Pomiar czasu przeszukania listy jednokierunkowej

Generated by Doxygen 1.8.6

Sun Mar 20 2016 15:56:27

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 jądro 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 ILista< ListType > Class Template Reference

Interfejs dla pojemnika Lista.

`#include <ILista.h>`

Protected Member Functions

- virtual void add (ListType item, uint index)=0

  Wstawia element w dowolnym miejscu listy.
- virtual bool remove (uint index)=0

  Usuwa element z dowolnego miejsca listy.
- virtual bool isEmpty ()=0

  Sprawdza czy lista jest pusta.
- virtual ListType get (uint index)=0

  Zwraca element z dowolnego miejsca listy.
- virtual uint size ()=0

  Zwraca rozmiar listy.

5.1.1 Detailed Description

template<class ListType>class ILista< ListType >

Interfejs dla pojemnika Lista.

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

5.1.2 Member Function Documentation

5.1.2.1 template<class ListType> virtual void ILista< ListType >::add (ListType *item,* uint *index*) [protected], [pure virtual]

Wstawia element w dowolnym miejscu listy.

Wstawia element typu Type w miejsce wskazywane przez zmienna index.

Parameters

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

Implemented in Lista< Type >.

5.1.2.2 template<class ListType> virtual ListType IListA< ListType >::get (uint *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 >.

5.1.2.3 template<class ListType> virtual bool IListA< ListType >::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 >.

5.1.2.4 template<class ListType> virtual bool IListA< ListType >::remove (uint *index*) [protected], [pure virtual]

Usuwa element z dowolnego miejsca listy.

Usuwa element z miejsca wskazywanego przez zmienna index.

Returns

Zwraca element typu Type.

Implemented in Lista< Type >.

5.1.2.5 template<class ListType> virtual uint IListA< ListType >::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:

- inc/IListA.h

5.2 IPojemnik< Type > Class Template Reference

Interfejs dla każdego pojemnika.

`#include <IPojemnik.h>`

Protected Member Functions

- virtual Type add (Type item, uint index)=0

  Dodaje element w okreslonym miejscu.
- virtual Type push (uint index)=0

  Usuwa element z okreslonego miejsca.
- virtual Type pop (uint index)=0

  Usuwa element z pojemnika.
- virtual bool empty ()=0

  Sprawdza czy pojemnika jest pusty.
- virtual uint size ()=0

  Zwraca aktualny rozmiar pojemnika.

5.2.1 Detailed Description

template<class Type>class IPojemnik< Type >

Interfejs dla każdego pojemnika.

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

5.2.2 Member Function Documentation

5.2.2.1 template<class Type > virtual Type IPojemnik< Type >::add (Type *item*, uint *index*) [protected], [pure virtual]

Dodaje element w okreslonym miejscu.

Metoda czysto wirtualna. Dodaje element item w miejscu index pojemnika.

Parameters

in	*item*	Dana, ktora ma byc wlozona.
in	*index*	Indeks, w ktorym ma znalezc sie nowa dana.

5.2.2.2 template<class Type > virtual bool IPojemnik< 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.

5.2.2.3 template<class Type > virtual Type IPojemnik< Type >::pop (uint *index*) [protected], [pure virtual]

Usuwa element z pojemnika.

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

Returns

Usuniety element.

5.2.2.4 template<class Type > virtual Type IPojemnik< Type >::push (uint *index*) [protected], [pure virtual]

Usuwa element z okreslonego miejsca.

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

Parameters

in	*index*	Indeks,z ktorego ma zostac usunieta dana.

5.2.2.5 template<class Type > virtual uint IPojemnik< 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.

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

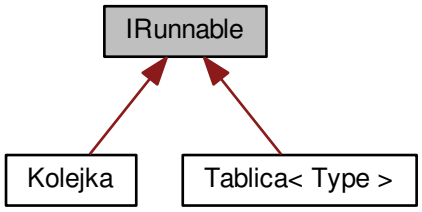
- inc/IPojemnik.h

5.3 IRunnable Class Reference

Interfejs dla biegacza.

`#include <IRunnable.h>`

Inheritance diagram for IRunnable:



Protected Member Functions

- virtual void prepare (unsigned int size)=0
- virtual void run ()=0

5.3.1 Detailed Description

Interfejs dla biegacza.

Klasa abstrakcyjna z metodami czysto wirtualnymi.

5.3.2 Member Function Documentation

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

Implemented in Tablica< Type >.

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

Implemented in Tablica< Type >.

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

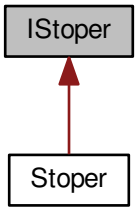
- inc/IRunnable.h

5.4 IStoper Class Reference

Interfejs dla stopera.

`#include <IStoper.h>`

Inheritance diagram for IStoper:



Protected Member Functions

- virtual void start ()=0
- virtual void stop ()=0
- virtual double getElapsedTime ()=0
- virtual void dumpToFile (std::string file_name)=0

5.4.1 Detailed Description

Interfejs dla stopera.

Klasa abstrakcyjna z metodami czysto wirtualnymi.

5.4.2 Member Function Documentation

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

Implemented in Stoper.

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

Implemented in Stoper.

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

Implemented in Stoper.

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

Implemented in Stoper.

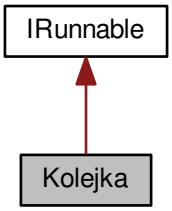
The documentation for this class was generated from the following file:

- inc/IStoper.h

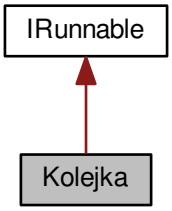
5.5 Kolejka Class Reference

`#include <Kolejka.h>`

Inheritance diagram for Kolejka:



Collaboration diagram for Kolejka:



Additional Inherited Members

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

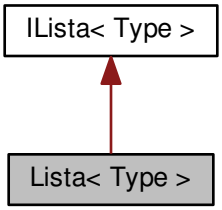
- inc/Kolejka.h

5.6 Lista< Type > Class Template Reference

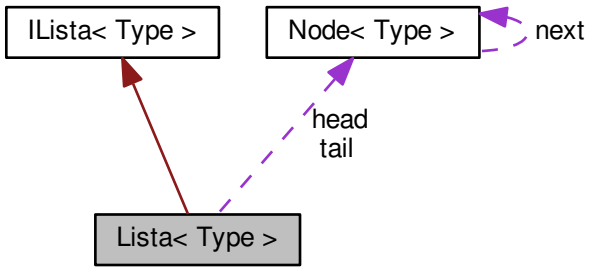
Klasa Lista, w ktorej odbywa sie zapis dynamiczny elementow typu int.

`#include <Lista.h>`

Inheritance diagram for Lista< Type >:



Collaboration diagram for Lista< Type >:



Public Member Functions

- Lista ()

  Konstruktor.
- ~Lista ()

  Destruktor.
- virtual void add (Type item, uint index)

  Wstawia element w dowolnym miejscu listy.
- virtual bool remove (uint index)

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

  Sprawdza czy lista jest pusta.
- virtual Type get (uint index)

  Zwraca element z dowolnego miejsca listy.
- virtual uint size ()

  Zwraca rozmiar listy.
- virtual uint run (Type desired_element)

  Szuka elementu.
- virtual void prepare (uint desired_size)

Zapisuje liste slowami.

- void print ()

  Wypisuje zawartosc listy.

Private Attributes

- Node< Type > * head

  Pierwszy element listy.

- Node< Type > * tail

  Ostatni element listy.

- uint size_of_list

  Przechowuje rozmiar listy.

Additional Inherited Members

5.6.1 Detailed Description

template<class Type>class Lista< Type >

Klasa Lista, w ktorej odbywa sie zapis dynamiczny elementow typu int.

Implementuje metody interfejsu IRunnable. Zajmuje sie dynamiczna alokacja pamieci. Lista jest dwukierunkowa oraz posiada mechanizm, dzieki ktoremu mamy dostep zarowno do pierwszego jak i ostatniego elementu.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 template<class Type > Lista< Type >::Lista ( )

Konstruktor.

Tworzy poczatek listy.

5.6.2.2 template<class Type > Lista< Type >::~Lista ( )

Destruktor.

Usuwa cala pamiec listy "skaczac" po jej elementach.

5.6.3 Member Function Documentation

5.6.3.1 template<class Type > void Lista< Type >::add (Type *item*, uint *index*) [virtual]

Wstawia element w dowolnym miejscu listy.

Wstawia element typu Type w miejsce wskazywane przez zmienna index.

Parameters

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

Implements ILista< Type >.

5.6.3.2 template<class Type > Type Lista< Type >::get (uint *index*) [virtual]

Zwraca element z dowolnego miejsca listy.

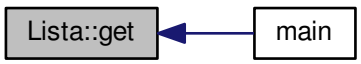
Zwraca element z miejsca wskazywanego przez zmienna index. Wyjatki są typu: const char * "Empty list" - pusta lista "Index out of bounds" - przekroczono zakres, nie ma tylu elementow

Returns

Zwraca element typu Type.

Implements ILista< Type >.

Here is the caller graph for this function:



5.6.3.3 template<class Type > bool Lista< Type >::isEmpty ( ) [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 >.

5.6.3.4 template<class Type > void Lista< Type >::prepare (uint *desired_size*) [virtual]

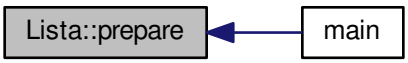
Zapisuje liste slowami.

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

Parameters

in	*desired_size*	Ile elementow ma zostac wczytanych.

Here is the caller graph for this function:

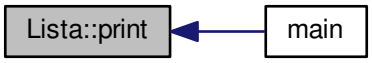


5.6.3.5 template<class Type > void Lista< Type >::print ( )

Wypisuje zawartosc listy.

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

Here is the caller graph for this function:



5.6.3.6 template<class Type > bool Lista< Type >::remove (uint *index*) [virtual]

Usuwa element z dowolnego miejsca listy.

Usuwa element z miejsca wskazywanego przez zmienna index.

Return values

true	Udalo sie usunac element o podanym indeksie.
false	Nie udalo sie usunac elementu o podanym indeksie.

Implements ILista< Type >.

5.6.3.7 template<class Type > uint Lista< Type >::run (Type *desired_element*) [virtual]

Szuka elementu.

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

Parameters

in	*desired_element*	Poszukiwana fraza.

Return values

index>0	Znalazl element i wyswietlil.
1199999999	Nie znalazl i nie wyswietlil elementu. Wybrana wartosc, poniewaz nigdy tak duzej liczby elementow nie wczytamy. 10 cyfr.
1198989898	Lista pusta. Wybrana wartosc, poniewaz nigdy tak duzej liczby elementow nie wczytamy. 10 cyfr.

5.6.3.8 template<class Type > uint Lista< Type >::size ( ) [virtual]

Zwraca rozmiar listy.

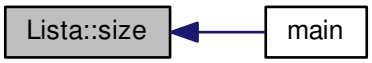
Zwraca ilosc elementow w liscie.

Returns

Rozmiar listy.

Implements ILista< Type >.

Here is the caller graph for this function:



5.6.4 Member Data Documentation

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

Pierwszy element listy.

Wskazuje na pierwszy element listy.

5.6.4.2 template<class Type> uint Lista< Type >::size_of_list [private]

Przechowuje rozmiar listy.

Dzieki zastosowaniu tej zmiennej, o wiele latwiej poslugiwac sie lista.

5.6.4.3 template<class Type> Node<Type>* Lista< Type >::tail [private]

Ostatni element listy.

Wskazuje na ostatni element listy.

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

- inc/Lista.h

5.7 Node< NodeType > Class Template Reference

Imlementacja wezlow dla listy.

`#include <IListа.h>`

Public Member Functions

- NodeType getElem ()
  Dostep do pola element.
- Node getNext ()
  Dostep do nastepnego wezla.
- void setElem (const NodeType t)
  Ustawia pole element.

- void setNext (Node< NodeType > *t)

  Ustawia nastepny wezel.

Public Attributes

- NodeType element

  Element w wezle.
- Node< NodeType > * next

  Wskaznik na nastepny wezel.

Friends

- template<class ListType >
  class ILista

  Zaprzyjaznienie interfejsu ILista.

5.7.1 Detailed Description

template<class NodeType>class Node< NodeType >

Imlementacja wezlow dla listy.

Potrzebne do implementacji interfejsu listy.

5.7.2 Member Function Documentation

5.7.2.1 template<class NodeType> NodeType Node< NodeType >::getElem ( ) [inline]

Dostep do pola element.

Wymuszone poprzez hermetyzacje.

Returns

Zwraca element typu Type.

5.7.2.2 template<class NodeType> Node Node< NodeType >::getNext ( ) [inline]

Dostep do nastepnego wezla.

Wymuszone poprzez hermetyzacje.

Returns

Zwraca element typu Node.

5.7.2.3 template<class NodeType> void Node< NodeType >::setElem (const NodeType *t*) [inline]

Ustawia pole element.

Wymuszone poprzez hermetyzacje.

Parameters

in	*Wartosc,ktora*	ma zostac zapisana do pola element.

5.7.2.4 template<class NodeType> void Node< NodeType >::setNext (Node< NodeType > * *t*) [inline]

Ustawia nastepny wezel.

Wymuszone poprzez hermetyzacje.

Parameters

in	*t*	Wezel, ktory ma zostac przypisany do pola next.

5.7.3 Friends And Related Function Documentation

5.7.3.1 template<class NodeType> template<class ListType > friend class IListа [friend]

Zaprzyjaznienie interfejsu ILista.

Umozliwia dostep do wezlow dla listy.

5.7.4 Member Data Documentation

5.7.4.1 template<class NodeType> NodeType Node< NodeType >::element

Element w wezle.

Co jest w wezle.

5.7.4.2 template<class NodeType> Node<NodeType>* Node< NodeType >::next

Wskaznik na nastepny wezel.

Wskazuje na nastepny wezel.

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

- inc/ILista.h

5.8 Sedzia Class Reference

Implementacja klasy Sedzia.

`#include <Sedzia.h>`

Public Member Functions

- bool setOff (unsigned int how_many)
  Funkcja, w ktorej odbywa sie bieg.

5.8.1 Detailed Description

Implementacja klasy Sedzia.

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

5.8.2 Member Function Documentation

5.8.2.1 bool Sedzia::setOff (unsigned int *how_many*)

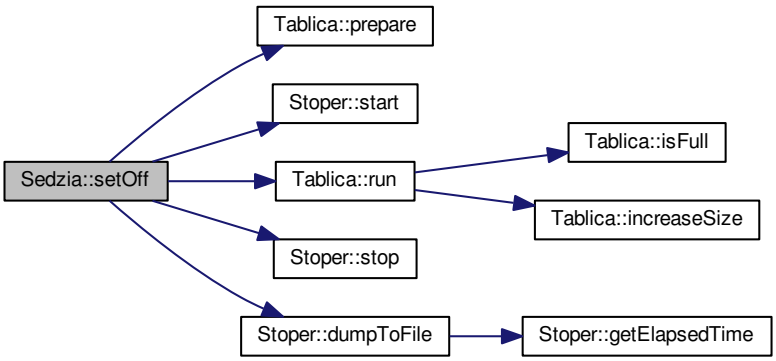
Funkcja, w ktorej odbywa sie bieg.

Podczas wykonywania tej funkcji uruchamiany jest Stoper oraz wypelniany jest element klasy TablicA po uprzednim jej przygotowaniu.

Parameters

how_many	Informacja iloma elementami ma zostac wypelniona tablica.

Here is the call graph for this function:



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

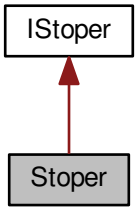
- inc/Sedzia.h
- src/Sedzia.cpp

5.9 Stoper Class Reference

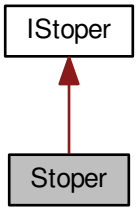
Implementacja klasy Stoper.

`#include <Stoper.h>`

Inheritance diagram for Stoper:



Collaboration diagram for Stoper:



Public Member Functions

- 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

- clock_t start_time

  Moment startu stopera.
- clock_t stop_time

  Moment zatrzymania stopera.
- std::fstream my_file

  Strumien zapisu do pliku.

Additional Inherited Members

5.9.1 Detailed Description

Implementacja klasy Stoper.

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

5.9.2 Member Function Documentation

5.9.2.1 void Stoper::dumpToFile (std::string *file_name*) [virtual]

Implementacja funkcji dumpToFile() z interfejsu IStoper.

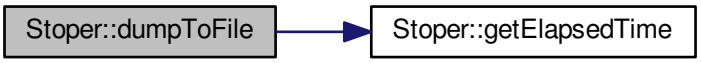
Zapisuje zmierzony czas do pliku o nazwie "${file_name}.csv". Plik otwierany w trybie dopisywania (append) oraz wyjsciowym (out). Plik .csv to tzw. Comma-Separated Values - latwo je potem zaimportowac do arkusza kalkulacyjnego oraz sa zgodne z ogolno przyjetym standardem.

Parameters

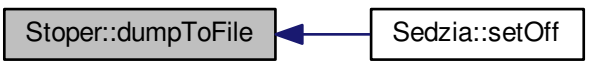
file_name	Nazwa pliku, do ktorego beda zapisane dane. Nazwa nie nie powinna zawierac rozszerzenia. Rozszerzenie jest dodawane w funkcji.

Implements IStoper.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.2 double Stoper::getElapsedTime ( ) [virtual]

Implementacja funkcji getElapse() z interfejsu IStoper.

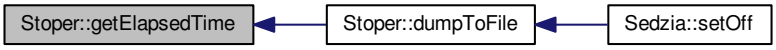
Oblicza czas pomiedzy czasem zapisanym w zmiennych start_time i stop_time.

Returns

Zwraca zmierzony czas - roznica pomiedzy polem start_time a polem stop_time.

Implements IStoper.

Here is the caller graph for this function:



5.9.2.3 void Stoper::start ( ) [virtual]

Implementacja funkcji start() z interfejsu IStoper.

Zapisuje moment uruchomienia stopera.

Implements IStoper.

Here is the caller graph for this function:

5.9.2.4 void Stoper::stop ( ) [virtual]

Implementacja funkcji stop() z interfejsu IStoper.

Zapisuje moment zatrzymania stopera.

Implements IStoper.

Here is the caller graph for this function:

5.9.3 Member Data Documentation

5.9.3.1 std::fstream Stoper::my_file [private]

Strumien zapisu do pliku.

Pole ulatwiajace zapis do pliku.

5.9.3.2 clock_t Stoper::start_time [private]

Moment startu stopera.

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

5.9.3.3 clock_t Stoper::stop_time [private]

Moment zatrzymania stopera.

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

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

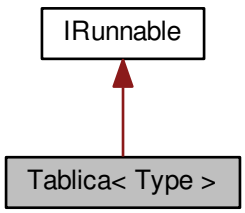
- inc/Stoper.h
- src/Stoper.cpp

5.10 Tablica< Type > Class Template Reference

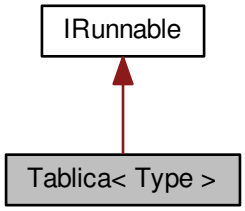
Klasa Tablica, w ktorej odbywa sie zapis dynamiczny elementow.

`#include <Tablica.h>`

Inheritance diagram for Tablica< Type >:



Collaboration diagram for Tablica< Type >:



Public Member Functions

- Tablica (uint x=10)

  Konstruktor parametryczny.
- ~Tablica ()

  Destruktor.
- virtual void prepare (uint size)

  Implementacja funkcji prepare() interfesju IRunnable.
- virtual void run ()

  Implementacja funkcji run() interfesju IRunnable.
- uint getSize ()

  Zwraca aktualny rozmiar tablicy dynamicznej.

Private Member Functions

- bool isFull ()

  Pozwala prosto okreslic, czy nalezy przydzielic pamiec.
- void increaseSize ()

  Zwieksza rozmiar przydzielonej pamieci na stercie.

Private Attributes

- Type * elements

  Wskaznik do poczatku tablicy dynamicznej.
- uint current_size

  Okresla aktualny rozmiar stosu.
- uint desired_size

  Okresla pozadany rozmiar stosu.
- unsigned int index

  Okresla aktualny indeks.

5.10.1 Detailed Description

template<class Type>class Tablica< Type >

Klasa Tablica, w ktorej odbywa sie zapis dynamiczny elementow.

Implementuje metody interfejsu IRunnable. Zajmuje sie dynamiczna alokacja pamieci. Elastyczna a propos typow wskutek zastosowania szablonow.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 template<class Type> Tablica< Type >::Tablica (uint *x* = 10) [inline]

Konstruktor parametryczny.

Umozliwia okreslenie poczatkowego rozmiaru tablicy. W przypadku braku okreslenia tego rozmiaru przyjmuje domyslna wartosc rowna 10.

Parameters

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

5.10.2.2 template<class Type> Tablica< Type >::~Tablica ( ) [inline]

Destruktor.

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

5.10.3 Member Function Documentation

5.10.3.1 template<class Type> uint Tablica< Type >::getSize ( ) [inline]

Zwraca aktualny rozmiar tablicy dynamicznej.

Zwraca wartosc pola current_size.

Returns

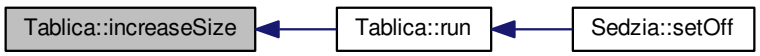
Zwraca wartosc typu unsigned int, gdyz takiego typu jest zmienna current_size.

5.10.3.2 template<class Type> void Tablica< Type >::increaseSize ( ) [inline],[private]

Zwieksza rozmiar przydzielonej pamieci na stercie.

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

Here is the caller graph for this function:



5.10.3.3 template<class Type> bool Tablica< Type >::isFull ( ) [inline], [private]

Pozwala prosto okreslic, czy nalezy przydzielic pamiec.

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

Returns

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

Here is the caller graph for this function:

5.10.3.4 template<class Type> virtual void Tablica< Type >::prepare (uint *size*) [inline], [virtual]

Implementacja funkcji prepare() interfesju IRunnable.

Zapisuje pozadany rozmiar do pola desired_size.

Parameters

size	Parametr typu unsigned int, gdyz rozmiar nie powinien nigdy byc ujemny. Jego wartosc zapisywana jest do pola desired_size.

Implements IRunnable.

Here is the caller graph for this function:

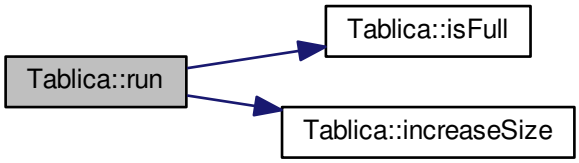
5.10.3.5 template<class Type> virtual void Tablica< Type >::run ( ) [inline], [virtual]

Implementacja funkcji run() interfesju IRunnable.

Uruchamia "bieg", w ktorym nastepuje zapis elementow do poszczegolnych elementow tablicy dynamicznej. Tam odbywa sie alokacja pamieci oraz instrukcje warunkowe.

Implements IRunnable.

Here is the call graph for this function:



Here is the caller graph for this function:

5.10.4 Member Data Documentation

5.10.4.1 template<class Type> uint Tablica< Type >::current_size [private]

Okresla aktualny rozmiar stosu.

Pole prywatne typu unsigned int, gdyz rozmiar nigdy nie powinien byc ujemny.

5.10.4.2 template<class Type> uint Tablica< Type >::desired_size [private]

Okresla pozadany rozmiar stosu.

Pole prywatne typu unsigned int, gdyz rozmiar nigdy nie powinien byc ujemny. Zadawane w funkcji prepare().

5.10.4.3 template<class Type> Type* Tablica< Type >::elements [private]

Wskaznik do poczatku tablicy dynamicznej.

Wskazuje na adres w pamieci sterty. Pole prywatne.

5.10.4.4 template<class Type> unsigned int Tablica< Type >::index [private]

Okresla aktualny indeks.

Pole prywatne typu unsigned int, gdyz indeks nigdy nie powinien byc ujemny. Przechowuje indeks, pierwszej wolnej komorki pamieci, do ktorego mozliwy bedzie zapis.

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

- inc/Tablica.h

Chapter 6

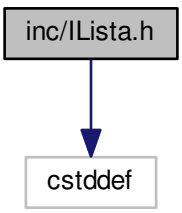
File Documentation

6.1 inc/ILista.h File Reference

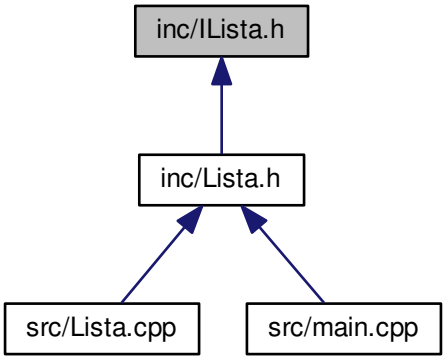
Plik zawiera interfejs dla pojemnika Lista oraz dla klasy Wezel.

```
#include <cstddef>
```

Include dependency graph for ILista.h:



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



Classes

- class Node< NodeType >
  Imlementacja wezlow dla listy.
- class ILista< ListType >
  Interfejs dla pojemnika Lista.

Typedefs

- typedef unsigned int uint
  Skraca zapis.

6.1.1 Detailed Description

Plik zawiera interfejs dla pojemnika Lista oraz dla klasy Wezel. Wskutek zastosowania szablonow wszystkie definicje musza znajdowac sie w pliku naglowkowym, a nie zrodlowym. Wezel jest elementem listy.

Author

Kamil Kuczaj.

6.1.2 Typedef Documentation

6.1.2.1 typedef unsigned int uint

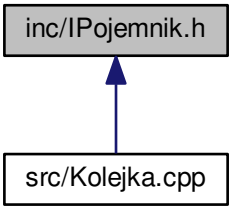
Skraca zapis.

Zdefiniowanie wlasnego typu - pozwala na krotszy zapis.

6.2 inc/IPojemnik.h File Reference

Plik zawiera interfejs dla pojemnika Stos, Kolejka oraz Tablica.

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



Classes

- class IPojemnik< Type >

  Interfejs dla każdego pojemnika.

Typedefs

- typedef unsigned int uint

  Skraca zapis.

6.2.1 Detailed Description

Plik zawiera interfejs dla pojemnika Stos, Kolejka oraz Tablica. Wskutek zastosowania szablonow wszystkie definicje musza znajdowac sie w pliku naglowkowym, a nie zrodlowym.

Author

Kamil Kuczaj.

6.2.2 Typedef Documentation

6.2.2.1 typedef unsigned int uint

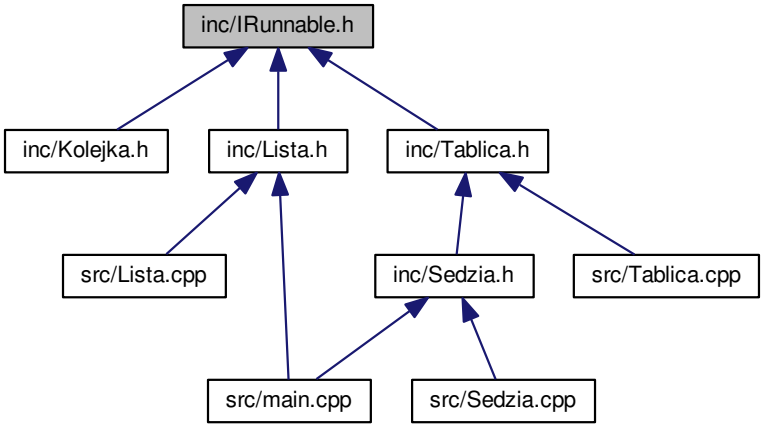
Skraca zapis.

Zdefiniowanie wlasnego typu - pozwala na krotszy zapis.

6.3 inc/IRunnable.h File Reference

Naglowek zawierajacy interfejs dla biegacza.

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



Classes

- class IRunnable

  Interfejs dla biegacza.

6.3.1 Detailed Description

Naglowek zawierajacy interfejs dla biegacza.

Author

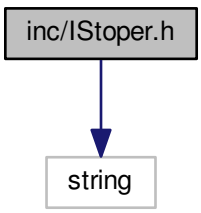
Kamil Kuczaj

6.4 inc/IStoper.h File Reference

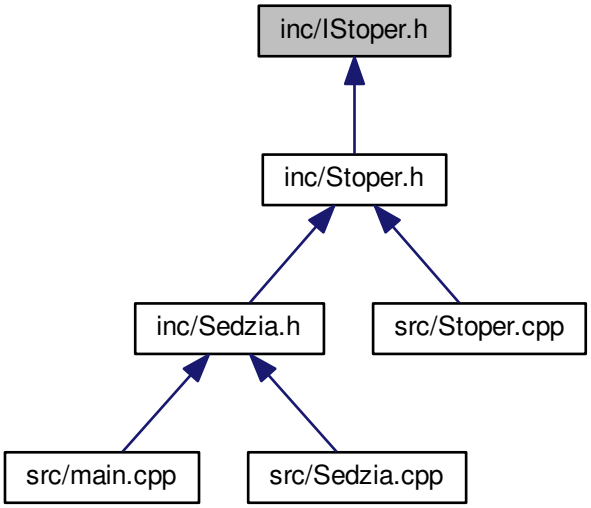
Naglowek zawierajacy interfejs dla stopera.

```
#include <string>
```

Include dependency graph for IStoper.h:



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



Classes

- class IStoper

  Interfejs dla stopera.

6.4.1 Detailed Description

Naglowek zawierajacy interfejs dla stopera.

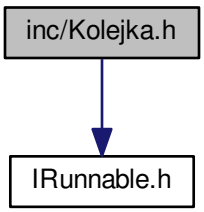
Author

Kamil Kuczaj

6.5 inc/Kolejka.h File Reference

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

Include dependency graph for Kolejka.h:



Classes

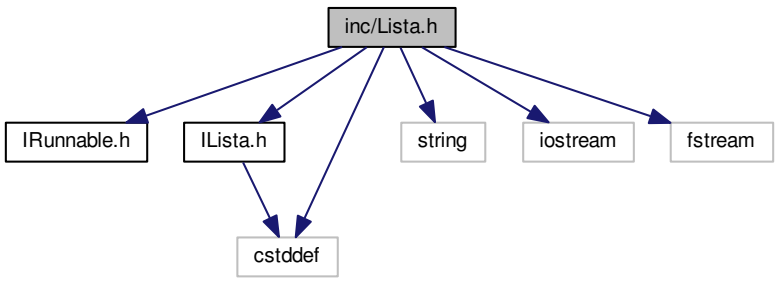
- class Kolejka

6.6 inc/Lista.h File Reference

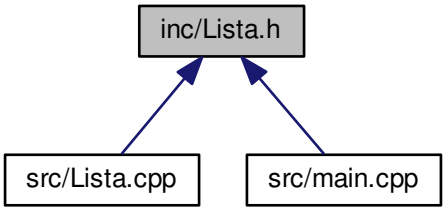
Implementacja jednokierunkowej listy.

```
#include "IRunnable.h"
#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, w ktorej odbywa sie zapis dynamiczny elementow typu int.

Typedefs

- typedef unsigned int uint

  Skraca zapis.

6.6.1 Detailed Description

Implementacja jednokierunkowej listy. Wskutek zastosowania szablonow wszystkie definicje musza znajdowac sie w pliku naglowkowym, a nie zrodlowym.

Author

Kamil Kuczaj.

6.6.2 Typedef Documentation

6.6.2.1 typedef unsigned int uint

Skraca zapis.

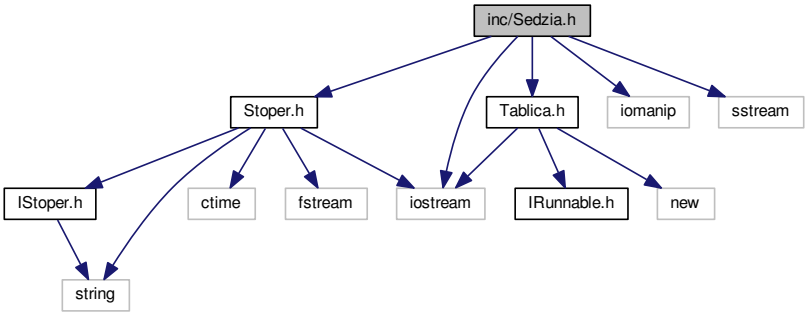
Zdefiniowanie wlasnego typu - pozwala na krotszy zapis

6.7 inc/Sedzia.h File Reference

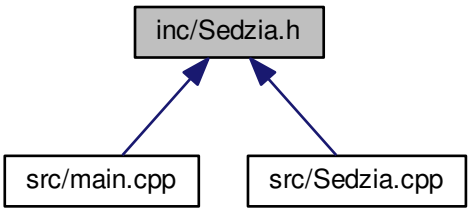
Naglowek opisujacy implementacje Sedziego.

```
#include "Stoper.h"
#include "Tablica.h"
#include <iostream>
#include <iomanip>
#include <sstream>
```

Include dependency graph for Sedzia.h:



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



Classes

- class Sedzia

  Implementacja klasy Sedzia.

6.7.1 Detailed Description

Naglowek opisujacy implementacje Sedziego.

Author

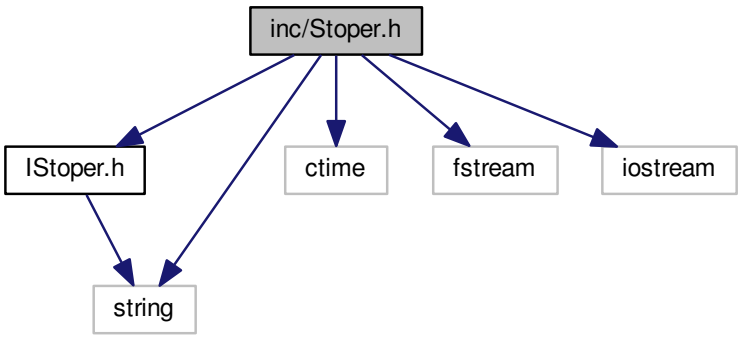
Kamil Kuczaj

6.8 inc/Stoper.h File Reference

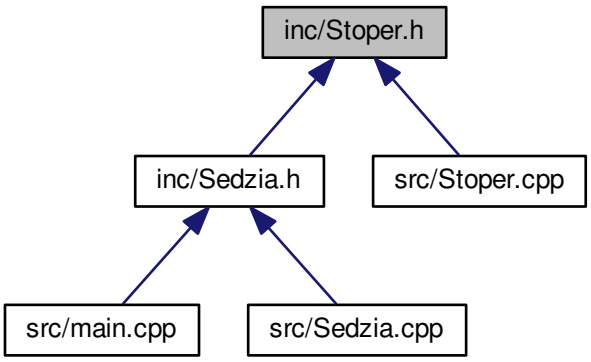
Implementacja interfejsu IStoper w klasie Stoper.

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

6.8.1 Detailed Description

Implementacja interfejsu IStoper w klasie Stoper.

Author

Kamil Kuczaj

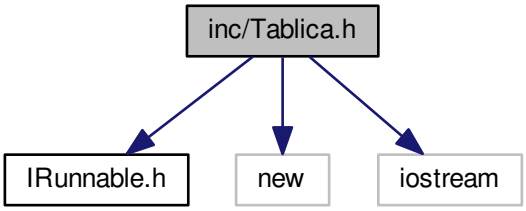
6.9 inc/Stos.h File Reference

6.10 inc/Tablica.h File Reference

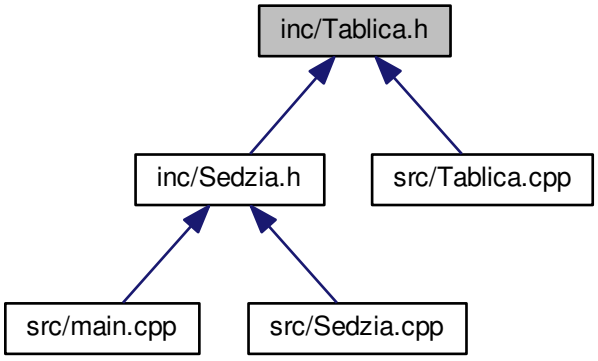
Implementacja interfesju IRunnable.

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

Include dependency graph for Tablica.h:



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



Classes

- class Tablica< Type >

  Klasa Tablica, w ktorej odbywa sie zapis dynamiczny elementow.

Typedefs

- typedef unsigned int uint

  Skraca zapis.

6.10.1 Detailed Description

Implementacja interfesju IRunnable.

Author

Kamil Kuczaj

6.10.2 Typedef Documentation

6.10.2.1 typedef unsigned int uint

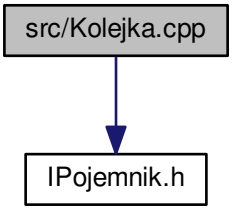
Skraca zapis.

Zdefiniowanie wlasnego typu - pozwala na krotszy zapis

6.11 src/Kolejka.cpp File Reference

```
#include "IPojemnik.h"
```

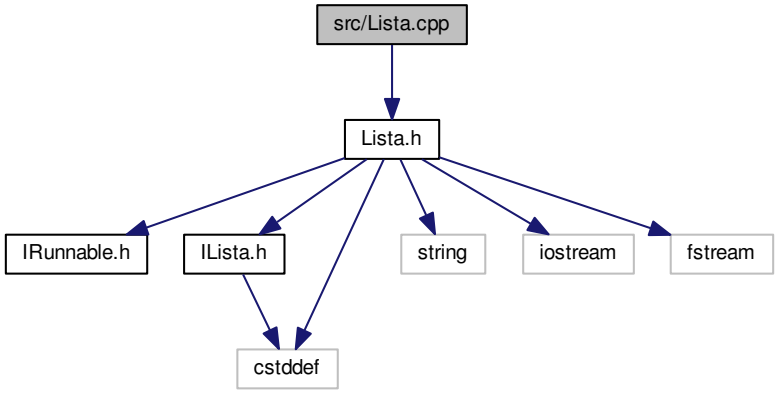
Include dependency graph for Kolejka.cpp:



6.12 src/Lista.cpp File Reference

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

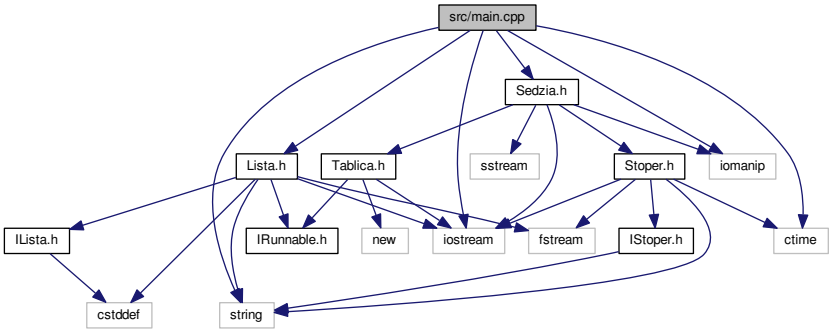
Include dependency graph for Lista.cpp:



6.13 src/main.cpp File Reference

```
#include "Sedzia.h"
#include "Lista.h"
#include <iostream>
#include <ctime>
#include <iomanip>
#include <string>
```

Include dependency graph for main.cpp:



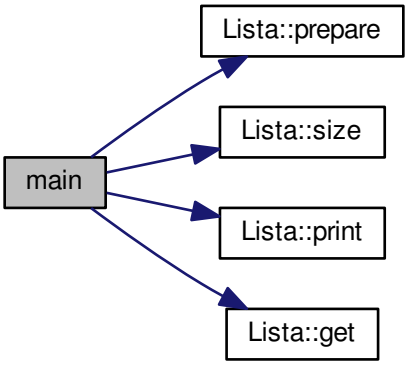
Functions

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

6.13.1 Function Documentation

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

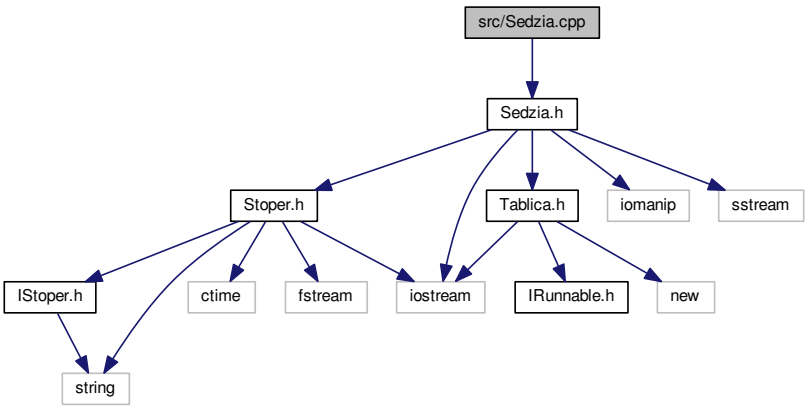
Here is the call graph for this function:



6.14 src/Sedzia.cpp File Reference

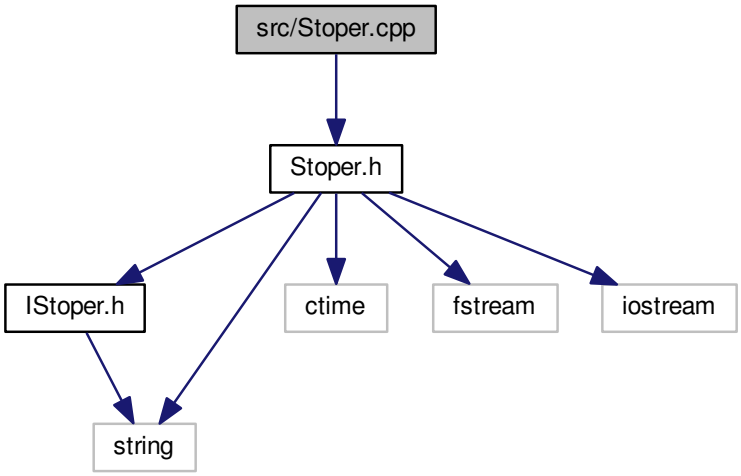
```
#include "Sedzia.h"
```

Include dependency graph for Sedzia.cpp:



6.15 src/Stoper.cpp File Reference

```
#include "Stoper.h"
```
Include dependency graph for Stoper.cpp:



6.16 src/Stos.cpp File Reference

6.17 src/Tablica.cpp File Reference

```
#include "Tablica.h"
```
Include dependency graph for Tablica.cpp: