SList Search

Wygenerowano przez Doxygen 1.8.6

N, 20 mar 2016 16:48:26

Contents

1 Indeks hierarchiczny

1.1 Hierarchia klas

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

2 Indeks klas

2.1 Lista klas

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

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

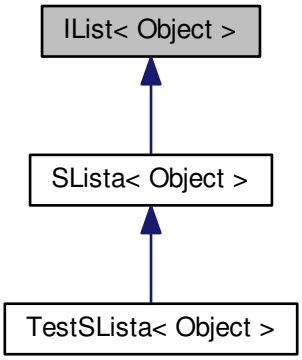
4 Dokumentacja klas

4.1 Dokumentacja szablonu klasy IList< Object >

Interfejs listy jednokierunkowej.

`#include <IList.hh>`

Diagram dziedziczenia dla IList< Object >



Metody publiczne

- virtual bool IsEmpty ()=0

  Metoda sprawdzająca, czy lista jest pusta.
- virtual const Object & Front ()=0

  Metoda zwracająca pierwszy element listy.
- virtual void AddFront (const Object newItem)=0

  Metoda dodająca element na początek listy.
- virtual void RemoveFront ()=0

  Metoda usuwająca element z początku listy.

4.1.1 Opis szczegółowy

template<typename Object>class IList< Object >

Interfejs listy jednokierunkowej.

Definiuje ADT dla listy jednokierunkowej.

Lista może przechowywać dowolny typ danych dzięki zastosowaniu szablonu.

Definicja w linii 22 pliku IList.hh.

4.1.2 Dokumentacja funkcji składowych

4.1.2.1 template<typename Object > virtual void IList< Object >::AddFront (const Object *newItem*) [pure virtual]

Metoda dodająca element na początek listy.

Parametry

>p0.10	>p0.15	p0.678
`in` *newItem* - element do dodania

Implementowany w SLista< Object >.

4.1.2.2 template<typename Object > virtual const Object& IList< Object >::Front ( ) [pure virtual]

Metoda zwracająca pierwszy element listy.

Zwraca

pierwszy element listy

Implementowany w SLista< Object >.

4.1.2.3 template<typename Object > virtual bool IList< Object >::IsEmpty ( ) [pure virtual]

Metoda sprawdzająca, czy lista jest pusta.

Zwracane wartości

>p0.25	p0.705
true - jeśli lista jest pusta
truefalse - jeśli nie jest pusta

Implementowany w SLista< Object >.

4.1.2.4 template<typename Object > virtual void IList< Object >::RemoveFront ( ) [pure virtual]

Metoda usuwająca element z początku listy.

Implementowany w SLista< Object >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

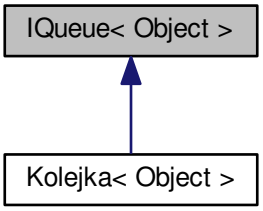
- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/IList.hh

4.2 Dokumentacja szablonu klasy IQueue< Object >

Klasa modelująca interfejs kolejki.

`#include <IQueue.hh>`

Diagram dziedziczenia dla IQueue< Object >



Metody publiczne

- virtual bool IsEmpty ()=0

  Metoda sprawdzająca, czy lista jest pusta.
- virtual int Size ()=0

  Metoda obliczająca rozmiar kolejki.
- virtual Object & Front ()=0

  Metoda zwracająca pierwszy element kolejki.
- virtual void Enqueue (Object item)=0

  Metoda dodająca element do kolejki.
- virtual Object & Dequeue ()=0

  Metoda usuwająca element kolejki.

4.2.1 Opis szczegółowy

template<typename Object>class IQueue< Object >

Klasa modelująca interfejs kolejki.

Definiuje ADT dla kolejki.

Kolejka może przechowywać dowolny typ danych dzięki zastosowaniu szablonu.

Definicja w linii 22 pliku IQueue.hh.

4.2.2 Dokumentacja funkcji składowych

4.2.2.1 template<typename Object > virtual Object& IQueue< Object >::Dequeue ( ) [pure virtual]

Metoda usuwająca element kolejki.

Usuwa element z początki kolejki.

Zwraca

pierwszy element kolejki

Implementowany w Kolejka< Object >.

4.2.2.2 template<typename Object > virtual void IQueue< Object >::Enqueue (Object *item*) [pure virtual]

Metoda dodająca element do kolejki.

Ustawia element na koniec kolejki.

Parametry

>p0.10	>p0.15	p0.678
`in` *item*	- element do dodania

Implementowany w Kolejka< Object >.

4.2.2.3 template<typename Object > virtual Object& IQueue< Object >::Front ( ) [pure virtual]

Metoda zwracająca pierwszy element kolejki.

Zwraca

pierwszy element kolejki

Implementowany w Kolejka< Object >.

4.2.2.4 template<typename Object > virtual bool IQueue< Object >::IsEmpty ( ) [pure virtual]

Metoda sprawdzająca, czy lista jest pusta.

true - jeśli lista jest pusta false - jeśli nie jest pusta

Implementowany w Kolejka< Object >.

4.2.2.5 template<typename Object > virtual int IQueue< Object >::Size ( ) [pure virtual]

Metoda obliczająca rozmiar kolejki.

Zwraca

liczba elementów w kolejce

Implementowany w Kolejka< Object >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

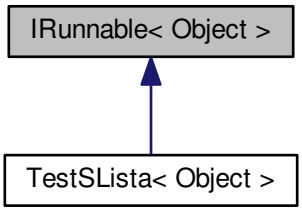
- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/IQueue.hh

4.3 Dokumentacja szablonu klasy IRunnable< Object >

Klasa szablonowa modelująca interfejs "Biegacza".

`#include <IRunnable.hh>`

Diagram dziedziczenia dla IRunnable< Object >



Metody publiczne

- Object & Parametr ()

  Metoda zwracająca zdefiniowany parametr.
- virtual bool Prepare (int rozmiar)=0

  Metoda przygotowująca obiekt do operacji.
- virtual bool PrepareUsingFile (string nazwaPliku, int rozmiar)=0

  Metoda przygotowująca obiekt do operacji z wykorzystaniem pliku.
- virtual bool Run (Object track)=0

  Metoda uruchamiająca zdefiniowaną operację

Atrybuty chronione

- Object parametr

4.3.1 Opis szczegółowy

template<typename Object>class IRunnable< Object >

Klasa szablonowa modelująca interfejs "Biegacza".

Klasa jest abstrakcyjnym uogólnieniem obiektu, na którym można wykonać zdefiniowane operacje, którym z kolei można zmierzyć czas wykonywania.

Definicja w linii 22 pliku IRunnable.hh.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 template<typename Object > Object& IRunnable< Object >::Parametr ( ) [inline]

Metoda zwracająca zdefiniowany parametr.

Zwraca

wartość parametru obiektu

Definicja w linii 32 pliku IRunnable.hh.

4.3.2.2 template<typename Object > virtual bool IRunnable< Object >::Prepare (int *rozmiar*) [pure virtual]

Metoda przygotowująca obiekt do operacji.

Parametry

>p0.10	>p0.15	p0.678
in *rozmiar* - liczba elementów do przygotowania;

Zwracane wartości

>p0.25	p0.705
true - jeśli przygotowanie się powiodło
false - jeśli wystąpił jakiś błąd

4.3.2.3 template<typename Object > virtual bool IRunnable< Object >::PrepareUsingFile (string *nazwaPliku,* int *rozmiar*) [pure virtual]

Metoda przygotowująca obiekt do operacji z wykorzystaniem pliku.

Parametry

>p0.10	>p0.15	p0.678
in *nazwaPliku* - plik do pobrania danych
in *rozmiar* - liczba elementów do przygotowania;

Zwracane wartości

>p0.25	p0.705
true - jeśli przygotowanie się powiodło
false - jeśli wystąpił jakiś błąd

Implementowany w TestSLista< Object >.

4.3.2.4 template<typename Object > virtual bool IRunnable< Object >::Run (Object *track*) [pure virtual]

Metoda uruchamiająca zdefiniowaną operację

Parametry

>p0.10	>p0.15	p0.678
in *track* - parametr wykonania operacji

Zwracane wartości

>p0.25	p0.705
true - jeśli operacja zakończyła się pomyślnie
false - jeśli wystąpił jakiś błąd

Implementowany w TestSLista< Object >.

4.3.3 Dokumentacja atrybutów składowych

4.3.3.1 template<typename Object > Object IRunnable< Object >::parametr [protected]

Definicja w linii 25 pliku IRunnable.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

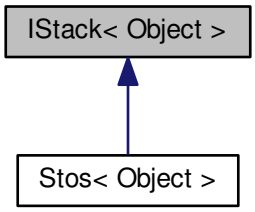
- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/IRunnable.hh

4.4 Dokumentacja szablonu klasy IStack< Object >

Klasa szablonowa modelująca interfejs stosu.

`#include <IStack.hh>`

Diagram dziedziczenia dla IStack< Object >



Metody publiczne

- virtual bool IsEmpty ()=0

  Metoda sprawdzająca, czy stos jest pusty.
- virtual int Size ()=0

  Metoda obliczająca rozmiar stosu.
- virtual Object & Top ()=0

  Metoda zwracająca wierzchołek stosu.
- virtual void Push (Object item)=0

  Metoda dodająca element na stos.
- virtual Object & Pop ()=0

  Metoda zrzucająca element ze stosu.

4.4.1 Opis szczegółowy

template<typename Object>class IStack< Object >

Klasa szablonowa modelująca interfejs stosu.

Definiuje ADT dla stosu.

Stos może przechowywać dowolny typ danych dzięki zastosowaniu szablonu.

Definicja w linii 22 pliku IStack.hh.

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 template<typename Object > virtual bool IStack< Object >::IsEmpty ( ) [pure virtual]

Metoda sprawdzająca, czy stos jest pusty.

true - jeśli stos jest pustay false - jeśli stos nie jest pusty

Implementowany w Stos< Object >.

4.4.2.2 template<typename Object > virtual Object& IStack< Object >::Pop ( ) [pure virtual]

Metoda zrzucająca element ze stosu.

Usuwa wierzchołek ze stosu i zwraca jego wartość.

Zwraca

element na wierzchu stosu

Implementowany w Stos< Object >.

4.4.2.3 template<typename Object > virtual void IStack< Object >::Push (Object *item*) [pure virtual]

Metoda dodająca element na stos.

Wrzuca element na wierzchołek stosu.

Parametry

>p0.10	>p0.15	p0.678
in *item*	- element do dodania

Implementowany w Stos< Object >.

4.4.2.4 template<typename Object > virtual int IStack< Object >::Size ( ) [pure virtual]

Metoda obliczająca rozmiar stosu.

Zwraca

liczba elementów na stos

Implementowany w Stos< Object >.

4.4.2.5 template<typename Object > virtual Object& IStack< Object >::Top ( ) [pure virtual]

Metoda zwracająca wierzchołek stosu.

Zwraca

element na wierzchu stosu

Implementowany w Stos< Object >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

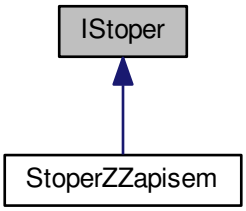
- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/IStack.hh

4.5 Dokumentacja klasy IStoper

Klasa modelująca interfejs stopera.

`#include <IStoper.hh>`

Diagram dziedziczenia dla IStoper



Metody publiczne

- virtual void Start ()

  Metoda rozpoczynająca pomiar czasu.
- virtual void Stop ()

  Metoda kończąca pomiar czasu.
- virtual double GetElapsedTime ()

  Metoda podająca zmierzony czas.

Atrybuty chronione

- clock_t start
- clock_t koniec

4.5.1 Opis szczegółowy

Klasa modelująca interfejs stopera.

Klasa jest modelelem stopera o podstawowych funkcjach.

Definicja w linii 20 pliku IStoper.hh.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 double IStoper::GetElapsedTime ( ) [virtual]

Metoda podająca zmierzony czas.

Zwraca

zmierzony czas w sekundach

Definicja w linii 12 pliku IStoper.cpp.

4.5.2.2 void IStoper::Start ( ) [virtual]

Metoda rozpoczynająca pomiar czasu.

Definicja w linii 4 pliku IStoper.cpp.

4.5.2.3 void IStoper::Stop ( ) [virtual]

Metoda kończąca pomiar czasu.

Definicja w linii 8 pliku IStoper.cpp.

4.5.3 Dokumentacja atrybutów składowych

4.5.3.1 clock_t IStoper::koniec [protected]

Definicja w linii 23 pliku IStoper.hh.

4.5.3.2 clock_t IStoper::start [protected]

Definicja w linii 23 pliku IStoper.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/IStoper.hh
- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/IStoper.cpp

4.6 Dokumentacja szablonu klasy Kolejka< Object >

Klasa szablonowa implementująca kolejkę

`#include <Kolejka.hh>`

Diagram dziedziczenia dla Kolejka< Object >

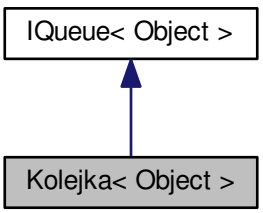
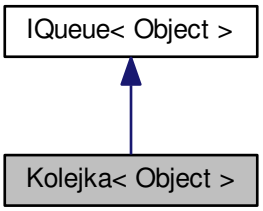


Diagram współpracy dla Kolejka< Object >:



Metody publiczne

- Kolejka ()
  Konstruktor bezargumentowy kolejki.
- Kolejka (int ile)
  Konstruktor kolejki.
- ~Kolejka ()
  Destruktor kolejki.
- int & Pojemnosc ()
  Metoda sprawdzająca pojemność kolejki.
- virtual bool IsEmpty ()
  Metoda sprawdzająca, czy kolejka jest pusta.
- virtual int Size ()
  Metoda obliczająca rozmiar kolejki.
- virtual Object & Front ()
  Metoda zwracająca pierwszy element kolejki.
- virtual void Enqueue (Object item)
  Metoda dodająca element do kolejki.
- virtual Object & Dequeue ()
  Metoda usuwająca element kolejki.

4.6.1 Opis szczegółowy

template<typename Object>class Kolejka< Object >

Klasa szablonowa implementująca kolejkę

Kolejka zbudowana jest w oparciu o dynamiczną tablicę.

Kolejka może przechowywać dowolny typ danych dzięki zastosowaniu szablonu.

Definicja w linii 23 pliku Kolejka.hh.

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 template<typename Object > Kolejka< Object >::Kolejka ( )

Konstruktor bezargumentowy kolejki.

Inicjuje Kolejkę poprzez zaalokowanie tablicy o rozmiarze 1. Ustawia indeksy f (front) i r (rear) na 0.

Definicja w linii 118 pliku Kolejka.hh.

4.6.2.2 template<typename Object > Kolejka< Object >::Kolejka (int *ile*)

Konstruktor kolejki.

Inicjuje Kolejkę poprzez zaalokowanie tablicy o rozmiarze podanym w argumencie konstruktora (o ile jest różny od 0). Ustawia indeksy f (front) i r (rear) na 0.

Parametry

>p0.10	>p0.15	p0.678
in *ile* - początkowa pojemność kolejki;

Definicja w linii 128 pliku Kolejka.hh.

4.6.2.3 template<typename Object > Kolejka< Object >::~Kolejka ( )

Destruktor kolejki.

Zwalnia pamięć zajmowaną przez kolejkę. Ustawia wskaźnik tablicy na NULL.

Definicja w linii 139 pliku Kolejka.hh.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 template<typename Object > Object & Kolejka< Object >::Dequeue ( ) [virtual]

Metoda usuwająca element kolejki.

Sprawdza, czy kolejka jest pusta. Jeśli tak, wyrzuca wyjątek. Jeśli nie, usuwa element z początki kolejki o indeksie f (front), a następnie przesuwa indeks f o jedno miejsce dalej.

Zwraca

pierwszy element kolejki

Implementuje IQueue< Object >.

Definicja w linii 192 pliku Kolejka.hh.

4.6.3.2 template<typename Object > void Kolejka< Object >::Enqueue (Object *item*) [virtual]

Metoda dodająca element do kolejki.

Ustawia element na koniec kolejki

- komórka tablicy o indeksie r (rear). Jeśli nie ma już miejsca w kolejce tablica jest powiększana 2 razy. Następnie element zostaje prawidłowo dodany do kolejki, a indeks r zostaje przesunięty o jedno miejsce dalej.

Parametry

>p0.10	>p0.15	p0.678
in *item* - element do dodania

Implementuje IQueue< Object >.

Definicja w linii 170 pliku Kolejka.hh.

4.6.3.3 template<typename Object > Object & Kolejka< Object >::Front ( ) [virtual]

Metoda zwracająca pierwszy element kolejki.

Jeśli kolejka jest pusta, wyrzuca wyjątek. Zwraca element tablicy o indeksie f (front).

Zwraca

pierwszy element kolejki

Implementuje IQueue< Object >.

Definicja w linii 162 pliku Kolejka.hh.

4.6.3.4 template<typename Object > bool Kolejka< Object >::IsEmpty ( ) [virtual]

Metoda sprawdzająca, czy kolejka jest pusta.

Kolejka jest pusta, jeśli wartości indeksów f (front) i r (rear) są sobie równe.

Zwracane wartości

|>p0.25|p0.705|

true - jeśli lista jest pusta

false - jeśli nie jest pusta

Implementuje IQueue< Object >.

Definicja w linii 146 pliku Kolejka.hh.

4.6.3.5 template<typename Object > int& Kolejka< Object >::Pojemnosc ( ) [inline]

Metoda sprawdzająca pojemność kolejki.

Zwraca parametr opisujący pojemność tablicy dynamicznej implementującej kolejkę.

Zwraca

pojemność kolejki

Definicja w linii 64 pliku Kolejka.hh.

4.6.3.6 template<typename Object > int Kolejka< Object >::Size ( ) [virtual]

Metoda obliczająca rozmiar kolejki.

Rozmiar obliczany jest przez odjęcie wartości indeksów f (front) i r (rear).

Zwraca

liczba elementów w kolejce

Implementuje IQueue< Object >.

Definicja w linii 155 pliku Kolejka.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/Kolejka.hh

4.7 Dokumentacja szablonu klasy SLista< Object >

Szablonowa klasa implementująca listę jednokierunkową

`#include <SLista.hh>`

Diagram dziedziczenia dla SLista< Object >

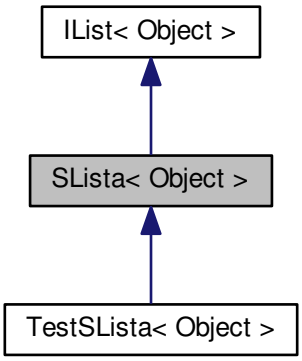
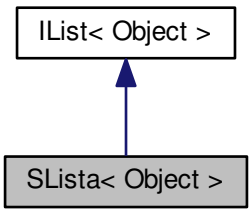


Diagram współpracy dla SLista< Object >:



Metody publiczne

- SLista ()

  Konstruktor listy jednokierunkowaj.
- ~SLista ()

  Destruktor listy jednokierunkowaj.
- SNode< Object > * Head ()

  Metoda zwracająca głowę listy.
- SNode< Object > * Find (Object k)

  Metoda wyszukująca element na liście.
- virtual bool IsEmpty ()

  Metoda sprawdzająca, czy lista jest pusta.
- virtual const Object & Front ()

  Metoda zwracająca pierwszy element listy.
- virtual void AddFront (const Object newItem)

  Metoda dodająca element na początek listy.

- virtual void RemoveFront ()

  Metoda usuwająca element z początku listy.
- void printl ()

  Metoda wypisująca zawartość listy na standardowe wyjście.

4.7.1 Opis szczegółowy

template<typename Object>class SLista< Object >

Szablonowa klasa implementująca listę jednokierunkową

SLista jest zbudowana w oparciu o węzły SNode oraz operacje na wskaźnikach.

SLista może przechowywać dowolny typ danych dzięki zastosowaniu szablonu.

Definicja w linii 25 pliku SLista.hh.

4.7.2 Dokumentacja konstruktora i destruktora

4.7.2.1 template<typename Object > SLista< Object >::SLista ( )

Konstruktor listy jednokierunkowaj.

Inicjuje SListę poprzez ustawienie wskaźnika NULL jako początek (head) tej listy.

Definicja w linii 131 pliku SLista.hh.

4.7.2.2 template<typename Object > SLista< Object >::~SLista ( )

Destruktor listy jednokierunkowaj.

Usuwa SListę poprzez ustawienie wskaźnika NULL jako początek (head) tej listy.

Definicja w linii 135 pliku SLista.hh.

4.7.3 Dokumentacja funkcji składowych

4.7.3.1 template<typename Object > void SLista< Object >::AddFront (const Object *newItem*) [virtual]

Metoda dodająca element na początek listy.

Alokuje nowy węzeł, dodaje nowy element, dodaje powiązanie tak, aby węzeł wskazywał na stary head, uaktualnia head.

Parametry

>p0.10	>p0.15	p0.678
in *newItem*	- element do dodania	

Implementuje IList< Object >.

Definicja w linii 169 pliku SLista.hh.

4.7.3.2 template<typename Object > SNode< Object > * SLista< Object >::Find (Object *k*)

Metoda wyszukująca element na liście.

Implementuje algorytm liniowego przeszukiwania listy.

Parametry

>p0.10	>p0.15	p0.678

in *k* - element do wyszukania

Zwraca

Wskaźnik do znalezionego elementu lub NULL, gdy nie znaleziono.

Definicja w linii 121 pliku SLista.hh.

4.7.3.3 template<typename Object > const Object & SLista< Object >::Front ( ) [virtual]

Metoda zwracająca pierwszy element listy.

Sprawdza, czy lista jest pusta i zwraca dane pierwszego węzła listy. Jeśli lista jest pusta, wyrzuca wyjątek.

Zwraca

dane pierwszego węzła listy

Implementuje IList< Object >.

Definicja w linii 160 pliku SLista.hh.

4.7.3.4 template<typename Object > SNode< Object > * SLista< Object >::Head ( )

Metoda zwracająca głowę listy.

Zwraca wskaźnik do początku listy lub NULL, jeśli lista jest pusta.

Zwraca

Wskaźnik do głowy listy.

Definicja w linii 151 pliku SLista.hh.

4.7.3.5 template<typename Object > bool SLista< Object >::IsEmpty ( ) [virtual]

Metoda sprawdzająca, czy lista jest pusta.

Sprawdza, czy head wskazuje na coś innego niż NULL. Implementacja metody wirtualnej z interfejsu IList.

Zwraca

true - jeśli lista jest pusta, false - jeśli nie

Implementuje IList< Object >.

Definicja w linii 142 pliku SLista.hh.

4.7.3.6 template<typename Object > void SLista< Object >::printl ( )

Metoda wypisująca zawartość listy na standardowe wyjście.

Definicja w linii 113 pliku SLista.hh.

4.7.3.7 template<typename Object > void SLista< Object >::RemoveFront ( ) [virtual]

Metoda usuwająca element z początku listy.

Uaktualnia head, aby wskazywał na kolejny element na liście, po czym usuwa stary węzeł.

Parametry

|>p0.10|>p0.15|p0.678|

in *newItem* - element do dodania

Implementuje IList< Object >.

Definicja w linii 178 pliku SLista.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/SLista.hh

4.8 Dokumentacja szablonu klasy SNode< Object >

Klasa implementująca węzeł listę jednokierunkową

`#include <SNode.hh>`

Metody publiczne

- Object GetElement ()

  Metoda zwracająca obecny element węzła.
- SNode< Object > * GetNext ()

  Metoda zwracająca wskaźnik do następnego węzła.
- void SetElement (Object newItem)

  Metoda przypisująca element do węzła.
- void SetNext (SNode< Object > *newItem)

  Metoda przypisująca następny element do węzła.

4.8.1 Opis szczegółowy

template<typename Object>class SNode< Object >

Klasa implementująca węzeł listę jednokierunkową

Węzeł może przechowywać dowolny typ danych dzięki zastosowaniu szablonu.

Definicja w linii 20 pliku SNode.hh.

4.8.2 Dokumentacja funkcji składowych

4.8.2.1 template<typename Object> Object SNode< Object >::GetElement ( ) [inline]

Metoda zwracająca obecny element węzła.

Zwraca

element w polu item węzła

Definicja w linii 31 pliku SNode.hh.

4.8.2.2 template<typename Object> SNode<Object>* SNode< Object >::GetNext ( ) [inline]

Metoda zwracająca wskaźnik do następnego węzła.

Zwraca

wskaźnik w polu next węzła

Definicja w linii 38 pliku SNode.hh.

4.8.2.3 template<typename Object> void SNode< Object >::SetElement (Object *newItem*) [inline]

Metoda przypisująca element do węzła.

Ustawia podany element w polu item węzła.

Parametry

>p0.10	>p0.15	p0.678
in *newItem* - element do ustawienia w węźle

Definicja w linii 46 pliku SNode.hh.

4.8.2.4 template<typename Object> void SNode< Object >::SetNext (SNode< Object > * *newItem*) [inline]

Metoda przypisująca następny element do węzła.

Ustawia podany wskaźnik w polu next węzła.

Parametry

>p0.10	>p0.15	p0.678
in *newItem* - wskaźnik do następnego węzła

Definicja w linii 54 pliku SNode.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/SNode.hh

4.9 Dokumentacja klasy StoperZZapisem

Klasa implementująca rozbudowany stoper.

`#include <StoperZZapisem.hh>`

Diagram dziedziczenia dla StoperZZapisem

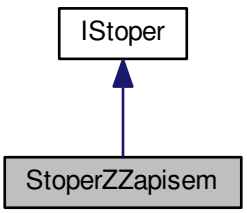
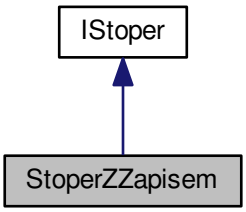


Diagram współpracy dla StoperZZapisem:



Metody publiczne

- StoperZZapisem ()
- ~StoperZZapisem ()
- int & Rozmiar ()
- double *& Poczatek ()
- double *& PoczatekBufora ()
- bool SaveElapsedTime (double rekord)

  Metoda zapisująca wartość pomiaru czasu okrążenia.
- bool SaveAverageTimeToBuffer (double rekord)

  Metoda zapisująca średni czas okrążenia do bufora plikowego.
- void ShowMemory ()

  Metoda wypisująca zawartość pamięci stopera.
- void CleanMemory ()

  Metoda usuwająca zawartość pamięci stopera.
- void CleanBuffer ()

  Metoda usuwająca zawartość bufora plikowego stopera.
- double SeriesAverage ()

  Metoda wyliczająca średni czas okrążenia.
- bool DumpTimesToFile (ofstream &plik)

  Metoda zapisująca zawartość bufora plikowego do pliku.
- bool DumpOneTimeToFile (ofstream &plik, double rekord)

  Metoda zapisująca pojedynczy rekord bufora plikowego do pliku.

Dodatkowe Dziedziczone Składowe

4.9.1 Opis szczegółowy

Klasa implementująca rozbudowany stoper.

Klasa jest modelelem stopera z funkcją zapisu czasu okrążeń, liczeniem średniego czasu kilku okrążeń, zapisu zmierzonych czasów do pliku.

Definicja w linii 26 pliku StoperZZapisem.hh.

Parametry

Zwracane wartości

>p0.25	p0.705
true - jeśli udało się zapisać
false - jeśli udało się zapisać

Definicja w linii 83 pliku StoperZZapisem.cpp.

4.9.3.4 bool StoperZZapisem::DumpTimesToFile (ofstream & *plik*)

Metoda zapisująca zawartość bufora plikowego do pliku.

Dokonuje zapisu rekordów w buforze do pliku.

Parametry

>p0.10	>p0.15	p0.678
`in` *plik* - dowiązanie do pliku, do którego wykona zapis

Zwracane wartości

>p0.25	p0.705
true - jeśli udało się zapisać

false - jeśli udało się zapisać

Definicja w linii 68 pliku StoperZZapisem.cpp.

4.9.3.5 double* & StoperZZapisem::Poczatek ( ) [inline]

Definicja w linii 35 pliku StoperZZapisem.hh.

4.9.3.6 double* & StoperZZapisem::PoczatekBufora ( ) [inline]

Definicja w linii 36 pliku StoperZZapisem.hh.

4.9.3.7 int& StoperZZapisem::Rozmiar ( ) [inline]

Definicja w linii 34 pliku StoperZZapisem.hh.

4.9.3.8 bool StoperZZapisem::SaveAverageTimeToBuffer (double *rekord*)

Metoda zapisująca średni czas okrążenia do bufora plikowego.

Dodaje podany czas do pamięci stopera, z której można dokonać zapisu do pliku.

Parametry

>p0.10	>p0.15	p0.678
in *rekord* - wartość pomiaru czasu	

Zwracane wartości

>p0.25	p0.705
true - jeśli udało się zapisać	
false - jeśli udało się zapisać	

Definicja w linii 26 pliku StoperZZapisem.cpp.

4.9.3.9 bool StoperZZapisem::SaveElapsedTime (double *rekord*)

Metoda zapisująca wartość pomiaru czasu okrążenia.

Dodaje podany czas do tablicy czasów okrążeń.

Parametry

>p0.10	>p0.15	p0.678
in *rekord* - wartość pomiaru czasu	

Zwracane wartości

>p0.25	p0.705
true - jeśli udało się zapisać	
false - jeśli udało się zapisać	

Definicja w linii 15 pliku StoperZZapisem.cpp.

4.9.3.10 double StoperZZapisem::SeriesAverage ( )

Metoda wyliczająca średni czas okrążenia.

Definicja w linii 55 pliku StoperZZapisem.cpp.

4.9.3.11 void StoperZZapisem::ShowMemory ( )

Metoda wypisująca zawartość pamięci stopera.

Definicja w linii 35 pliku StoperZZapisem.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/StoperZZapisem.hh
- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/StoperZZapisem.cpp

4.10 Dokumentacja szablonu klasy Stos< Object >

Klasa szablonowa implementująca stos.

`#include <Stos.hh>`

Diagram dziedziczenia dla Stos< Object >

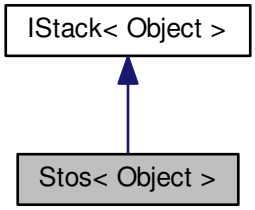
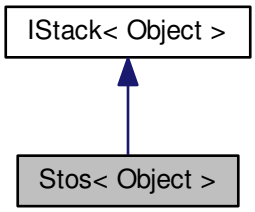


Diagram współpracy dla Stos< Object >:



Metody publiczne

- Stos ()

  Konstruktor bezargumentowy stosu.
- Stos (int ile)

  Konstruktor stosu.
- ~Stos ()

Destruktor stosu.

- int & Pojemnosc ()

  Metoda sprawdzająca pojemność stosu.

- virtual bool IsEmpty ()

  Metoda sprawdzająca, czy stos jest pusty.

- virtual int Size ()

  Metoda obliczająca rozmiar stosu.

- virtual Object & Top ()

  Metoda zwracająca wierzchołek stosu.

- virtual void Push (Object item)

  Metoda dodająca element na stos.

- virtual Object & Pop ()

  Metoda zrzucająca element ze stosu.

4.10.1 Opis szczegółowy

template<typename Object>class Stos< Object >

Klasa szablonowa implementująca stos.

Stos zbudowany jest w oparciu o dynamiczną tablicę.

Stos może przechowywać dowolny typ danych dzięki zastosowaniu szablonu.

Definicja w linii 22 pliku Stos.hh.

4.10.2 Dokumentacja konstruktora i destruktora

4.10.2.1 template<typename Object > Stos< Object >::Stos ( )

Konstruktor bezargumentowy stosu.

Inicjuje stos poprzez zaalokowanie tablicy o rozmiarze 1. Ustawia indeks top na -1.

Definicja w linii 119 pliku Stos.hh.

4.10.2.2 template<typename Object > Stos< Object >::Stos (int *ile*)

Konstruktor stosu.

Inicjuje stos poprzez zaalokowanie tablicy o rozmiarze podanym w argumencie konstruktora (o ile jest różny od 0). Ustawia indeks top na -1.

Parametry

>p0.10	>p0.15	p0.678
`in` *ile* - początkowa pojemność kolejki;

Definicja w linii 128 pliku Stos.hh.

4.10.2.3 template<typename Object > Stos< Object >::~Stos ( )

Destruktor stosu.

Zwalnia pamięć zajmowaną przez stosu. Ustawia wskaźnik tablicy dynamicznej na NULL.

Definicja w linii 140 pliku Stos.hh.

4.10.3 Dokumentacja funkcji składowych

4.10.3.1 template<typename Object > bool Stos< Object >::IsEmpty ( ) [virtual]

Metoda sprawdzająca, czy stos jest pusty.

Jeśli indeks top<0, stos jest pusty. W przeciwnym wypadku, stos nie jest pusty. true - jeśli stos jest pusty false - jeśli stos nie jest pusty

Implementuje IStack< Object >.

Definicja w linii 147 pliku Stos.hh.

4.10.3.2 template<typename Object > int& Stos< Object >::Pojemnosc ( ) [inline]

Metoda sprawdzająca pojemność stosu.

Zwraca parametr opisujący pojemność tablicy dynamicznej implementującej stos.

Zwraca

pojemność stosu

Definicja w linii 63 pliku Stos.hh.

4.10.3.3 template<typename Object > Object & Stos< Object >::Pop ( ) [virtual]

Metoda zrzucająca element ze stosu.

Jeśli stos jest pusty, wyrzuca wyjątek. Zwraca element tablicy o indeksie top. Zmniejsza indeks top o 1.

Zwraca

element na wierzchu stosu

Implementuje IStack< Object >.

Definicja w linii 190 pliku Stos.hh.

4.10.3.4 template<typename Object > void Stos< Object >::Push (Object *item*) [virtual]

Metoda dodająca element na stos.

Wrzuca element na wierzchołek stosu. Indeks top zostaje przesunięty o jedno miejsce dalej. Jeśli nie ma już miejsca na stosie tablica jest powiększana 2 razy. Następnie element zostaje prawidłowo wrzucony na stos, a indeks top zostaje przesunięty o jedno miejsce dalej.

Parametry

|>p0.10|>p0.15|p0.678|

`in` *item* - element do dodania

Implementuje IStack< Object >.

Definicja w linii 172 pliku Stos.hh.

4.10.3.5 template<typename Object > int Stos< Object >::Size ( ) [virtual]

Metoda obliczająca rozmiar stosu.

Rozmiar obliczany jest przez działanie 'top+1'.

Zwraca

liczba elementów na stosie

Implementuje IStack< Object >.

Definicja w linii 156 pliku Stos.hh.

4.10.3.6 template<typename Object > Object & Stos< Object >::Top ( ) [virtual]

Metoda zwracająca wierzchołek stosu.

Jeśli stos jest pusty, wyrzuca wyjątek. Zwraca element tablicy o indeksie top.

Zwraca

element na wierzchu stosu

Implementuje IStack< Object >.

Definicja w linii 164 pliku Stos.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

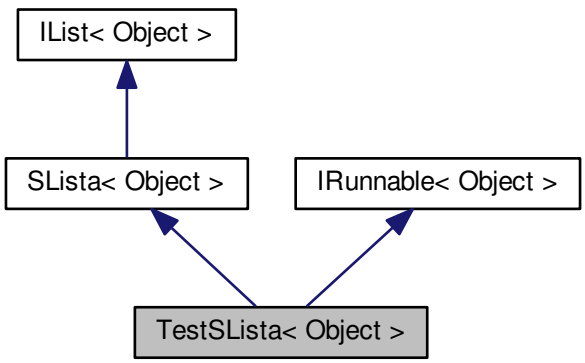
- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/Stos.hh

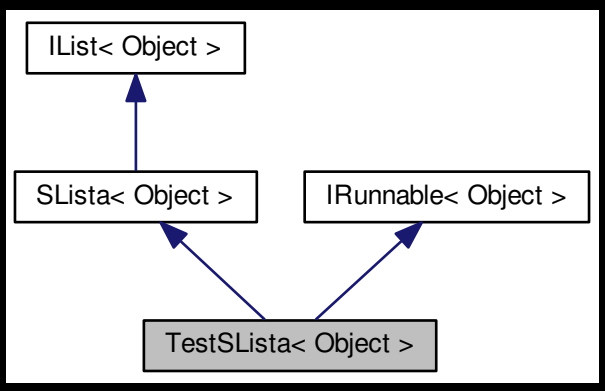
4.11 Dokumentacja szablonu klasy TestSLista< Object >

Szablonowa klasa listy jednokierunkowej implementująca IRunnable.

`#include <TestSLista.hh>`

Diagram dziedziczenia dla TestSLista< Object >





PrepareUsingFile

Run

template<typename Object>class TestSLista< Object >

Szablonowa klasa listy jednokierunkowej implementująca IRunnable.

TestSLista jest zbudowana w oparciu o węzły SNode oraz operacje na wskaźnikach.

TestSLista może przechowywać dowolny typ danych dzięki zastosowaniu szablonu.

Definicja w linii 29 pliku TestSLista.hh.

4.11.2 Dokumentacja funkcji składowych

4.11.2.1 template<typename Object > bool TestSLista< Object >::PrepareUsingFile (string *nazwaPliku,* int *rozmiar*) [virtual]

Metoda przygotowująca obiekt do operacji z wykorzystaniem pliku.

Jeśli list nie jest pusta, metoda ją opróżnia. Następnie wypełnia listę z podanego pliku daną liczbę elementów.

Parametry

>p0.10	>p0.15	p0.678
in	*nazwaPliku*	- plik do pobrania danych
in	*rozmiar*	- liczba elementów do przygotowania

Zwracane wartości

>p0.25	p0.705
true - jeśli przygotowanie się powiodło
false - jeśli wystąpił jakiś błąd

Implementuje IRunnable< Object >.

Definicja w linii 60 pliku TestSLista.hh.

4.11.2.2 template<typename Object > bool TestSLista< Object >::Run (Object *track*) [virtual]

Metoda uruchamiająca wyszukiwanie podanego obiektu.

Wywołuje metodę wyszukującą podany element na liście. Następnie wypisuje na standardowe wyjście "Znaleziono!" bądź "Nie znaleziono!".

Parametry

>p0.10	>p0.15	p0.678
`in` *track* - element do wyszukania

Zwracane wartości

>p0.25	p0.705
true - operacja zakończyła się niezależnie od tego, czy znaleziono, czy nie

Implementuje IRunnable< Object >.

Definicja w linii 88 pliku TestSLista.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

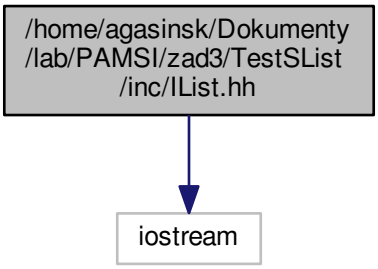
- /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/TestSLista.hh

5 Dokumentacja plików

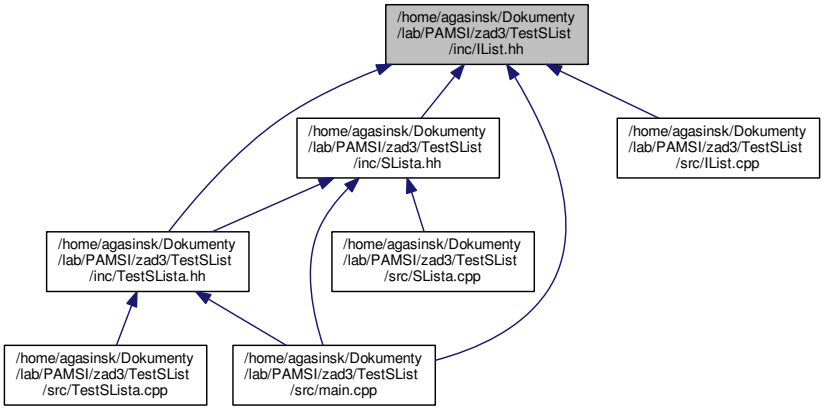
5.1 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/I-List.hh

```
#include <iostream>
```

Wykres zależności załączania dla IList.hh:



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



Komponenty

- class IList< Object >

  Interfejs listy jednokierunkowej.

5.1.1 Opis szczegółowy

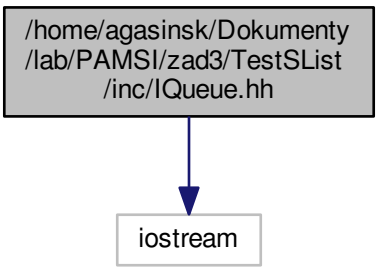
Plik zawiera interfejs listy jednokierunkową

Definicja w pliku IList.hh.

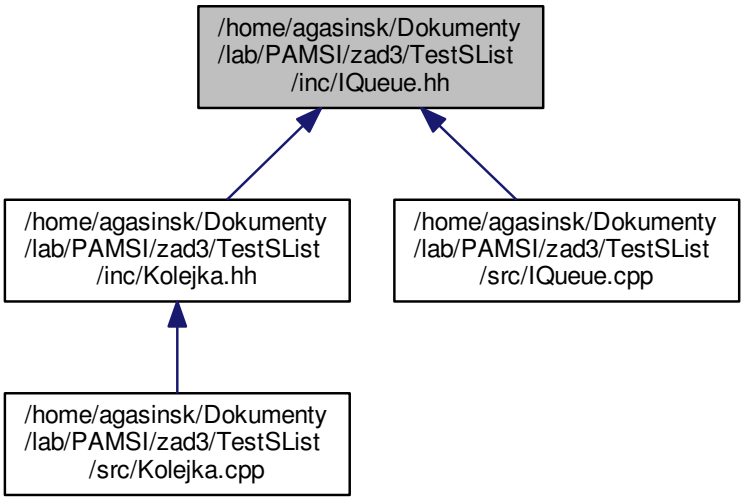
5.2 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/IQueue.hh

```
#include <iostream>
```

Wykres zależności załączania dla IQueue.hh:



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



Komponenty

- class IQueue< Object >

  Klasa modelująca interfejs kolejki.

5.2.1 Opis szczegółowy

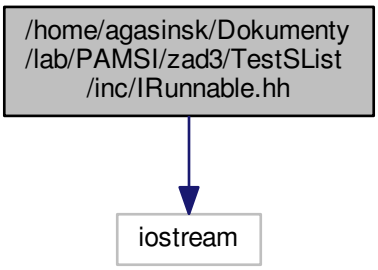
Plik zawiera interfejs kolejki

Definicja w pliku IQueue.hh.

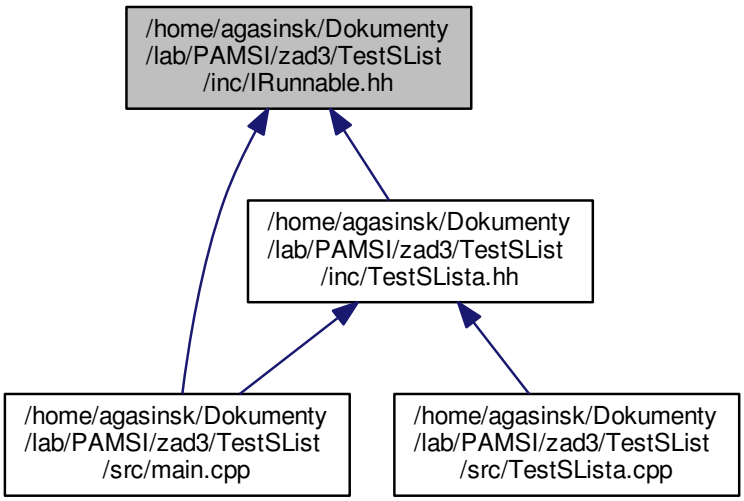
5.3 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/IRunnable.hh

```
#include <iostream>
```

Wykres zależności załączania dla IRunnable.hh:



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



Komponenty

- class IRunnable< Object >

  Klasa szablonowa modelująca interfejs "Biegacza".

5.3.1 Opis szczegółowy

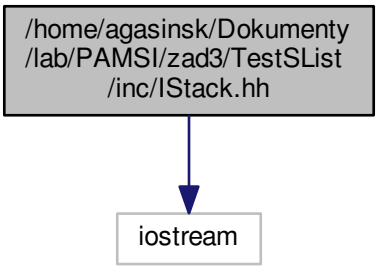
Plik zawiera interfejs obiektu, który można poddawać pomiarom czasu działania.

Definicja w pliku IRunnable.hh.

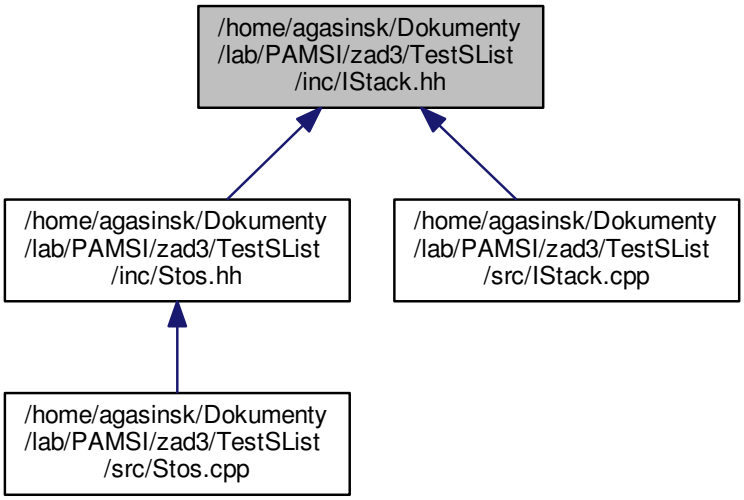
5.4 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/IStack.hh

```
#include <iostream>
```

Wykres zależności załączania dla IStack.hh:



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



Komponenty

- class IStack< Object >

  Klasa szablonowa modelująca interfejs stosu.

5.4.1 Opis szczegółowy

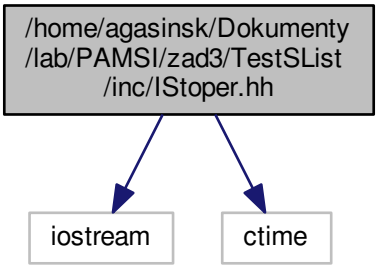
Plik zawiera interfejs stosu

Definicja w pliku IStack.hh.

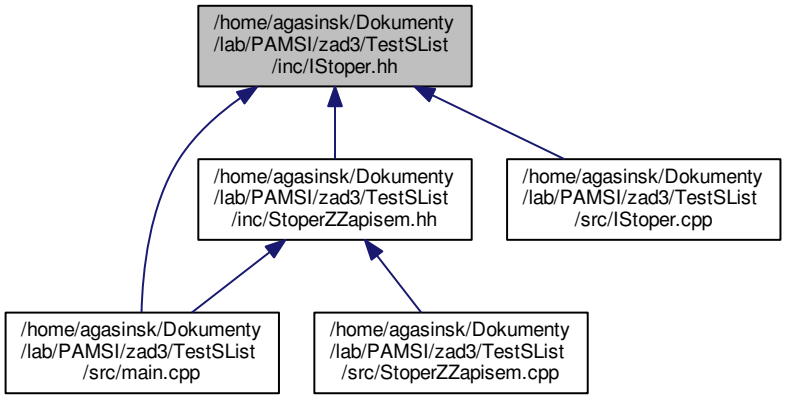
5.5 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/IStoper.hh

```
#include <iostream>
#include <ctime>
```

Wykres zależności załączania dla IStoper.hh:



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



Komponenty

- class IStoper

  Klasa modelująca interfejs stopera.

5.5.1 Opis szczegółowy

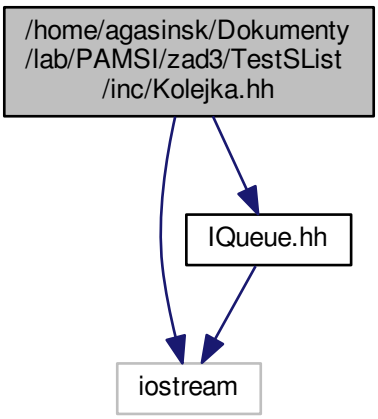
Plik zawiera definicję interfejsu podstawowego stopera.

Definicja w pliku IStoper.hh.

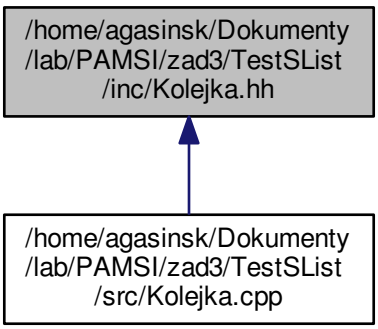
5.6 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/-Kolejka.hh

```
#include <iostream>
#include "IQueue.hh"
```

Wykres zależności załączania dla Kolejka.hh:



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



Komponenty

- class Kolejka< Object >

  Klasa szablonowa implementująca kolejkę

5.6.1 Opis szczegółowy

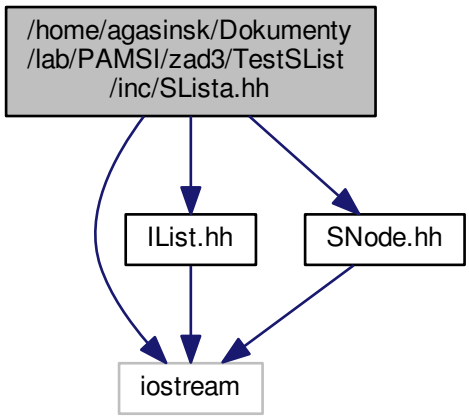
Plik zawiera implementację interfejsu kolejki

Definicja w pliku Kolejka.hh.

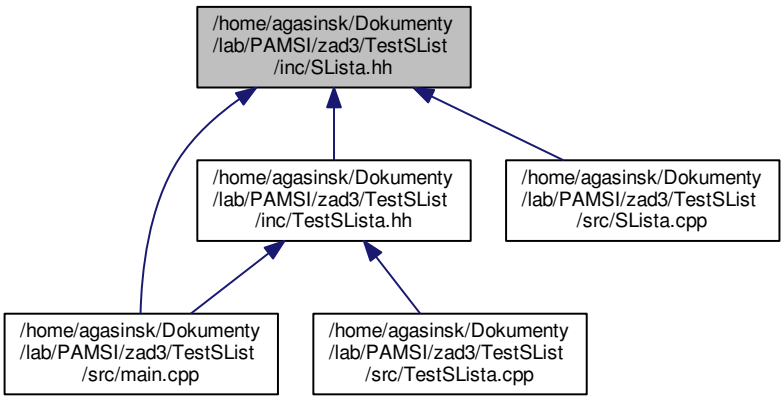
5.7 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/SLista.hh

```
#include <iostream>
#include "IList.hh"
#include "SNode.hh"
```

Wykres zależności załączania dla SLista.hh:



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



Komponenty

- class SLista< Object >

  Szablonowa klasa implementująca listę jednokierunkową

5.7.1 Opis szczegółowy

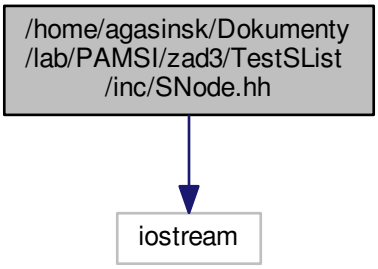
Plik zawiera definicję klasy implementującej listę jednokierunkową.

Definicja w pliku SLista.hh.

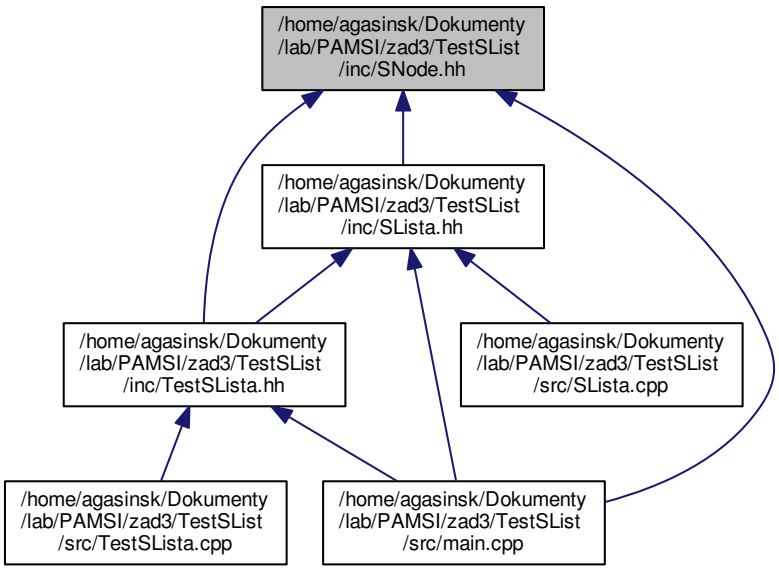
5.8 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/SNode.hh

```
#include <iostream>
```

Wykres zależności załączania dla SNode.hh:



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



Komponenty

- class SNode< Object >

  Klasa implementująca węzeł listę jednokierunkową

5.8.1 Opis szczegółowy

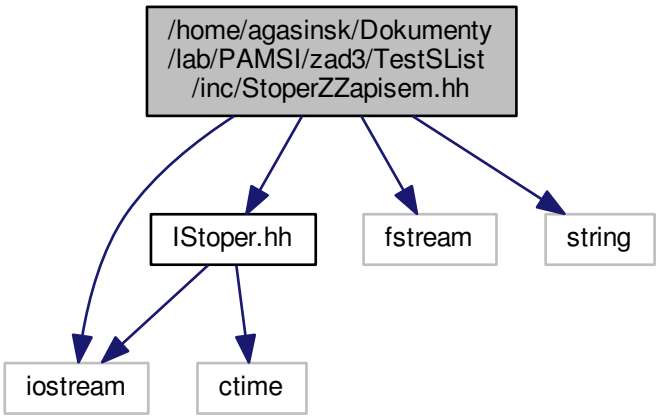
Plik zawiera definicję węzła listy jednokierunkowej.

Definicja w pliku SNode.hh.

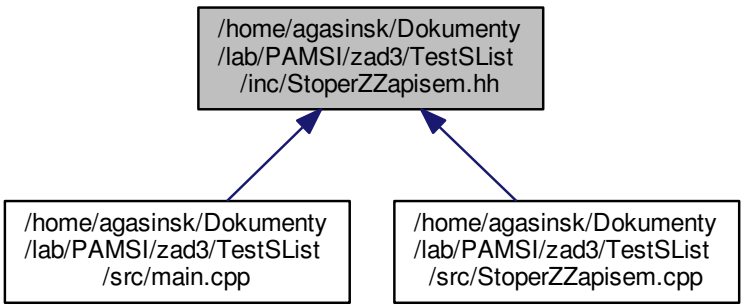
5.9 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/StoperZZapisem.hh

```
#include "IStoper.hh"
#include <iostream>
#include <fstream>
#include <string>
```

Wykres zależności załączania dla StoperZZapisem.hh:



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



Komponenty

- class StoperZZapisem

  Klasa implementująca rozbudowany stoper.

Definicje

- #define POJEMNOSC 35
- #define BUFOR 6

5.9.1 Opis szczegółowy

Plik zawiera implementację rozbudowanego stopera.

Definicja w pliku StoperZZapisem.hh.

5.9.2 Dokumentacja definicji

5.9.2.1 #define BUFOR 6

Definicja w linii 11 pliku StoperZZapisem.hh.

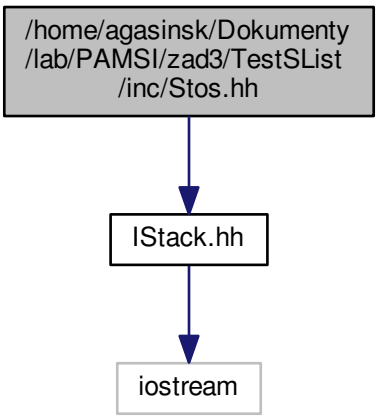
5.9.2.2 #define POJEMNOSC 35

Definicja w linii 10 pliku StoperZZapisem.hh.

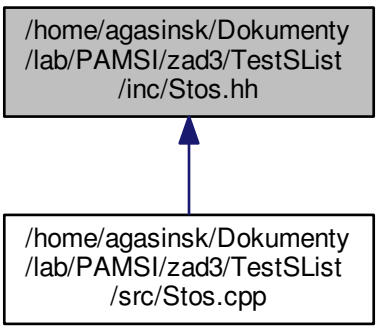
5.10 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/-Stos.hh

```
#include "IStack.hh"
```

Wykres zależności załączania dla Stos.hh:



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



Komponenty

- class Stos< Object >

  Klasa szablonowa implementująca stos.

5.10.1 Opis szczegółowy

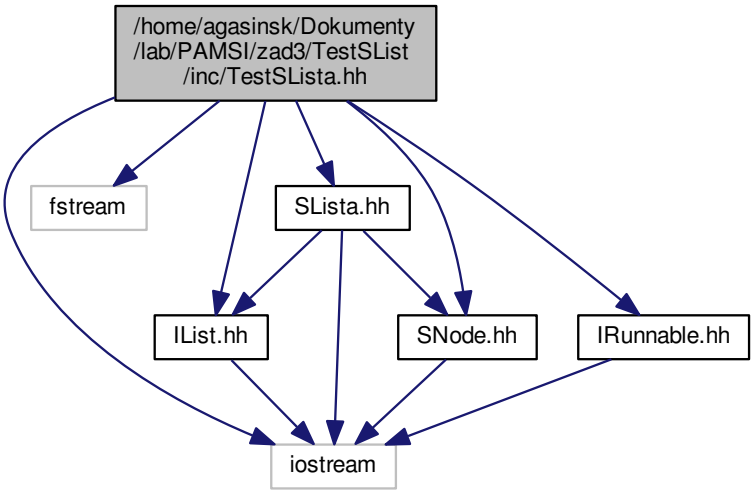
Plik zawiera implementację interfejsu stosu

Definicja w pliku Stos.hh.

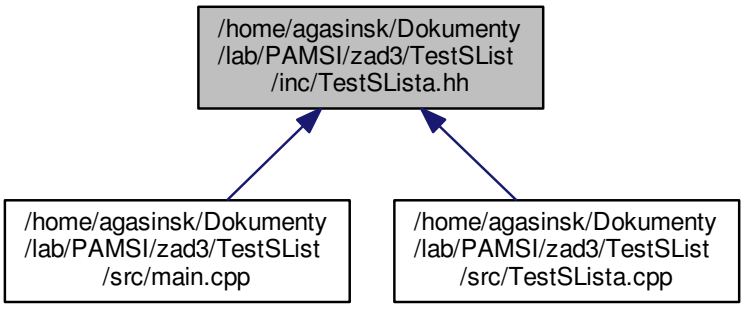
5.11 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/inc/TestSLista.hh

```
#include <iostream>
#include <fstream>
#include "IList.hh"
#include "SNode.hh"
#include "SLista.hh"
#include "IRunnable.hh"
```

Wykres zależności załączania dla TestSLista.hh:



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



Komponenty

- class TestSLista< Object >

  Szablonowa klasa listy jednokierunkowej implementująca IRunnable.

5.11.1 Opis szczegółowy

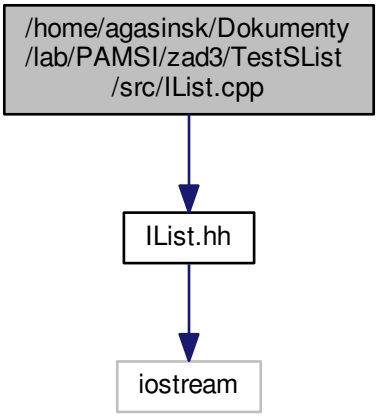
Plik zawiera definicję klasy implementującej listę jednokierunkową oraz interfejs IRunnable

Definicja w pliku TestSLista.hh.

5.12 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/IList.cpp

```
#include "IList.hh"
```

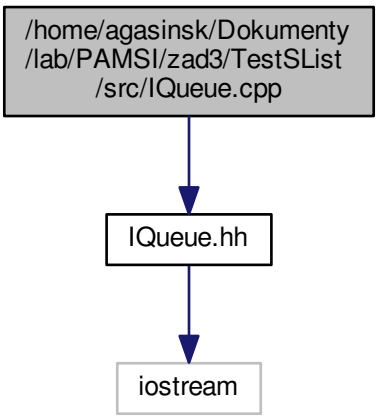
Wykres zależności załączania dla IList.cpp:



5.13 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/IQueue.cpp

```
#include "IQueue.hh"
```

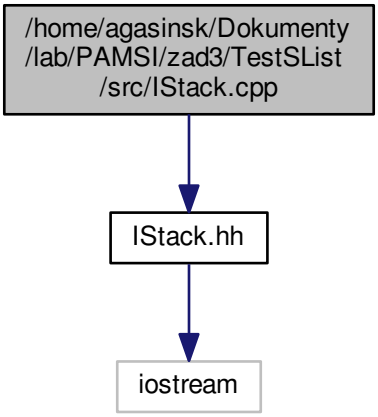
Wykres zależności załączania dla IQueue.cpp:



5.14 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/IStack.cpp

```
#include "IStack.hh"
```

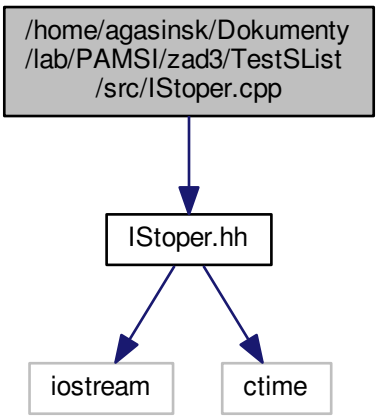
Wykres zależności załączania dla IStack.cpp:



5.15 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/IStoper.cpp

```
#include "IStoper.hh"
```

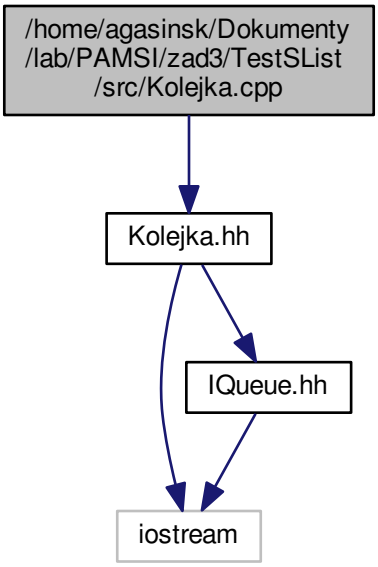
Wykres zależności załączania dla IStoper.cpp:



5.16 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/-Kolejka.cpp

```
#include "Kolejka.hh"
```

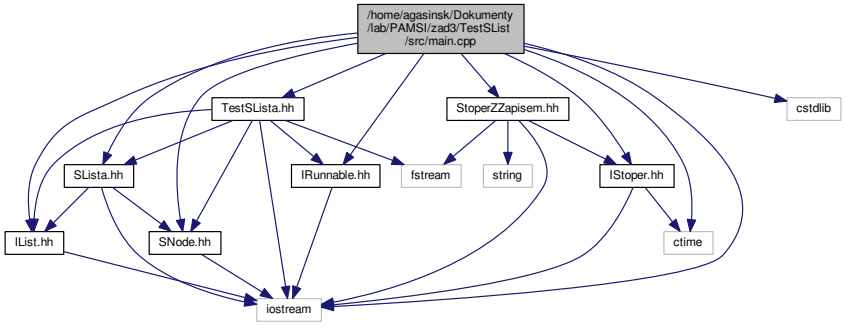
Wykres zależności załączania dla Kolejka.cpp:



5.17 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/main.cpp

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include "IRunnable.hh"
#include "IList.hh"
#include "SNode.hh"
#include "SLista.hh"
#include "TestSLista.hh"
#include "IStoper.hh"
#include "StoperZZapisem.hh"
```

Wykres zależności załączania dla main.cpp:



Definicje

- #define LOSOWANE 1000

Funkcje

- int main (int argc, char ∗argv[])

5.17.1 Opis szczegółowy

Zawiera kod źródłowy głównego modułu programu.

Definicja w pliku main.cpp.

5.17.2 Dokumentacja definicji

5.17.2.1 #define LOSOWANE 1000

Definicja w linii 13 pliku main.cpp.

5.17.3 Dokumentacja funkcji

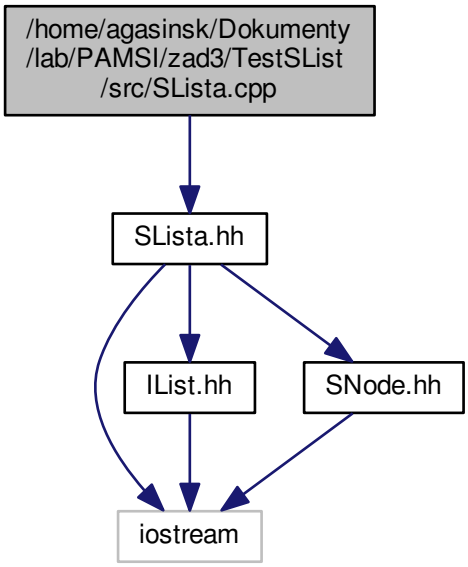
5.17.3.1 int main (int *argc,* char ∗ *argv[]*)

Definicja w linii 20 pliku main.cpp.

5.18 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/SLista.cpp

```
#include "SLista.hh"
```

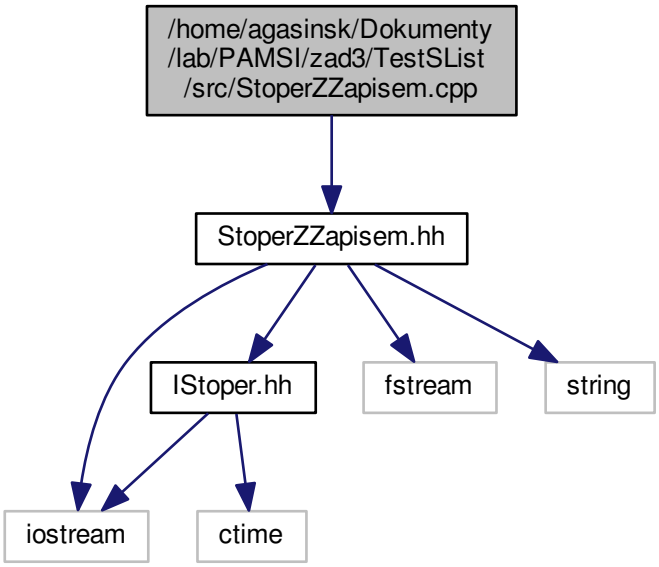
Wykres zależności załączania dla SLista.cpp:



5.19 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/-StoperZZapisem.cpp

```
#include "StoperZZapisem.hh"
```

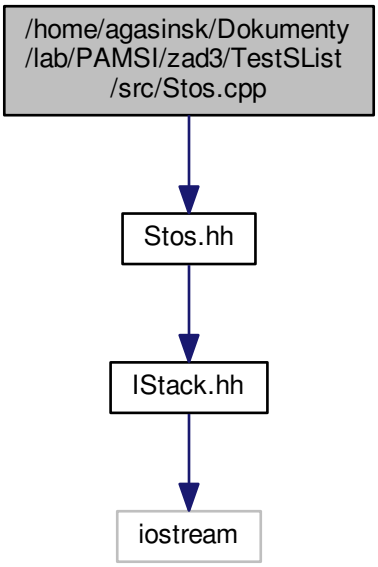
Wykres zależności załączania dla StoperZZapisem.cpp:



5.20 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/-Stos.cpp

```
#include "Stos.hh"
```

Wykres zależności załączania dla Stos.cpp:



5.21 Dokumentacja pliku /home/agasinsk/Dokumenty/lab/PAMSI/zad3/TestSList/src/-TestSLista.cpp

```
#include "TestSLista.hh"
```

Wykres zależności załączania dla TestSLista.cpp: