

Pomiar czasu algorytmów DFS i BFS grafu nieskierowanego.

Generated by Doxygen 1.8.6

Sun May 15 2016 18:51:10

Contents

Chapter 1

Opis programu

Author

Kamil Kuczaj 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:

IGraph	??
Graph	??
IKolejka< Type >	??
Kolejka< Type >	??
ILista< Type >	??
Lista< Type >	??
IRunnable	??
GraphBFS	??
GraphDFS	??
IStoper	??
Stoper	??
IStos< Type >	??
Stos< Type >	??
ListNode	??
Lista< Type >::Node	??
Sedzia	??

Chapter 3

Class Index

3.1 Class List

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

Graph	Graf oparty o liste sasiedztwa	??
GraphBFS	??
GraphDFS	??
IGraph	??
IKolejka< Type >	Interfejs dla kolejki	??
ILista< Type >	Interfejs dla pojemnika Lista	??
IRunnable	Interfejs dla biegacza	??
IStoper	Interfejs dla stopera	??
IStos< Type >	Interfejs dla każdego pojemnika	??
Kolejka< Type >	Implementacja interfejsu IKolejka w postaci klasy Kolejka	??
Lista< Type >	Klasa Lista , która symuluje zachowanie klasy list z biblioteki STL	??
ListNode	??
Lista< Type >::Node	Implementacja wezlow dla listy	??
Sedzia	Implementacja klasy Sedzia	??
Stoper	Implementacja klasy Stoper	??
Stos< Type >	Implementacja klasy Stos , złożonej z intow	??

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Graph.h	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/GraphBFS.h	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/GraphDFS.h	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IGraph.h	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IKolejka.h	
Plik zawiera interfejs dla pojemnika Kolejka	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/ILista.h	
Plik zawiera interfejs dla pojemnika Lista oraz dla klasy Wezel	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IRunnable.h	
Naglowek zawierajacy interfejs dla biegacza	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IStoper.h	
Naglowek zawierajacy interfejs dla stopera	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IStos.h	
Plik zawiera interfejs dla pojemnika Stos	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Kolejka.h	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Lista.h	
Implementacja jednokierunkowej listy	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Sedzia.h	
Naglowek opisujacy implementacje Sedziego	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Stoper.h	
Implementacja interfejsu IStoper w klasie Stoper	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Stos.h	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/src/Graph.cpp	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/src/GraphBFS.cpp	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/src/GraphDFS.cpp	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/src/Kolejka.cpp	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/src/Lista.cpp	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/src/main.cpp	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/src/Sedzia.cpp	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/src/Stoper.cpp	??
/home/kkuczaj/PAMSI/lab08_25.04/prj/src/Stos.cpp	??

Chapter 5

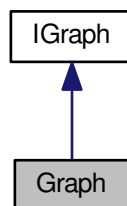
Class Documentation

5.1 Graph Class Reference

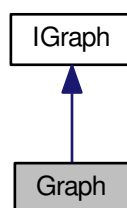
Graf oparty o liste sasiedztwa.

```
#include <Graph.h>
```

Inheritance diagram for Graph:



Collaboration diagram for Graph:



Public Member Functions

- [Graph](#) (int how_many)
- virtual void [addVertex](#) (int x)
- virtual void [addEdge](#) (int x, int y, int weight=1)
- virtual void [removeVertex](#) (int x)
- virtual void [removeEdge](#) (int x, int y)
- virtual [Lista](#)< int > [getNeighbours](#) (int x)
- void [print](#) ()
For debug.
- int [size](#) ()
Returns the number of vertices.
- [Lista](#)< int > [front](#) ()
Zwraca sasiadujące wierzchołki pierwszego wierzchołka grafu.
- [Lista](#)< int > [back](#) ()
Zwraca sasiadujące wierzchołki ostatniego elementu grafu.
- [Lista](#)< int > [operator\[\]](#) (int n)
Zwraca sasiadujące wierzchołki n-tego wierzchołka grafu.
- bool [isEdge](#) (int u, int v)
Sprawdza czy wierzchołki połączone są krawędzią.

Private Attributes

- std::vector< [Lista](#)< int > > [graph](#)
Pole, które będzie reprezentować graf..

5.1.1 Detailed Description

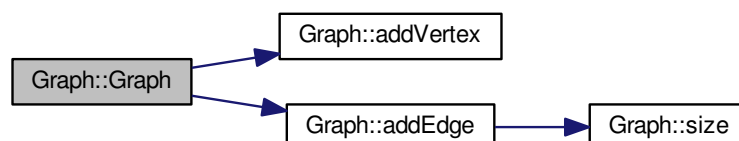
Graf oparty o listę sąsiedztwa.

[Lista](#) sąsiedztwa została wybrana jako sposób implementacji grafu, gdyż algorytmy przeszukania grafu BFS oraz DFS są na niej szybsze. Dodatkowo, dodaje wierzchołki już posortowane. Dzięki temu, pojemnik jest bardziej czytelny.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 [Graph::Graph](#) (int *how_many*) [inline]

Here is the call graph for this function:



5.1.3 Member Function Documentation

5.1.3.1 `virtual void Graph::addEdge (int x, int y, int weight = 1)` `[inline]`, `[virtual]`

Implements [IGraph](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3.2 `virtual void Graph::addVertex (int x)` `[inline]`, `[virtual]`

Implements [IGraph](#).

Here is the caller graph for this function:



5.1.3.3 `Lista<int> Graph::back ()` `[inline]`

Zwraca sąsiadujące wierzchołki ostatniego elementu grafu.

Here is the call graph for this function:



5.1.3.4 `Lista<int> Graph::front ()` `[inline]`

Zwraca sąsiadujące wierzchołki pierwszego wierzchołka grafu.

5.1.3.5 `virtual Lista<int> Graph::getNeighbours (int x)` `[inline]`, `[virtual]`

Implements [IGraph](#).

5.1.3.6 `bool Graph::isEdge (int u, int v)` `[inline]`

Sprawdza czy wierzchołki połączone są krawędzią.

5.1.3.7 `Lista<int> Graph::operator[] (int n)` `[inline]`

Zwraca sąsiadujące wierzchołki n-tego wierzchołka grafu.

5.1.3.8 `void Graph::print ()` `[inline]`

For debug.

5.1.3.9 `virtual void Graph::removeEdge (int x, int y)` `[inline]`, `[virtual]`

Implements [IGraph](#).

5.1.3.10 `virtual void Graph::removeVertex (int x)` `[inline]`, `[virtual]`

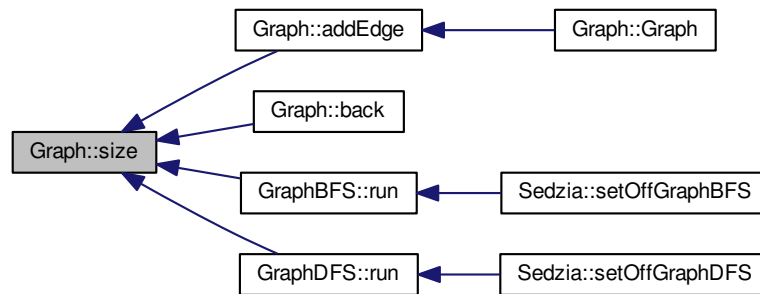
Implements [IGraph](#).

5.1.3.11 `int Graph::size ()` `[inline]`

Returns the number of vertices.

Return the size of vector.

Here is the caller graph for this function:



5.1.4 Member Data Documentation

5.1.4.1 `std::vector< Lista<int> > Graph::graph` [private]

Pole, ktore bedzie reprezentowac graf..

Uzylem elementow biblioteki STL, gdyz sa lepsze od moich. Tzn. nie musze skupiac sie na poprawianiu starych struktur, tylko moze skupic sie na implementacji grafu.

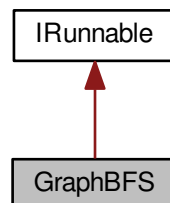
The documentation for this class was generated from the following file:

- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Graph.h

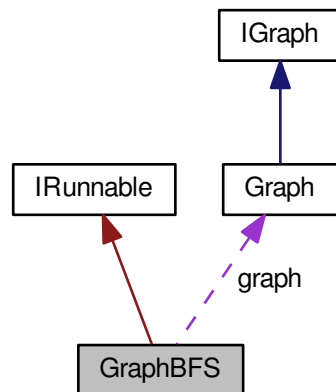
5.2 GraphBFS Class Reference

```
#include <GraphBFS.h>
```

Inheritance diagram for GraphBFS:



Collaboration diagram for GraphBFS:



Public Member Functions

- [GraphBFS](#) (int how_many)
- virtual void [run](#) ()
Odpalenie badanej czynnosci.
- virtual void [prepare](#) (int how_many)
Przygotowuje pojemnik przed wykonaniem czynnosci.

Private Attributes

- [Graph](#) graph
Testowany graf.

Additional Inherited Members

5.2.1 Constructor & Destructor Documentation

5.2.1.1 `GraphBFS::GraphBFS (int how_many) [inline]`

5.2.2 Member Function Documentation

5.2.2.1 `virtual void GraphBFS::prepare (int size) [inline],[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

<i>in</i>	<i>size</i>	llosc elementow.
-----------	-------------	------------------

Implements [IRunnable](#).

Here is the caller graph for this function:



5.2.2.2 virtual void GraphBFS::run () [inline],[virtual]

Odpalenie badanej czynnosci.

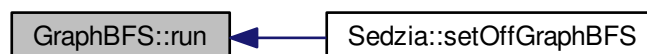
Funkcja, ktorej ciałem maja byc instrukcje, ktorych czas chcemy zmierzyc.

Implements [IRunnable](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3 Member Data Documentation

5.2.3.1 Graph GraphBFS::graph [private]

Testowany graf.

W celu dalszych informacji na temat tego pola prosze odwolac sie do klasy [Graph](#).

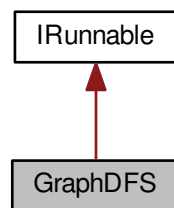
The documentation for this class was generated from the following file:

- [/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/GraphBFS.h](#)

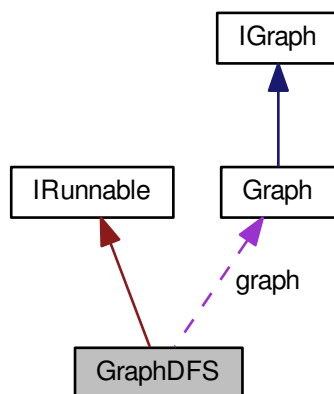
5.3 GraphDFS Class Reference

```
#include <GraphDFS.h>
```

Inheritance diagram for GraphDFS:



Collaboration diagram for GraphDFS:



Public Member Functions

- [GraphDFS](#) (int how_many)
- virtual void [run](#) ()
Odpalenie badanej czynnosci.
- virtual void [prepare](#) (int how_many)
Przygotowuje pojemnik przed wykonaniem czynnosci.

Private Attributes

- [Graph graph](#)

Testowany graf.

Additional Inherited Members

5.3.1 Constructor & Destructor Documentation

5.3.1.1 `GraphDFS::GraphDFS (int how_many)` `[inline]`

5.3.2 Member Function Documentation

5.3.2.1 `virtual void GraphDFS::prepare (int size)` `[inline],[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.
-----------------	-------------------	------------------

Implements [IRunnable](#).

Here is the caller graph for this function:



5.3.2.2 `virtual void GraphDFS::run ()` `[inline],[virtual]`

Odpalenie badanej czynnosci.

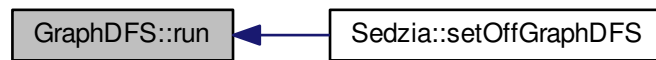
Funkcja, której ciałem mają być instrukcje, których czas chcemy zmierzyć.

Implements [IRunnable](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.3 Member Data Documentation

5.3.3.1 Graph GraphDFS::graph [private]

Testowany graf.

W celu dalszych informacji na temat tego pola prosze odwolac sie do klasy [Graph](#).

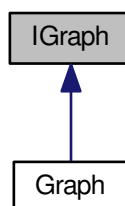
The documentation for this class was generated from the following file:

- `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/GraphDFS.h`

5.4 IGraph Class Reference

```
#include <IGraph.h>
```

Inheritance diagram for IGraph:



Public Member Functions

- virtual void [addVertex](#) (int x)=0
- virtual void [addEdge](#) (int x, int y, int weight=1)=0
- virtual void [removeVertex](#) (int x)=0
- virtual void [removeEdge](#) (int x, int y)=0
- virtual [Lista](#)< int > [getNeighbours](#) (int x)=0

5.4.1 Member Function Documentation

5.4.1.1 `virtual void IGraph::addEdge (int x, int y, int weight = 1) [pure virtual]`

Implemented in [Graph](#).

5.4.1.2 `virtual void IGraph::addVertex (int x) [pure virtual]`

Implemented in [Graph](#).

5.4.1.3 `virtual Lista<int> IGraph::getNeighbours (int x) [pure virtual]`

Implemented in [Graph](#).

5.4.1.4 `virtual void IGraph::removeEdge (int x, int y) [pure virtual]`

Implemented in [Graph](#).

5.4.1.5 `virtual void IGraph::removeVertex (int x) [pure virtual]`

Implemented in [Graph](#).

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

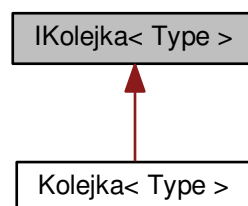
- [/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IGraph.h](#)

5.5 IKolejka< Type > Class Template Reference

Interfejs dla kolejki.

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

Inheritance diagram for IKolejka< Type >:



Protected Member Functions

- `virtual void push (Type element)=0`
Dodaje element na poczatek.
- `virtual Type 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

`template<class Type>class IKolejka< Type >`

Interfejs dla kolejki.

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

5.5.2 Member Function Documentation

5.5.2.1 `template<class Type > virtual bool IKolejka< Type >::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< Type >`.

5.5.2.2 `template<class Type > virtual Type IKolejka< Type >::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< Type >`.

5.5.2.3 `template<class Type > virtual void IKolejka< Type >::push (Type 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< Type >`.

5.5.2.4 `template<class Type > virtual int IKolejka< Type >::size ()` `[protected]`, `[pure virtual]`

Zwraca aktualny rozmiar pojemnika.

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

Returns

Ilosc elementow w pojemniku.

Implemented in [Kolejka< Type >](#).

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

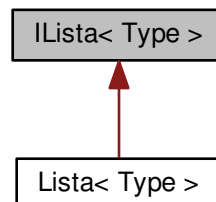
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/[IKolejka.h](#)

5.6 ILista< Type > Class Template Reference

Interfejs dla pojemnika [Lista](#).

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

Inheritance diagram for ILista< Type >:



Protected Member Functions

- virtual void [add](#) (Type item, int weight, int index)=0
Wstawia element w dowolnym miejscu listy.
- virtual Type [remove](#) (int index)=0
Usuwa element z dowolnego miejsca listy.
- virtual bool [isEmpty](#) ()=0
Sprawdza czy lista jest pusta.
- virtual Type [get](#) (int index)=0
Zwraca element z dowolnego miejsca listy.
- virtual int [getWeight](#) (int n)=0
- virtual int [size](#) ()=0
Zwraca rozmiar listy.

5.6.1 Detailed Description

```
template<class Type>class ILista< Type >
```

Interfejs dla pojemnika [Lista](#).

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

5.6.2 Member Function Documentation

5.6.2.1 `template<class Type > virtual void ILista< Type >::add (Type item, int weight, int index)` [protected], [pure virtual]

Wstawia element w dowolnym miejscu listy.

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

Parameters

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

Implemented in [Lista< Type >](#).

5.6.2.2 `template<class Type > virtual Type ILista< Type >::get (int index)` [protected], [pure virtual]

Zwraca element z dowolnego miejsca listy.

Zwraca element z miejsca wskazywanego przez zmienna `index`.

Returns

Zwraca element typu `Type`.

Implemented in [Lista< Type >](#).

5.6.2.3 `template<class Type > virtual int ILista< Type >::getWeight (int n)` [protected], [pure virtual]

Implemented in [Lista< Type >](#).

5.6.2.4 `template<class Type > virtual bool ILista< Type >::isEmpty ()` [protected], [pure virtual]

Sprawdza czy lista jest pusta.

Sprawdza czy w liście są jakieś elementy.

Return values

<code>true</code>	Lista jest pusta.
<code>false</code>	Lista nie jest pusta.

Implemented in [Lista< Type >](#).

5.6.2.5 `template<class Type > virtual Type ILista< Type >::remove (int index)` [protected], [pure virtual]

Usuwa element z dowolnego miejsca listy.

Usuwa element z miejsca wskazywanego przez zmienna `index`.

Returns

Zwraca zawartość komórki o tej indeksie.

Implemented in [Lista< Type >](#).

5.6.2.6 `template<class Type > virtual int ILista< Type >::size () [protected],[pure virtual]`

Zwraca rozmiar listy.

Zwraca ilosc elementow w liscie.

Returns

Rozmiar listy.

Implemented in [Lista< Type >](#).

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

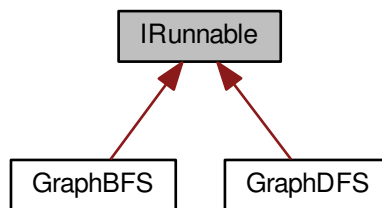
- `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/ILista.h`

5.7 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.

5.7.1 Detailed Description

Interfejs dla biegacza.

Klasa abstrakcyjna z metodami czysto wirtualnymi.

5.7.2 Member Function Documentation

5.7.2.1 `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>	liczba elementów.
-----------------	-------------------	-------------------

Implemented in [GraphDFS](#), and [GraphBFS](#).

5.7.2.2 `virtual void IRunnable::run ()` `[protected]`, `[pure virtual]`

Odpalenie badanej czynności.

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

Implemented in [GraphBFS](#), and [GraphDFS](#).

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

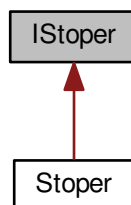
- `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IRunnable.h`

5.8 IStoper Class Reference

Interfejs dla stopera.

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

Inheritance diagram for IStoper:



Protected Member Functions

- `virtual void start ()=0`
Ma symulować moment startu stopera.
- `virtual void stop ()=0`
- `virtual double getElapsedTime ()=0`
Ma symulować rezultat pokazania wyniku pomiaru czasu na stoperze.
- `virtual void dumpToFile (std::string file_name)=0`
Ma symulować moment zapisu zmierzonego czasu na kartkę papieru.

5.8.1 Detailed Description

Interfejs dla stopera.

Klasa abstrakcyjna z metodami czysto wirtualnymi.

5.8.2 Member Function Documentation

5.8.2.1 `virtual void IStoper::dumpToFile (std::string file_name)` [protected],[pure virtual]

Ma symulowac moment zapisu zmierzonego czasu na kartke papieru.

Metoda czysto wirtualna.

Parameters

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

Implemented in [Stoper](#).

5.8.2.2 `virtual double IStoper::getElapsedTime ()` [protected],[pure virtual]

Ma symulowac rezultat pokazania wyniku pomiaru czasu na stoperze.

Metoda czysto wirtualna.

Implemented in [Stoper](#).

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

Ma symulowac moment startu stopera.

Metoda czysto wirtualna.

Implemented in [Stoper](#).

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

Implemented in [Stoper](#).

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

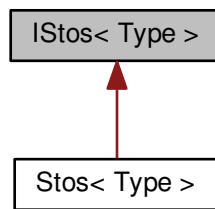
- [/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IStoper.h](#)

5.9 IStos< Type > Class Template Reference

Interfejs dla każdego pojemnika.

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

Inheritance diagram for `IStos< Type >`:



Protected Member Functions

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

5.9.1 Detailed Description

```
template<class Type>class IStos< Type >
```

Interfejs dla każdego pojemnika.

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

5.9.2 Member Function Documentation

5.9.2.1 `template<class Type> virtual bool IStos< Type >::empty () [protected], [pure virtual]`

Sprawdza czy pojemnika jest pusty.

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

Return values

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

Implemented in `Stos< Type >`.

5.9.2.2 `template<class Type> virtual Type IStos< Type >::pop () [protected], [pure virtual]`

Usuwa element z pojemnika.

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

Returns

Usuniety element.

Implemented in [Stos< Type >](#).

5.9.2.3 `template<class Type > virtual void IStos< Type >::push (Type element)` `[protected], [pure virtual]`

Dodaje element na poczatek.

Dodaje element na poczatek pojemnika.

Parameters

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

Implemented in [Stos< Type >](#).

5.9.2.4 `template<class Type > virtual int IStos< Type >::size ()` `[protected], [pure virtual]`

Zwraca aktualny rozmiar pojemnika.

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

Returns

Ilosc elementow w pojemniku.

Implemented in [Stos< Type >](#).

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

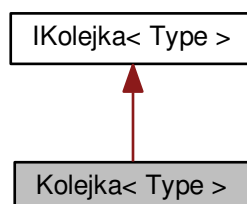
- [/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IStos.h](#)

5.10 Kolejka< Type > Class Template Reference

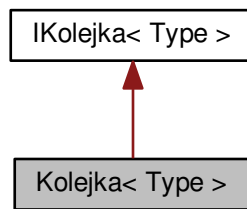
Implementacja interfejsu [IKolejka](#) w postaci klasy [Kolejka](#).

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

Inheritance diagram for Kolejka< Type >:



Collaboration diagram for Kolejka< Type >:



Public Member Functions

- virtual void `push` (Type element)
Dodaje element na poczatek.
- virtual Type `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< Type > queue`
Zawartosc kolejki.

Additional Inherited Members

5.10.1 Detailed Description

```
template<class Type>class Kolejka< Type >
```

Implementacja interfejsu `IKolejka` w postaci klasy `Kolejka`.

Korzysta z klasy `Lista`, jako jej prywatne pole oraz calej jej funkcjonalnosci. W celu zrozumienia pelnej funkcjonalnosci klasy `Kolejka`, prosze odwolac sie do dokumentacji klasy `Lista`.

5.10.2 Member Function Documentation

5.10.2.1 `template<class Type > virtual bool Kolejka< Type >::empty () [inline], [virtual]`

Sprawdza czy pojemnika jest pusty.

Sprawdza czy znajduja sie jakies elementy w pojemniku. Metoda czysto wirtualna.

Return values

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

Implements [IKolejka< Type >](#).

5.10.2.2 `template<class Type > virtual Type Kolejka< Type >::pop () [inline],[virtual]`

Usuwa element z pojemnika.

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

Returns

Usuniety element.

Implements [IKolejka< Type >](#).

5.10.2.3 `template<class Type > void Kolejka< Type >::print () [inline]`

Wyswietla zawartosc kolejki.

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

5.10.2.4 `template<class Type > virtual void Kolejka< Type >::push (Type element) [inline],[virtual]`

Dodaje element na poczatek.

Dodaje element na poczatek pojemnika.

Parameters

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

Implements [IKolejka< Type >](#).

5.10.2.5 `template<class Type > virtual int Kolejka< Type >::size () [inline],[virtual]`

Zwraca aktualny rozmiar pojemnika.

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

Returns

Ilosc elementow w pojemniku.

Implements [IKolejka< Type >](#).

5.10.3 Member Data Documentation

5.10.3.1 `template<class Type > Lista<Type> Kolejka< Type >::queue [private]`

Zawartosc kolejki.

Symuluje kolejke, poniewac jest to bardzo prosta implementacja.

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

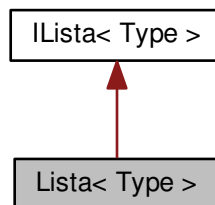
- `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Kolejka.h`

5.11 Lista< Type > Class Template Reference

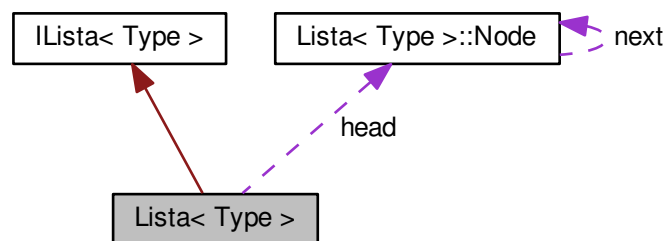
Klasa [Lista](#), która symuluje zachowanie klasy list z biblioteki STL.

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

Inheritance diagram for Lista< Type >:



Collaboration diagram for Lista< Type >:



Classes

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

Public Member Functions

- [Lista](#) ()
Konstruktor.
- [Lista](#) (Type x)
Parametryczny konstruktor.
- virtual void [add](#) (Type item, int w, int n)
Wstawia element w dowolnym miejscu listy.
- void [clear](#) ()
- virtual Type [remove](#) (int n)

Usuwa element z dowolnego miejsca listy.

- virtual bool `isEmpty` ()

Sprawdza czy lista jest pusta.

- virtual Type `get` (int n)

Zwraca element z dowolnego miejsca listy.

- virtual int `getWeight` (int n)
- virtual int `size` ()

Zwraca rozmiar listy.

- void `print` ()

Wypisuje zawartosc listy.

- int `search` (Type searched_word)

Wyszukuje podane slowo i zwraca jego indeks.

- Type `min` (Lista other_than)

Choose min other than specified as an argument.

Private Attributes

- Node * `head`

Pierwszy element listy.

- int `size_of_list`

Przechowuje rozmiar listy.

Additional Inherited Members

5.11.1 Detailed Description

`template<class Type>class Lista< Type >`

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

5.11.2 Constructor & Destructor Documentation

5.11.2.1 `template<class Type > Lista< Type >::Lista () [inline]`

Konstruktor.

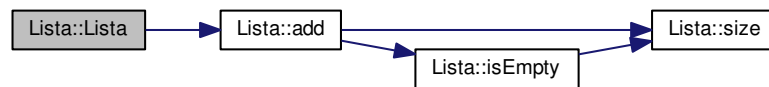
Tworzy poczatek listy. Alokuje dla niego pamiec.

5.11.2.2 `template<class Type > Lista< Type >::Lista (Type x) [inline]`

Parametryczny konstruktor.

Konstruktor dodany na potrzeby implementacji grafu.

Here is the call graph for this function:



5.11.3 Member Function Documentation

5.11.3.1 `template<class Type > virtual void Lista< Type >::add (Type item, int w, int n)` `[inline]`, `[virtual]`

Wstawia element w dowolnym miejscu listy.

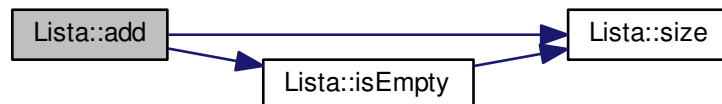
Wstawia element typu `Type` w miejsce wskazywane przez zmienna `index`.

Parameters

<code>in</code>	<code>item</code>	Element wstawiany. Słowo typu <code>string</code> .
<code>in</code>	<code>w</code>	Waga krawedzi.
<code>in</code>	<code>n</code>	Miejsce, w które ma być wstawiony element <code>item</code> .

Implements [ILista< Type >](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.2 `template<class Type > void Lista< Type >::clear ()` `[inline]`

5.11.3.3 `template<class Type > virtual Type Lista< Type >::get (int n) [inline],[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< Type >](#).

Here is the call graph for this function:



5.11.3.4 `template<class Type > virtual int Lista< Type >::getWeight (int n) [inline],[virtual]`

Implements [ILista< Type >](#).

Here is the call graph for this function:



5.11.3.5 `template<class Type > virtual bool Lista< Type >::isEmpty () [inline],[virtual]`

Sprawdza czy lista jest pusta.

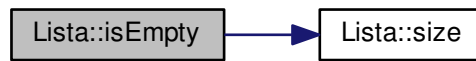
Sprawdza czy w liście są jakieś elementy.

Return values

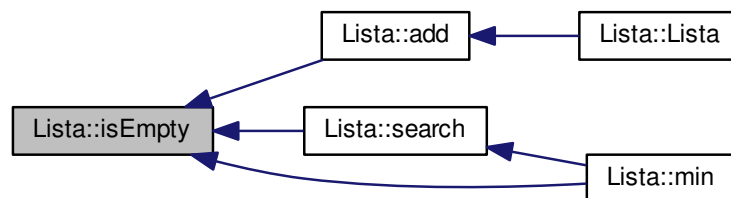
<i>true</i>	Lista jest pusta.
<i>false</i>	Lista nie jest pusta.

Implements [ILista< Type >](#).

Here is the call graph for this function:



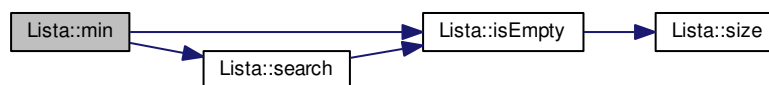
Here is the caller graph for this function:



5.11.3.6 `template<class Type > Type Lista< Type >::min (Lista< Type > other_than) [inline]`

Choose min other than specified as an argument.

Here is the call graph for this function:



5.11.3.7 `template<class Type > void Lista< Type >::print () [inline]`

Wypisuje zawartosc listy.

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

5.11.3.8 `template<class Type > virtual Type Lista< Type >::remove (int n) [inline], [virtual]`

Usuwa element z dowolnego miejsca listy.

Usuwa element z miejsca wskazywanego przez zmienna index.

Returns

Zwraca slowo, ktore znajdowalo sie na tym indeksie.

Implements [ILista< Type >](#).

Here is the call graph for this function:



5.11.3.9 `template<class Type > int Lista< Type >::search (Type searched_word)` `[inline]`

Wyszukuje podane slowo i zwraca jego indeks.

Wyszukuje w liscie podane slowo ypu `std::string`. Zwraca liczbe, ktora reprezentuje indeks z podanym slowem.\

Parameters

<code>in</code>	<code>searched_word</code>	Szukane slowo.
-----------------	----------------------------	----------------

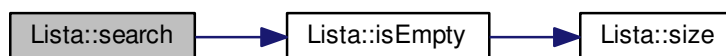
Return values

-1	Lista pusta.
-2	Nie ma takiego elementu w liscie.

Returns

Indeks, na ktorym znajduje sie szukane slowo.

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.10 `template<class Type> virtual int Lista< Type>::size () [inline],[virtual]`

Zwraca rozmiar listy.

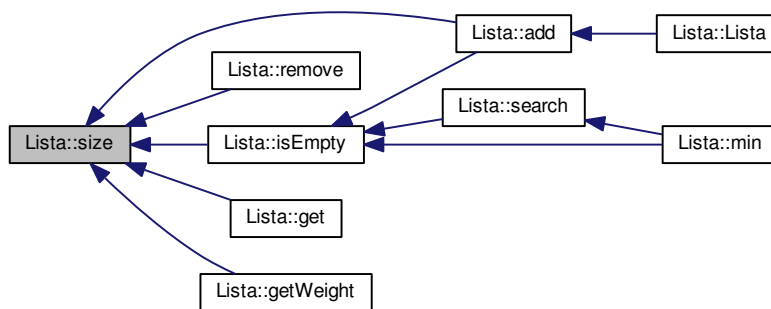
Zwraca ilosc elementow w liscie.

Returns

Rozmiar listy.

Implements [ILista< Type>](#).

Here is the caller graph for this function:



5.11.4 Member Data Documentation

5.11.4.1 `template<class Type> Node* Lista< Type>::head [private]`

Pierwszy element listy.

Wskazuje na pierwszy element listy.

5.11.4.2 `template<class Type> int Lista< Type>::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 file:

- `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Lista.h`

5.12 ListNode Struct Reference

```
#include <IGraph.h>
```

Public Member Functions

- [ListNode](#) (int y, int weight)

Public Attributes

- int [key](#)
- int [w](#)

5.12.1 Constructor & Destructor Documentation

5.12.1.1 `ListNode::ListNode (int y, int weight) [inline]`

5.12.2 Member Data Documentation

5.12.2.1 int `ListNode::key`

5.12.2.2 int `ListNode::w`

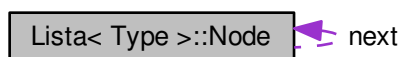
The documentation for this struct was generated from the following file:

- `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IGraph.h`

5.13 Lista< Type >::Node Struct Reference

Implementacja wezlow dla listy.

Collaboration diagram for Lista< Type >::Node:



Public Attributes

- Type [element](#)
Element w wezle.
- int [weight](#)
Waga krawedzi.
- [Node](#) * [next](#)
Wskaźnik na następny wezel.

5.13.1 Detailed Description

```
template<class Type>struct Lista< Type >::Node
```

Implementacja wezlow dla listy.

Potrzebne do implementacji interfejsu listy. Zawiera pole typu string.

5.13.2 Member Data Documentation

5.13.2.1 `template<class Type > Type Lista< Type >::Node::element`

Element w wezle.

Co jest w wezle. Ma przechowywac pojedyncze slowo.

5.13.2.2 `template<class Type > Node* Lista< Type >::Node::next`

Wskaznik na nastepny wezel.

Wskazuje na nastepny wezel.

5.13.2.3 `template<class Type > int Lista< Type >::Node::weight`

Waga krawedzi.

Pole stworzone w celu implementacji grafu. Reprezentuje wage krawedzi pomiedzy wezlami.

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

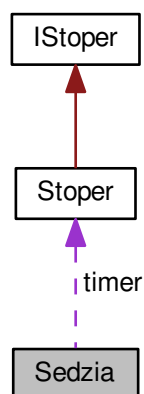
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Lista.h

5.14 Sedzia Class Reference

Implementacja klasy [Sedzia](#).

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

Collaboration diagram for Sedzia:



Public Member Functions

- void [setOffGraphDFS](#) (int &how_many, int &trials_count)
Funkcja, gdzie odbywa sie zapis phonebook'a do tablicy haszowej.
- void [setOffGraphBFS](#) (int &how_many, int &trials_count)

Private Attributes

- [Stoper timer](#)

5.14.1 Detailed Description

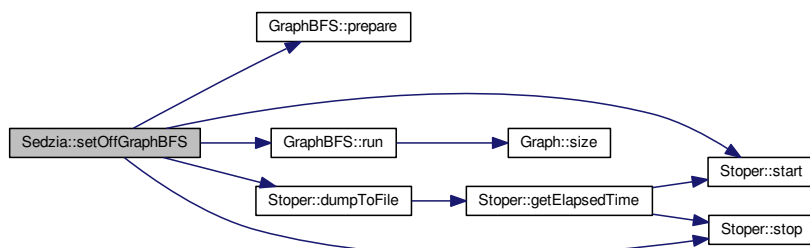
Implementacja klasy [Sedzia](#).

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

5.14.2 Member Function Documentation

5.14.2.1 void Sedzia::setOffGraphBFS (int & *how_many*, int & *trials_count*)

Here is the call graph for this function:



5.14.2.2 void Sedzia::setOffGraphDFS (int & *how_many*, int & *trials_count*)

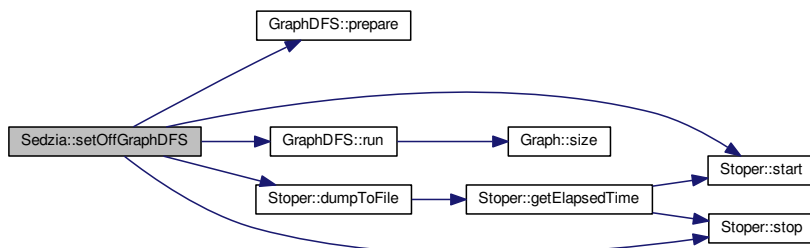
Funkcja, gdzie odbywa sie zapis phonebook'a do tablicy haszowej.

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

Parameters

in	<i>how_many</i>	Informacja iloma elementami ma zostac wypelniona tablica.
----	-----------------	---

Here is the call graph for this function:



5.14.3 Member Data Documentation

5.14.3.1 Stoper Sedzia::timer [private]

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

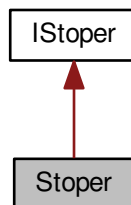
- [/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Sedzia.h](#)
- [/home/kkuczaj/PAMSI/lab08_25.04/prj/src/Sedzia.cpp](#)

5.15 Stoper Class Reference

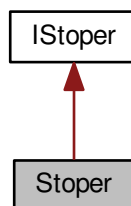
Implementacja klasy [Stoper](#).

```
#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

5.15.1 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>`.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 [Stoper::Stoper](#) ()

Konstruktor bezparametryczny.

Alokuje pamięć dla pól, ponieważ są wskaźnikami.

5.15.2.2 [Stoper::~~Stoper](#) ()

Destruktor.

Zwalnia pamięć po polach.

5.15.3 Member Function Documentation

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

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

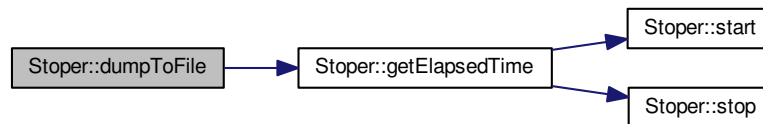
Zapisuje zmierzony czas do pliku o nazwie `"${file_name}.csv"`. Plik otwierany w trybie dopisywania (append) oraz wyjściowym (out). Plik .csv to tzw. Comma-Separated Values - łatwo je potem zaimportować do arkusza kalkulacyjnego oraz są zgodne z ogólnie przyjętym standardem.

Parameters

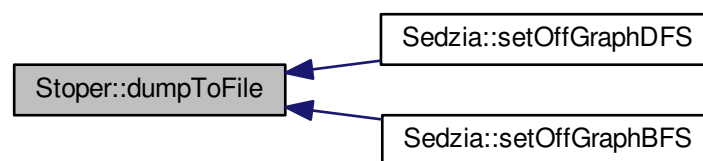
<i>file_name</i>	Nazwa pliku, do ktorego beda zapisane dane. Nazwa nie powinna zawierac rozszerzenia. Rozszerzenie jest dodawane w funkcji.
------------------	--

Implements [IStoper](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.15.3.2 `double Stoper::getElapsedTime () [virtual]`

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

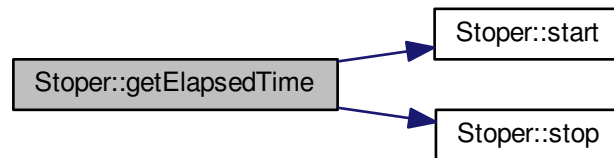
Oblicza czas pomiedzy czasem zapisanym w zmiennych `start_time` i `stop_time`.

Returns

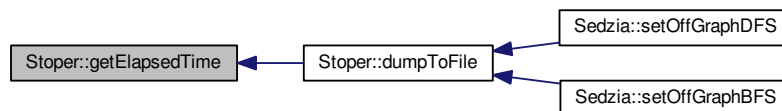
Zwraca zmierzony czas - roznica pomiedzy polem start_time a polem stop_time. Zwraca wynik w mikrosekundach.

Implements [IStoper](#).

Here is the call graph for this function:



Here is the caller graph for this function:



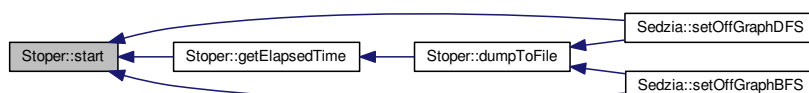
5.15.3.3 void Stoper::start () [virtual]

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

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

Implements [IStoper](#).

Here is the caller graph for this function:



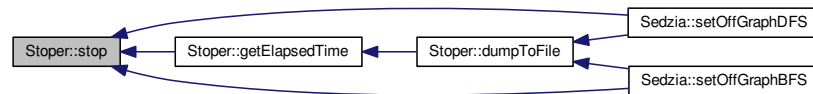
5.15.3.4 void Stoper::stop () [virtual]

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

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

Implements [IStoper](#).

Here is the caller graph for this function:



5.15.4 Member Data Documentation

5.15.4.1 `timeval* Stoper::start_time` [private]

Moment startu stopera.

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

5.15.4.2 `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:

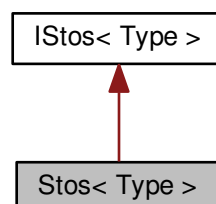
- `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Stoper.h`
- `/home/kkuczaj/PAMSI/lab08_25.04/prj/src/Stoper.cpp`

5.16 `Stos< Type >` Class Template Reference

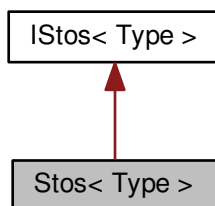
Implementacja klasy [Stos](#), złożonej z intów.

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

Inheritance diagram for `Stos< Type >`:



Collaboration diagram for Stos< Type >:



Public Member Functions

- `Stos ()`
Bezparametryczny konstruktor.
- `~Stos ()`
Destruktor.
- virtual void `push` (Type item)
Usuwa element z okreslonego miejsca.
- virtual Type `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< Type > stack`
Zawartosc stosu.

Additional Inherited Members

5.16.1 Detailed Description

```
template<class Type>class Stos< Type >
```

Implementacja klasy `Stos`, zlozonej z intow.

Implementacja pojemnika, gdzie dostepny jest jedynie element bedacy "na gorze". Jej skladowe elementy to stringi. Zdecydowalem sie nie stosowac szablonow ze wzgledu na niepotrzebna komplikacje. Zdecydowalem sie na uzycie listy jako elementu klasy, poniewaz byl to wymog prowadzacego. Nie ma ograniczen rozmiaru.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 `template<class Type > Stos< Type >::Stos () [inline]`

Bezparametryczny konstruktor.

Inicjalizuje wierzcholek *top jak wskaźnik na NULL.

5.16.2.2 `template<class Type > Stos< Type >::~~Stos () [inline]`

Destruktor.

Popuje wszystkie elementy.

Here is the call graph for this function:



5.16.3 Member Function Documentation

5.16.3.1 `template<class Type > virtual bool Stos< Type >::empty () [inline],[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< Type >](#).

5.16.3.2 `template<class Type > virtual Type Stos< Type >::pop () [inline],[virtual]`

Usuwa element z pojemnika.

Usuwa element z pojemnika i zwraca go użytkownikowi.

Returns

Usuniety element.

Implements [IStos< Type >](#).

Here is the caller graph for this function:



5.16.3.3 `template<class Type> void Stos< Type>::print () [inline]`

Wyswietla elementy stosu.

Wyswietla cala zawartosc stosu. Nie jest czescia interfesju.

5.16.3.4 `template<class Type> virtual void Stos< Type>::push (Type item) [inline],[virtual]`

Usuwa element z okreslonego miejsca.

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

Parameters

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

Implements [IStos< Type >](#).

5.16.3.5 `template<class Type> virtual int Stos< Type>::size () [inline],[virtual]`

Zwraca aktualny rozmiar pojemnika.

Zwraca wartosc, ktora reprezentuje obecna ilosc elementow w pojemniku.

Returns

Ilosc elementow w pojemniku.

Implements [IStos< Type >](#).

5.16.4 Member Data Documentation

5.16.4.1 `template<class Type> Lista<Type> Stos< Type>::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 file:

- `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Stos.h`

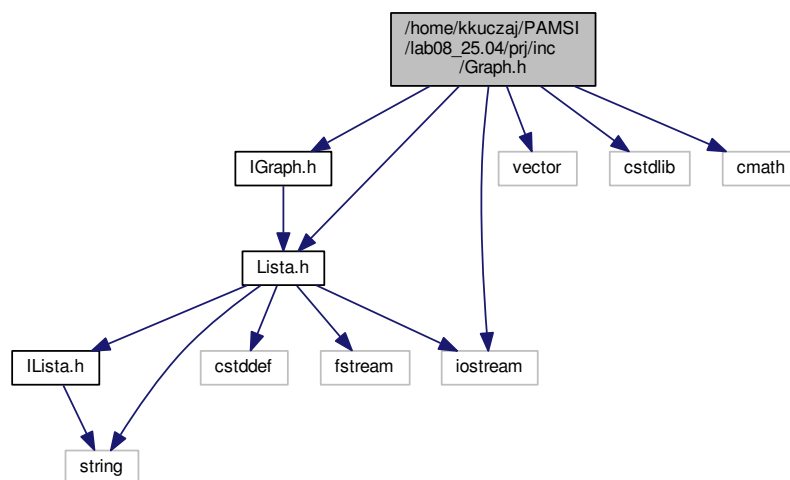
Chapter 6

File Documentation

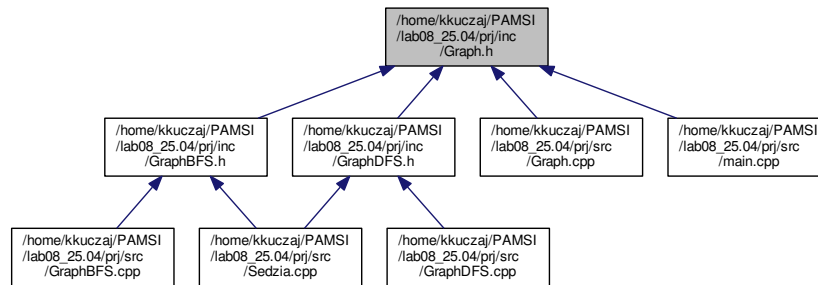
6.1 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Graph.h File Reference

```
#include "IGraph.h"  
#include "Lista.h"  
#include <vector>  
#include <iostream>  
#include <cstdlib>  
#include <cmath>
```

Include dependency graph for Graph.h:



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



Classes

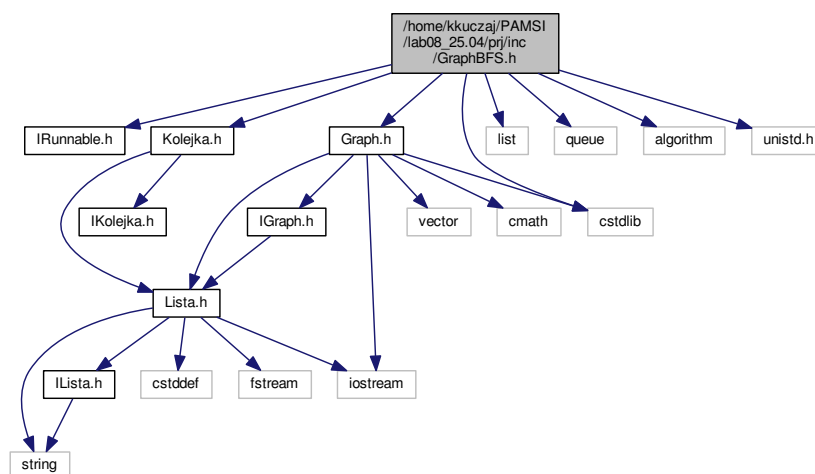
- class [Graph](#)

Graf oparty o liste sasiedztwa.

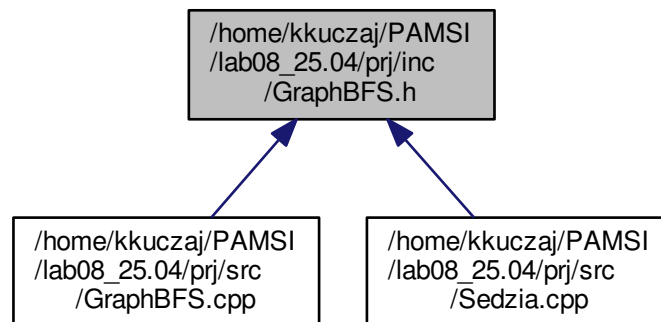
6.2 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/GraphBFS.h File Reference

```
#include "IRunnable.h"
#include "Graph.h"
#include "Kolejka.h"
#include <list>
#include <queue>
#include <algorithm>
#include <cstdlib>
#include <unistd.h>
```

Include dependency graph for GraphBFS.h:



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



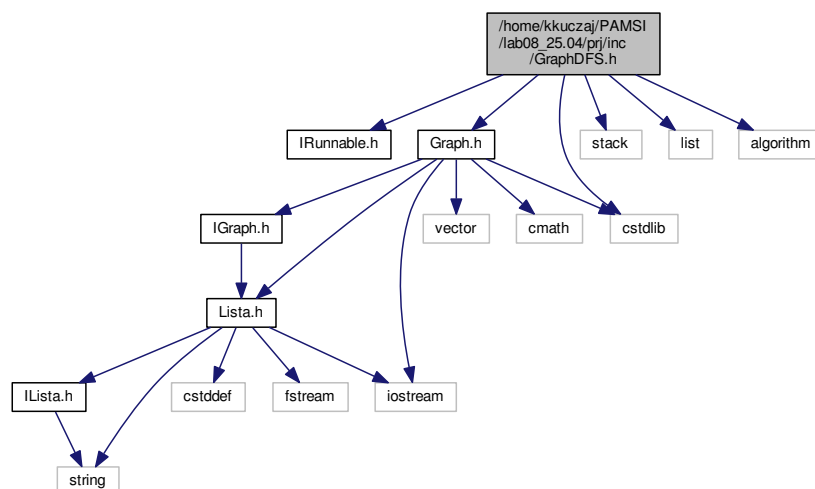
Classes

- class [GraphBFS](#)

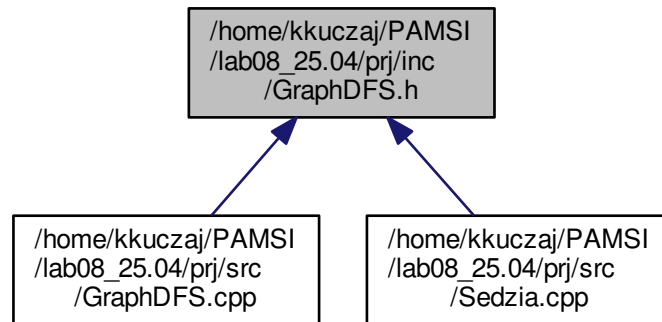
6.3 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/GraphDFS.h File Reference

```
#include "IRunnable.h"
#include "Graph.h"
#include <cstdlib>
#include <stack>
#include <list>
#include <algorithm>
```

Include dependency graph for GraphDFS.h:



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



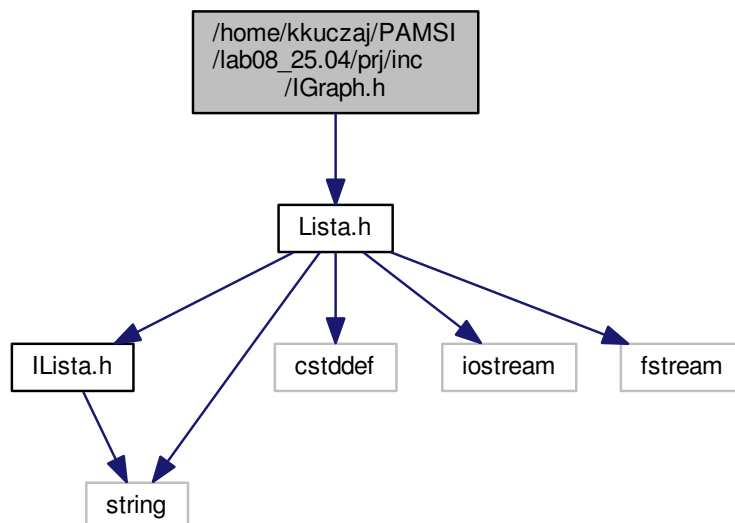
Classes

- class [GraphDFS](#)

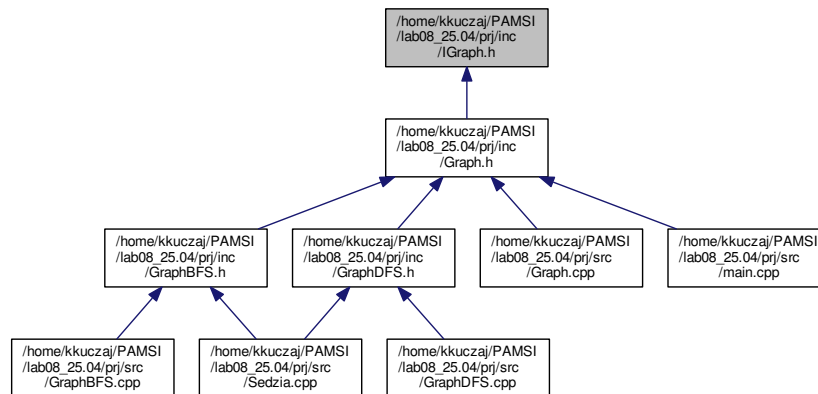
6.4 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IGraph.h File Reference

```
#include "Lista.h"
```

Include dependency graph for IGraph.h:



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



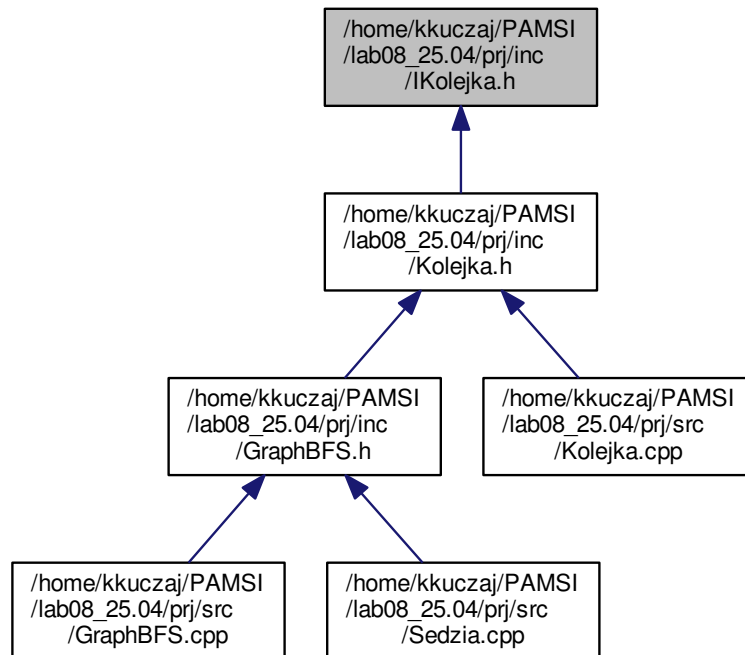
Classes

- struct [ListNode](#)
- class [IGraph](#)

6.5 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IKolejka.h File Reference

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

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



Classes

- class `IKolejka< Type >`

Interfejs dla kolejki.

6.5.1 Detailed Description

Plik zawiera interfejs dla pojemnika `Kolejka`. Nie zdecydowano się na użycie szablonów, gdyż zbyt komplikuje to budowę programu.

Author

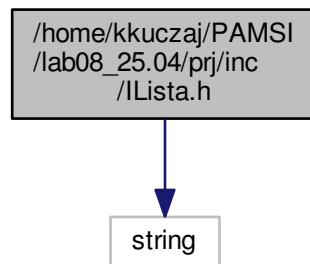
- Kamil Kuczaj.

6.6 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/ILista.h File Reference

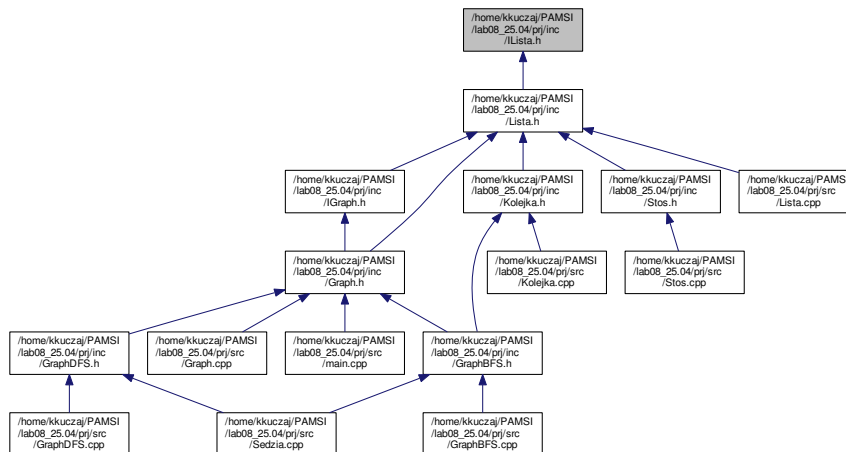
Plik zawiera interfejs dla pojemnika [Lista](#) oraz dla klasy [Wezel](#).

```
#include <string>
```

Include dependency graph for ILista.h:



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



Classes

- class [ILista< Type >](#)

Interfejs dla pojemnika [Lista](#).

6.6.1 Detailed Description

Plik zawiera interfejs dla pojemnika [Lista](#) oraz dla klasy `Wezel`. `Wezel` jest elementem listy. Użycie szablonów zbyt komplikuje implementację, więc odrzuciłem ich zastosowanie.

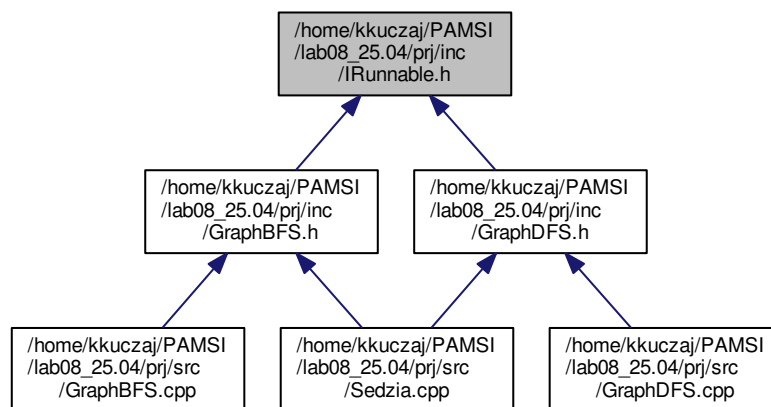
Author

Kamil Kuczaj.

6.7 `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IRunnable.h` File Reference

Nagłówek zawierający interfejs dla biegacza.

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



Classes

- class `IRunnable`

Interfejs dla biegacza.

6.7.1 Detailed Description

Nagłówek zawierający interfejs dla biegacza.

Author

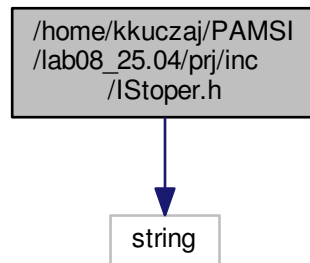
Kamil Kuczaj

6.8 `/home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IStoper.h` File Reference

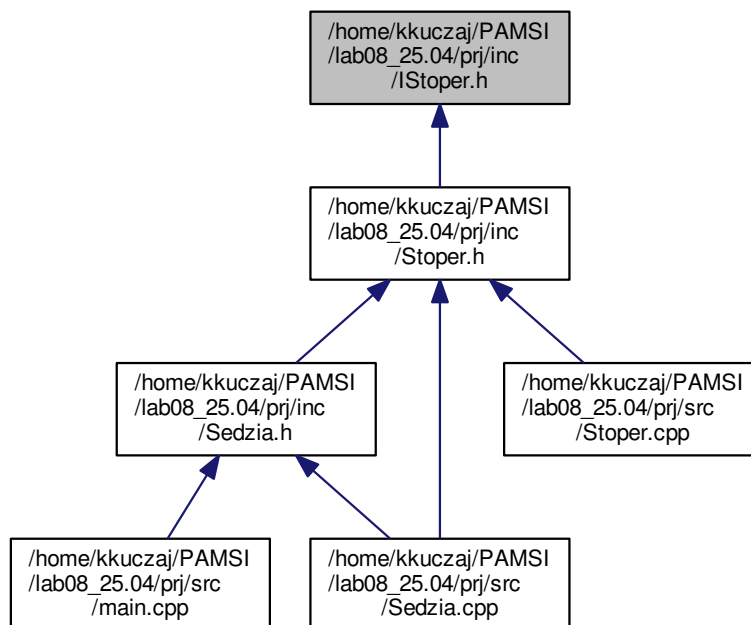
Nagłówek zawierający interfejs dla stopera.

```
#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.

6.8.1 Detailed Description

Naglowek zawierajacy interfejs dla stopera.

Author

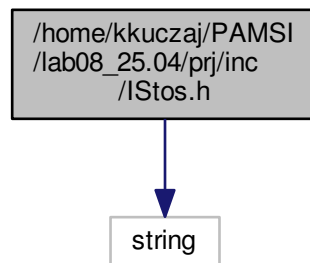
Kamil Kuczaj

6.9 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IStos.h File Reference

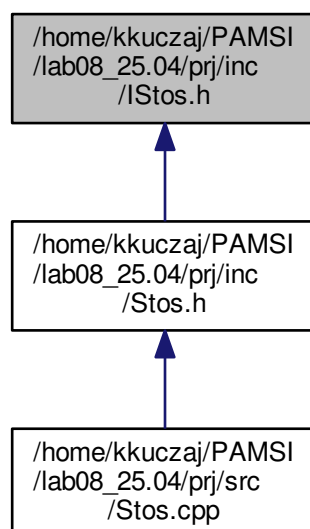
Plik zawiera interfejs dla pojemnika [Stos](#).

```
#include <string>
```

Include dependency graph for IStos.h:



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



Classes

- class [IStos< Type >](#)

Interfejs dla każdego pojemnika.

6.9.1 Detailed Description

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

Author

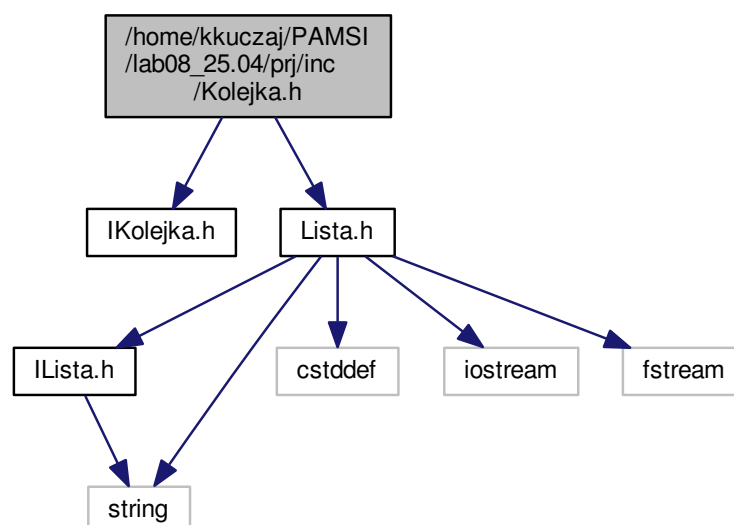
Kamil Kuczaj.

6.10 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Kolejka.h File Reference

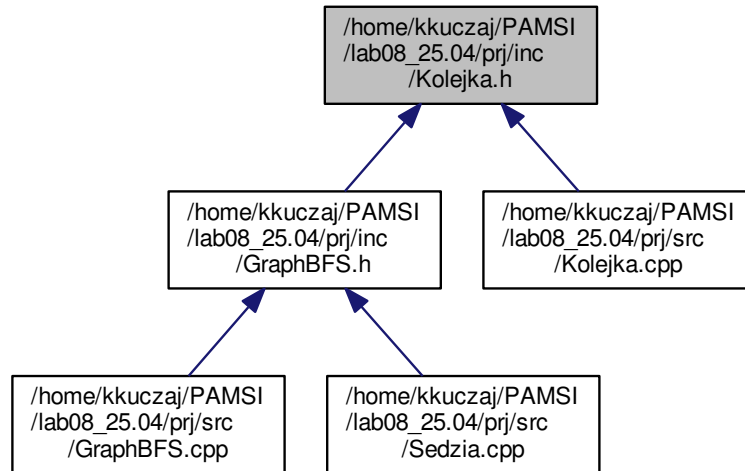
```
#include "IKolejka.h"
```

```
#include "Lista.h"
```

Include dependency graph for Kolejka.h:



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



Classes

- class [Kolejka< Type >](#)

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

6.11 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Lista.h File Reference

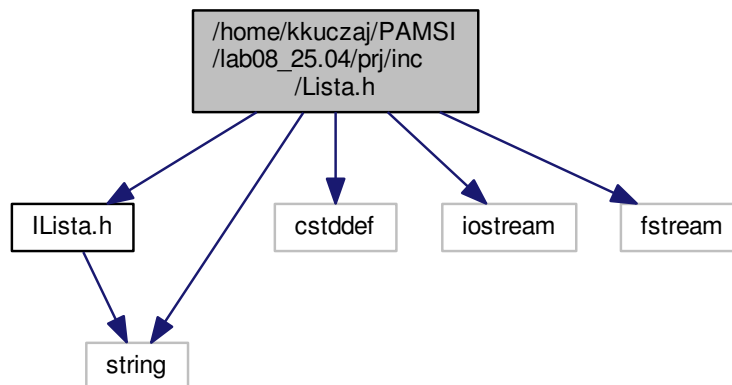
Implementacja jednokierunkowej listy.

```

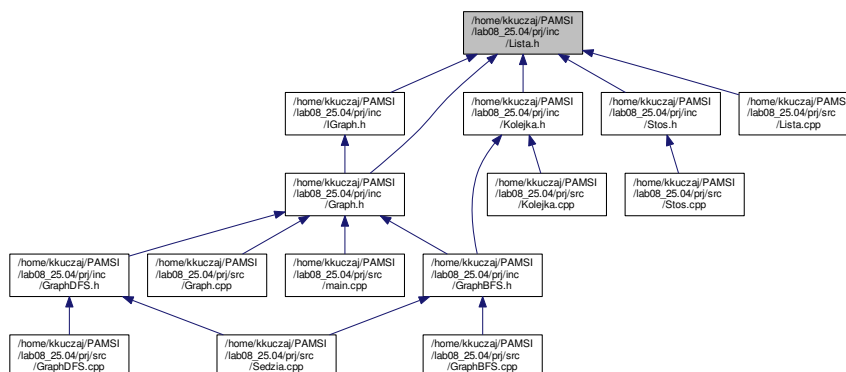
#include "ILista.h"
#include <cstdlib>
#include <string>
#include <iostream>
#include <fstream>

```


Include dependency graph for Lista.h:



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



Classes

- class `Lista< Type >`
Klasa `Lista`, która symuluje zachowanie klasy `list` z biblioteki `STL`.
- struct `Lista< Type >::Node`
Implementacja węzłów dla listy.

6.11.1 Detailed Description

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

Author

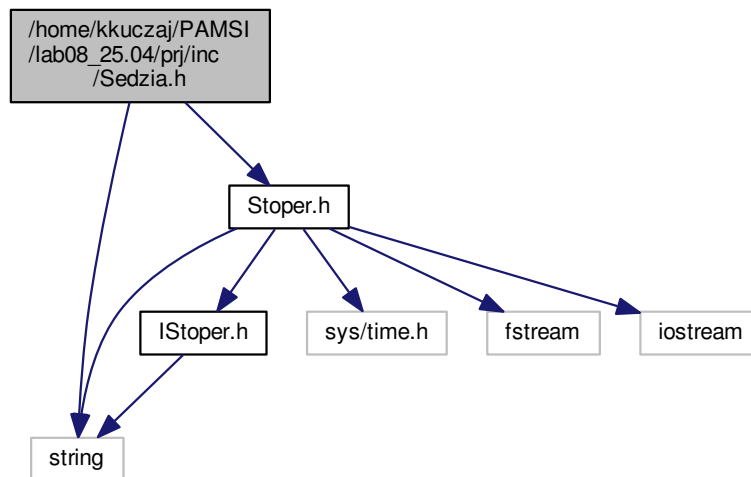
Kamil Kuczaj.

6.12 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Sedzia.h File Reference

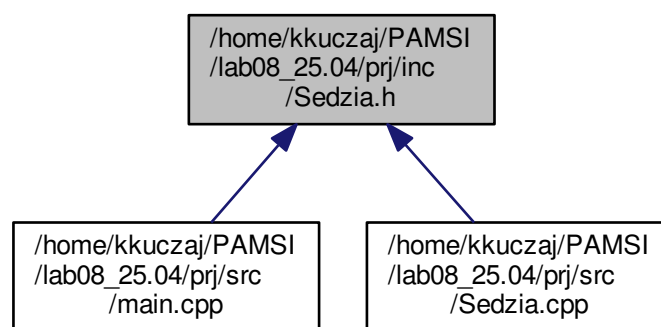
Naglowek opisujacy implementacje Sedziego.

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

Include dependency graph for Sedzia.h:



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



Classes

- class [Sedzia](#)

Implementacja klasy [Sedzia](#).

6.12.1 Detailed Description

Naglowek opisujący implementację Sedziego.

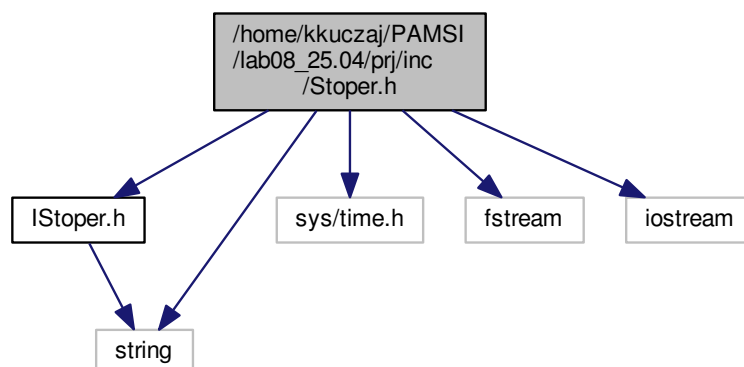
Author

Kamil Kuczaj

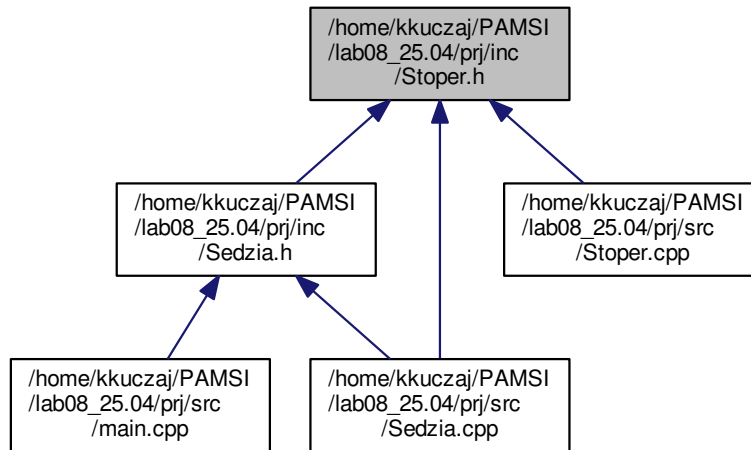
6.13 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Stoper.h File Reference

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](#).

6.13.1 Detailed Description

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

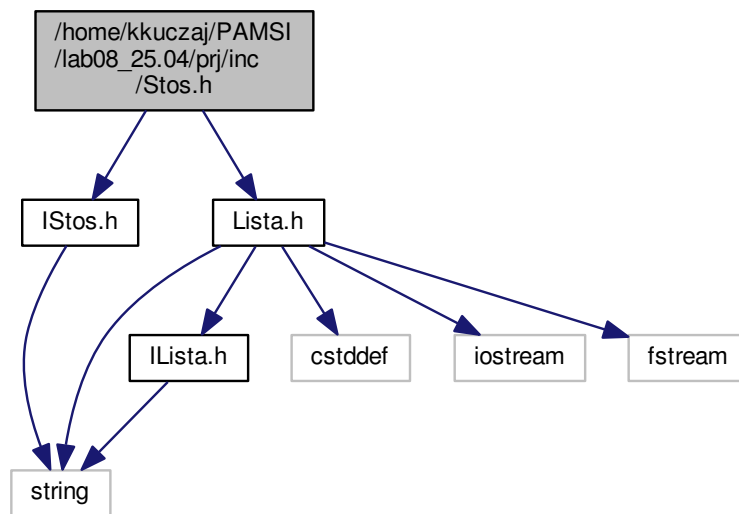
Author

Kamil Kuczaj

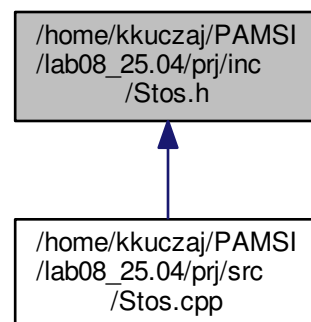
6.14 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Stos.h File Reference

```
#include "IStos.h"
#include "Lista.h"
```

Include dependency graph for Stos.h:



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



Classes

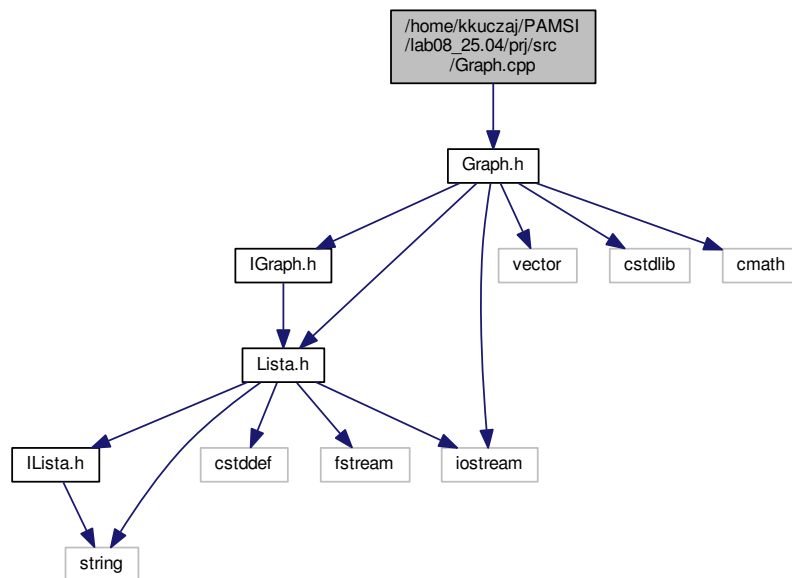
- class `Stos< Type >`

Implementacja klasy `Stos`, złożonej z intów.

6.15 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/Graph.cpp File Reference

```
#include "Graph.h"
```

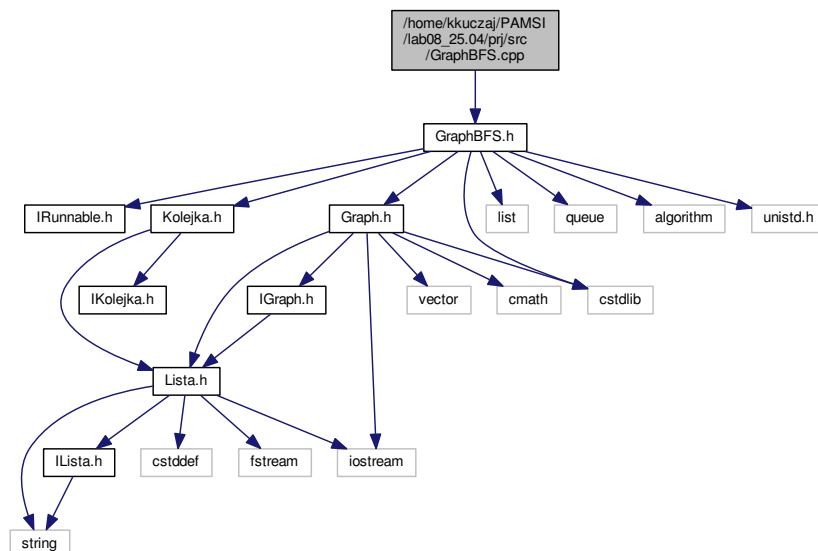
Include dependency graph for Graph.cpp:



6.16 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/GraphBFS.cpp File Reference

```
#include "GraphBFS.h"
```

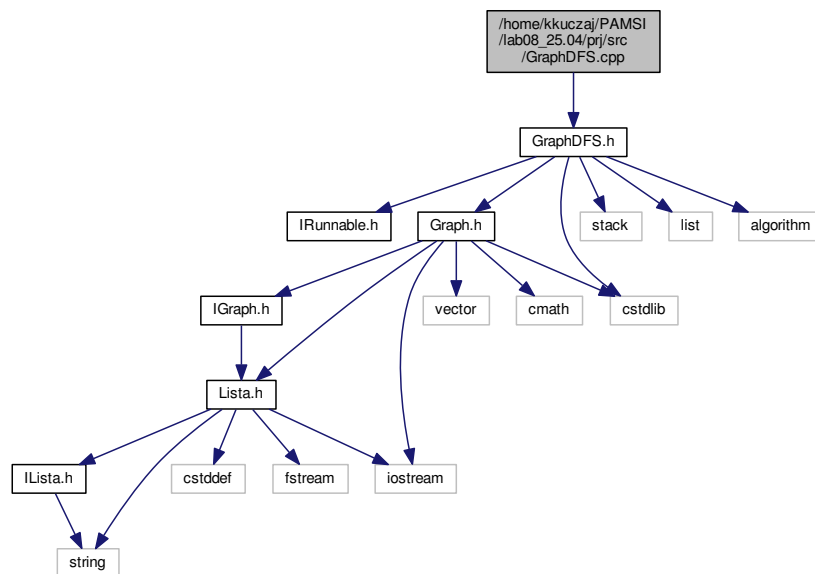
Include dependency graph for GraphBFS.cpp:



6.17 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/GraphDFS.cpp File Reference

```
#include "GraphDFS.h"
```

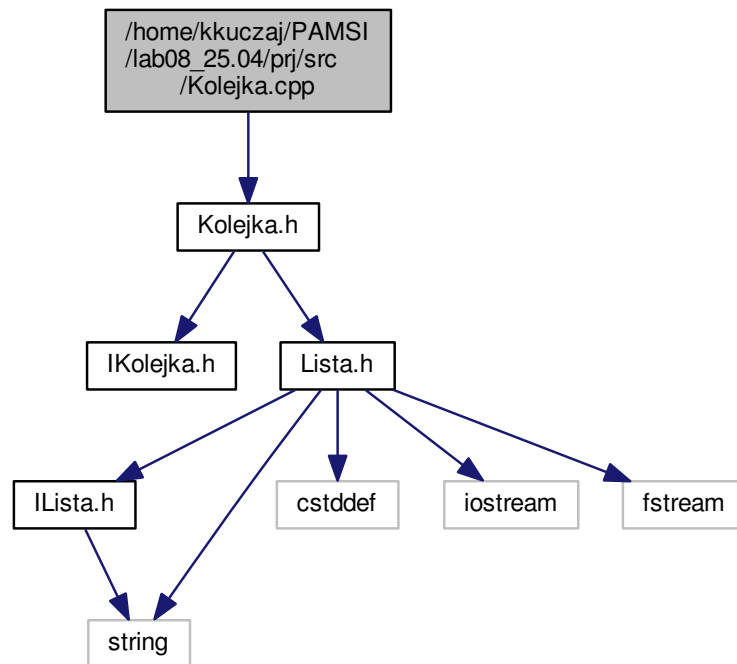
Include dependency graph for GraphDFS.cpp:



6.18 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/Kolejka.cpp File Reference

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

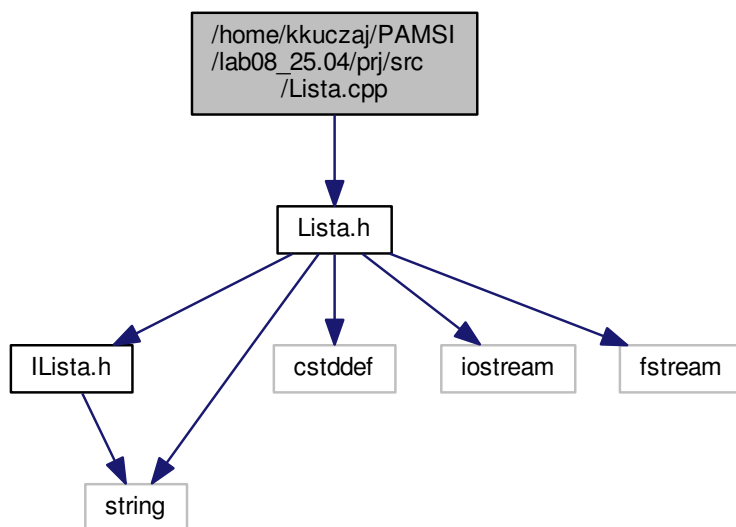
Include dependency graph for Kolejka.cpp:



6.19 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/List.cpp File Reference

```
#include "Lista.h"
```


Include dependency graph for Lista.cpp:



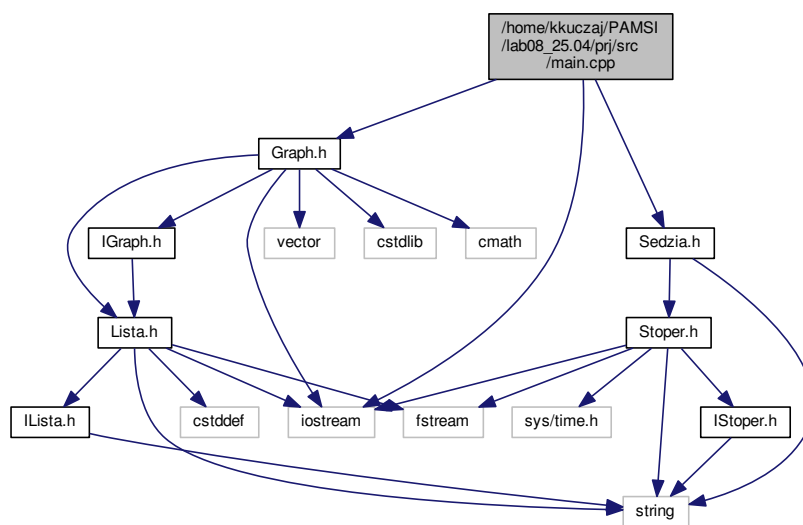
6.20 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/main.cpp File Reference

```

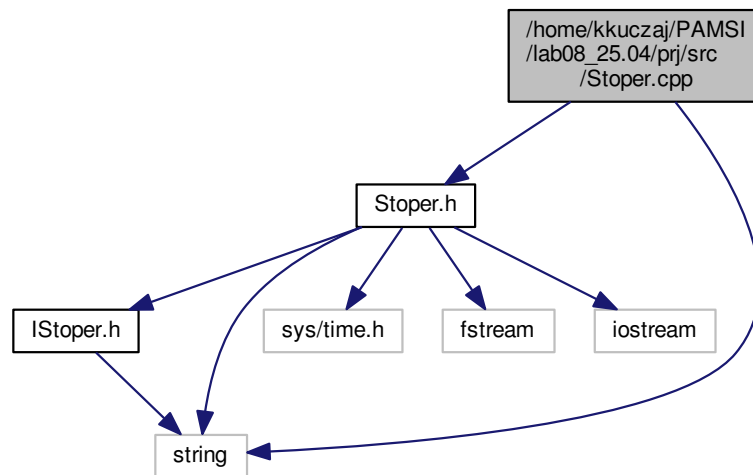
#include "Sedzia.h"
#include <iostream>
#include "Graph.h"

```

Include dependency graph for main.cpp:



Include dependency graph for Stoper.cpp:



6.23 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/Stos.cpp File Reference

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

Include dependency graph for Stos.cpp:

