Pomiar czasu algorytmów DFS i BFS grafu nieskierowanego

Generated by Doxygen 1.8.6

Sun May 8 2016 23:09:27

Contents

Chapter 1

Opis programu

Author

Kamil Kuczaj 218478@student.pwr.edu.pl

1.1 Wstep

Program zostal zbudowany modulowo. W folderze inc/ znajduja sie pliki naglowkowe. Folder src/ zawiera pliki zrodlowe. W glownym folderze zbudowany zostal Makefile. Pliki obiektowe sa budowane w folderze obj/ a nastepnie linkowane do glownego folderu (prj/). Testowano przy wykorzystaniu kompilatora g++ w wersji 4.8.4 na systemie Linux Ubuntu 14.04.04 opartego o jadro 4.2.0-30-generic.

1.2 Licencja

Program udostepniam na licencji GPLv3.

1.3 Instalacja

Aby zbudowac i jednoczesnie odpalic program: $ make

Aby pozbyc sie plikow z koncowka $*\sim$ lub zaczynajacych sie na #$*$: $ make order

Aby pozbyc sie programu wykonywalnego oraz plikow obiektowych: $ make clean

Aby wyswietlic pomoc do pliku Makefile: $ make help

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

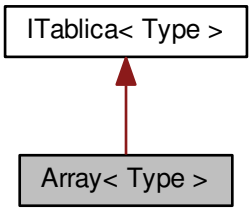
Class Documentation

5.1 Array< Type > Class Template Reference

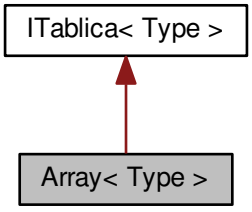
Klasa Tablica, w ktorej odbywa sie zapis dynamiczny elementow.

`#include <Tablica.h>`

Inheritance diagram for Array< Type >:



Collaboration diagram for Array< Type >:



Public Member Functions

- virtual bool isFull ()
  Pozwala prosto okreslic, czy nalezy przydzielic pamiec.
- virtual void increaseSize ()
  Zwieksza rozmiar przydzielonej pamieci na stercie.
- Array (int x=10)
  Konstruktor parametryczny.
- ~Array ()
  Destruktor.
- virtual int getSize ()
  Zwraca aktualny rozmiar tablicy dynamicznej.
- void decreaseSize (int n)
  Zmniejsza zmienna przechowujaca rozmiar tablicy.
- virtual int getDesiredSize () const
  Zwraca wartosc desired_size.
- virtual void setDesiredSize (int t)
  Ustawia wartosc desired_size.
- virtual Type operator[] (int i) const
  Akcesor do tablicy.
- virtual Type & operator[] (int i)
  Modyfikator do tablicy.
- void bubbleSort ()
  Sortowanie babelkowe.

Private Attributes

- Type * elements
  Wskaznik do poczatku tablicy dynamicznej.
- int current_size
  Okresla aktualny rozmiar stosu.
- int desired_size
  Okresla pozadany rozmiar stosu.
- int index
  Okresla aktualny indeks.

Additional Inherited Members

5.1.1 Detailed Description

template<class Type>class Array< Type >

Klasa Tablica, w ktorej odbywa sie zapis dynamiczny elementow.

Implementuje metody interfejsu ITablica. Zajmuje sie dynamiczna alokacja pamieci.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 template<class Type > Array< Type >::Array (int *x* = 10) [inline], [explicit]

Konstruktor parametryczny.

Umozliwia okreslenie poczatkowego rozmiaru tablicy. W przypadku braku okreslenia tego rozmiaru przyjmuje domyslna wartosc rowna 10. Explicit oznacza tyle, ze nie moge stworzyc tablicy w ten sposob: Tablica t = 10;

Parameters

x	Okresla poczatkowa wielkosc przydzielonej pamieci. Domyslna wartosc w przypadku braku podania to 10.

5.1.2.2 template<class Type > Array< Type >::~Array ( ) [inline]

Destruktor.

Usuwa pamiec przypisana komorce, na ktora wskazuje pole ∗elements.

5.1.3 Member Function Documentation

5.1.3.1 template<class Type > void Array< Type >::bubbleSort ( ) [inline]

Sortowanie babelkowe.

Sortuje elementy metoda babelkowa. Zlozonosc obliczeniowa n^2.

Here is the call graph for this function:

5.1.3.2 template<class Type > void Array< Type >::decreaseSize (int *n*) [inline]

Zmniejsza zmienna przechowujaca rozmiar tablicy.

Zmniejsza rozmiar, zmienna current_size o n. Stworzenie tej funkcji zostalo wymuszone przez implementacje listy. Uzywanie funkcji remove(int n) z klasy Lista powodowalo to, ze jej rozmiar faktycznie malal o jeden element, ale klasa Tablica o tym nie wiedziala.

Parameters

in	*n*	O ile zmniejszyc zmienna current_size.

5.1.3.3 template<class Type > virtual int Array< Type >::getDesiredSize ( ) const [inline], [virtual]

Zwraca wartosc desired_size.

Zwraca rozmiar, ktory ma osiagnac tablica. Moze byc wieksza niz desired_size.

Implements ITablica< Type >.

5.1.3.4 template<class Type > virtual int Array< Type >::getSize ( ) [inline], [virtual]

Zwraca aktualny rozmiar tablicy dynamicznej.

Zwraca wartosc pola current_size.

Returns

Zwraca wartosc typu int. Reprezentuje ilosc danych w tablilcy.

Implements ITablica< Type >.

Here is the caller graph for this function:

5.1.3.5 template<class Type > virtual void Array< Type >::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< Type >.

5.1.3.6 template<class Type > virtual bool Array< Type >::isFull ( ) [inline], [virtual]

Pozwala prosto okreslic, czy nalezy przydzielic pamiec.

Metoda prywatna. Sluzy do okreslania czy nalezy wywolac metode increaseSize().

Return values

true	Pamiec pelna. Nalezy zwiekszyc rozmiar.
false	Jest jeszcze wolne miejsce.

Implements ITablica< Type >.

5.1.3.7 template<class Type > virtual Type Array< Type >::operator[] (int *i*) const [inline], [virtual]

Akcesor do tablicy.

Umozliwia dostep do tablicy.

Parameters

in	*i*	Indeks, w ktorym wartosc tablicy ma zostac zwrocona.

Returns

Wartosc komorki tablicy, wskazywana przez i-ty indeks.

Implements ITablica< Type >.

5.1.3.8 template<class Type > virtual Type& Array< Type >::operator[] (int *i*) [inline], [virtual]

Modyfikator do tablicy.

Umozliwia dostep do zmiany i-tego elementu w tablicy.

Parameters

in	*i*	Wskazuje element, ktory ma zostac zmieniony.

Returns

Referencja do i-tego elementu.

Implements ITablica< Type >.

5.1.3.9 template<class Type > virtual void Array< Type >::setDesiredSize (int *t*) [inline], [virtual]

Ustawia wartosc desired_size.

Ustawia rozmiar, ktory ma osiagnac tablica.

Implements ITablica< Type >.

5.1.4 Member Data Documentation

5.1.4.1 template<class Type > int Array< Type >::current_size [private]

Okresla aktualny rozmiar stosu.

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

5.1.4.2 template<class Type > int Array< Type >::desired_size [private]

Okresla pozadany rozmiar stosu.

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

5.1.4.3 template<class Type > Type* Array< Type >::elements [private]

Wskaznik do poczatku tablicy dynamicznej.

Wskazuje na adres w pamieci sterty. Pole prywatne.

5.1.4.4 template<class Type > int Array< Type >::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:

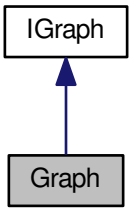
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Tablica.h

5.2 Graph Class Reference

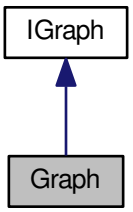
Graf oparty o liste sasiedztwa.

`#include <Graph.h>`

Inheritance diagram for Graph:



Collaboration diagram for Graph:



Public Member Functions

- 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 sasiadujace wierczholki pierwszego wierzcholka grafu.
- Lista< int > back ()

  Zwraca sasiadujace wierczholki ostatniego elementu grafu.
- Lista< int > operator[] (int n)

  Zwraca sasiadujace wierzcholki n-tego wierzcholka grafu.
- bool isEdge (int u, int v)

Private Attributes

- std::vector< Lista< int > > graph

  Pole, ktore bedzie reprezentowac graf..

5.2.1 Detailed Description

Graf oparty o liste sasiedztwa.

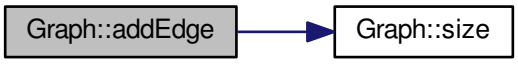
Lista sasiedztwa zostala wybrana jako sposob implementacji grafu, gdyz algorytmy przeszukania grafu BFS oraz DFS sa na niej szybsze. Dodatkowo, dodaje wierzcholki juz posortowane. Dzieki temu, pojemnik jest bardziej czytelny.

5.2.2 Member Function Documentation

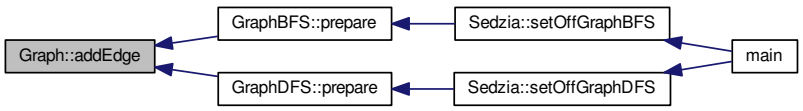
5.2.2.1 virtual void Graph::addEdge (int *x*, int *y*, int *weight* = 1) [inline], [virtual]

Implements IGraph.

Here is the call graph for this function:



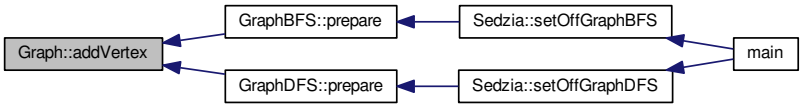
Here is the caller graph for this function:



5.2.2.2 virtual void Graph::addVertex (int *x*) [inline], [virtual]

Implements IGraph.

Here is the caller graph for this function:



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

Zwraca sasiadujace wierczholki ostatniego elementu grafu.

Here is the call graph for this function:

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

Zwraca sasiadujace wierczholki pierwszego wierzcholka grafu.

5.2.2.5 virtual Lista<int> Graph::getNeighbours (int *x*) [inline], [virtual]

Implements IGraph.

5.2.2.6 bool Graph::isEdge (int *u*, int *v*) [inline]

Here is the caller graph for this function:

5.2.2.7 Lista<int> Graph::operator[] (int *n*) [inline]

Zwraca sasiadujace wierzcholki n-tego wierzcholka grafu.

5.2.2.8 void Graph::print ( ) [inline]

For debug.

5.2.2.9 virtual void Graph::removeEdge (int *x*, int *y*) [inline], [virtual]

Implements IGraph.

5.2.2.10 virtual void Graph::removeVertex (int *x*) [inline], [virtual]

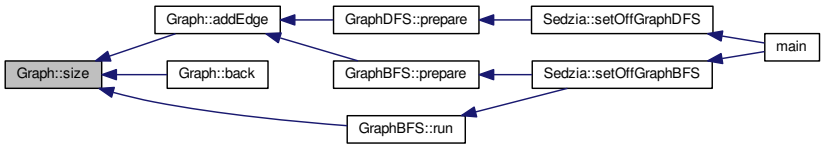
Implements IGraph.

5.2.2.11 int Graph::size ( ) [inline]

Returns the number of vertices.

Return the size of vector.

Here is the caller graph for this function:



5.2.3 Member Data Documentation

5.2.3.1 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 moge skupic sie na implementacji grafu.

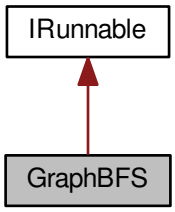
The documentation for this class was generated from the following file:

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

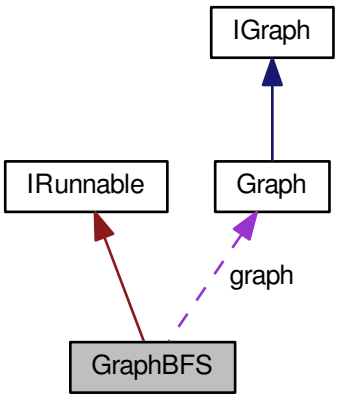
5.3 GraphBFS Class Reference

`#include <GraphBFS.h>`

Inheritance diagram for GraphBFS:



Collaboration diagram for GraphBFS:



Public Member Functions

- virtual void run ()

  Odpalenie badanej czynnosci.
- virtual void prepare (int how_many)

  Przygotowuje pojemnik przed wykonaniem czynnosci.

Private Attributes

- Graph graph

  Testowany graf.
- std::list< int > lista

  Przechowuje liste wierzcholkow.

Additional Inherited Members

5.3.1 Member Function Documentation

5.3.1.1 virtual void GraphBFS::prepare (int *size*) [inline], [virtual]

Przygotowuje pojemnik przed wykonaniem czynnosci.

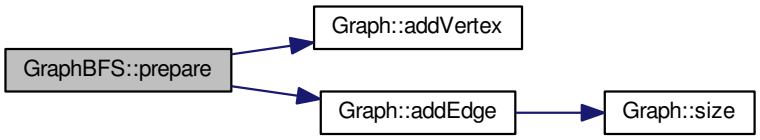
Funkcja, ktora ma wykonac wszystkie dodatkowe czynnosci, ktorych czasu nie bedziemy mierzyc. Polega ona na wczytaniu konkretnej ilosci elementow.

Parameters

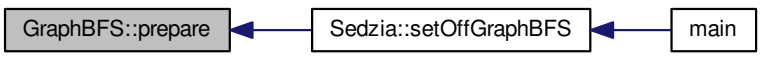
in	*size*	Ilosc elementow.

Implements IRunnable.

Here is the call graph for this function:



Here is the caller graph for this function:



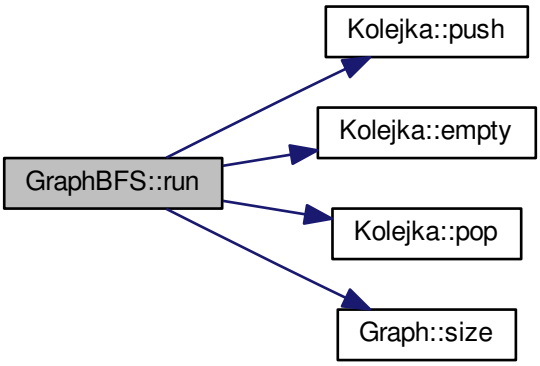
5.3.1.2 virtual void GraphBFS::run ( ) [inline], [virtual]

Odpalenie badanej czynnosci.

Funkcja, ktorej cialem 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:

5.3.2 Member Data Documentation

5.3.2.1 Graph GraphBFS::graph [private]

Testowany graf.

W celu dalszych informacji na temat tego pola prosze odwolac sie do klasy Graph.

5.3.2.2 std::list<int> GraphBFS::lista [private]

Przechowuje liste wierzcholkow.

Przechowuje liste wierzcholkow badanego grafu.

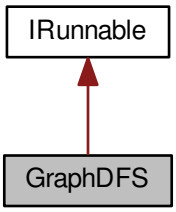
The documentation for this class was generated from the following file:

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

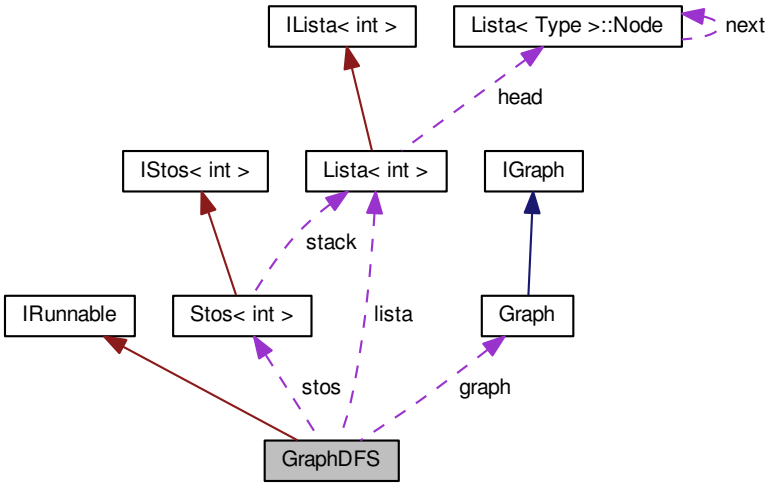
5.4 GraphDFS Class Reference

`#include <GraphDFS.h>`

Inheritance diagram for GraphDFS:



Collaboration diagram for GraphDFS:



Public Member Functions

- void DFS ()
- void runDFS (int u, int state[])
- virtual void run ()

  Odpalenie badanej czynnosci.
- virtual void prepare (int how_many)

  Przygotowuje pojemnik przed wykonaniem czynnosci.

Private Types

- enum VertexState { White, Gray, Black }

Private Attributes

- Graph graph

  Testowany graf.
- Lista< int > lista

  Przechowuje liste wierzcholkow.
- Stos< int > stos

  Przechowuje tymczasowe odwiedzone wierzcholki.
- int graph_size

Additional Inherited Members

5.4.1 Member Enumeration Documentation

5.4.1.1 enum GraphDFS::VertexState [private]

Enumerator

White

Gray

Black

5.4.2 Member Function Documentation

5.4.2.1 void GraphDFS::DFS ( ) [inline]

Here is the call graph for this function:

Here is the caller graph for this function:

5.4.2.2 virtual void GraphDFS::prepare (int *size*) [inline], [virtual]

Przygotowuje pojemnik przed wykonaniem czynnosci.

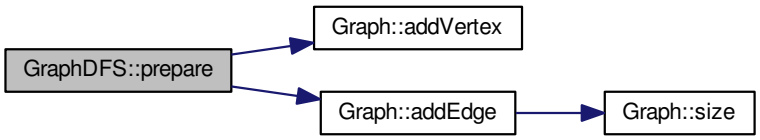
Funkcja, ktora ma wykonac wszystkie dodatkowe czynnosci, ktorych czasu nie bedziemy mierzyc. Polega ona na wczytaniu konkretnej ilosci elementow.

Parameters

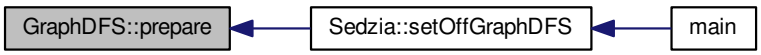
in	*size*	Ilosc elementow.

Implements IRunnable.

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.3 virtual void GraphDFS::run ( ) [inline], [virtual]

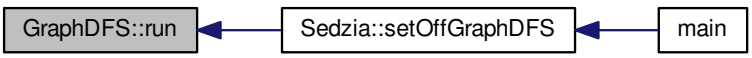
Odpalenie badanej czynnosci.

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

Implements IRunnable.

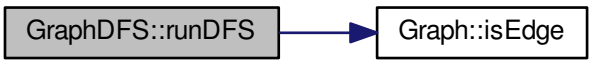
Here is the call graph for this function:

Here is the caller graph for this function:

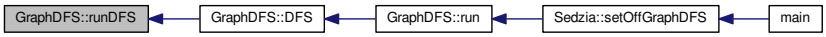


5.4.2.4 void GraphDFS::runDFS (int *u*, int *state[]*) [inline]

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3 Member Data Documentation

5.4.3.1 Graph GraphDFS::graph [private]

Testowany graf.

W celu dalszych informacji na temat tego pola prosze odwolac sie do klasy Graph.

5.4.3.2 int GraphDFS::graph_size [private]

5.4.3.3 Lista<int> GraphDFS::lista [private]

Przechowuje liste wierzcholkow.

Przechowuje liste wierzcholkow badanego grafu.

5.4.3.4 Stos<int> GraphDFS::stos [private]

Przechowuje tymczasowe odwiedzone wierzcholki.

Warunek DFSu.

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

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

5.5 IBinaryTree< Type > Class Template Reference

Interfejs drzewa binarnego.

`#include <IBinaryTree.h>`

Public Member Functions

- virtual void put (Type element)=0
- virtual bool search (Type element) const =0
- virtual void rebalance ()=0
- virtual void print ()=0

5.5.1 Detailed Description

template<class Type>class IBinaryTree< Type >

Interfejs drzewa binarnego.

Zawiera ogolne zalozenia tej struktury danych. Uzyto szablonow. W przypadku wrzucenia dwoch takich samych elementow - nadpisuja sie.

5.5.2 Member Function Documentation

5.5.2.1 template<class Type > virtual void IBinaryTree< Type >::print ( ) [pure virtual]

5.5.2.2 template<class Type > virtual void IBinaryTree< Type >::put (Type *element*) [pure virtual]

5.5.2.3 template<class Type > virtual void IBinaryTree< Type >::rebalance ( ) [pure virtual]

5.5.2.4 template<class Type > virtual bool IBinaryTree< Type >::search (Type *element*) const [pure virtual]

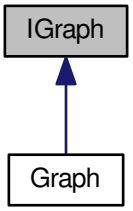
The documentation for this class was generated from the following file:

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

5.6 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

5.6.1 Member Function Documentation

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

Implemented in Graph.

5.6.1.2 virtual void IGraph::addVertex (int *x*) [pure virtual]

Implemented in Graph.

5.6.1.3 virtual Lista<int> IGraph::getNeighbours (int *x*) [pure virtual]

Implemented in Graph.

5.6.1.4 virtual void IGraph::removeEdge (int *x*, int *y*) [pure virtual]

Implemented in Graph.

5.6.1.5 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

5.7 IHashTable Class Reference

Interfejs tablicy z hashowaniem.

`#include <IHashTable.h>`

Public Member Functions

- virtual int operator[] (std::string key) const =0

  Getter do tablicy.
- virtual void put (std::string key, int value)=0

  Setter do tablicy.

5.7.1 Detailed Description

Interfejs tablicy z hashowaniem.

Interfejs tablicy z hashowaniem, ktory opisuje jakie funkcje beda znajdywaly sie w implementacji oraz co beda zwracac.

5.7.2 Member Function Documentation

5.7.2.1 virtual int IHashTable::operator[] (std::string *key*) const [pure virtual]

Getter do tablicy.

Po podaniu klucza, w tym przypadku nazwiska, funkcja zwroci nam numer telefonu, ktory jest przypisany do nazwiska. Zupelnie jak w rzeczywistej ksiazce telefonicznej. W funkcji obowiazuja dwa wyjatki:

- EmptyList ==> rzucany w przypadku gdy nie ma zadnego rekordu, tj. lista (tom) jest pusta
- KeyIsNotThere ==> rzucany w przypadku gdy nie znalezlismy podanego klucza, a lista (tom) nie jest pusta.

Parameters

in	*key*	Klucz, po ktorym wyszukiwana jest odpowiednia lista.

5.7.2.2 virtual void IHashTable::put (std::string *key,* int *value*) [pure virtual]

Setter do tablicy.

Dodaje element do tomu, pod warunkiem, ze nie ma takiego elementu. W przypadku gdy znajdzie juz taki rzuca wyjatek:

- DuplicateElement ==> rzucany w przypadku, gdy podany klucz jest juz w bazie.

Parameters

in	*key*	Klucz, po ktorym wyszukiwana jest odpowiednia lista. Musi byc inny niz te, ktore znajduja sie juz w tabliicy.
in	*value*	Numer telefonu przypisany do nazwiska. To ten element bedzie zwracany przy wyszukiwaniu.

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

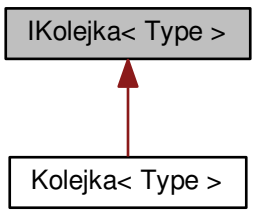
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IHashTable.h

5.8 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.

5.8.1 Detailed Description

template<class Type>class IKolejka< Type >

Interfejs dla kolejki.

Abstrakcyjna klasa, ktora zostala utworzona na potrzeby ADT Abstract Data Types.

5.8.2 Member Function Documentation

5.8.2.1 template<class Type > virtual bool IKolejka< Type >::empty ( ) [protected], [pure virtual]

Sprawdza czy pojemnika jest pusty.

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

Return values

true	Pojemnik pusty.
false	Pojemnik nie jest pusty.

Implemented in Kolejka< Type >.

5.8.2.2 template<class Type > virtual Type IKolejka< Type >::pop ( ) [protected], [pure virtual]

Usuwa element z pojemnika.

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

Returns

Usuniety element.

Implemented in Kolejka< Type >.

5.8.2.3 template<class Type > virtual void IKolejka< Type >::push (Type *element*) [protected], [pure virtual]

Dodaje element na poczatek.

Dodaje element na poczatek pojemnika.

Parameters

in	*element*	"Wpychany" element typu string.

Implemented in Kolejka< Type >.

5.8.2.4 template<class Type > virtual int IKolejka< 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 Kolejka< Type >.

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

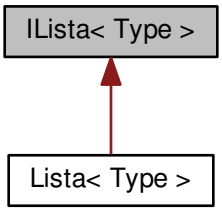
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IKolejka.h

5.9 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.

5.9.1 Detailed Description

template<class Type>class ILista< Type >

Interfejs dla pojemnika Lista.

Abstrakcyjna klasa, ktora zostala utworzona na potrzeby ADT Abstract Data Types.

5.9.2 Member Function Documentation

5.9.2.1 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

in	*item*	Element wstawiany. Slowo.
in	*index*	Miejsce, w ktore ma byc wstawiony element item.

Implemented in Lista< Type >, and Lista< int >.

5.9.2.2 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 >, and Lista< int >.

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

Implemented in Lista< Type >, and Lista< int >.

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

Sprawdza czy lista jest pusta.

Sprawdza czy w liscie sa jakies elementy.

Return values

true	Lista jest pusta.
false	Lista nie jest pusta.

Implemented in Lista< Type >, and Lista< int >.

5.9.2.5 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 zawartosc komorki o tej indeksie.

Implemented in Lista< Type >, and Lista< int >.

5.9.2.6 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 >, and Lista< int >.

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

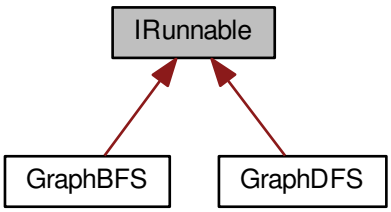
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/ILista.h

5.10 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.

5.10.1 Detailed Description

Interfejs dla biegacza.

Klasa abstrakcyjna z metodami czysto wirtualnymi.

5.10.2 Member Function Documentation

5.10.2.1 virtual void IRunnable::prepare (int *size*) [protected], [pure virtual]

Przygotowuje pojemnik przed wykonaniem czynnosci.

Funkcja, ktora ma wykonac wszystkie dodatkowe czynnosci, ktorych czasu nie bedziemy mierzyc. Polega ona na wczytaniu konkretnej ilosci elementow.

Parameters

in	*size*	Ilosc elementow.

Implemented in GraphDFS, and GraphBFS.

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

Odpalenie badanej czynnosci.

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

Implemented in GraphDFS, and GraphBFS.

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

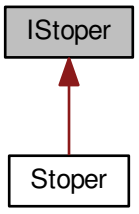
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IRunnable.h

5.11 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.

5.11.1 Detailed Description

Interfejs dla stopera.

Klasa abstrakcyjna z metodami czysto wirtualnymi.

5.11.2 Member Function Documentation

5.11.2.1 virtual void IStoper::dumpToFile (std::string *file_name*) [protected], [pure virtual]

Ma symulowac moment zapisu zmierzonego czasu na kartke papieru.

Metoda czysto wirtualna.

Parameters

file_name	Nazwa pliku. Obiekt klasy string.

Implemented in Stoper.

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

Ma symulowac rezultat pokazania wyniku pomiaru czasu na stoperze.

Metoda czysto wirtualna.

Implemented in Stoper.

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

Ma symulowac moment startu stopera.

Metoda czysto wirtualna.

Implemented in Stoper.

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

Implemented in Stoper.

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

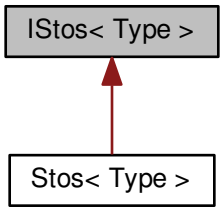
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IStoper.h

5.12 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.

5.12.1 Detailed Description

template<class Type>class IStos< Type >

Interfejs dla każdego pojemnika.

Abstrakcyjna klasa, ktora zostala utworzona na potrzeby ADT Abstract Data Types.

5.12.2 Member Function Documentation

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

Sprawdza czy pojemnika jest pusty.

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

Return values

true	Pojemnik pusty.
false	Pojemnik nie jest pusty.

Implemented in Stos< Type >, and Stos< int >.

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

Usuwa element z pojemnika.

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

Returns

Usuniety element.

Implemented in Stos< Type >, and Stos< int >.

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

Dodaje element na poczatek.

Dodaje element na poczatek pojemnika.

Parameters

in	*element*	"Wpychany" element typu std::string.

Implemented in Stos< Type >, and Stos< int >.

5.12.2.4 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 >, and Stos< int >.

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

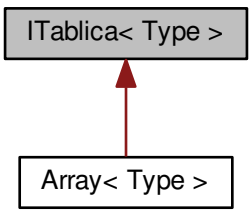
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IStos.h

5.13 ITablica< Type > Class Template Reference

Interfejs tablicy.

`#include <ITablica.h>`

Inheritance diagram for ITablica< Type >:



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 Type operator[] (int i) const =0

  Akcesor to i-tego elementu tablicy.
- virtual Type & operator[] (int i)=0

  Modyfikator do i-tego elementu tablicy.

5.13.1 Detailed Description

template<class Type>class ITablica< Type >

Interfejs tablicy.

Wymuszony poprzez ISP (programowanie obiektowe SOLID).

5.13.2 Member Function Documentation

5.13.2.1 template<class Type > virtual int ITablica< Type >::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, satysfakcjonujaca ilosc elementow.

Implemented in Array< Type >.

5.13.2.2 template<class Type > virtual int ITablica< Type >::getSize ( ) [protected], [pure virtual]

Zwraca ilosc zapisanych elementow.

Zwraca ilosc elementow, ktore sa w tablicy. Nie uwzglednia pustych komorek. Pamietaj, ze indeksujemy od zera, wiec ostatni element ma indeks (getSize() - 1)

Returns

Ilosc zapisanych elementow.

Implemented in Array< Type >.

5.13.2.3 template<class Type > virtual void ITablica< Type >::increaseSize ( ) [protected], [pure virtual]

Zwieksza rozmiar tablicy.

Alokuje pamiec dla nowej tablicy dynamicznej oraz kopiuje elementy starej tablicy do nowej. Nastepnie usuwa pamiec dla starej tablicy.

Implemented in Array< Type >.

5.13.2.4 template<class Type > virtual bool ITablica< Type >::isFull ( ) [protected], [pure virtual]

Sprawdza czy tablica jest pelna.

Sprawdza czy sa jeszcze wolne komorki pamieci przydzielone tablicy.

Return values

true	Tablica pelna. Nalezy zaalokowac nowa pamiec.
false	Jest jeszcze miejsce.

Implemented in Array< Type >.

5.13.2.5 template<class Type > virtual Type ITablica< Type >::operator[] (int *i*) const [protected], [pure virtual]

Akcesor to i-tego elementu tablicy.

Umozliwia dostep do i-tego elementu. Nie mozemy ta metoda zmieniac wartosci tego elementu, lecz mozemy go odczytac.

Returns

i-ty element

Implemented in Array< Type >.

5.13.2.6 template<class Type > virtual Type& ITablica< Type >::operator[] (int *i*) [protected], [pure virtual]

Modyfikator do i-tego elementu tablicy.

Umozliwia dostep do i-tego elementu. Mozemy ta metoda jedynie zmieniac wartosc i-tego elementu, gdyz odwolujemy sie do niego poprzez referencje.

Returns

Referencja do i-tego elementu

Implemented in Array< Type >.

5.13.2.7 template<class Type > virtual void ITablica< Type >::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 Array< Type >.

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

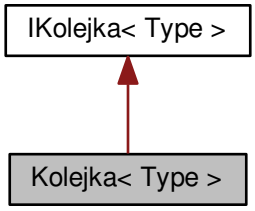
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/ITablica.h

5.14 Kolejka< Type > Class Template Reference

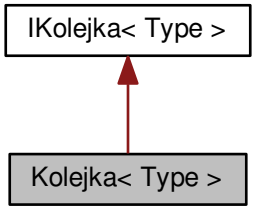
Implementacja interfejsu IKolejka w postaci klasy Kolejka.

`#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

5.14.1 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.

5.14.2 Member Function Documentation

5.14.2.1 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

true	Pojemnik pusty.
false	Pojemnik nie jest pusty.

Implements IKolejka< Type >.

Here is the caller graph for this function:

5.14.2.2 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 >.

Here is the caller graph for this function:

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

Wyswietla zawartosc kolejki.

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

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

Dodaje element na poczatek.

Dodaje element na poczatek pojemnika.

Parameters

in	*element*	"Wpychany" element typu string.

Implements IKolejka< Type >.

Here is the caller graph for this function:

5.14.2.5 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 >.

5.14.3 Member Data Documentation

5.14.3.1 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:

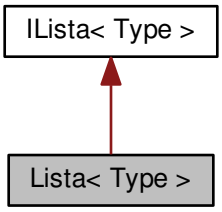
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Kolejka.h

5.15 Lista< Type > Class Template Reference

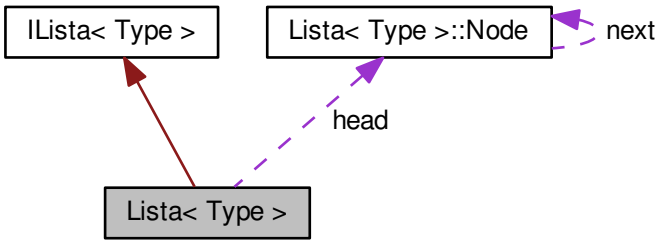
Klasa Lista, ktora symuluje zachowanie klasy list z biblioteki STL.

`#include <Lista.h>`

Inheritance diagram for Lista< Type >:



Collaboration diagram for Lista< Type >:



Classes

- struct Node

  Imlementacja wezlow 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.
- 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

5.15.1 Detailed Description

template<class Type>class Lista< Type >

Klasa Lista, ktora symuluje zachowanie klasy list z biblioteki STL.

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

5.15.2 Constructor & Destructor Documentation

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

Konstruktor.

Tworzy poczatek listy. Alokuje dla niego pamiec.

5.15.2.2 template<class Type> Lista< Type >::Lista (Type *x*) [inline]

Parametryczny konstruktor.

Konstruktor dodany na potrzeby implementacji grafu.

5.15.3 Member Function Documentation

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

Wstawia element w dowolnym miejscu listy.

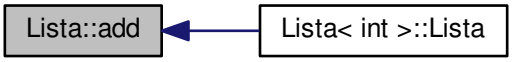
Wstawia element typu Type w miejsce wskazywane przez zmienna index.

Parameters

in	*item*	Element wstawiany. Slowo typu string.
in	*w*	Waga krawedzi.
in	*n*	Miejsce, w ktore ma byc wstawiony element item.

Implements ILista< Type >.

Here is the caller graph for this function:



5.15.3.2 template<class Type> virtual Type Lista< Type >::get (int *n*) [inline], [virtual]

Zwraca element z dowolnego miejsca listy.

Zwraca element z miejsca wskazywanego przez zmienna index. Wyjatki sa typu: const char * "Empty list" - pusta lista "Index out of bounds" - przekroczono zakres, nie ma tylu elementow

Returns

Zwraca element typu std::string.

Implements ILista< Type >.

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

Implements ILista< Type >.

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

Sprawdza czy lista jest pusta.

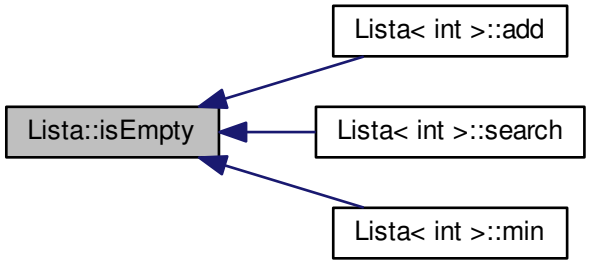
Sprawdza czy w liscie sa jakies elementy.

Return values

true	Lista jest pusta.
false	Lista nie jest pusta.

Implements ILista< Type >.

Here is the caller graph for this function:



5.15.3.5 template<class Type> Type Lista< Type >::min (Lista< Type > *other_than*) [inline]

Choose min other than specified as an argument.

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

Wypisuje zawartosc listy.

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

5.15.3.7 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 >.

5.15.3.8 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

in	*searched_word*	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 caller graph for this function:

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

Zwraca rozmiar listy.

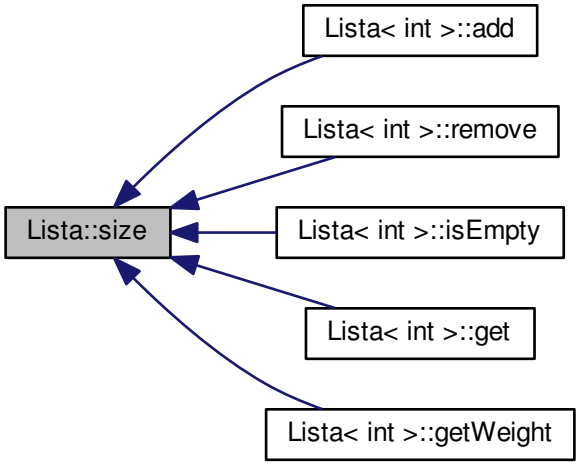
Zwraca ilosc elementow w liscie.

Returns

Rozmiar listy.

Implements ILista< Type >.

Here is the caller graph for this function:



5.15.4 Member Data Documentation

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

Pierwszy element listy.

Wskazuje na pierwszy element listy.

5.15.4.2 template<class Type> int Lista< Type >::size_of_list [private]

Przechowuje rozmiar listy.

Dzieki zastosowaniu tej zmiennej, o wiele latwiej debugowac Lista. Pozwala to na kontrole mechanizmow sprawdzania. Powinien byc zawsze dodatni.

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

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

5.16 ListNode Struct Reference

`#include <IGraph.h>`

Public Member Functions

- ListNode (int y, int weight)

Public Attributes

- int key

- int w

5.16.1 Constructor & Destructor Documentation

5.16.1.1 ListNode::ListNode (int *y*, int *weight*) [inline]

5.16.2 Member Data Documentation

5.16.2.1 int ListNode::key

5.16.2.2 int ListNode::w

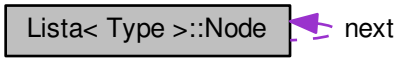
The documentation for this struct was generated from the following file:

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

5.17 Lista< Type >::Node Struct Reference

Imlementacja wezlow dla listy.

Collaboration diagram for Lista< Type >::Node:



Public Attributes

- Type element

  Element w wezle.
- int weight

  Waga krawedzi.
- Node * next

  Wskaznik na nastepny wezel.

5.17.1 Detailed Description

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

Imlementacja wezlow dla listy.

Potrzebne do implementacji interfejsu listy. Zawiera pole typu string.

5.17.2 Member Data Documentation

5.17.2.1 template<class Type> Type Lista< Type >::Node::element

Element w wezle.

Co jest w wezle. Ma przechowywac pojedyncze slowo.

5.17.2.2 template<class Type> Node* Lista< Type >::Node::next

Wskaznik na nastepny wezel.

Wskazuje na nastepny wezel.

5.17.2.3 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:

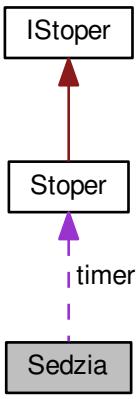
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Lista.h

5.18 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

5.18.1 Detailed Description

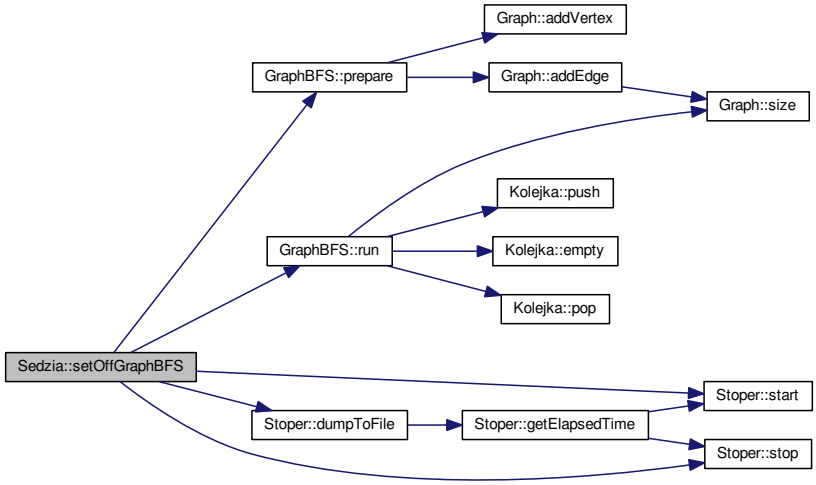
Implementacja klasy Sedzia.

Sedzia wykorzystuje elementy klasy Stoper oraz klasy Tablica. Mierzy czas wypelniania elemntow Tablicy.

5.18.2 Member Function Documentation

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

Here is the call graph for this function:



Here is the caller graph for this function:

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

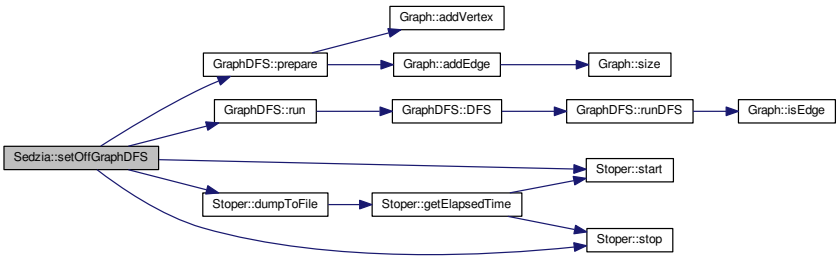
Funkcja, gdzie odbywa sie zapis phonebook'a do tablicy haszowej.

Podczas wykonywania tej funkcji uruchamiany jest Stoper oraz wypelniany jest element klasy HashTable po uprzednim jej przygotowaniu. Slowa pobiera z tego samego slownika co lista.

Parameters

in	*how_many*	Informacja iloma elementami ma zostac wypelniona tablica.

Here is the call graph for this function:



Here is the caller graph for this function:

5.18.3 Member Data Documentation

5.18.3.1 Stoper Sedzia::timer [private]

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

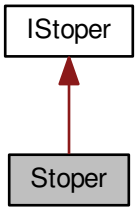
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Sedzia.h
- /home/kkuczaj/PAMSI/lab08_25.04/prj/src/Sedzia.cpp

5.19 Stoper Class Reference

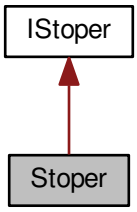
Implementacja klasy Stoper.

`#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

5.19.1 Detailed Description

Implementacja klasy Stoper.

W klasie Stoper zostaly zaimplemetowane metody pozwalajace na pomiar czasu. Pomiar czasu odbywa sie dzieki bibliotece <sys/time.h> a zapis do pliku korzysta z biblioteki <fstream>.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 Stoper::Stoper ( )

Konstruktor bezparametryczny.

Alokuje pamiec dla pol, poniewaz sa wskaznikami.

5.19.2.2 Stoper::~Stoper ( )

Destruktor.

Zwalniam pamiec po polach.

5.19.3 Member Function Documentation

5.19.3.1 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 wyjsciowym (out). Plik .csv to tzw. Comma-Separated Values - latwo je potem zaimportowac do arkusza kalkulacyjnego oraz sa zgodne z ogolno przyjetym standardem.

Parameters

file_name	Nazwa pliku, do ktorego beda zapisane dane. Nazwa nie 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:

5.19.3.2 double Stoper::getElapsedTime ( ) [virtual]

Implementacja funkcji getElapse() z interfejsu IStoper.

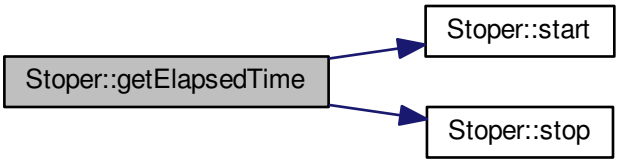
Oblicza czas pomiedzy czasem zapisanym w zmiennych start_time i stop_time.

Returns

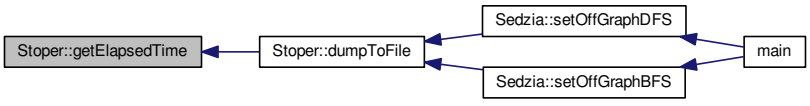
Zwraca zmierzony czas - roznica pomiedzy polem start_time a polem stop_time. Zwraca wynik w mikrosekundach.

Implements IStoper.

Here is the call graph for this function:



Here is the caller graph for this function:



5.19.3.3 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:

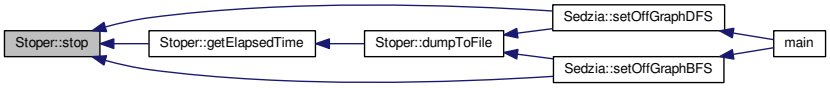
5.19.3.4 void Stoper::stop ( ) [virtual]

Implementacja funkcji stop() z interfejsu IStoper.

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

Implements IStoper.

Here is the caller graph for this function:



5.19.4 Member Data Documentation

5.19.4.1 timeval* Stoper::start_time [private]

Moment startu stopera.

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

5.19.4.2 timeval* Stoper::stop_time [private]

Moment zatrzymania stopera.

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

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

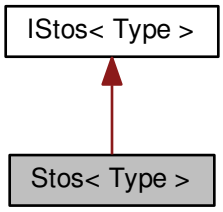
- /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Stoper.h
- /home/kkuczaj/PAMSI/lab08_25.04/prj/src/Stoper.cpp

5.20 Stos< Type > Class Template Reference

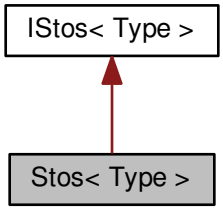
Implementacja klasy Stos, zlozonej z intow.

`#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

5.20.1 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.

5.20.2 Constructor & Destructor Documentation

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

Bezparametryczny konstruktor.

Inicjalizuje wierzcholek *top jak wskaznik na NULL.

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

Destruktor.

Popuje wszystkie elementy.

5.20.3 Member Function Documentation

5.20.3.1 template<class Type> virtual bool Stos< Type >::empty ( ) [inline], [virtual]

Sprawdza czy pojemnika jest pusty.

Sprawdza czy znajduja sie jakies elementy w pojemniku.

Return values

true	Pojemnik pusty.
false	Pojemnik nie jest pusty.

Implements IStos< Type >.

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

Usuwa element z pojemnika.

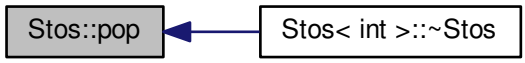
Usuwa element z pojemnika i zwraca go uzytkownikowi.

Returns

Usuniety element.

Implements IStos< Type >.

Here is the caller graph for this function:



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

Wyswietla elementy stosu.

Wyswietla cala zawartosc stosu. Nie jest czescia interfesju.

5.20.3.4 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

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

Implements IStos< Type >.

5.20.3.5 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 >.

5.20.4 Member Data Documentation

5.20.4.1 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

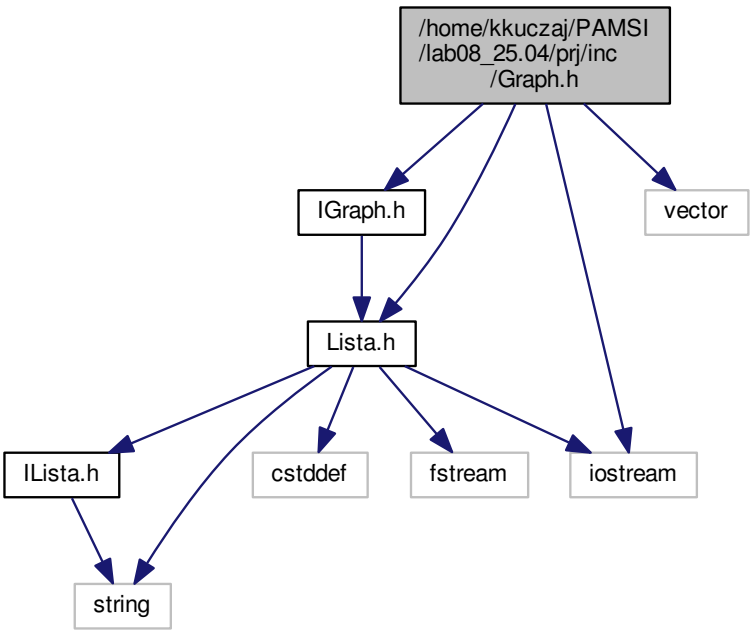
Chapter 6

File Documentation

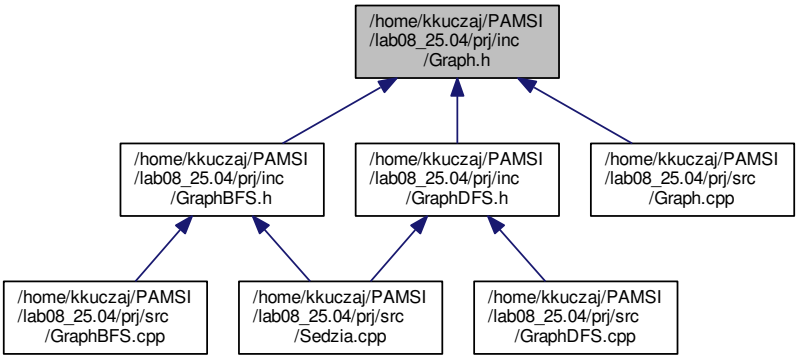
6.1 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Graph.h File Reference

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

Include dependency graph for Graph.h:



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



Classes

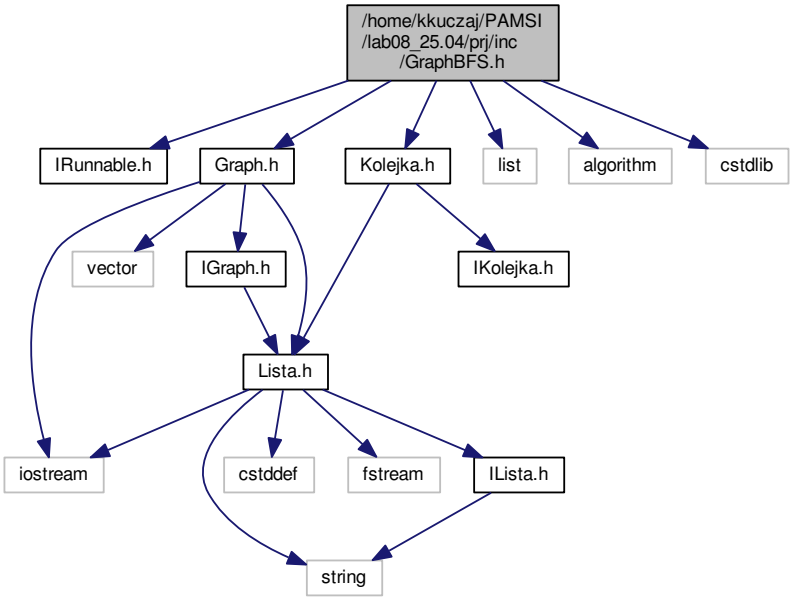
- class Graph

  Graf oparty o liste sasiedztwa.

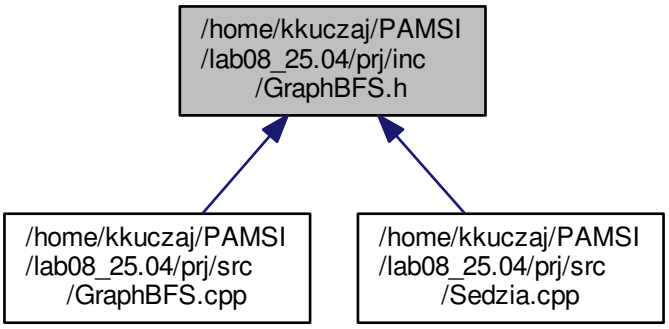
6.2 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/GraphBFS.h File Reference

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

Include dependency graph for GraphBFS.h:



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



Classes

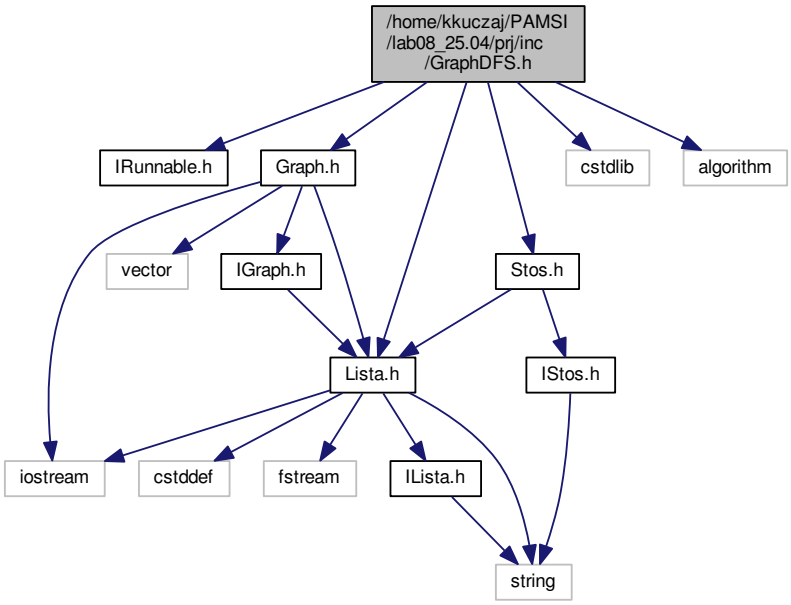
- class GraphBFS

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

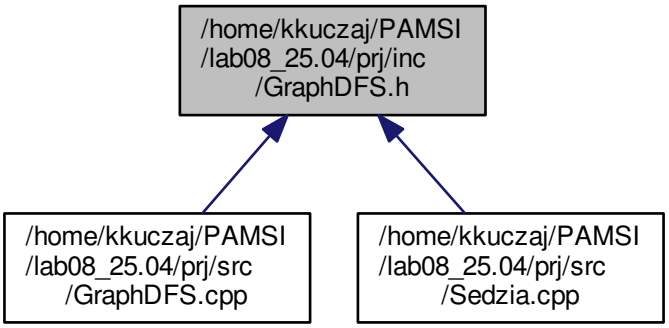
```
#include "IRunnable.h"
```

```
#include "Graph.h"
#include "Stos.h"
#include "Lista.h"
#include <cstdlib>
#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/IBinaryTree.h File Reference

Classes

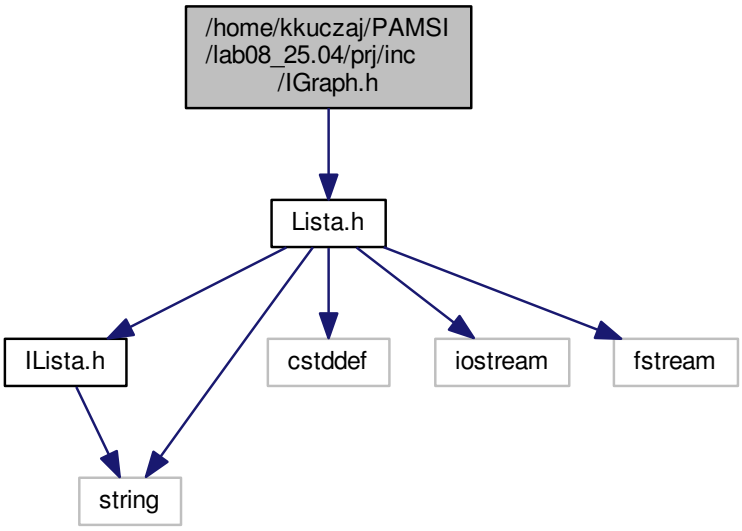
- class IBinaryTree< Type >

  Interfejs drzewa binarnego.

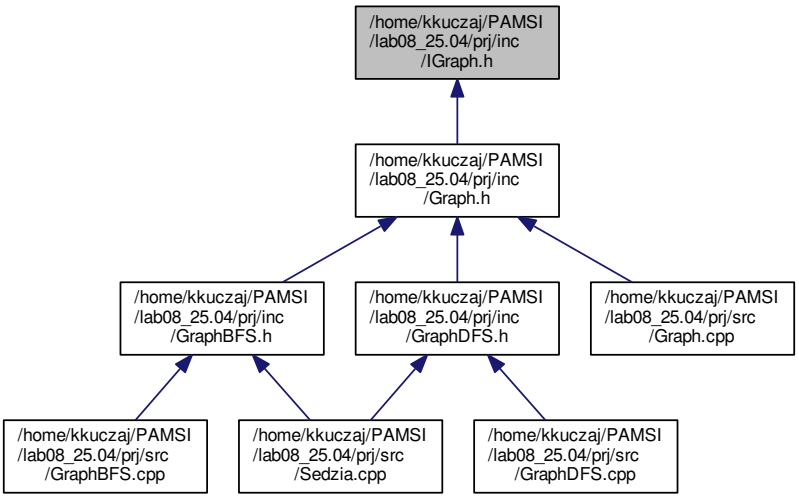
6.5 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/IGraph.h File Reference

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

Include dependency graph for IGraph.h:



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

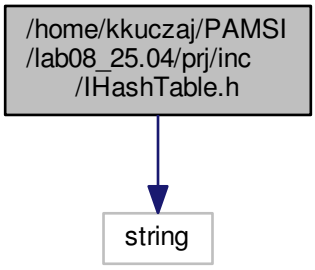


Classes

- struct ListNode
- class IGraph

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

`#include <string>`
Include dependency graph for IHashTable.h:



Classes

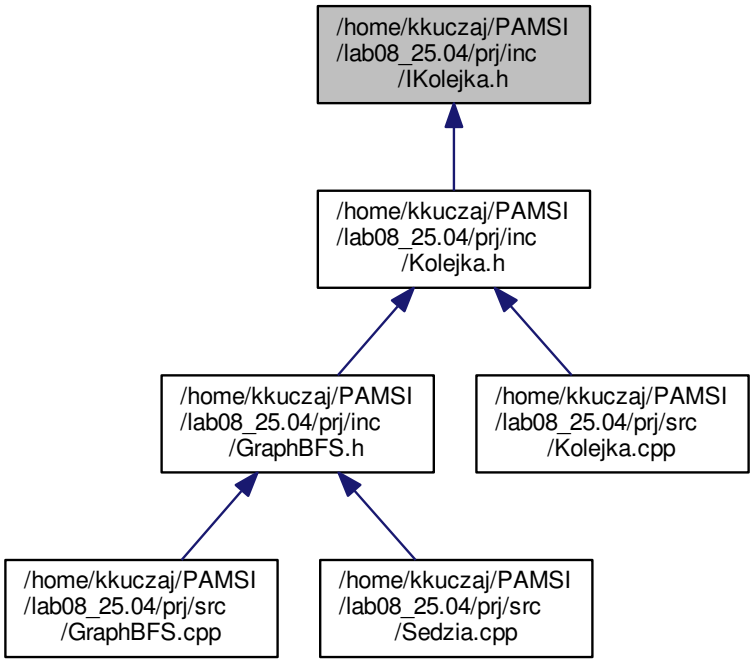
- class IHashTable

  Interfejs tablicy z hashowaniem.

6.7 /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.

6.7.1 Detailed Description

Plik zawiera interfejs dla pojemnika Kolejka. Nie zdecydowano sie na uzycie szablonow, gdyz zbyt komplikuje to budowe programu.

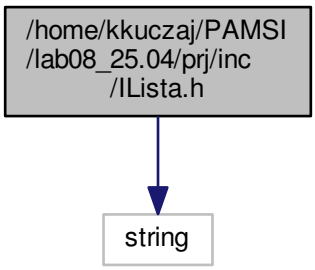
Author

- Kamil Kuczaj.

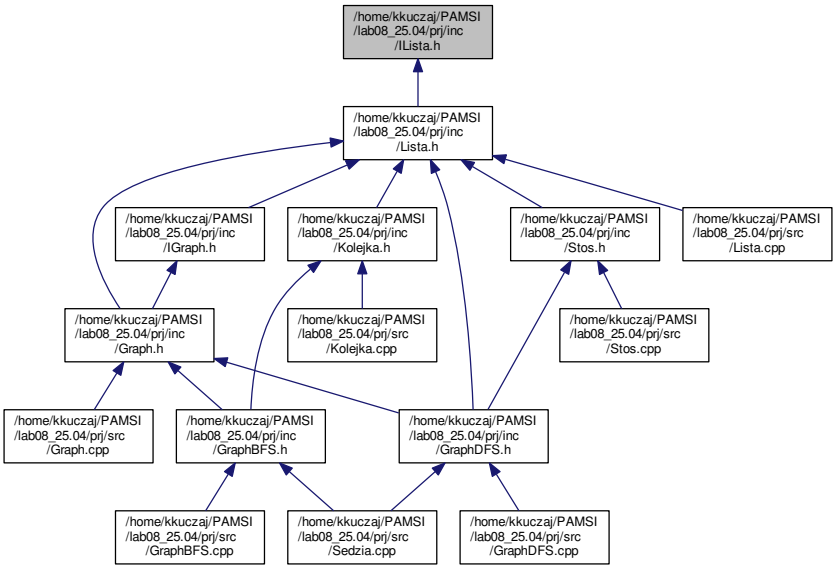
6.8 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/ILista.h File Reference

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.

6.8.1 Detailed Description

Plik zawiera interfejs dla pojemnika Lista oraz dla klasy Wezel. Wezel jest elementem listy. Uzycie szablonow zbytnio komplikuje implementacje, wiec odrzucilem ich zastosowanie.

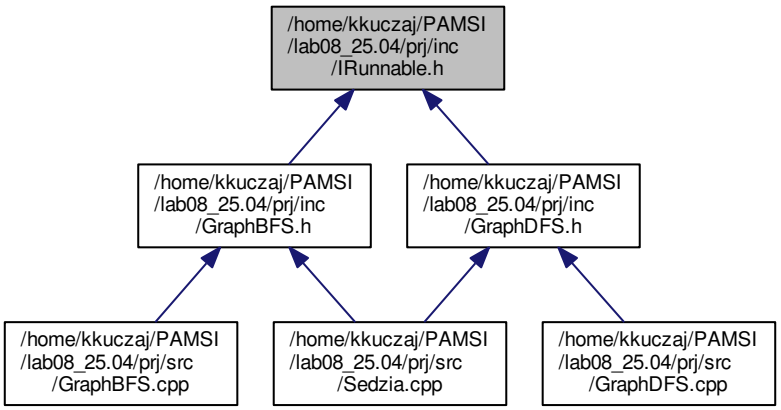
Author

Kamil Kuczaj.

6.9 /home/kkuczaj/PAMSI/lab08_25.04/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.

6.9.1 Detailed Description

Naglowek zawierajacy interfejs dla biegacza.

Author

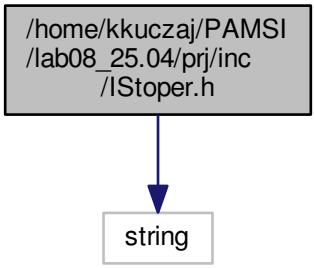
Kamil Kuczaj

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

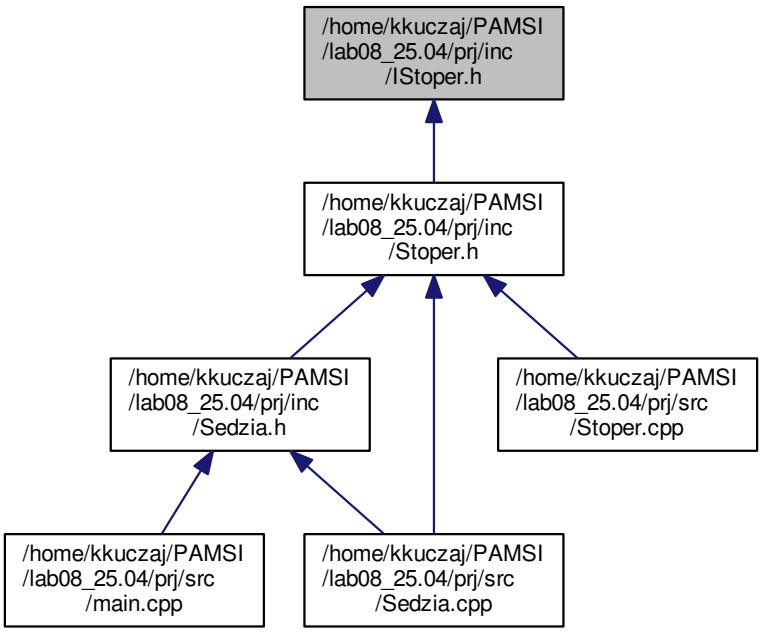
Naglowek zawierajacy interfejs dla stopera.

```
#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.

6.10.1 Detailed Description

Naglowek zawierajacy interfejs dla stopera.

Author

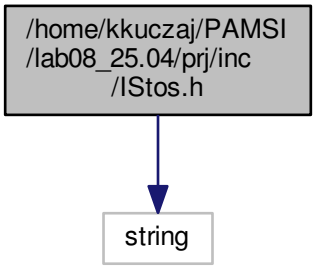
Kamil Kuczaj

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

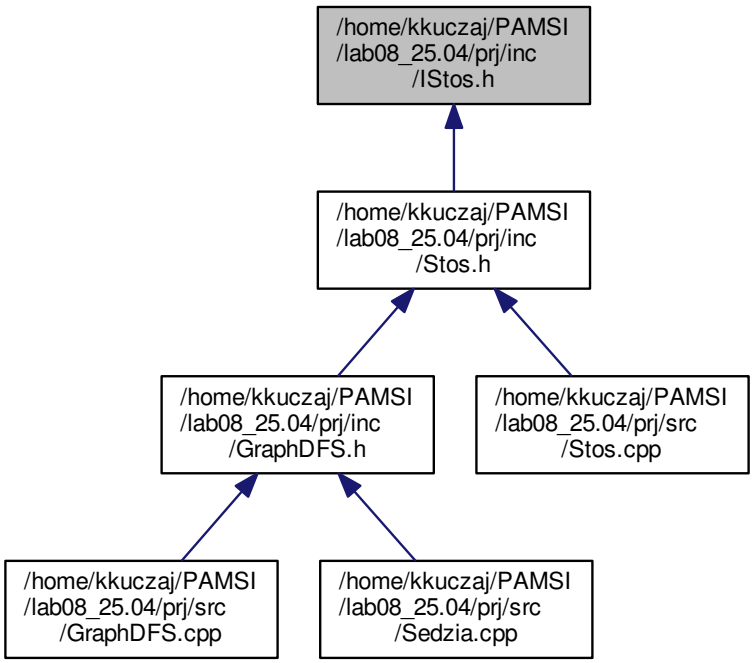
Plik zawiera interfejs dla pojemnika Stos.

```
#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.

6.11.1 Detailed Description

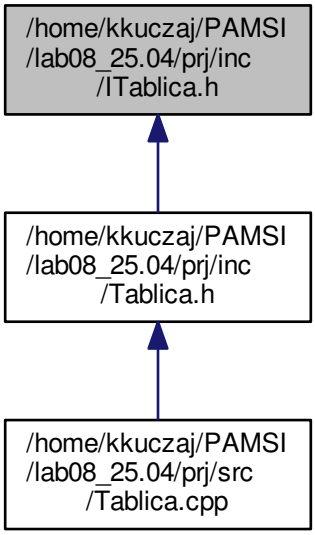
Plik zawiera interfejs dla pojemnika Stos. Nie zdecydowano sie na uzycie szablonow, gdyz zbyt komplikuje to budowe programu.

Author

Kamil Kuczaj.

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

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



Classes

- class ITablica< Type >

  Interfejs tablicy.

6.12.1 Detailed Description

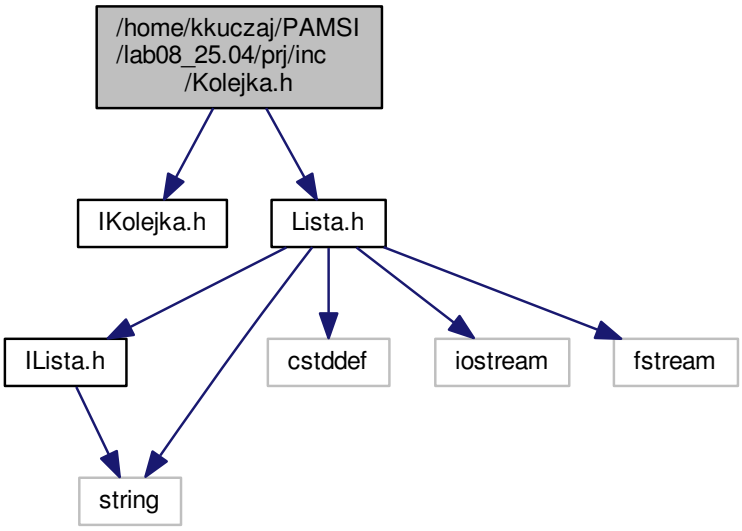
Author

Kamil Kuczaj

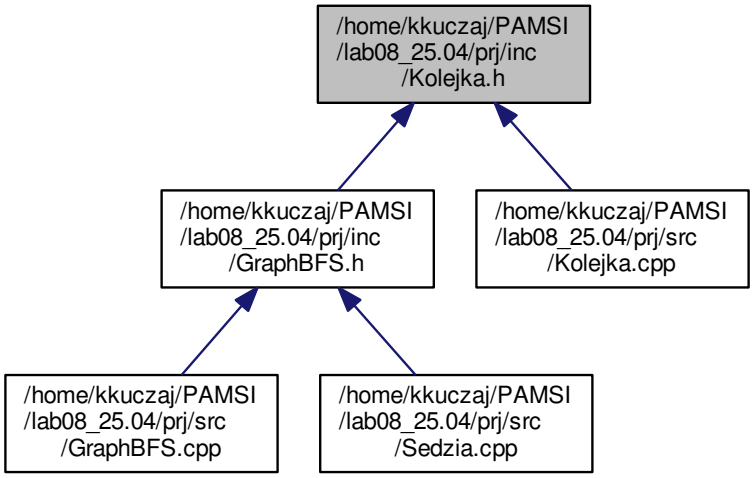
6.13 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Kolejka.h File Reference

```
#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 >

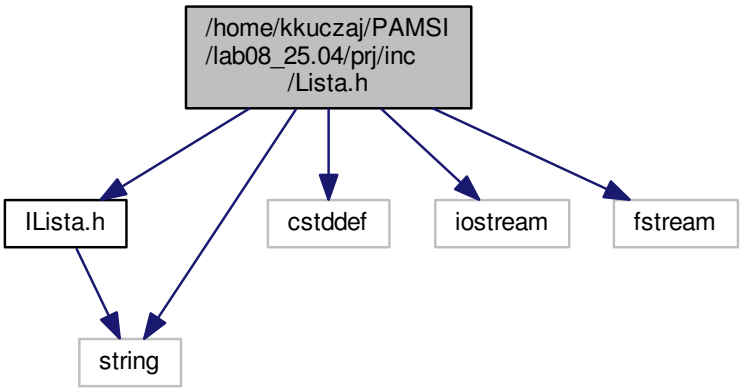
  Implementacja interfejsu IKolejka w postaci klasy Kolejka.

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

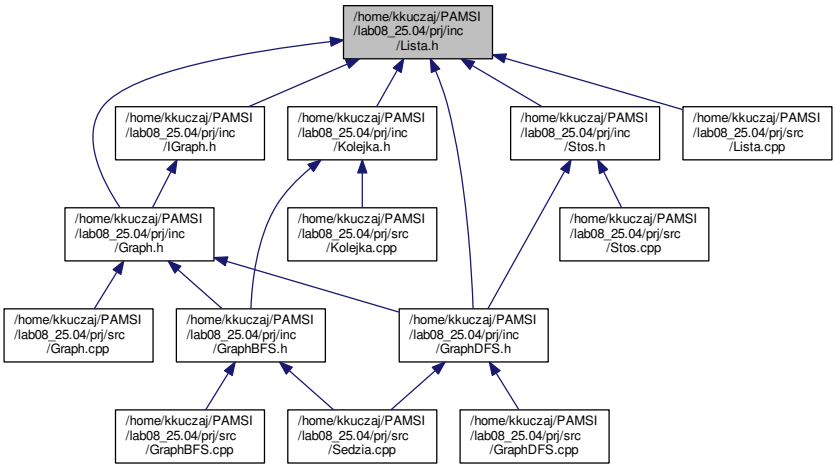
Implementacja jednokierunkowej listy.

```
#include "ILista.h"
#include <cstddef>
#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, ktora symuluje zachowanie klasy list z biblioteki STL.
- struct Lista< Type >::Node

  Imlementacja wezlow dla listy.

6.14.1 Detailed Description

Implementacja jednokierunkowej listy. Ze wzledu na komplikacje implementacji mechanizmow przy uzyciu szablonow, zdecydowalem sie je usunac z konstrukcji programu.

Author

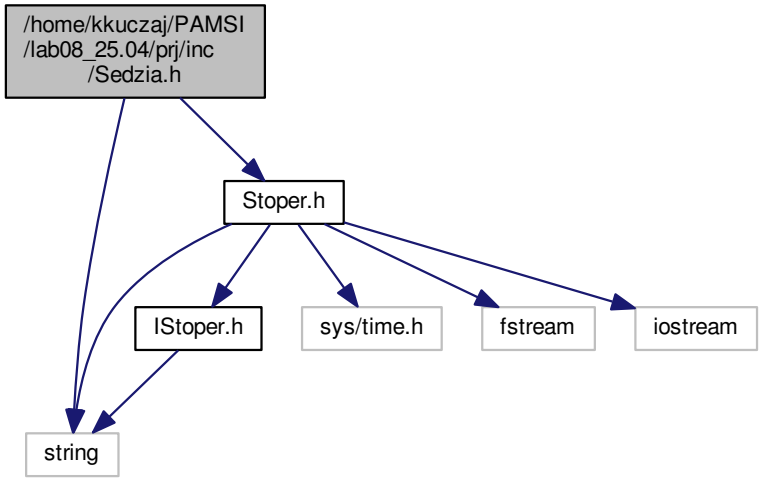
Kamil Kuczaj.

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

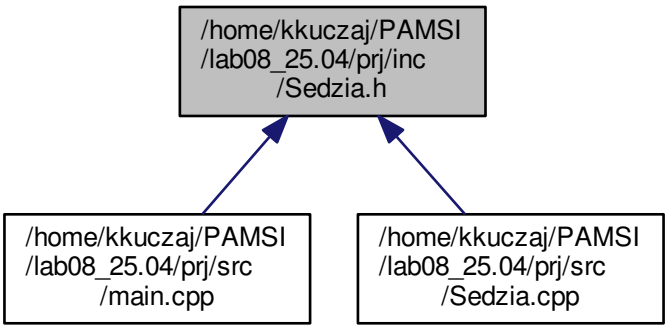
Naglowek opisujacy implementacje Sedziego.

```
#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.

6.15.1 Detailed Description

Naglowek opisujacy implementacje Sedziego.

Author

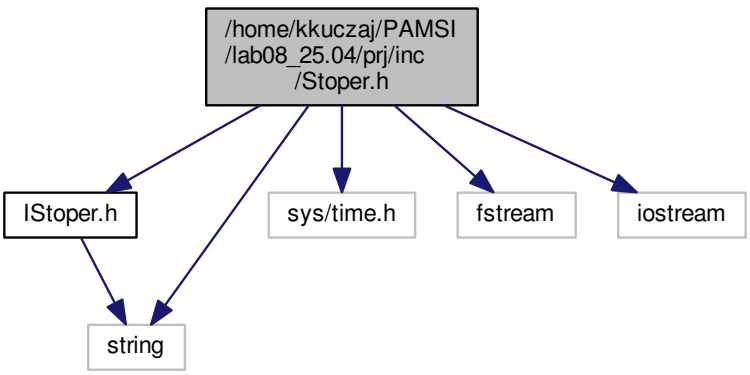
Kamil Kuczaj

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

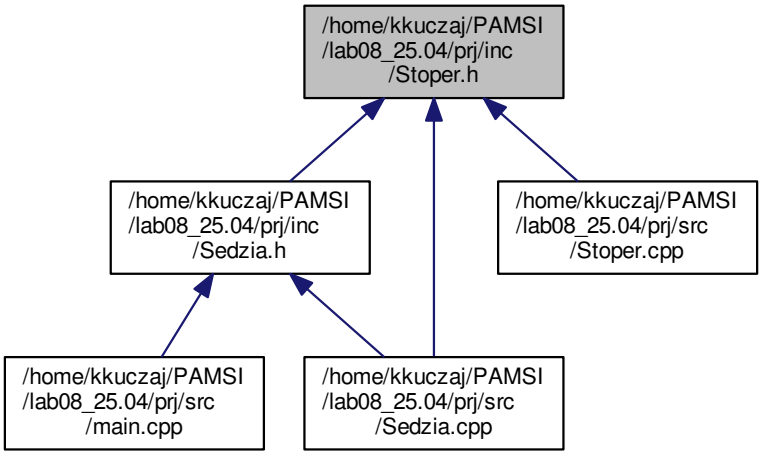
Implementacja interfejsu IStoper w klasie Stoper.

```
#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.

6.16.1 Detailed Description

Implementacja interfejsu IStoper w klasie Stoper.

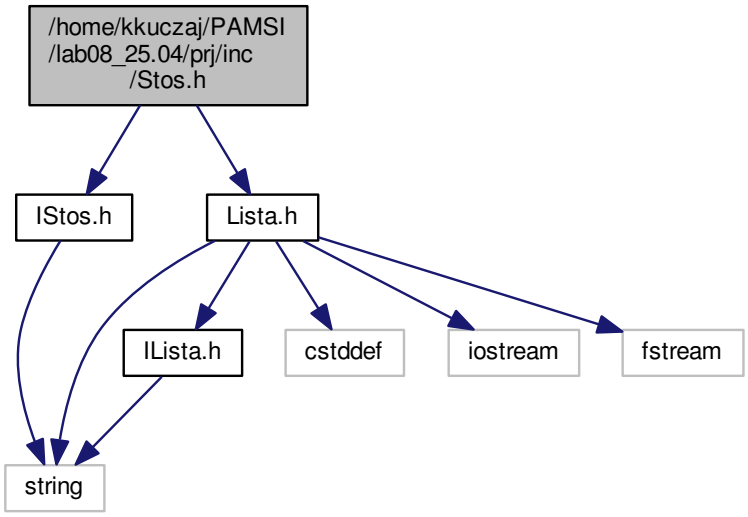
Author

Kamil Kuczaj

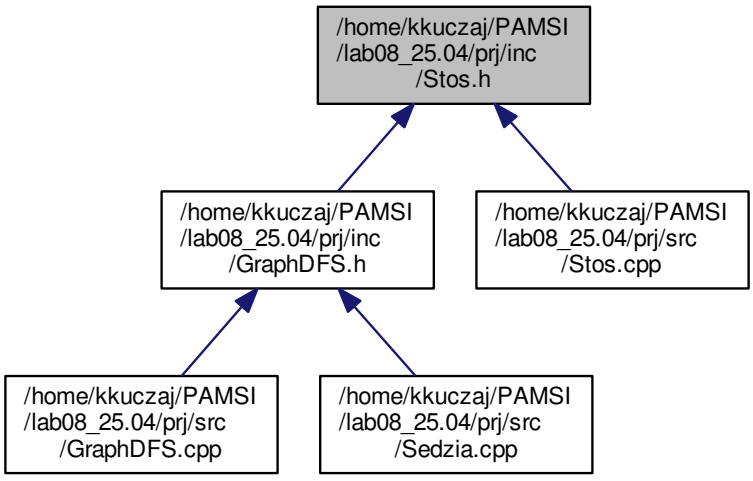
6.17 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Stos.h File Reference

```
#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 >

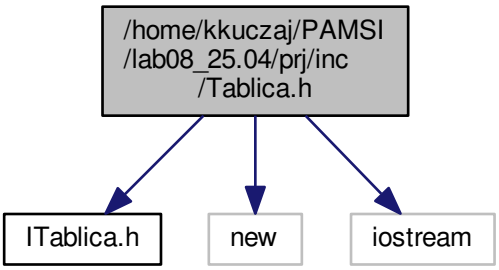
  Implementacja klasy Stos, zlozonej z intow.

6.18 /home/kkuczaj/PAMSI/lab08_25.04/prj/inc/Tablica.h File Reference

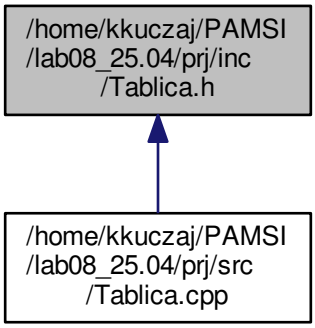
Implementacja interfesju ITablca. Po konsultacji z prowadzacym zdecydowalem sie nie wykorzystywac szablonow.

```
#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 Array< Type >

  Klasa Tablica, w ktorej odbywa sie zapis dynamiczny elementow.

6.18.1 Detailed Description

Implementacja interfesju ITablca. Po konsultacji z prowadzacym zdecydowalem sie nie wykorzystywac szablonow.

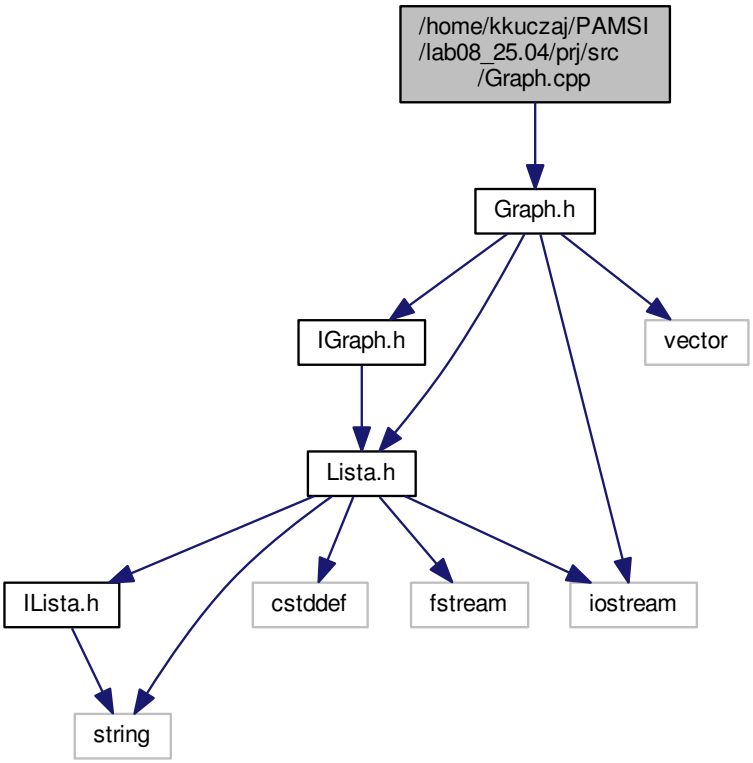
Author

Kamil Kuczaj

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

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

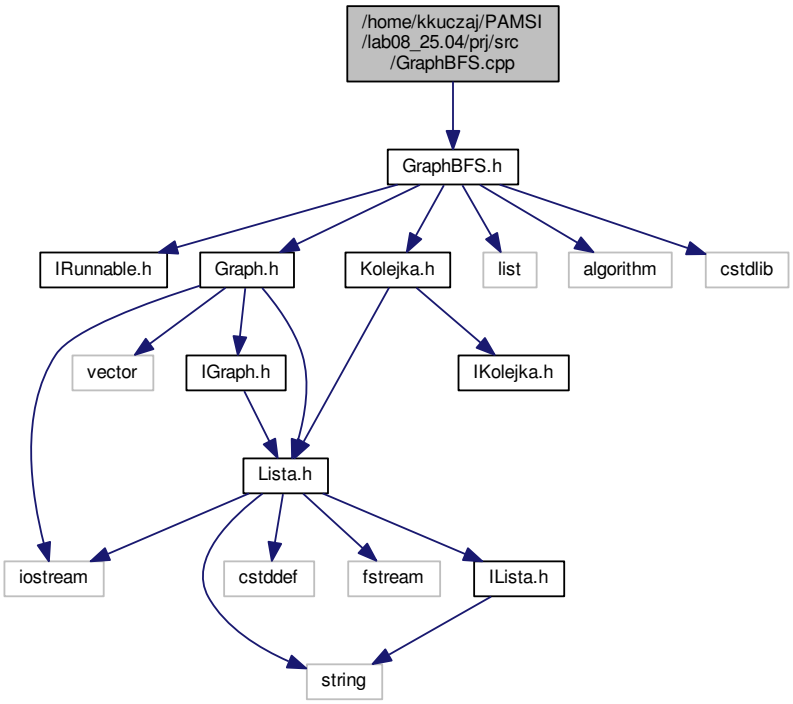
Include dependency graph for Graph.cpp:



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

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

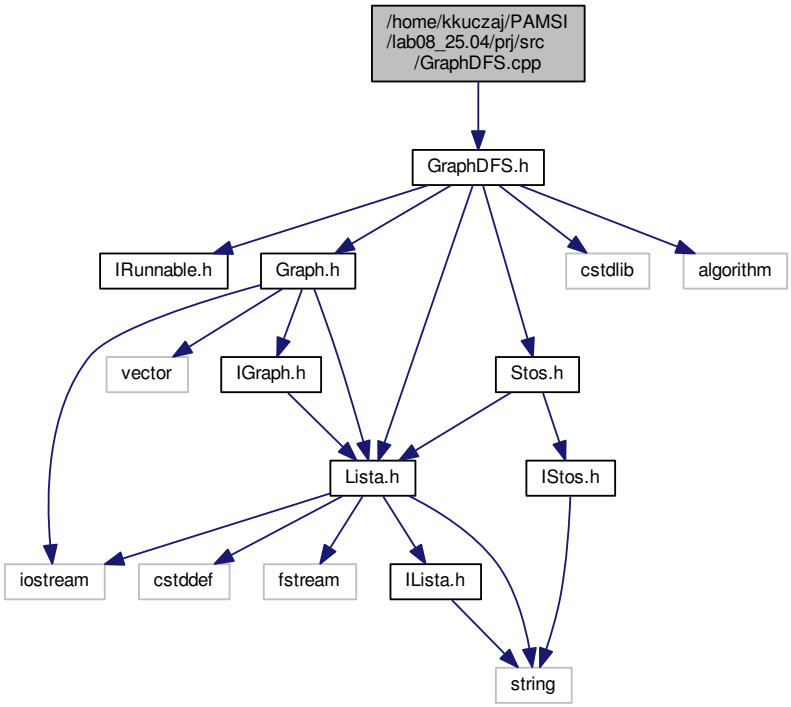
Include dependency graph for GraphBFS.cpp:



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

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

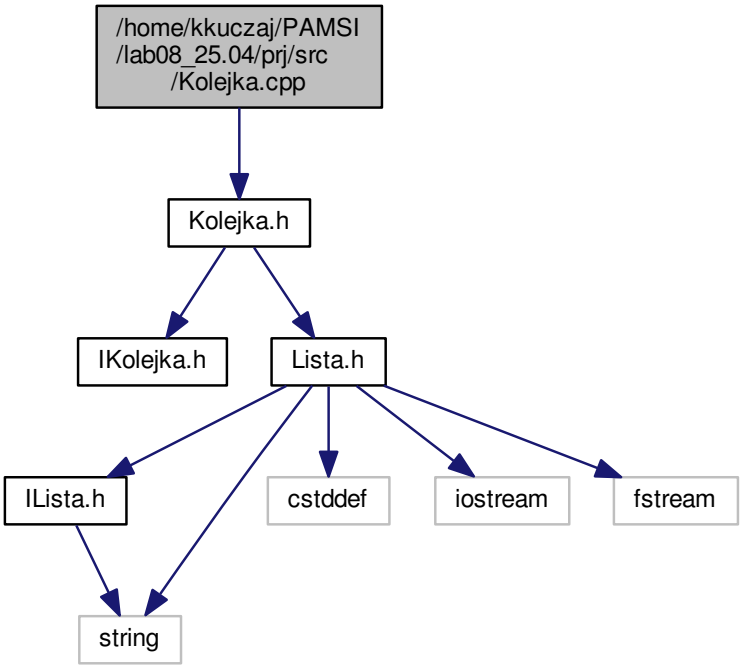
Include dependency graph for GraphDFS.cpp:



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

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

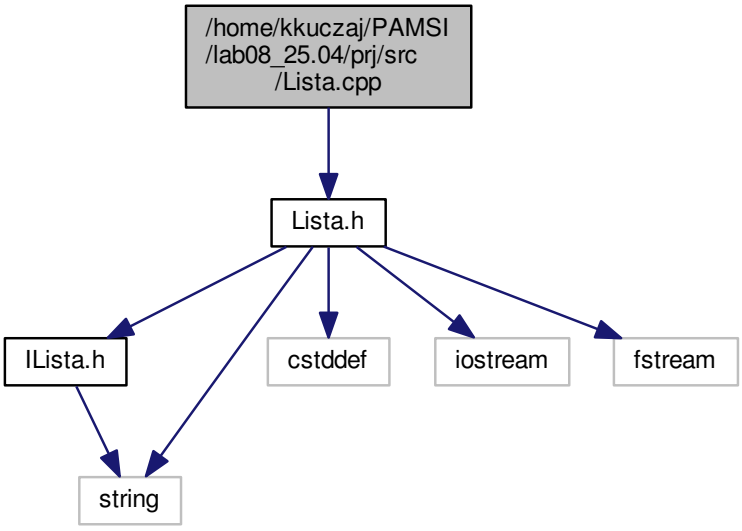
Include dependency graph for Kolejka.cpp:



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

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

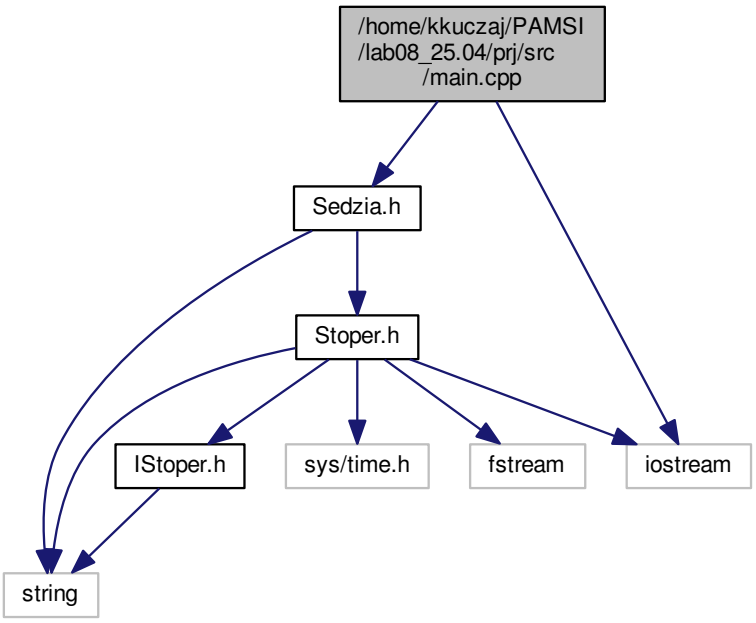
Include dependency graph for Lista.cpp:



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

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

Include dependency graph for main.cpp:



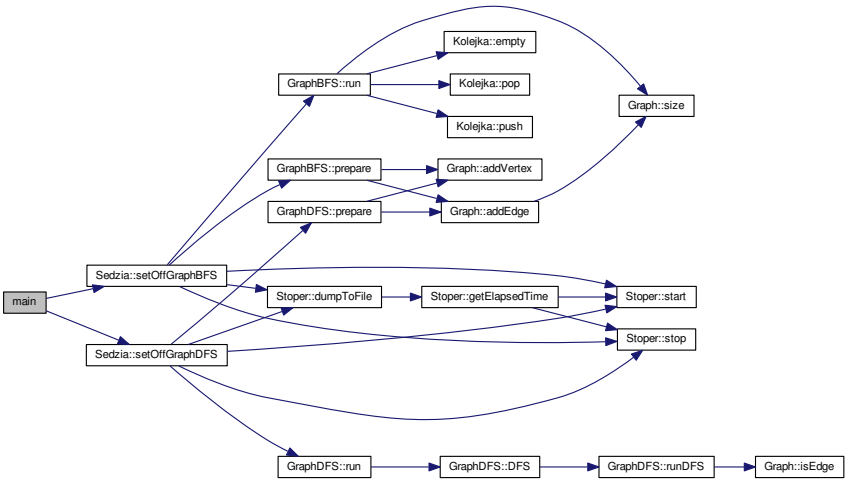
Functions

- int main (int argc, char **argv)

6.24.1 Function Documentation

6.24.1.1 int main (int *argc*, char ** *argv*)

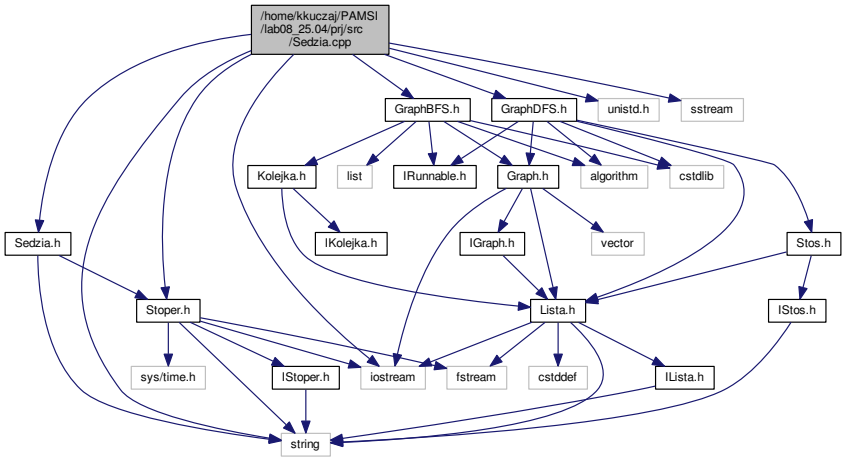
Here is the call graph for this function:



6.25 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/Sedzia.cpp File Reference

```
#include "Sedzia.h"
#include "Stoper.h"
#include "GraphBFS.h"
#include "GraphDFS.h"
#include <unistd.h>
#include <sstream>
#include <string>
#include <iostream>
```

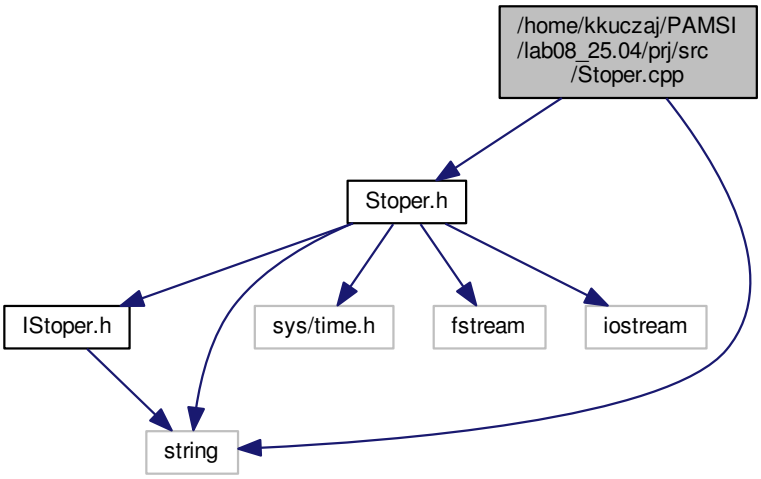
Include dependency graph for Sedzia.cpp:



6.26 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/Stoper.cpp File Reference

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

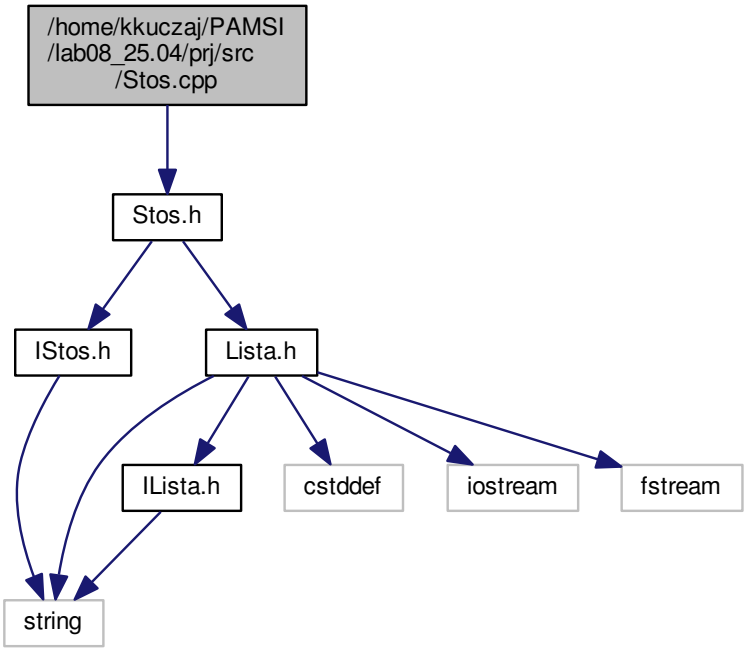
Include dependency graph for Stoper.cpp:



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

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

Include dependency graph for Stos.cpp:



6.28 /home/kkuczaj/PAMSI/lab08_25.04/prj/src/Tablica.cpp File Reference

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

Include dependency graph for Tablica.cpp: