista::search ( std::string searched_word )`

Wyszukuje podane słowo i zwraca jego indeks.

Wyszukuje w liście podane słowo typu `std::string`. Zwraca liczbę, która reprezentuje indeks z podanym słowem.\

#### Parameters

in	<i>searched_word</i>	Szukane słowo.
----	----------------------	----------------

#### Return values

-1	<a href="#">Lista</a> pusta.
-2	Nie ma takiego elementu w liście.



#### 5.9.4.1 Node\* Lista::head [private]

Pierwszy element listy.

Wskazuje na pierwszy element listy.

#### 5.9.4.2 int Lista::size\_of\_list [private]

Przechowuje rozmiar listy.

Dzięki zastosowaniu tej zmiennej, o wiele łatwiej debugować [Lista](#). Pozwala to na kontrole mechanizmów sprawdzania. Powinien być zawsze dodatni.

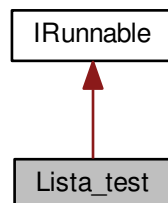
The documentation for this class was generated from the following files:

- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/inc/Lista.h](/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Lista.h)
- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/src/Lista.cpp](/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Lista.cpp)

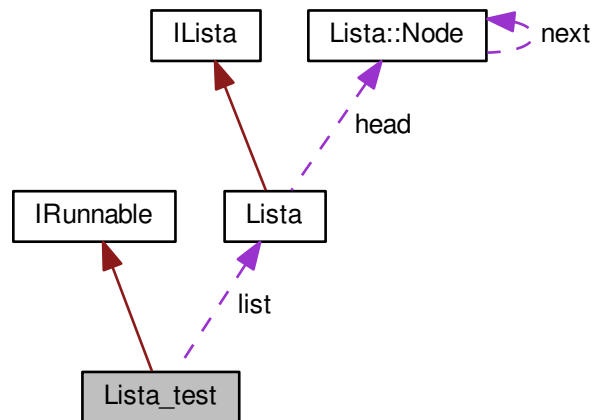
## 5.10 Lista\_test Class Reference

```
#include <Lista_test.h>
```

Inheritance diagram for Lista\_test:



Collaboration diagram for Lista\_test:



## Public Member Functions

- virtual void [run](#) ()  
*Szuka elementu.*
- virtual void [prepare](#) (int desired\_size)  
*Zapisuje liste slowami.*
- std::string [getRandomWordFromTheDict](#) ()  
*Losuje slowo ze slownika.*

## Private Attributes

- [Lista](#) *list*

## Additional Inherited Members

### 5.10.1 Member Function Documentation

#### 5.10.1.1 std::string Lista\_test::getRandomWordFromTheDict ( )

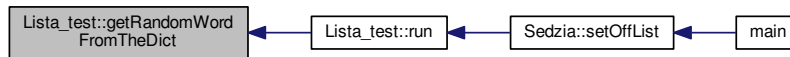
Losuje slowo ze slownika.

Wybiera slow z pelnego zakresu slownika.

## Returns

Losowe slowo typu string.

Here is the caller graph for this function:



### 5.10.1.2 void Lista\_test::prepare ( int *desired\_size* ) [virtual]

Zapisuje liste slowami.

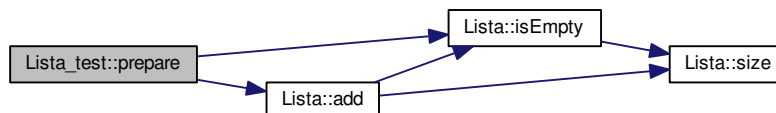
Zapisuje liste slowami zaczerpnietych ze slownika. !!! WAZNE !!!! Funkcja powinna byc uzyta tylko na poczatku, gdy cala lista jest pusta. Inaczej nastapi nadpisanie elementow poczatkowych.

## Parameters

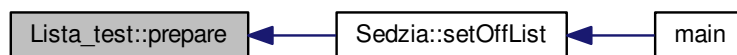
in	<i>desired_size</i>	Ile elementow ma zostac wczytanych.
----	---------------------	-------------------------------------

Implements [IRunnable](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.10.1.3 void Lista\_test::run ( ) [virtual]

Szuka elementu.

Szuka elementu wskazanego przez uzytkownika. W funkcji nastepuje Segmentation fault, gdy probujemy znalezc element, ktorego tam nie ma. W ogole sposob kodowania bledow gdy na nie napotka jest debilny ale nie mialem wystarczajaco duzo czasu aby to przerobic.

## Parameters

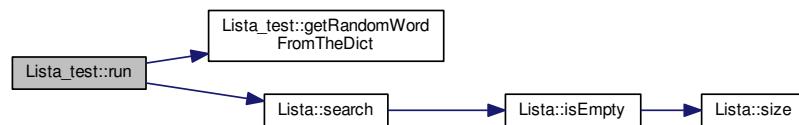
<i>in</i>	<i>desired_element</i>	Poszukiwana fraza.
-----------	------------------------	--------------------

## Return values

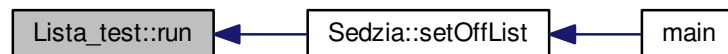
>0	Znalazl element i wyswietlil.
-1	Nie znalazl i nie wyswietlil elementu.
-2	<a href="#">Lista</a> pusta.

Implements [IRunnable](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.10.2 Member Data Documentation

### 5.10.2.1 Lista Lista\_test::list [private]

The documentation for this class was generated from the following files:

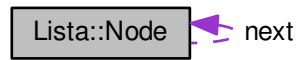
- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/inc/Lista\\_test.h](/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Lista_test.h)
- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/src/Lista\\_test.cpp](/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Lista_test.cpp)

## 5.11 Lista::Node Struct Reference

Implementacja wezlow dla listy.



Collaboration diagram for Lista::Node:



## Public Attributes

- `std::string element`  
*Element w wezle.*
- `Node * next`  
*Wskaźnik na następny wezel.*

### 5.11.1 Detailed Description

Implementacja wezlow dla listy.

Potrzebne do implementacji interfejsu listy. Zawiera pole typu string.

### 5.11.2 Member Data Documentation

#### 5.11.2.1 `std::string Lista::Node::element`

Element w wezle.

Co jest w wezle. Ma przechowywac pojedyncze slowo.

#### 5.11.2.2 `Node* Lista::Node::next`

Wskaźnik na następny wezel.

Wskazuje na następny wezel.

The documentation for this struct was generated from the following file:

- `/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Lista.h`

## 5.12 Sedzia Class Reference

Implementacja klasy [Sedzia](#).

```
#include <Sedzia.h>
```

## Public Member Functions

- `void setOffsetTable (int how_many)`  
*Funkcja, w ktorej odbywa sie zapis intow.*

- void [setOffList](#) (int how\_many, int trials\_count)  
*Funkcja, w której odbywa się pomiar czasu szukania w liście.*
- void [setOffStack](#) (int how\_many)  
*Funkcja, w której odbywa się zapis stringów do stosu.*
- void [setOffQueue](#) (int how\_many)  
*Funkcja, w której odbywa się zapis stringów do kolejki.*

### 5.12.1 Detailed Description

Implementacja klasy [Sedzia](#).

[Sedzia](#) wykorzystuje elementy klasy [Stoper](#) oraz klasy [Tablica](#). Mierzy czas wypełniania elementów Tablicy.

### 5.12.2 Member Function Documentation

#### 5.12.2.1 void Sedzia::setOffList ( int how\_many, int trials\_count )

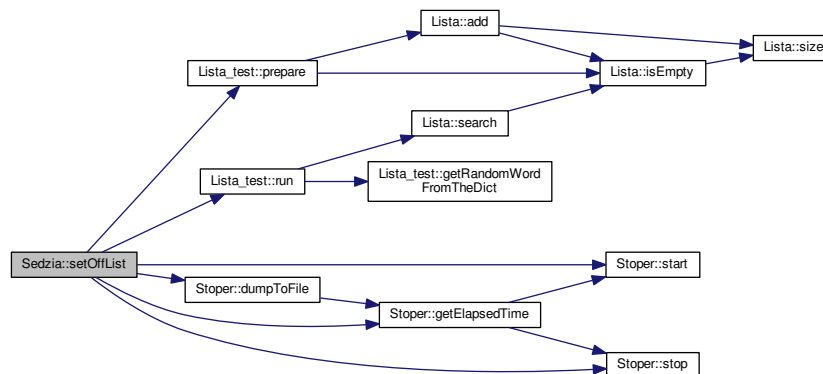
Funkcja, w której odbywa się pomiar czasu szukania w liście.

Losuje how\_many słów a potem znajduje wylosowany.

#### Parameters

in	<i>how_many</i>	Ilość słów jaka ma zostać wczytana do listy.
in	<i>trials_count</i>	Ile razy ma zostać wylosowane słowo ze słownika oraz ile razy ma zostać podjęta próba znalezienia go w liście.

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.2.2 void Sedzia::setOffQueue ( int *how\_many* )

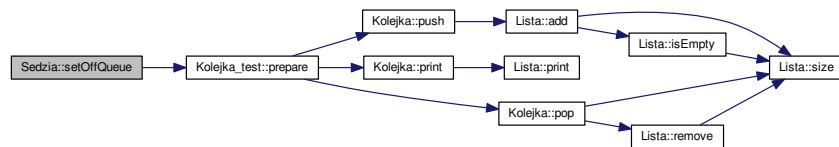
Funkcja, w której odbywa się zapis stringów do kolejki.

Podczas wykonywania tej funkcji uruchamiany jest [Stoper](#) oraz wypełniany jest element klasy [Stos](#) po uprzednim jej przygotowaniu. Słowa pobiera z tego samego słownika co lista.

## Parameters

in	<i>how_many</i>	Informacja iloma elementami ma zostać wypełniona tablica.
----	-----------------	---

Here is the call graph for this function:

5.12.2.3 void Sedzia::setOffStack ( int *how\_many* )

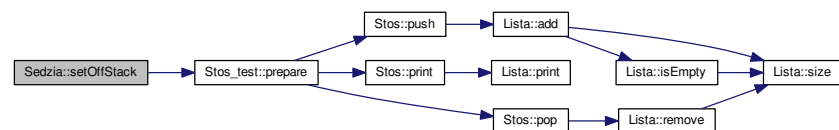
Funkcja, w której odbywa się zapis stringów do stosu.

Podczas wykonywania tej funkcji uruchamiany jest [Stoper](#) oraz wypełniany jest element klasy [Stos](#) po uprzednim jej przygotowaniu. Słowa pobiera z tego samego słownika co lista.

## Parameters

in	<i>how_many</i>	Informacja iloma elementami ma zostać wypełniona tablica.
----	-----------------	---

Here is the call graph for this function:

5.12.2.4 void Sedzia::setOffTable ( int *how\_many* )

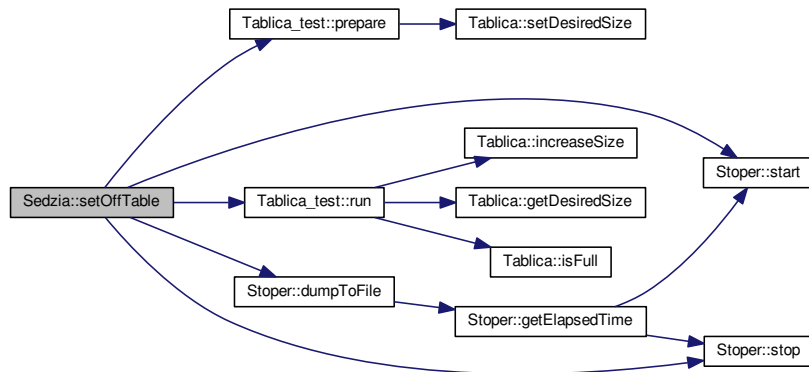
Funkcja, w której odbywa się zapis intów.

Podczas wykonywania tej funkcji uruchamiany jest [Stoper](#) oraz wypełniany jest element klasy [Tablica](#) po uprzednim jej przygotowaniu.

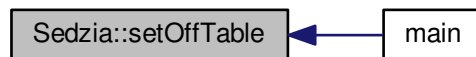
## Parameters

in	<i>how_many</i>	Informacja iloma elementami ma zostać wypełniona tablica.
----	-----------------	---

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

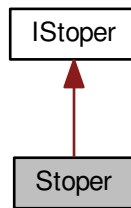
- `/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Sedzia.h`
- `/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Sedzia.cpp`

## 5.13 Stoper Class Reference

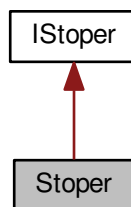
Implementacja klasy [Stoper](#).

```
#include <Stoper.h>
```

Inheritance diagram for Stoper:



Collaboration diagram for Stoper:



## Public Member Functions

- `Stoper ()`  
*Konstruktor bezparametryczny.*
- `~Stoper ()`  
*Destruktor.*
- `virtual void start ()`  
*Implementacja funkcji `start()` z interfejsu `IStoper`.*
- `virtual void stop ()`  
*Implementacja funkcji `stop()` z interfejsu `IStoper`.*
- `virtual double getElapsedTime ()`  
*Implementacja funkcji `getElapse()` z interfejsu `IStoper`.*
- `virtual void dumpToFile (std::string file_name)`  
*Implementacja funkcji `dumpToFile()` z interfejsu `IStoper`.*

## Private Attributes

- `timeval * start_time`  
*Moment startu stopera.*
- `timeval * stop_time`  
*Moment zatrzymania stopera.*

## Additional Inherited Members

### 5.13.1 Detailed Description

Implementacja klasy [Stoper](#).

W klasie [Stoper](#) zostały zaimplementowane metody pozwalające na pomiar czasu. Pomiar czasu odbywa się dzięki bibliotece `<sys/time.h>` a zapis do pliku korzysta z biblioteki `<fstream>`.

### 5.13.2 Constructor & Destructor Documentation

#### 5.13.2.1 `Stoper::Stoper ( )`

Konstruktor bezparametryczny.

Alokuje pamięć dla pól, ponieważ są wskaźnikami.

#### 5.13.2.2 `Stoper::~~Stoper ( )`

Destruktor.

Zwalnia pamięć po polach.

### 5.13.3 Member Function Documentation

#### 5.13.3.1 `void Stoper::dumpToFile ( std::string file_name ) [virtual]`

Implementacja funkcji [dumpToFile\(\)](#) z interfejsu [IStoper](#).

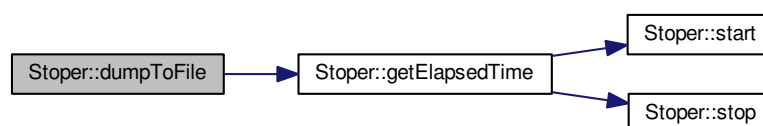
Zapisuje zmierzony czas do pliku o nazwie `"${file_name}.csv"`. Plik otwierany w trybie dopisywania (append) oraz wyjściowym (out). Plik .csv to tzw. Comma-Separated Values - łatwo je potem zaimportować do arkusza kalkulacyjnego oraz są zgodne z ogólnie przyjętym standardem.

##### Parameters

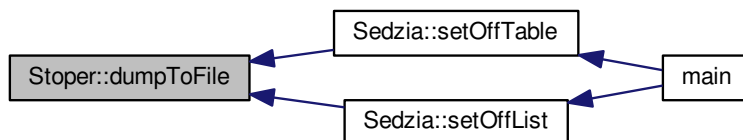
<i>file_name</i>	Nazwa pliku, do którego będą zapisane dane. Nazwa nie powinna zawierać rozszerzenia. Rozszerzenie jest dodawane w funkcji.
------------------	--

Implements [IStoper](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.13.3.2 double Stoper::getElapsedTime ( ) [virtual]

Implementacja funkcji `getElapse()` z interfejsu [IStoper](#).

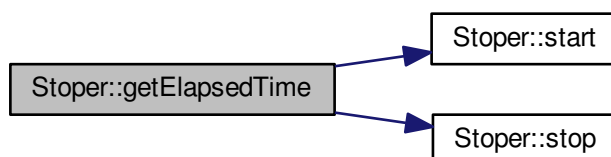
Oblicza czas pomiędzy czasem zapisanym w zmiennych `start_time` i `stop_time`.

##### Returns

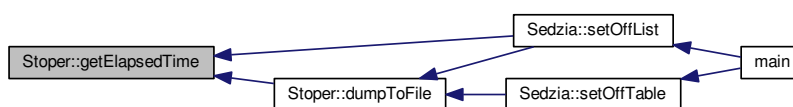
Zwraca zmierzony czas - roznica pomiędzy polem `start_time` a polem `stop_time`. Zwraca wynik w mikrosekundach.

Implements [IStoper](#).

Here is the call graph for this function:



Here is the caller graph for this function:



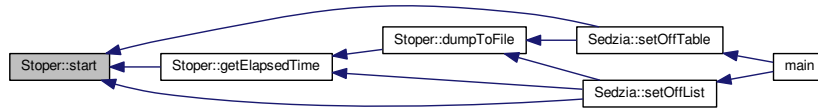
#### 5.13.3.3 void Stoper::start ( ) [virtual]

Implementacja funkcji `start()` z interfejsu [IStoper](#).

Zapisuje moment uruchomienia stopera. Korzysta z metody `gettimeofday()`.

Implements [IStoper](#).

Here is the caller graph for this function:



#### 5.13.3.4 void Stoper::stop ( ) [virtual]

Implementacja funkcji `stop()` z interfejsu [IStoper](#).

Zapisuje moment zatrzymania stopera. Korzysta z metody `gettimeofday()`.

Implements [IStoper](#).

Here is the caller graph for this function:



### 5.13.4 Member Data Documentation

#### 5.13.4.1 timeval\* Stoper::start\_time [private]

Moment startu stopera.

Element przechowujący informacje o czasie systemowym w momencie uruchomienia stopera. Element typu `timeval`. Nazwa zgodna konwencja podręcznika "Google C++ Style Guide".

#### 5.13.4.2 timeval\* Stoper::stop\_time [private]

Moment zatrzymania stopera.

Element przechowujący informacje o czasie systemowym w momencie zatrzymania stopera. Element typu `timeval`. Nazwa zgodna konwencja podręcznika "Google C++ Style Guide".

The documentation for this class was generated from the following files:

- `/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Stoper.h`
- `/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Stoper.cpp`

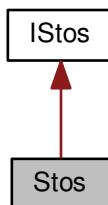
## 5.14 Stos Class Reference

Implementacja klasy [Stos](#), złożonej z intów.

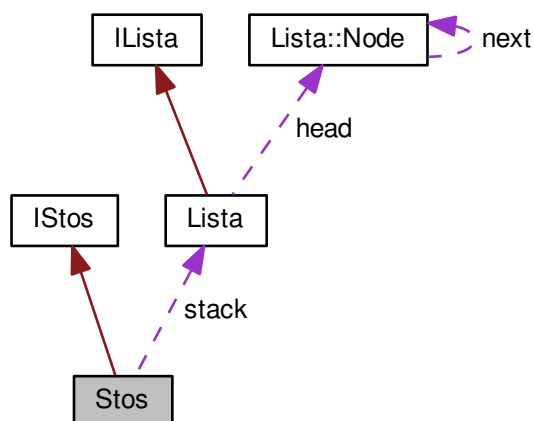


```
#include <Stos.h>
```

Inheritance diagram for Stos:



Collaboration diagram for Stos:



## Public Member Functions

- `Stos ()`  
*Bezparametryczny konstruktor.*
- `~Stos ()`  
*Destruktor.*
- `virtual void push (std::string item)`  
*Usuwa element z określonego miejsca.*
- `virtual std::string pop ()`  
*Usuwa element z pojemnika.*
- `virtual bool empty ()`  
*Sprawdza czy pojemnika jest pusty.*
- `virtual int size ()`

*Zwraca aktualny rozmiar pojemnika.*

- void [print](#) ()

*Wyswietla elementy stosu.*

## Private Attributes

- [Lista stack](#)

*Zawartosc stosu.*

## Additional Inherited Members

### 5.14.1 Detailed Description

Implementacja klasy [Stos](#), złożonej z intów.

Implementacja pojemnika, gdzie dostępny jest jedynie element będący "na gorze". Jej składowe elementy to stringi. Zdecydowałem się nie stosować szablonów ze względu na niepotrzebną komplikację. Zdecydowałem się na użycie listy jako elementu klasy, ponieważ był to wymóg prowadzącego. Nie ma ograniczeń rozmiaru.

### 5.14.2 Constructor & Destructor Documentation

#### 5.14.2.1 Stos::Stos ( )

Bezparametryczny konstruktor.

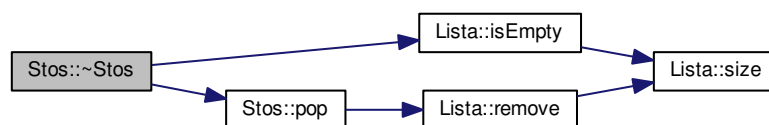
Inicjalizuje wierzchołek \*top jak wskaźnik na NULL.

#### 5.14.2.2 Stos::~~Stos ( )

Destruktor.

Popuje wszystkie elementy.

Here is the call graph for this function:



### 5.14.3 Member Function Documentation

#### 5.14.3.1 bool Stos::empty ( ) [virtual]

Sprawdza czy pojemnika jest pusty.

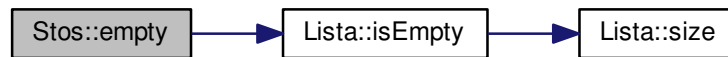
Sprawdza czy znajdują się jakieś elementy w pojemniku.

## Return values

<i>true</i>	Pojemnik pusty.
<i>false</i>	Pojemnik nie jest pusty.

Implements [IStos](#).

Here is the call graph for this function:

5.14.3.2 `std::string Stos::pop ( ) [virtual]`

Usuwa element z pojemnika.

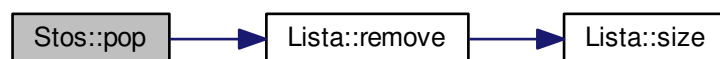
Usuwa element z pojemnika i zwraca go uzytkownikowi.

## Returns

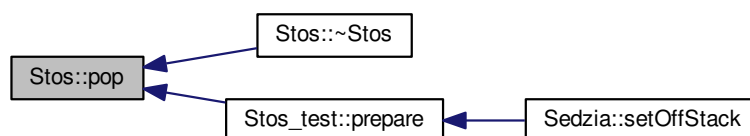
Usuniety element.

Implements [IStos](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.14.3.3 void Stos::print ( )

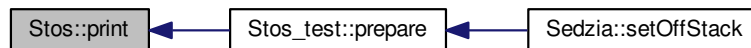
Wyswietla elementy stosu.

Wyswietla cala zawartosc stosu. Nie jest czescia interfesju.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.14.3.4 void Stos::push ( std::string item ) [virtual]

Usuwa element z okreslonego miejsca.

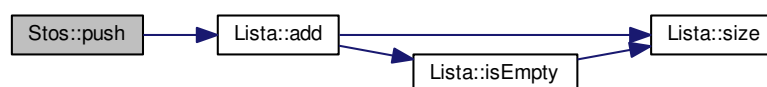
Usuwa i zwraca podany element znajdujacy sie w index-owym miejscu.

Parameters

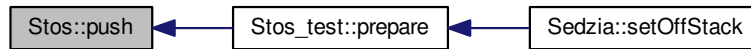
in	item	"Wpychany" element typu std::string.
----	------	--------------------------------------

Implements [IStos](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.14.3.5 int Stos::size ( ) [virtual]

Zwraca aktualny rozmiar pojemnika.

Zwraca wartosc, ktora reprezentuje obecna ilosc elementow w pojemniku.

##### Returns

Ilosc elementow w pojemniku.

Implements [IStos](#).

Here is the call graph for this function:



## 5.14.4 Member Data Documentation

### 5.14.4.1 Lista Stos::stack [private]

Zawartosc stosu.

Implementacja listy jako pole stosu jest wymogiem prowadzacego. Dodatkowo bardzo ulatiwa implementacje.

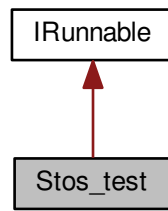
The documentation for this class was generated from the following files:

- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/inc/Stos.h](/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Stos.h)
- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/src/Stos.cpp](/home/kkuczaj/PAMSI/lab03_14.03/prj/src/Stos.cpp)

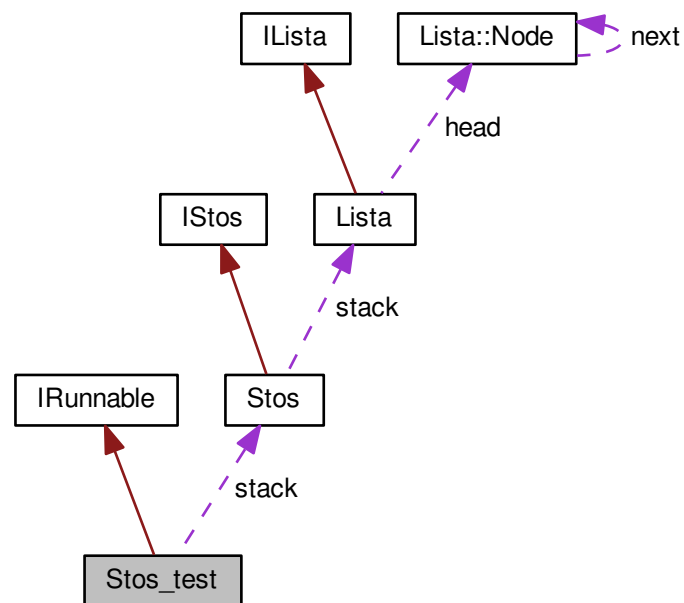
## 5.15 Stos\_test Class Reference

```
#include <Stos_test.h>
```

Inheritance diagram for Stos\_test:



Collaboration diagram for Stos\_test:



## Public Member Functions

- virtual void `prepare` (int size)  
*Przygotowuje pojemnik przed wykonaniem czynności.*
- virtual void `run` ()  
*Odpalenie badanej czynności.*

## Private Attributes

- `Stos stack`

## Additional Inherited Members

### 5.15.1 Member Function Documentation

#### 5.15.1.1 void Stos\_test::prepare ( int size ) [virtual]

Przygotowuje pojemnik przed wykonaniem czynności.

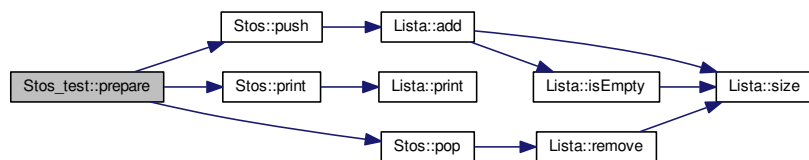
Funkcja, która ma wykonać wszystkie dodatkowe czynności, których czasu nie będziemy mierzyć. Polega ona na wczytaniu konkretnej ilości elementów.

##### Parameters

in	size	Ilość elementów.
----	------	------------------

Implements [IRunnable](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.15.1.2 void Stos\_test::run ( ) [virtual]

Odpalenie badanej czynności.

Funkcja, której ciałem mają być instrukcje, których czas chcemy zmierzyć.

Implements [IRunnable](#).

### 5.15.2 Member Data Documentation

#### 5.15.2.1 Stos Stos\_test::stack [private]

The documentation for this class was generated from the following files:

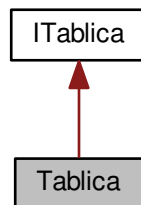
- /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Stos\_test.h
- /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Stos\_test.cpp

## 5.16 Tablica Class Reference

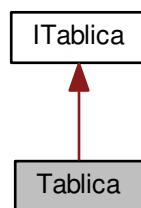
Klasa [Tablica](#), w której odbywa się zapis dynamiczny elementów.

```
#include <Tablica.h>
```

Inheritance diagram for Tablica:



Collaboration diagram for Tablica:



### Public Member Functions

- virtual bool [isFull](#) ()  
*Pozwala prosto okreslic, czy nalezy przydzielic pamiec.*
- virtual void [increaseSize](#) ()  
*Zwieksza rozmiar przydzielonej pamieci na stercie.*
- [Tablica](#) (int x=10)  
*Konstruktor parametryczny.*
- [~Tablica](#) ()  
*Destruktor.*
- virtual int [getSize](#) ()  
*Zwraca aktualny rozmiar tablicy dynamicznej.*
- virtual int [getDesiredSize](#) () const  
*Zwraca wartosc desired\_size.*
- virtual void [setDesiredSize](#) (int t)  
*Ustawia wartosc desired\_size.*



- virtual int [operator\[\]](#) (int i) const  
*Akcesor do tablicy.*
- virtual int & [operator\[\]](#) (int i)  
*Modyfikator do tablicy.*

### Private Attributes

- int \* [elements](#)  
*Wskaźnik do początku tablicy dynamicznej.*
- int [current\\_size](#)  
*Okresla aktualny rozmiar stosu.*
- int [desired\\_size](#)  
*Okresla pozadany rozmiar stosu.*
- int [index](#)  
*Okresla aktualny indeks.*

### Additional Inherited Members

#### 5.16.1 Detailed Description

Klasa [Tablica](#), w której odbywa się zapis dynamiczny elementów.

Implementuje metody interfejsu [ITablica](#). Zajmuje się dynamiczną alokacją pamięci.

#### 5.16.2 Constructor & Destructor Documentation

##### 5.16.2.1 `Tablica::Tablica ( int x = 10 ) [inline]`

Konstruktor parametryczny.

Umożliwia określenie początkowego rozmiaru tablicy. W przypadku braku określenia tego rozmiaru przyjmuje domyślną wartość równą 10.

##### Parameters

x	Okresla początkowa wielkość przydzielonej pamięci. Domyślna wartość w przypadku braku podania to 10.
---	--

##### 5.16.2.2 `Tablica::~Tablica ( ) [inline]`

Destruktor.

Usuwa pamięć przypisaną komórce, na którą wskazuje pole `*elements`.

#### 5.16.3 Member Function Documentation

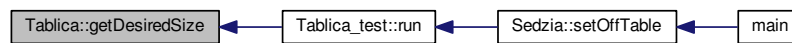
##### 5.16.3.1 `virtual int Tablica::getDesiredSize ( ) const [inline],[virtual]`

Zwraca wartość `desired_size`.

Zwraca rozmiar, który ma osiągnąć tablica. Może być większa niż `desired_size`.

Implementuje [ITablica](#).

Here is the caller graph for this function:



#### 5.16.3.2 `virtual int Tablica::getSize ( ) [inline],[virtual]`

Zwraca aktualny rozmiar tablicy dynamicznej.

Zwraca wartosc pola `current_size`.

##### Returns

Zwraca wartosc typu `int`. Reprezentuje ilosc danych w tablicy.

Implements [ITablica](#).

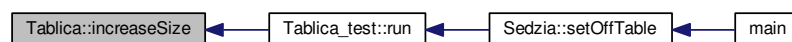
#### 5.16.3.3 `virtual void Tablica::increaseSize ( ) [inline],[virtual]`

Zwieksza rozmiar przydzielonej pamieci na stercie.

Metoda prywatna. Kopiuje elementy starej pamieci do komorki z nowo-przydzielona pamiecia. Usuwa stara pamiec.

Implements [ITablica](#).

Here is the caller graph for this function:



#### 5.16.3.4 `virtual bool Tablica::isFull ( ) [inline],[virtual]`

Pozwala prosto okreslic, czy nalezy przydzielic pamiec.

Metoda prywatna. Sluzy do okreslania czy nalezy wywolac metode [increaseSize\(\)](#).

##### Return values

<i>true</i>	Pamiec pelna. Nalezy zwiekszyć rozmiar.
<i>false</i>	Jest jeszcze wolne miejsce.

Implements [ITablica](#).

Here is the caller graph for this function:



#### 5.16.3.5 `virtual int Tablica::operator[] ( int i ) const [inline],[virtual]`

Akcesor do tablicy.

Umożliwia dostęp do tablicy.

Parameters

in	<i>i</i>	Indeks, w którym wartość tablicy ma zostać zwrócona.
----	----------	--

Returns

Wartość komórki tablicy, wskazywana przez *i*-ty indeks.

Implements [ITablica](#).

#### 5.16.3.6 `virtual int& Tablica::operator[] ( int i ) [inline],[virtual]`

Modyfikator do tablicy.

Umożliwia dostęp do zmiany *i*-tego elementu w tablicy.

Parameters

in	<i>i</i>	Wskazuje element, który ma zostać zmieniony.
----	----------	--

Returns

Referencja do *i*-tego elementu.

Implements [ITablica](#).

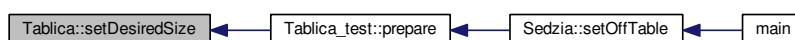
#### 5.16.3.7 `virtual void Tablica::setDesiredSize ( int t ) [inline],[virtual]`

Ustawia wartość `desired_size`.

Ustawia rozmiar, który ma osiągnąć tablica.

Implements [ITablica](#).

Here is the caller graph for this function:



## 5.16.4 Member Data Documentation

### 5.16.4.1 `int Tablica::current_size` [private]

Okresla aktualny rozmiar stosu.

Pole prywatne typu `int`. Rozmiar nigdy nie powinien byc ujemny.

### 5.16.4.2 `int Tablica::desired_size` [private]

Okresla pozadany rozmiar stosu.

Pole prywatne typu `int`. Rozmiar nigdy nie powinien byc ujemny. Zadawane w funkcji `prepare()`.

### 5.16.4.3 `int* Tablica::elements` [private]

Wskaźnik do początku tablicy dynamicznej.

Wskazuje na adres w pamieci sterty. Pole prywatne.

### 5.16.4.4 `int Tablica::index` [private]

Okresla aktualny indeks.

Pole prywatne typu `int`. Indeks nigdy nie powinien byc ujemny. Przechowuje indeks, pierwszego wolnego elementu tablicy, do ktorego mozliwy bedzie zapis.

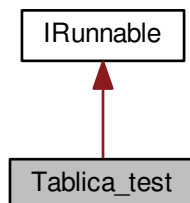
The documentation for this class was generated from the following file:

- `/home/kkuczaj/PAMSI/lab03_14.03/prj/inc/Tablica.h`

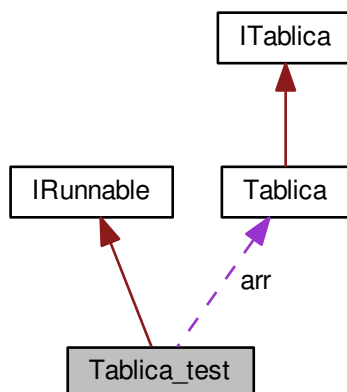
## 5.17 Tablica\_test Class Reference

```
#include <Tablica_test.h>
```

Inheritance diagram for `Tablica_test`:



Collaboration diagram for Tablica\_test:



## Public Member Functions

- virtual void [prepare](#) (int size)

Implementacja funkcji [prepare\(\)](#) interfesju [IRunnable](#).

- virtual void [run](#) ()

Implementacja funkcji [run\(\)](#) interfesju [IRunnable](#).

## Private Attributes

- [Tablica](#) arr

## Additional Inherited Members

### 5.17.1 Member Function Documentation

5.17.1.1 virtual void `Tablica_test::prepare ( int size )` `[inline]`,`[virtual]`

Implementacja funkcji [prepare\(\)](#) interfesju [IRunnable](#).

Zapisuje pozadany rozmiar do pola `desired_size`.

#### Parameters

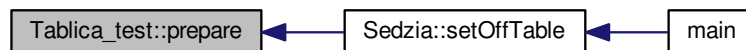
<i>size</i>	Parametr typu unsigned int, gdyz rozmiar nie powinien nigdy byc ujemny. Jego wartosc zapisywana jest do pola <code>desired_size</code> .
-------------	--

Implements [IRunnable](#).

Here is the call graph for this function:



Here is the caller graph for this function:



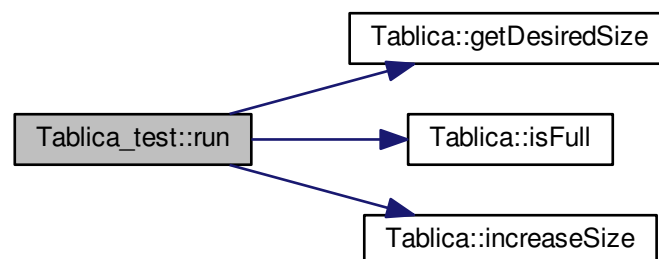
#### 5.17.1.2 `virtual void Tablica_test::run ( ) [inline],[virtual]`

Implementacja funkcji `run()` interfesju [IRunnable](#).

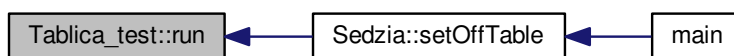
Uruchamia "bieg", w którym następuje zapis elementów do poszczególnych elementów tablicy dynamicznej. Tam odbywa się alokacja pamięci oraz instrukcje warunkowe.

Implements [IRunnable](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.17.2 Member Data Documentation

### 5.17.2.1 Tablica Tablica\_test::arr [private]

The documentation for this class was generated from the following file:

- [/home/kkuczaj/PAMSI/lab03\\_14.03/prj/inc/Tablica\\_test.h](#)





## Chapter 6

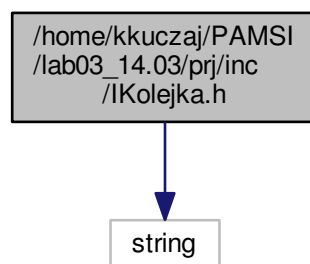
# File Documentation

### 6.1 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/IKolejka.h File Reference

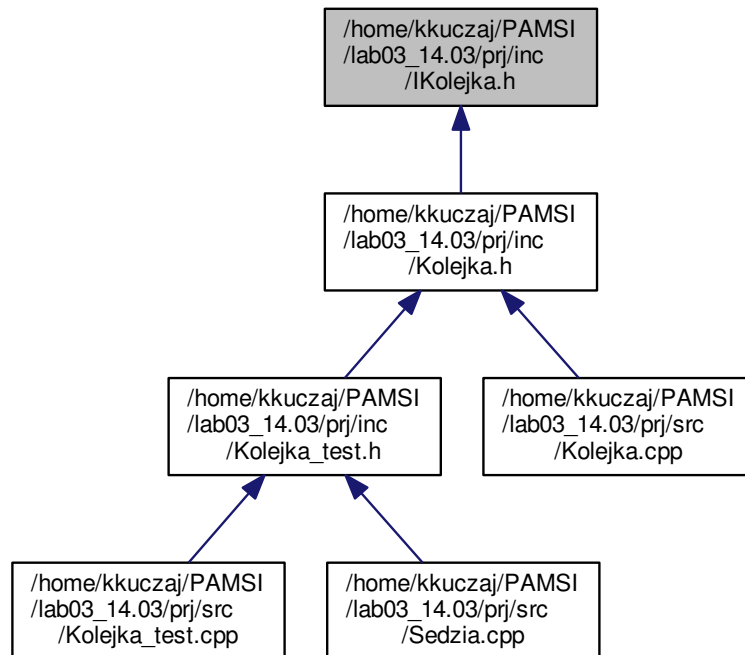
Plik zawiera interfejs dla pojemnika [Kolejka](#).

```
#include <string>
```

Include dependency graph for IKolejka.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [IKolejka](#)

*Interfejs dla kolejki.*

### 6.1.1 Detailed Description

Plik zawiera interfejs dla pojemnika [Kolejka](#). Nie zdecydowano się na użycie szablonów, gdyż zbyt komplikuje to budowę programu.

## Author

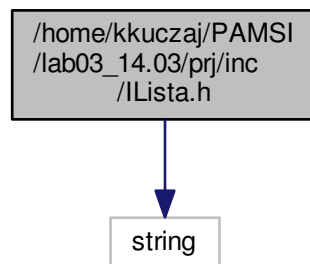
- Kamil Kuczaj.

## 6.2 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/ILista.h File Reference

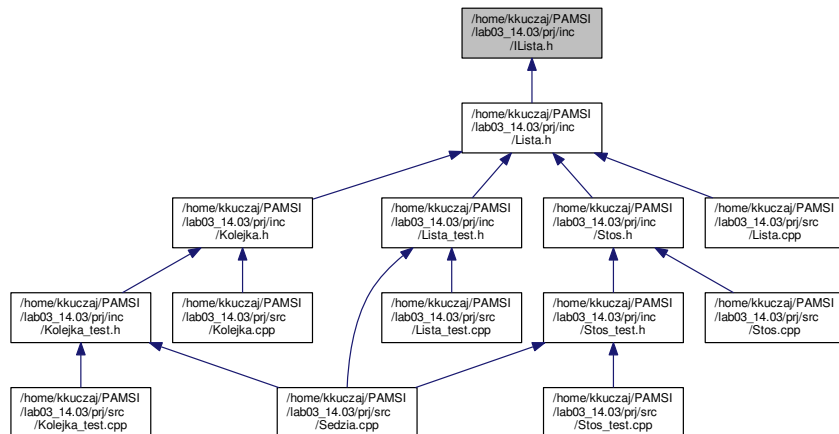
Plik zawiera interfejs dla pojemnika [Lista](#) oraz dla klasy Wezel.

```
#include <string>
```

Include dependency graph for ILista.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [ILista](#)

Interfejs dla pojemnika [Lista](#).

### 6.2.1 Detailed Description

Plik zawiera interfejs dla pojemnika [Lista](#) oraz dla klasy Wezel. Wezel jest elementem listy. Uzycie szablonow zbyttnio komplikuje implementacje, wiec odrzucilem ich zastosowanie.

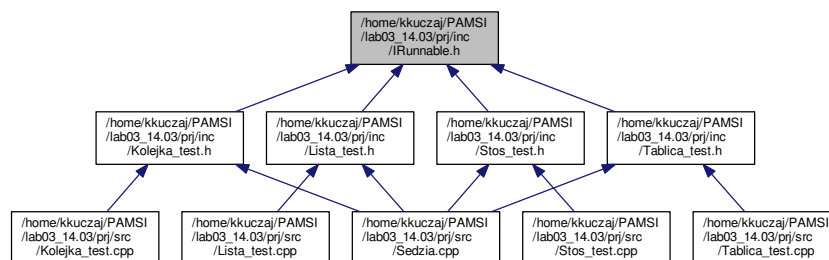
Author

Kamil Kuczaj.

## 6.3 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/IRunnable.h File Reference

Naglowek zawierajacy interfejs dla biegacza.

This graph shows which files directly or indirectly include this file:



## Classes

- class [IRunnable](#)

*Interfejs dla biegacza.*

### 6.3.1 Detailed Description

Naglowek zawierajacy interfejs dla biegacza.

Author

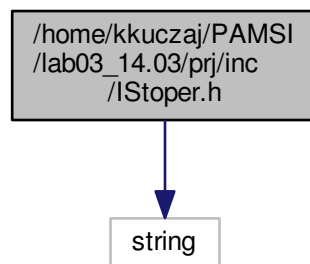
Kamil Kuczaj

## 6.4 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/IStoper.h File Reference

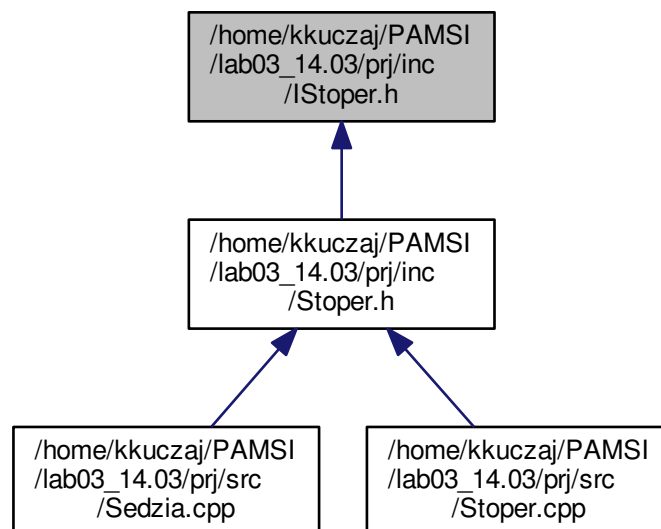
Naglowek zawierajacy interfejs dla stopera.

```
#include <string>
```

Include dependency graph for IStoper.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [IStoper](#)

*Interfejs dla stopera.*

### 6.4.1 Detailed Description

Naglowek zawierajacy interfejs dla stopera.

#### Author

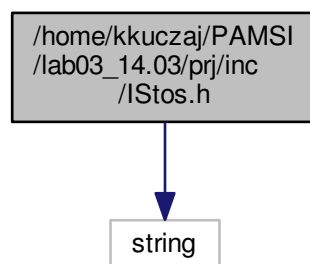
Kamil Kuczaj

## 6.5 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/IStos.h File Reference

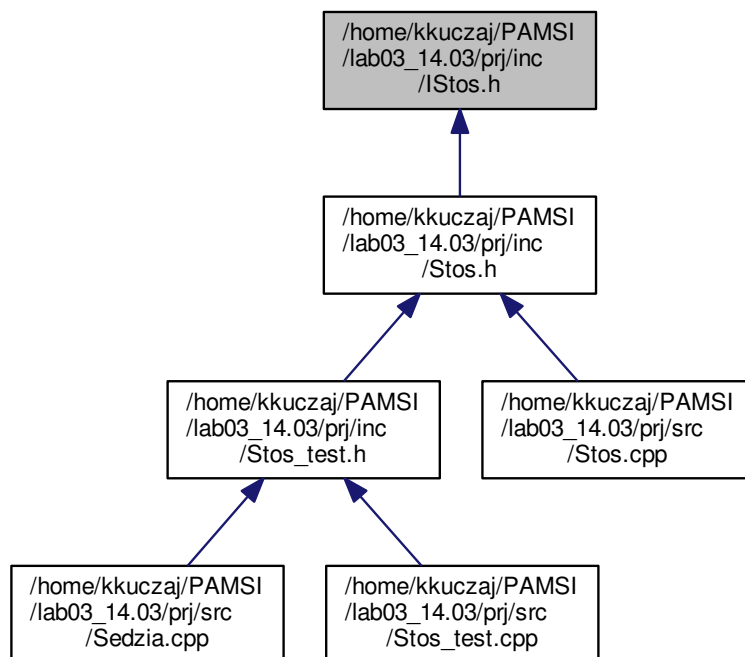
Plik zawiera interfejs dla pojemnika [Stos](#).

```
#include <string>
```

Include dependency graph for IStos.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [IStos](#)

*Interfejs dla każdego pojemnika.*

### 6.5.1 Detailed Description

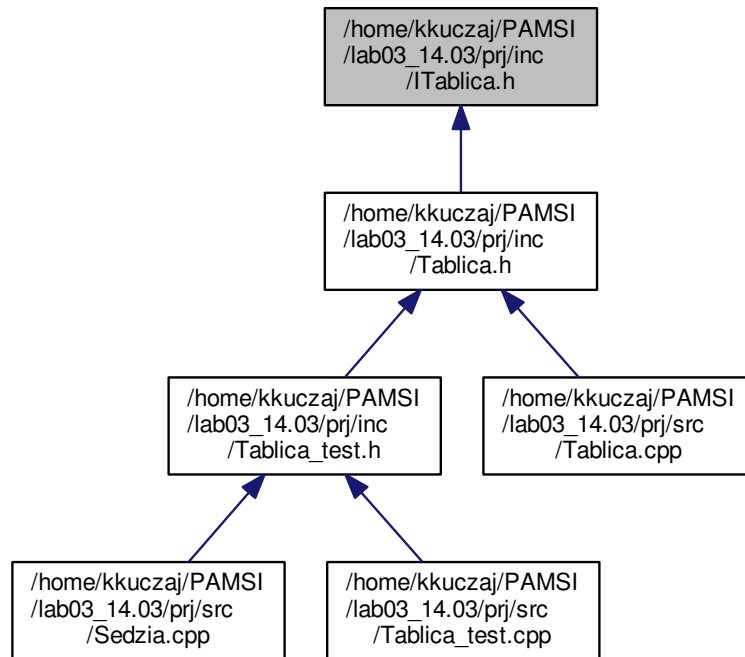
Plik zawiera interfejs dla pojemnika [Stos](#). Nie zdecydowano się na użycie szablonów, gdyż zbyt komplikuje to budowę programu.

Author

Kamil Kuczaj.

## 6.6 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/ITablica.h File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class [ITablica](#)  
*Interfejs tablicy.*

### 6.6.1 Detailed Description

Author

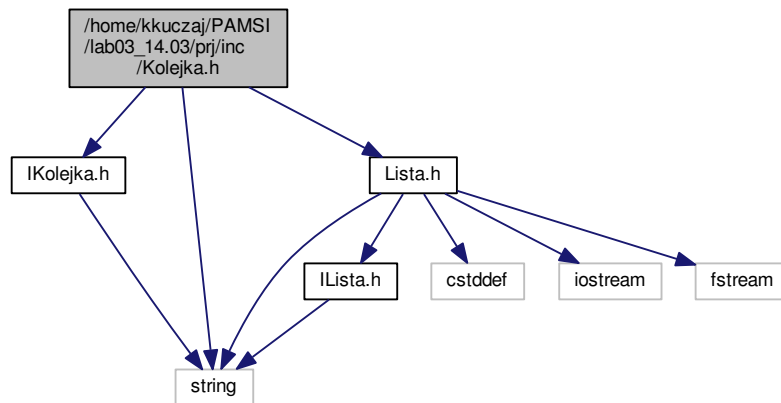
Kamil Kuczaj

## 6.7 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Kolejka.h File Reference

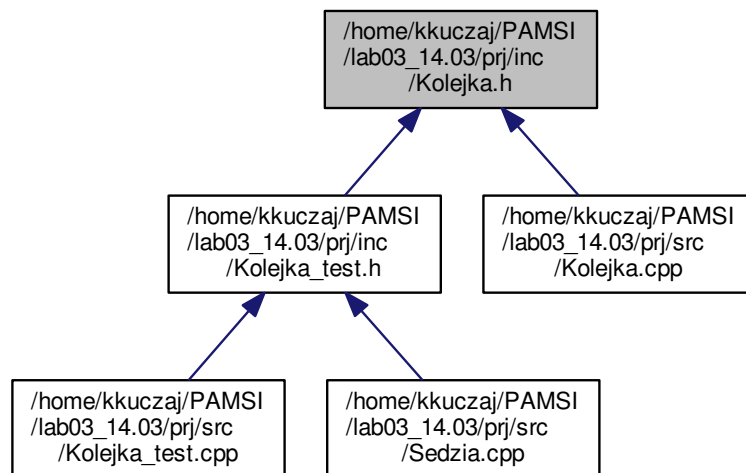
```
#include "IKolejka.h"  
#include "Lista.h"  
#include <string>
```



Include dependency graph for Kolejka.h:



This graph shows which files directly or indirectly include this file:



## Classes

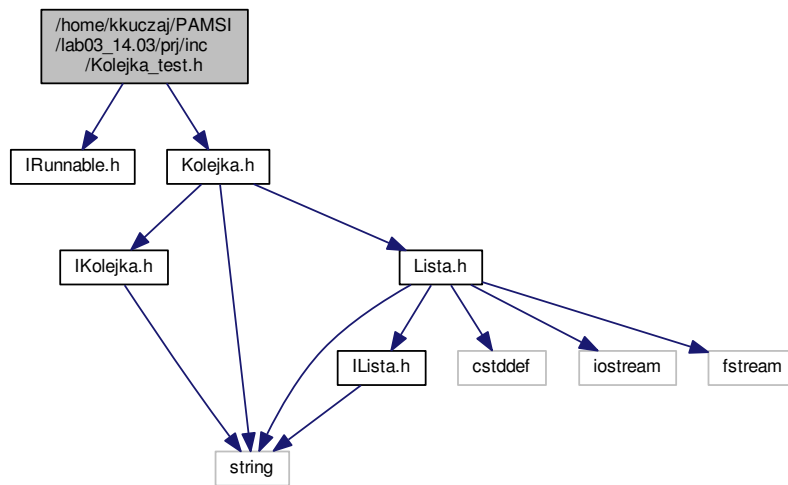
- class [Kolejka](#)

Implementacja interfejsu [IKolejka](#) w postaci klasy [Kolejka](#).

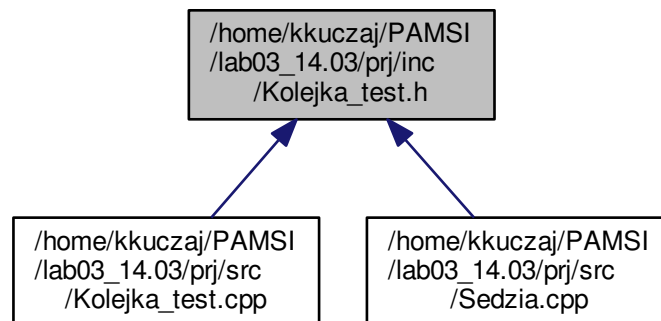
## 6.8 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Kolejka\_test.h File Reference

```
#include "IRunnable.h"
#include "Kolejka.h"
```

Include dependency graph for Kolejka\_test.h:



This graph shows which files directly or indirectly include this file:



## Classes

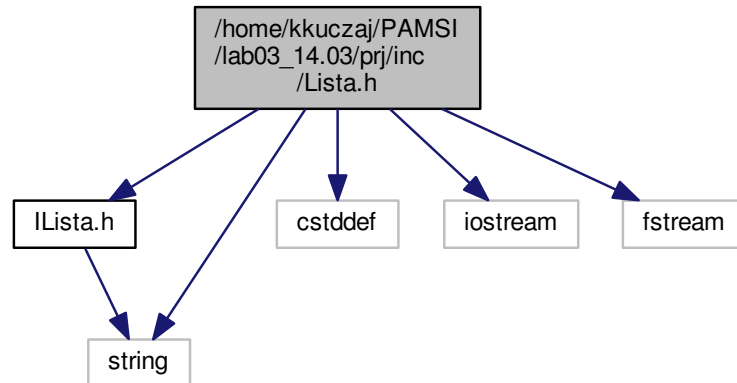
- class [Kolejka\\_test](#)

## 6.9 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Lista.h File Reference

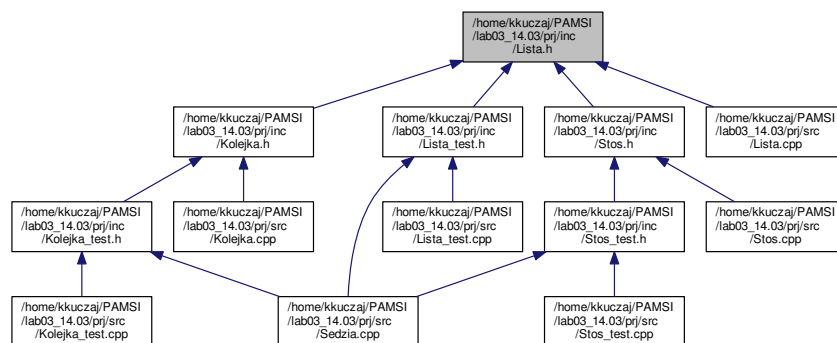
Implementacja jednokierunkowej listy.

```
#include "ILista.h"
#include <cstdlib>
#include <string>
#include <iostream>
#include <fstream>
```

Include dependency graph for Lista.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Lista](#)  
*Klasa [Lista](#), która symuluje zachowanie klasy list z biblioteki STL.*
- struct [Lista::Node](#)  
*Implementacja węzłów dla listy.*

### 6.9.1 Detailed Description

Implementacja jednokierunkowej listy. Ze względu na komplikacje implementacji mechanizmów przy użyciu szablonów, zdecydowałem się je usunąć z konstrukcji programu.

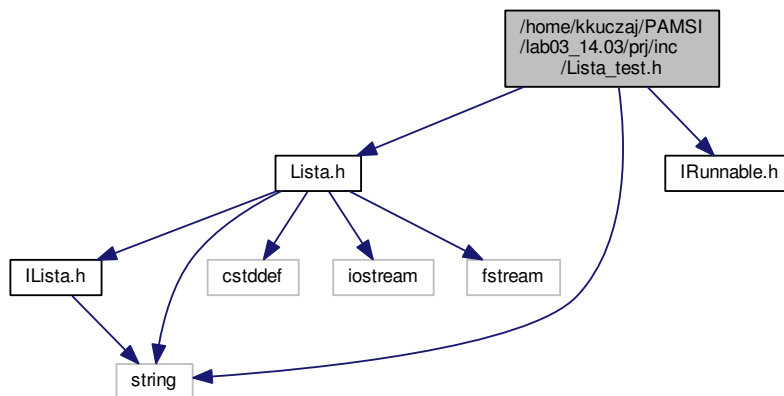
## Author

Kamil Kuczaj.

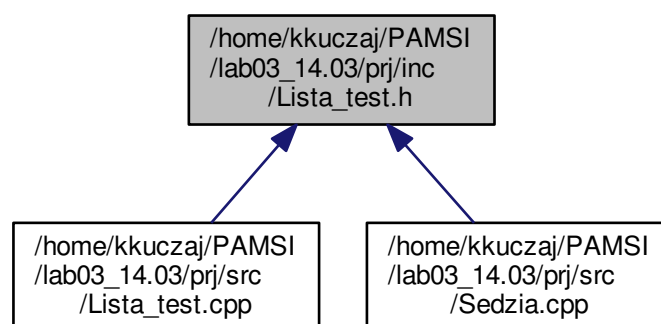
## 6.10 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Lista\_test.h File Reference

```
#include "Lista.h"  
#include "IRunnable.h"  
#include <string>
```

Include dependency graph for Lista\_test.h:



This graph shows which files directly or indirectly include this file:



## Classes

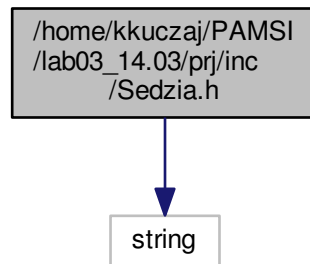
- class [Lista\\_test](#)

## 6.11 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Sedzia.h File Reference

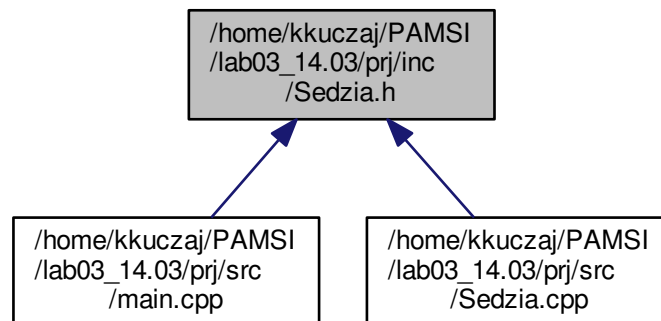
Nagłówek opisujący implementację Sedziego.

```
#include <string>
```

Include dependency graph for Sedzia.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Sedzia](#)  
*Implementacja klasy [Sedzia](#).*

#### 6.11.1 Detailed Description

Nagłówek opisujący implementację Sedziego.

Author

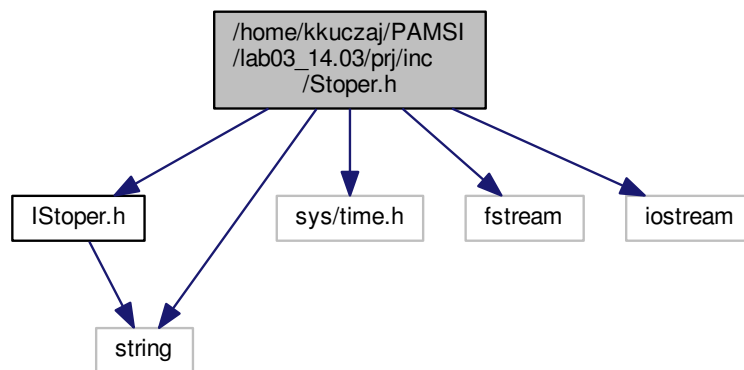
Kamil Kuczaj

## 6.12 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Stoper.h File Reference

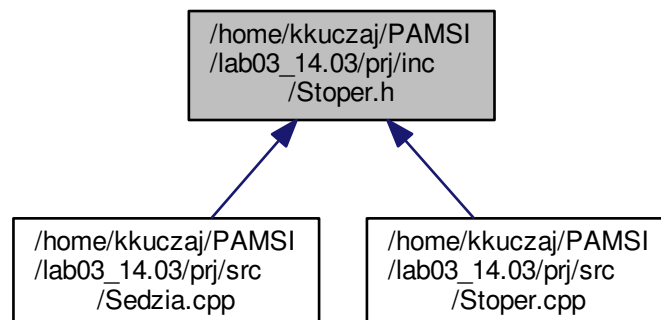
Implementacja interfejsu [IStoper](#) w klasie [Stoper](#).

```
#include "IStoper.h"  
#include <sys/time.h>  
#include <fstream>  
#include <iostream>  
#include <string>
```

Include dependency graph for Stoper.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Stoper](#)

*Implementacja klasy [Stoper](#).*

### 6.12.1 Detailed Description

Implementacja interfejsu [IStoper](#) w klasie [Stoper](#).

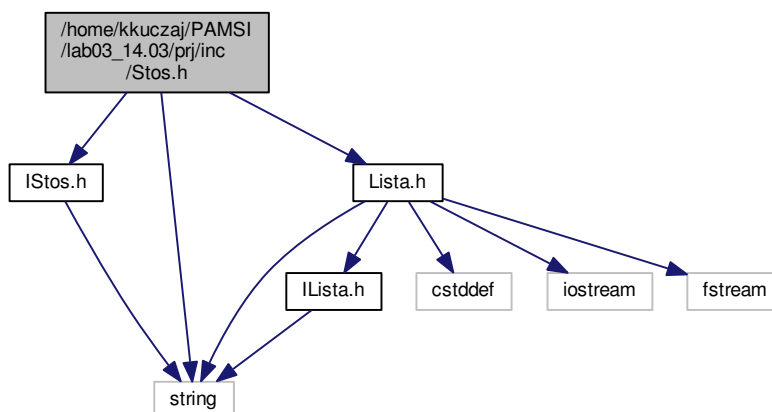
#### Author

Kamil Kuczaj

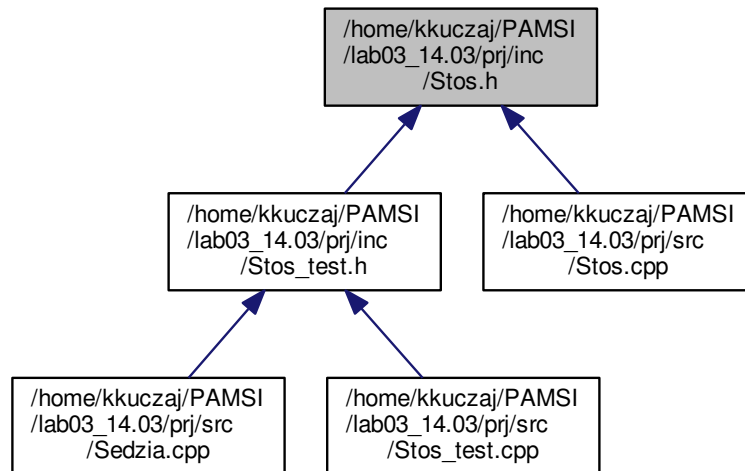
## 6.13 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Stos.h File Reference

```
#include "IStos.h"
#include "Lista.h"
#include <string>
```

Include dependency graph for Stos.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Stos](#)

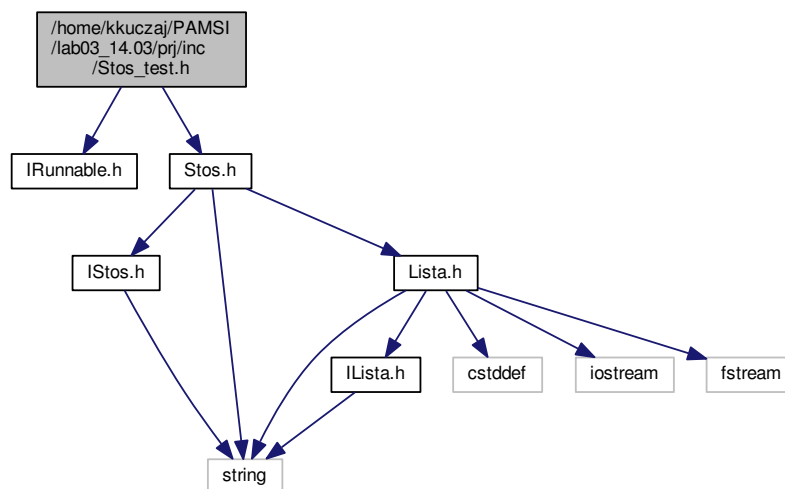
Implementacja klasy [Stos](#), zlozonej z intow.

## 6.14 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Stos\_test.h File Reference

```
#include "IRunnable.h"
```

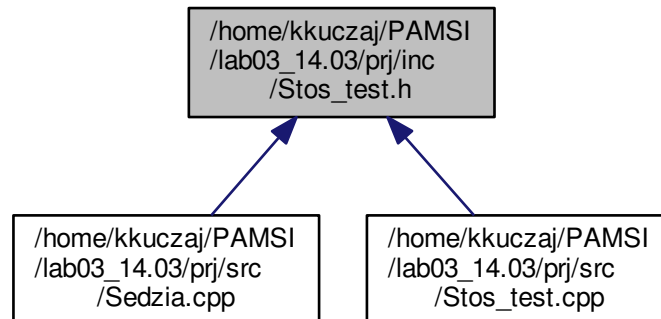
```
#include "Stos.h"
```

Include dependency graph for `Stos_test.h`:





This graph shows which files directly or indirectly include this file:



## Classes

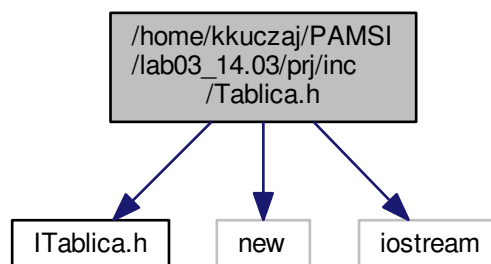
- class [Stos\\_test](#)

## 6.15 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Tablica.h File Reference

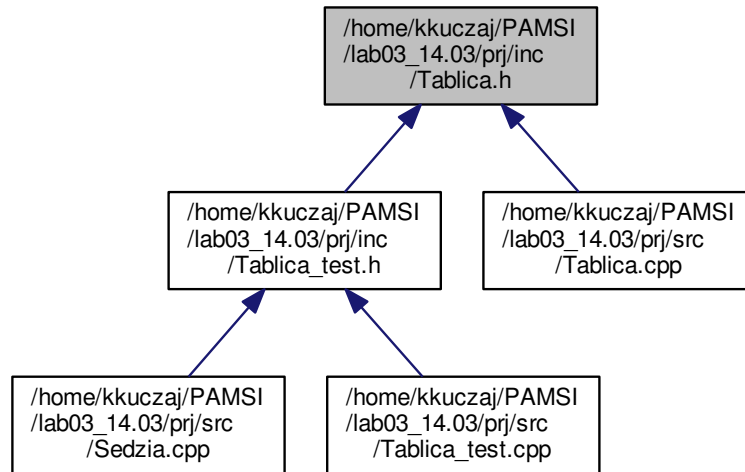
Implementacja interfejsu `ITablica`. Po konsultacji z prowadzącym zdecydowałem się nie wykorzystywać szablonów.

```
#include "ITablica.h"  
#include <new>  
#include <iostream>
```

Include dependency graph for `Tablica.h`:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Tablica](#)

*Klasa [Tablica](#), w której odbywa się zapis dynamiczny elementów.*

### 6.15.1 Detailed Description

Implementacja interfejsu ITablica. Po konsultacji z prowadzącym zdecydowałem się nie wykorzystywać szablonów.

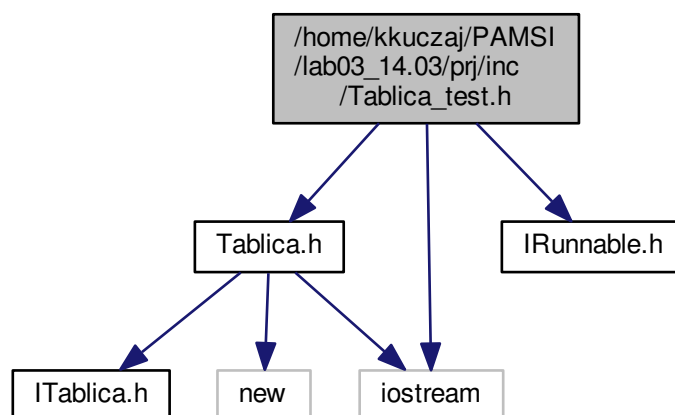
#### Author

Kamil Kuczaj

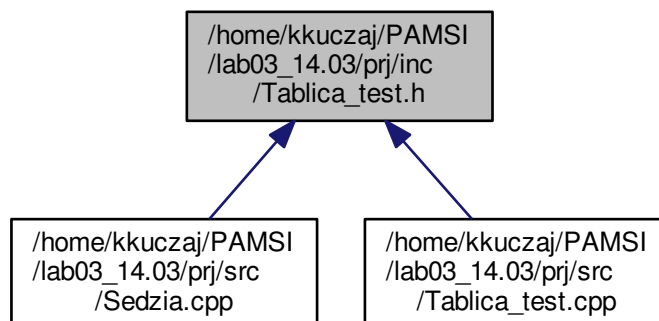
## 6.16 /home/kkuczaj/PAMSI/lab03\_14.03/prj/inc/Tablica\_test.h File Reference

```
#include "Tablica.h"  
#include "IRunnable.h"  
#include <iostream>
```

Include dependency graph for Tablica\_test.h:



This graph shows which files directly or indirectly include this file:



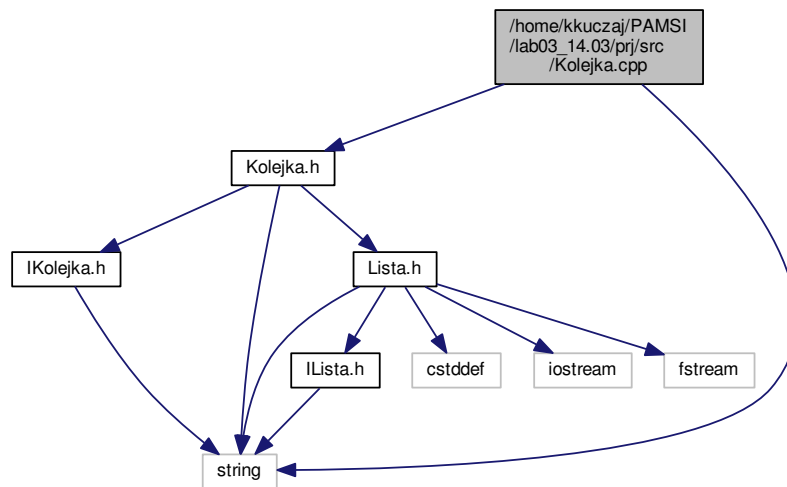
## Classes

- class [Tablica\\_test](#)

## 6.17 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Kolejka.cpp File Reference

```
#include "Kolejka.h"  
#include <string>
```

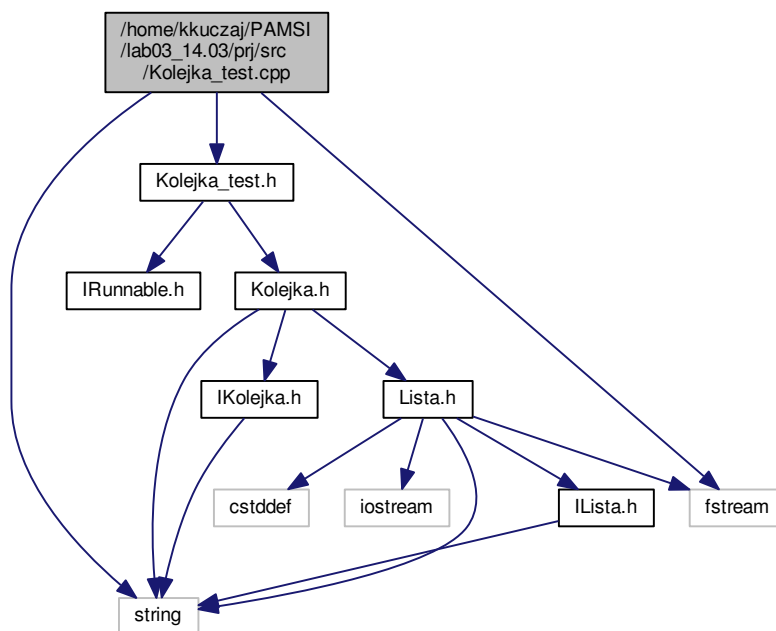
Include dependency graph for Kolejka.cpp:



## 6.18 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Kolejka\_test.cpp File Reference

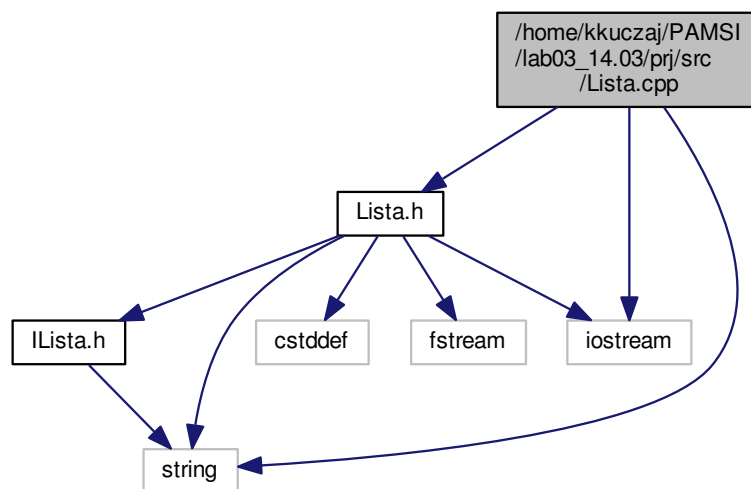
```
#include "Kolejka_test.h"
#include <string>
#include <fstream>
```

Include dependency graph for Kolejka\_test.cpp:



## 6.19 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Lista.cpp File Reference

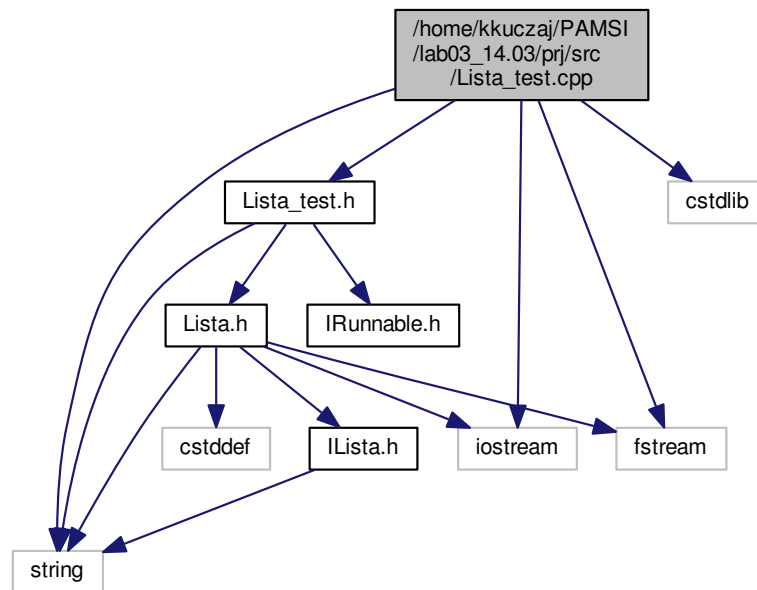
```
#include "Lista.h"  
#include <string>  
#include <iostream>  
Include dependency graph for Lista.cpp:
```



## 6.20 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Lista\_test.cpp File Reference

```
#include "Lista_test.h"  
#include <fstream>  
#include <cstdlib>  
#include <iostream>  
#include <string>
```

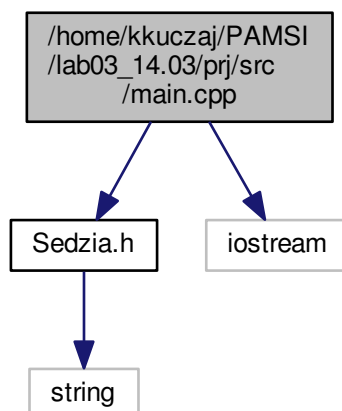
Include dependency graph for Lista\_test.cpp:



## 6.21 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/main.cpp File Reference

```
#include "Sedzia.h"
#include <iostream>
```

Include dependency graph for main.cpp:



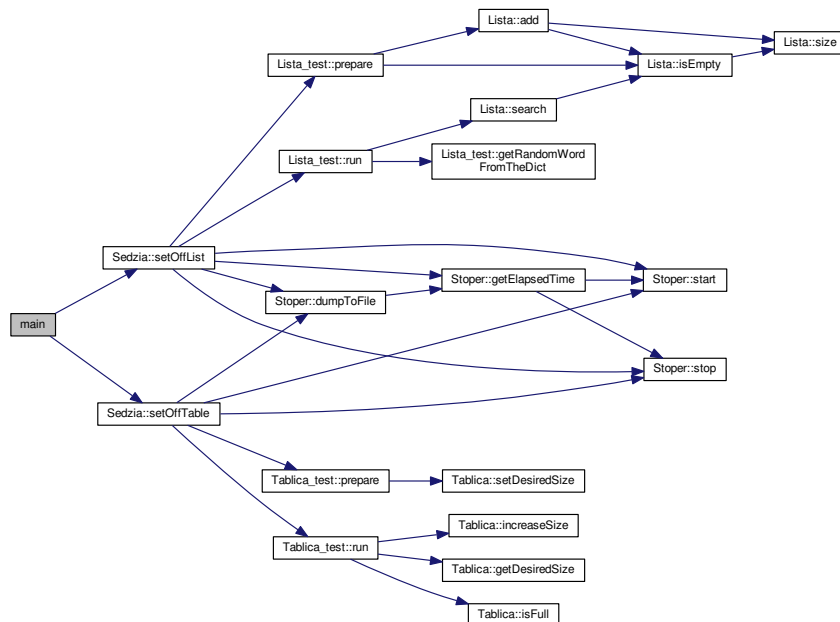
## Functions

- int [main](#) (int argc, char \*\*argv)

### 6.21.1 Function Documentation

#### 6.21.1.1 int main ( int argc, char \*\* argv )

Here is the call graph for this function:



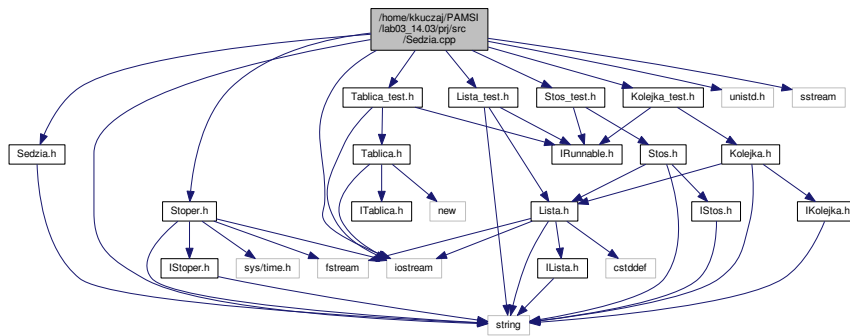
## 6.22 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Sedzia.cpp File Reference

```

#include "Sedzia.h"
#include "Stoper.h"
#include "Tablica_test.h"
#include "Lista_test.h"
#include "Stos_test.h"
#include "Kolejka_test.h"
#include <unistd.h>
#include <sstream>
#include <string>
#include <iostream>

```

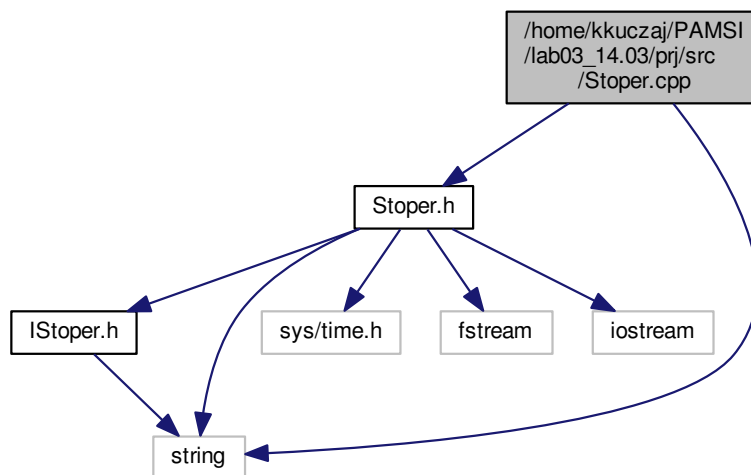
Include dependency graph for Sedzia.cpp:



## 6.23 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Stoper.cpp File Reference

```
#include "Stoper.h"
#include <string>
```

Include dependency graph for Stoper.cpp:

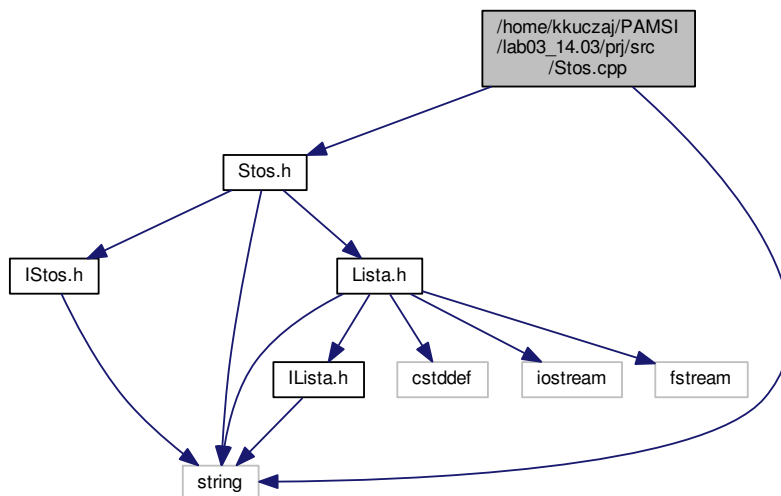


## 6.24 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Stos.cpp File Reference

```
#include "Stos.h"
#include <string>
```



Include dependency graph for Stos.cpp:



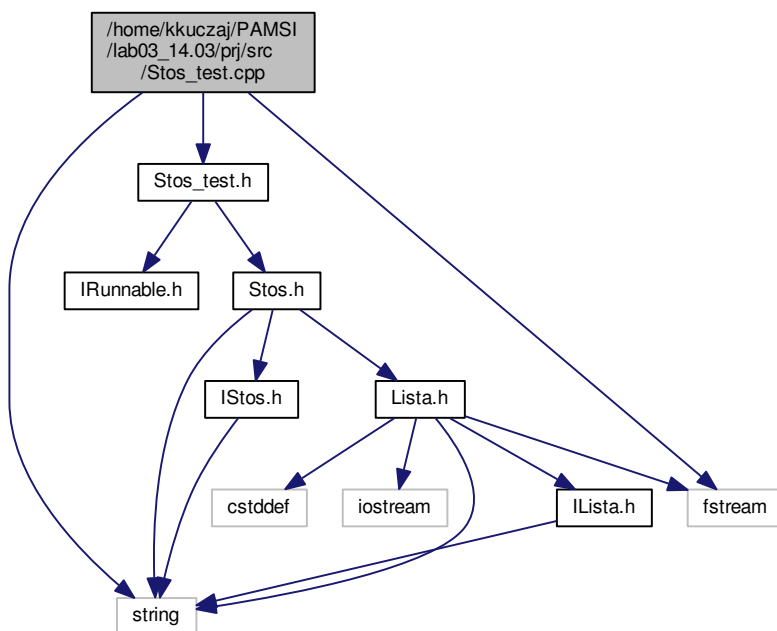
## 6.25 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Stos\_test.cpp File Reference

```

#include "Stos_test.h"
#include <string>
#include <fstream>

```

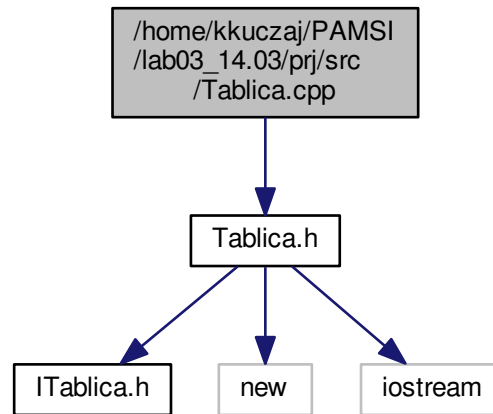
Include dependency graph for Stos\_test.cpp:



## 6.26 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Tablica.cpp File Reference

```
#include "Tablica.h"
```

Include dependency graph for Tablica.cpp:



## 6.27 /home/kkuczaj/PAMSI/lab03\_14.03/prj/src/Tablica\_test.cpp File Reference

```
#include "Tablica_test.h"
```

Include dependency graph for Tablica\_test.cpp:

