Tablica z haszowaniem jako tablica asocjacyjna

Wygenerowano przez Doxygen 1.8.6

N, 17 kwi 2016 16:37:33

Spis treści

4 Dokumentacja klas

4.1 Dokumentacja klasy AdvancedStopwatch

Klasa implementująca rozbudowany stoper.

`#include <AdvancedStopwatch.hh>`

Diagram dziedziczenia dla AdvancedStopwatch

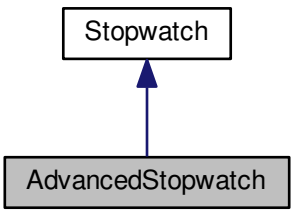
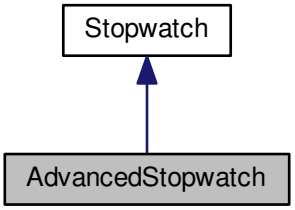


Diagram współpracy dla AdvancedStopwatch:



Metody publiczne

- AdvancedStopwatch ()
- ~AdvancedStopwatch ()
- unsigned & Rozmiar ()
- bool SaveElapsedTime (double rekord)

  Metoda zapisująca wartość pomiaru czasu okrążenia.
- double SeriesAverage ()

  Metoda wyliczająca średni czas okrążenia.
- bool SaveAverageTimeToBuffer (double rekord)

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

  Metoda wypisująca zawartość pamięci stopera.
- void CleanElapsedTimes ()

  Metoda usuwająca zawartość pamięci stopera.
- void CleanFileBuffer ()

  Metoda usuwająca zawartość bufora plikowego stopera.
- bool DumpFileBuffer (string nazwaPliku)

  Metoda zapisująca zawartość bufora plikowego do pliku.
- bool DumpToFile (string nazwaPliku, double rekord)

  Metoda zapisująca pojedynczy rekord bufora plikowego do pliku.

Dodatkowe Dziedziczone Składowe

4.1.1 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 27 pliku AdvancedStopwatch.hh.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 AdvancedStopwatch::AdvancedStopwatch ( )

Definicja w linii 4 pliku AdvancedStopwatch.cpp.

4.1.2.2 AdvancedStopwatch::∼AdvancedStopwatch ( )

Definicja w linii 15 pliku AdvancedStopwatch.cpp.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 void AdvancedStopwatch::CleanElapsedTimes ( )

Metoda usuwająca zawartość pamięci stopera.

Definicja w linii 66 pliku AdvancedStopwatch.cpp.

4.1.3.2 void AdvancedStopwatch::CleanFileBuffer ( )

Metoda usuwająca zawartość bufora plikowego stopera.

Definicja w linii 74 pliku AdvancedStopwatch.cpp.

4.1.3.3 bool AdvancedStopwatch::DumpFileBuffer (string *nazwaPliku*)

Metoda zapisująca zawartość bufora plikowego do pliku.

Dokonuje zapisu rekordów w buforze do pliku.

Parametry

>p0.10	>p0.15	p0.678
in *nazwaPliku* - nazwa pliku, do którego mają zostać zapisane czasy

Zwracane wartości

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

Definicja w linii 80 pliku AdvancedStopwatch.cpp.

4.1.3.4 bool AdvancedStopwatch::DumpToFile (string *nazwaPliku,* double *rekord*)

Metoda zapisująca pojedynczy rekord bufora plikowego do pliku.

Dokonuje zapisu wybranego rekordu w buforze do pliku.

Parametry

>p0.10	>p0.15	p0.678
in *nazwaPliku* - nazwa pliku, do którego ma zostać zapisany czas
in *rekord* - wartość pomiaru czasu, która ma być zapisana

Zwracane wartości

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

Definicja w linii 98 pliku AdvancedStopwatch.cpp.

4.1.3.5 void AdvancedStopwatch::PrintElapsedTimes ( )

Metoda wypisująca zawartość pamięci stopera.

Definicja w linii 58 pliku AdvancedStopwatch.cpp.

4.1.3.6 unsigned& AdvancedStopwatch::Rozmiar ( ) [inline]

Definicja w linii 35 pliku AdvancedStopwatch.hh.

4.1.3.7 bool AdvancedStopwatch::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 49 pliku AdvancedStopwatch.cpp.

4.1.3.8 bool AdvancedStopwatch::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 26 pliku AdvancedStopwatch.cpp.

4.1.3.9 double AdvancedStopwatch::SeriesAverage ( )

Metoda wyliczająca średni czas okrążenia.

Definicja w linii 37 pliku AdvancedStopwatch.cpp.

Dokumentacja dla tej klasy została wygenerowana z plików:

- prj/inc/AdvancedStopwatch.hh
- prj/src/AdvancedStopwatch.cpp

4.2 Dokumentacja szablonu klasy HashTab< Object >

Szablonowa klasa tablicy z haszowaniem.

`#include <HashTab.hh>`

Diagram dziedziczenia dla HashTab< Object >

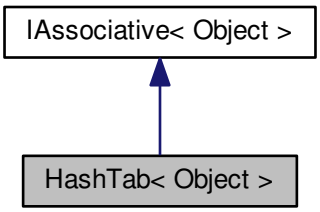
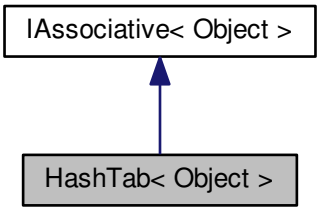


Diagram współpracy dla HashTab< Object >:



Metody publiczne

- HashTab ()

  Konstruktor tablicy z haszowaniem.
- ~HashTab ()

  Destruktor tablicy z haszowaniem.
- void ResetBuckets (int ile)

  Realokuje tablicę z haszowaniem dla podanej liczby elementów.
- unsigned HashFunction (string klucz)

  Funkcja skrótu.
- virtual Object & operator[] (string klucz)

  Przeciążenie operatora indeksującego tablicy z haszowaniem.
- virtual unsigned Insert (string klucz, Object newItem)

  Dodaje element o podanym kluczu do tablicy z haszowaniem.
- virtual Object Search (string klucz)

  Szuka elementu o podanym kluczu.

- virtual Object Delete (string klucz)

  Usuwa element o podanym kluczu.
- void PrintAll ()

  Drukuje całą tablicę z haszowaniem na standardowe wyjście.

4.2.1 Opis szczegółowy

template<typename Object>class HashTab< Object >

Szablonowa klasa tablicy z haszowaniem.

Klasa przechowuje elementy w tablicy N-elementowej, w której są umieszczone wskaźniki do listy węzłów w postaci: klucz, wartość. Możliwe jest odwołanie się do wartości za pośrednictwem klucza.

Definicja w linii 26 pliku HashTab.hh.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 template<typename Object > HashTab< Object >::HashTab ( )

Konstruktor tablicy z haszowaniem.

Inicjuje tablicę z haszowaniem poprzez zaolokowanie 5-elementowej tablicy list jednokierunkowych, w których można przechowywać elementy w postaci: (klucz (łańcuch znakowy), wartość).

Definicja w linii 128 pliku HashTab.hh.

4.2.2.2 template<typename Object > HashTab< Object >::~HashTab ( )

Destruktor tablicy z haszowaniem.

Zwalnia pamięć zajmowaną przez tablicę i ustawia wskaźnik na NULL.

Definicja w linii 135 pliku HashTab.hh.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 template<typename Object > Object HashTab< Object >::Delete (string *klucz*) [virtual]

Usuwa element o podanym kluczu.

Usuwa element z tablicy. Zwraca wartość pod podanym kluczem.

Parametry

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

`in` *klucz* - łańcuch znakowy do wyszukania

Zwraca

Wartość pod wskazanym kluczem.

Implementuje IAssociative< Object >.

Definicja w linii 209 pliku HashTab.hh.

4.2.3.2 template<typename Object > unsigned HashTab< Object >::HashFunction (string *klucz*)

Funkcja skrótu.

Oblicz indeks tablicy, do którego trafi element o podanym kluczu. Zamienia łańcuch znakowy na kod ASCII, jednocześnie wykonując działania arytmetyczne z dobranymi wartościami zmiennej a i stałej b (w tym dzielenie modulo

przez pojemność tablicy. Dzięki temu funkcja zapewnia równomierny rozkład kluczy w tablicy z haszowaniem. Jeśli w wyniku obliczeń indeks będzie ujemny, funkcja zwróci wynik dodawania ujemnego indeksu i pojemności tablicy.

Parametry

>p0.10	>p0.15	p0.678
in *klucz* - łańcuch znakowy, dla którego ma zostać wyliczony indeks

Definicja w linii 156 pliku HashTab.hh.

4.2.3.3 template<typename Object > unsigned HashTab< Object >::Insert (string *klucz,* Object *newItem*) [virtual]

Dodaje element o podanym kluczu do tablicy z haszowaniem.

Wywołuje funkcję skrótu, a następnie wstawia nowy element do tablicy pod otrzymanym indeksem. Jeśli otrzymany indeks jest już zajęty, element zostaje wstawiony na początek listy przechowywanej pod nim.

Parametry

>p0.10	>p0.15	p0.678
in *klucz* - łańcuch znakowy do dodania
in *newItem* - wartość dostępna pod podanym kluczem

Zwraca

Indeks w tablicy, pod którym znajduje się podany klucz.

Implementuje IAssociative< Object >.

Definicja w linii 190 pliku HashTab.hh.

4.2.3.4 template<typename Object > Object & HashTab< Object >::operator[] (string *klucz*) [virtual]

Przeciążenie operatora indeksującego tablicy z haszowaniem.

Umożliwia odczyt i zapis do tablicy z haszowaniem elementu o podanym kluczu. Wywołuje metodę skrótu, a następnie szuka elementu znajdującego się pod podanym kluczem na liście w otrzymanym przez funkcję skrótu indeksie.

Parametry

>p0.10	>p0.15	p0.678
in *klucz* - łańcuch znakowy do dodania lub odczytania

Zwraca

Element pod wskazanym kluczem.

Implementuje IAssociative< Object >.

Definicja w linii 182 pliku HashTab.hh.

4.2.3.5 template<typename Object > void HashTab< Object >::PrintAll ( )

Drukuje całą tablicę z haszowaniem na standardowe wyjście.

Elementy są wypisywane wg kolejnych indeksów tablicy tylko, jeśli dany indeks jest zajęty.

Definicja w linii 216 pliku HashTab.hh.

4.2.3.6 template<typename Object > void HashTab< Object >::ResetBuckets (int *ile*)

Realokuje tablicę z haszowaniem dla podanej liczby elementów.

Zwalnia pamięć zajmowaną przez starą tablicę, jeśli istniała wcześniej. Alokuje nową tablicę o pojemności 75% podanej wartości + 1.

Parametry

>p0.10	>p0.15	p0.678
`in` *ile* - liczba elementów, które mają być docelowo przechowywane w tablicy

Definicja w linii 144 pliku HashTab.hh.

4.2.3.7 template<typename Object > Object HashTab< Object >::Search (string *klucz*) [virtual]

Szuka elementu o podanym kluczu.

Wywołuje funkcję skrótu, a następnie przeszukuje listę pod otrzymanym indeksem. Zwraca element dostępny pod podanym kluczem (jeśli nie ma wartości, zwraca 0).

Parametry

>p0.10	>p0.15	p0.678
`in` *klucz* - łańcuch znakowy do wyszukania

Zwraca

Element pod wskazanym kluczem.

Implementuje IAssociative< Object >.

Definicja w linii 200 pliku HashTab.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

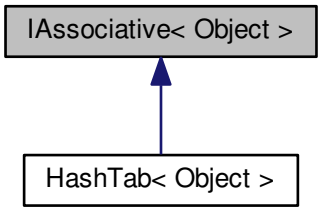
- prj/inc/HashTab.hh

4.3 Dokumentacja szablonu klasy IAssociative< Object >

Szablonowa klasa interfejsu tablicy asocjacyjnej.

`#include <IAssociative.hh>`

Diagram dziedziczenia dla IAssociative< Object >



Metody publiczne

- virtual Object & operator[] (string klucz)=0

Przeciążenie operatora indeksującego tablicy asocjacyjnej.

- virtual unsigned Insert (string klucz, Object newItem)=0

  Dodaje wartość o podanym kluczu.

- virtual Object Search (string klucz)=0

  Szuka wartości o podanym kluczu.

- virtual Object Delete (string klucz)=0

  Usuwa element o podanym kluczu.

4.3.1 Opis szczegółowy

template<typename Object>class IAssociative< Object >

Szablonowa klasa interfejsu tablicy asocjacyjnej.

Definiuje ADT dla tablicy asocjacyjnej.

Definicja w linii 19 pliku IAssociative.hh.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 template<typename Object > virtual Object IAssociative< Object >::Delete (string *klucz*) [pure virtual]

Usuwa element o podanym kluczu.

Usuwa element z tablicy. Zwraca wartość pod podanym kluczem.

Parametry

>p0.10	>p0.15	p0.678
in *klucz* - łańcuch znakowy do wyszukania

Zwraca

Wartość pod wskazanym kluczem.

Implementowany w HashTab< Object >.

4.3.2.2 template<typename Object > virtual unsigned IAssociative< Object >::Insert (string *klucz,* Object *newItem*) [pure virtual]

Dodaje wartość o podanym kluczu.

Parametry

>p0.10	>p0.15	p0.678
in *klucz* - łańcuch znakowy do dodania
in *newItem* - wartość dostępna pod podanym kluczem

Zwraca

Indeks w tablicy, pod którym znajdzie się podany klucz.

Implementowany w HashTab< Object >.

4.3.2.3 template<typename Object > virtual Object& IAssociative< Object >::operator[] (string *klucz*) [pure virtual]

Przeciążenie operatora indeksującego tablicy asocjacyjnej.

Umożliwia odczyt i zapis do tablicy z haszowaniem elementu o podanym kluczu.

Parametry

>p0.10	>p0.15	p0.678
`in` *klucz* - łańcuch znakowy do dodania lub odczytania

Zwraca

Wartość pod wskazanym kluczem.

Implementowany w HashTab< Object >.

4.3.2.4 template<typename Object > virtual Object IAssociative< Object >::Search (string *klucz*) [pure virtual]

Szuka wartości o podanym kluczu.

Zwraca wartość dostępną pod podanym kluczem.

Parametry

>p0.10	>p0.15	p0.678
`in` *klucz* - łańcuch znakowy do wyszukania

Zwraca

Wartość pod wskazanym kluczem.

Implementowany w HashTab< Object >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

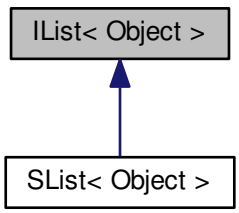
- prj/inc/IAssociative.hh

4.4 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.

4.4.1 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.

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 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 SList< Object >.

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

Metoda zwracająca pierwszy element listy.

Zwraca

pierwszy element listy

Implementowany w SList< Object >.

4.4.2.3 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 SList< Object >.

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

Metoda usuwająca element z początku listy.

Implementowany w SList< Object >.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- prj/inc/IList.hh

4.5 Dokumentacja szablonu klasy Node< Object >

Klasa implementująca węzeł listy jednokierunkowej.

`#include <Node.hh>`

Metody publiczne

- string GetKey ()

  Metoda zwracająca obecny klucz węzła.
- Object & GetValue ()

  Metoda zwracająca obecną wartość węzła.
- Node< Object > * GetNext ()

  Metoda zwracająca wskaźnik do następnego węzła.
- void SetKey (string newKey)

  Metoda przypisująca klucz do węzła.
- void SetValue (Object newValue)

  Metoda przypisująca wartość do węzła.
- void SetNext (Node< Object > *newItem)

  Metoda przypisująca następny element do węzła.

4.5.1 Opis szczegółowy

template<typename Object>class Node< Object >

Klasa implementująca węzeł listy jednokierunkowej.

Węzeł pozwalana przechowywać dane w formie: (klucz, wartość). Wartość może być dowolnego typu dzięki zastosowaniu szablonu.

Definicja w linii 21 pliku Node.hh.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 template<typename Object> string Node< Object >::GetKey ( ) [inline]

Metoda zwracająca obecny klucz węzła.

Zwraca

element w polu key węzła

Definicja w linii 33 pliku Node.hh.

4.5.2.2 template<typename Object> Node<Object>* Node< Object >::GetNext ( ) [inline]

Metoda zwracająca wskaźnik do następnego węzła.

Zwraca

wskaźnik w polu next węzła

Definicja w linii 45 pliku Node.hh.

4.5.2.3 template<typename Object> Object& Node< Object >::GetValue ( ) [inline]

Metoda zwracająca obecną wartość węzła.

Zwraca

element w polu value węzła

Definicja w linii 39 pliku Node.hh.

4.5.2.4 template<typename Object> void Node< Object >::SetKey (string *newKey*) [inline]

Metoda przypisująca klucz do węzła.

Ustawia podany element w polu key węzła.

Parametry

>p0.10	>p0.15	p0.678
in *newKey* - klucz do ustawienia w węźle

Definicja w linii 52 pliku Node.hh.

4.5.2.5 template<typename Object> void Node< Object >::SetNext (Node< 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 66 pliku Node.hh.

4.5.2.6 template<typename Object> void Node< Object >::SetValue (Object *newValue*) [inline]

Metoda przypisująca wartość do węzła.

Ustawia podany element w polu value węzła.

Parametry

>p0.10	>p0.15	p0.678
in *newValue* - wartość do ustawienia w węźle

Definicja w linii 59 pliku Node.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- prj/inc/Node.hh

4.6 Dokumentacja szablonu klasy SList< Object >

Szablonowa klasa implementująca listę jednokierunkową

`#include <SList.hh>`

Diagram dziedziczenia dla SList< Object >

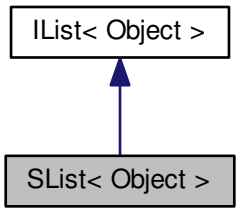
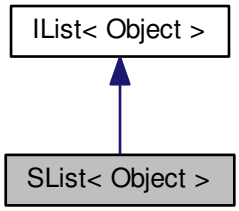


Diagram współpracy dla SList< Object >:



Metody publiczne

- SList ()

  Konstruktor listy jednokierunkowaj.
- ~SList ()

  Destruktor listy jednokierunkowaj.
- 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.
- void AddFront (const string newKey, const Object newItem)

  Metoda dodająca nowy węzeł na początek listy.
- virtual void RemoveFront ()

  Metoda usuwająca element z początku listy.

- Node< Object > * Head ()
  Metoda zwracająca głowę listy.
- Object Remove (string key)
  Metoda usuwająca węzeł o podanym kluczu.
- void Print ()
  Metoda wypisująca zawartość listy na standardowe wyjście.
- Node< Object > * Find (Object k)
  Metoda wyszukująca element na liście.
- Node< Object > * Find (string key)
  Metoda wyszukująca klucz na liście.
- Node< Object > * FindOrAdd (string key)
  Wyszukuje lub tworzy klucz na liście.

4.6.1 Opis szczegółowy

template<typename Object>class SList< 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 SList.hh.

4.6.2 Dokumentacja konstruktora i destruktora

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

Konstruktor listy jednokierunkowaj.

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

Definicja w linii 156 pliku SList.hh.

4.6.2.2 template<typename Object > SList< Object >::~SList ( )

Destruktor listy jednokierunkowaj.

Usuwa listę poprzez zwalnianie pamięci kolejnych początkowych węzłów.

Definicja w linii 160 pliku SList.hh.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 template<typename Object > void SList< 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 194 pliku SList.hh.

4.6.3.2 template<typename Object > void SList< Object >::AddFront (const string *newKey,* const Object *newItem*)

Metoda dodająca nowy węzeł na początek listy.

Alokuje nowy węzeł, przypisuje podane klucz i wartość, dodaje powiązanie tak, aby węzeł wskazywał na stary head, uaktualnia head.

Parametry

>p0.10	>p0.15	p0.678
in	*newKey*	- klucz do dodania
in	*newItem*	- wartość do dodania

Definicja w linii 203 pliku SList.hh.

4.6.3.3 template<typename Object > Node< Object > * SList< 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 254 pliku SList.hh.

4.6.3.4 template<typename Object > Node< Object > * SList< Object >::Find (string *key*)

Metoda wyszukująca klucz na liście.

Liniowo przeszukuje listę porównując kolejne klucze z podanym.

Parametry

>p0.10	>p0.15	p0.678
in	*key*	- klucz do wyszukania

Zwraca

Wskaźnik do znalezionego elementu lub NULL, gdy nie znaleziono.

Definicja w linii 265 pliku SList.hh.

4.6.3.5 template<typename Object > Node< Object > * SList< Object >::FindOrAdd (string *key*)

Wyszukuje lub tworzy klucz na liście.

Liniowo przeszukuje listę porównując kolejne klucze z podanym. Jeśli klucza nie znaleziono, na początek listy dodawany jest nowy węzeł o podanym kluczu. Jego wartość (domyślnie równą 0) można ustawić poprzez operację przypisania (za pomocą operatora []).

Parametry

>p0.10	>p0.15	p0.678
in	*key*	- klucz do wyszukania lub utworzenia

Zwraca

Wskaźnik do znalezionego lub nowoutworzonego elementu.

Definicja w linii 275 pliku SList.hh.

4.6.3.6 template<typename Object > const Object & SList< 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

element pierwszego węzła listy

Implementuje IList< Object >.

Definicja w linii 185 pliku SList.hh.

4.6.3.7 template<typename Object > Node< Object > * SList< 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 176 pliku SList.hh.

4.6.3.8 template<typename Object > bool SList< 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 167 pliku SList.hh.

4.6.3.9 template<typename Object > void SList< Object >::Print ( )

Metoda wypisująca zawartość listy na standardowe wyjście.

Definicja w linii 247 pliku SList.hh.

4.6.3.10 template<typename Object > Object SList< Object >::Remove (string *key*)

Metoda usuwająca węzeł o podanym kluczu.

Znajduje węzeł o podanym kluczu. Uaktualnia wskaźnik poprzednika na następny element listy. Usuwa go, zapamiętują wartość w nim przechowywaną.

Parametry

>p0.10	>p0.15	p0.678
in *key*	- klucz elementu do usunięcia	

Definicja w linii 225 pliku SList.hh.

4.6.3.11 template<typename Object > void SList< 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 213 pliku SList.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

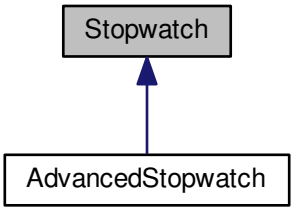
- prj/inc/SList.hh

4.7 Dokumentacja klasy Stopwatch

Klasa implementująca podstawowy stoper.

`#include <Stopwatch.hh>`

Diagram dziedziczenia dla Stopwatch



Metody publiczne

- virtual void Start ()

  Rozpoczyna pomiar czasu.
- virtual void Stop ()

  Kończy pomiar czasu.
- virtual double GetElapsedTime ()

  Oblicza czas na podstawie pól klasy.

Atrybuty chronione

- timeval start
- timeval stop

4.7.1 Opis szczegółowy

Klasa implementująca podstawowy stoper.

Klasa jest modelelem stopera z funkcjami start, stop i oblicz czas.

Definicja w linii 20 pliku Stopwatch.hh.

4.7.2 Dokumentacja funkcji składowych

4.7.2.1 double Stopwatch::GetElapsedTime ( ) [virtual]

Oblicza czas na podstawie pól klasy.

Odejmuje wartości zapisane w polach stop i start. Daje wynik w mikrosekundach.

Definicja w linii 12 pliku Stopwatch.cpp.

4.7.2.2 void Stopwatch::Start ( ) [virtual]

Rozpoczyna pomiar czasu.

Przypisuje wynik metody gettimeofday() do pola start.

Definicja w linii 4 pliku Stopwatch.cpp.

4.7.2.3 void Stopwatch::Stop ( ) [virtual]

Kończy pomiar czasu.

Przypisuje wynik metody gettimeofday() do pola stop.

Definicja w linii 8 pliku Stopwatch.cpp.

4.7.3 Dokumentacja atrybutów składowych

4.7.3.1 timeval Stopwatch::start [protected]

Definicja w linii 23 pliku Stopwatch.hh.

4.7.3.2 timeval Stopwatch::stop [protected]

struktury timeval start i stop

Definicja w linii 23 pliku Stopwatch.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

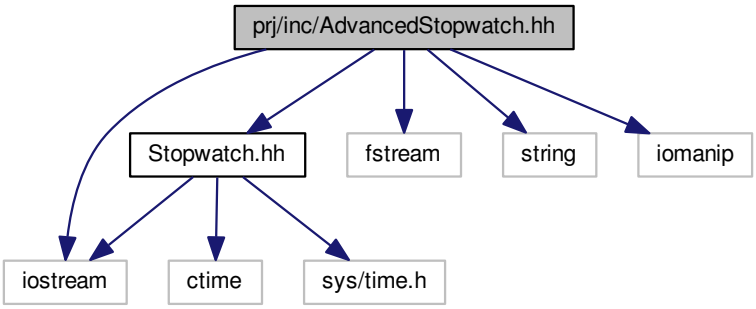
- prj/inc/Stopwatch.hh
- prj/src/Stopwatch.cpp

5 Dokumentacja plików

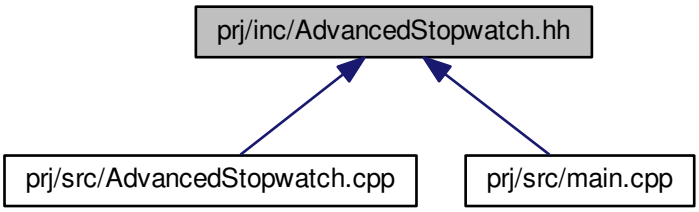
5.1 Dokumentacja pliku prj/inc/AdvancedStopwatch.hh

```
#include "Stopwatch.hh"
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
```

Wykres zależności załączania dla AdvancedStopwatch.hh:



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



Komponenty

- class AdvancedStopwatch

  Klasa implementująca rozbudowany stoper.

Definicje

- #define MAX_LAPS 100
- #define BUFOR 10

5.1.1 Opis szczegółowy

Plik zawiera implementację rozbudowanego stopera.

Definicja w pliku AdvancedStopwatch.hh.

5.1.2 Dokumentacja definicji

5.1.2.1 #define BUFOR 10

Definicja w linii 12 pliku AdvancedStopwatch.hh.

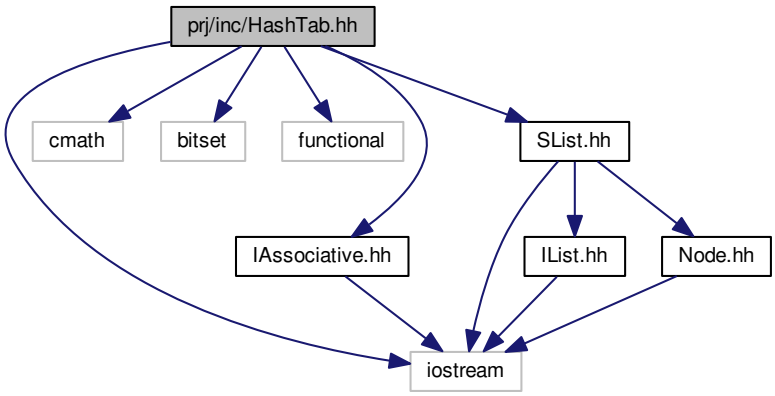
5.1.2.2 #define MAX_LAPS 100

Definicja w linii 11 pliku AdvancedStopwatch.hh.

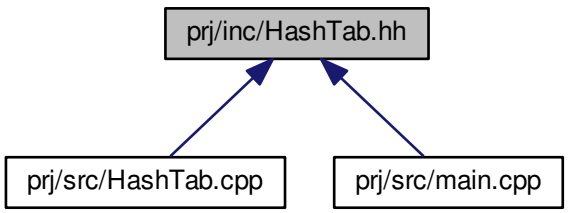
5.2 Dokumentacja pliku prj/inc/HashTab.hh

```
#include <iostream>
#include <cmath>
#include <bitset>
#include <functional>
#include "IAssociative.hh"
#include "SList.hh"
```

Wykres zależności załączania dla HashTab.hh:



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



Komponenty

- class HashTab< Object >

  Szablonowa klasa tablicy z haszowaniem.

5.2.1 Opis szczegółowy

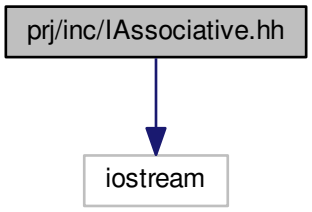
Plik zawiera implementację tablicy z haszowaniem.

Definicja w pliku HashTab.hh.

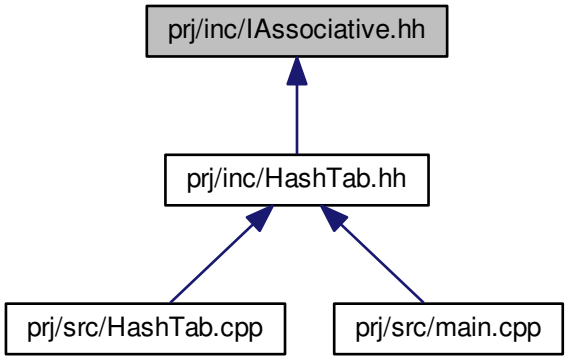
5.3 Dokumentacja pliku prj/inc/IAssociative.hh

```
#include <iostream>
```

Wykres zależności załączania dla IAssociative.hh:



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



Komponenty

- class IAssociative< Object >

Szablonowa klasa interfejsu tablicy asocjacyjnej.

5.3.1 Opis szczegółowy

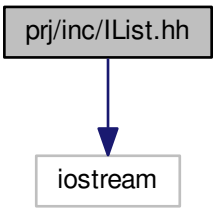
Plik zawiera interfejs tablicy asocjacyjnej.

Definicja w pliku IAssociative.hh.

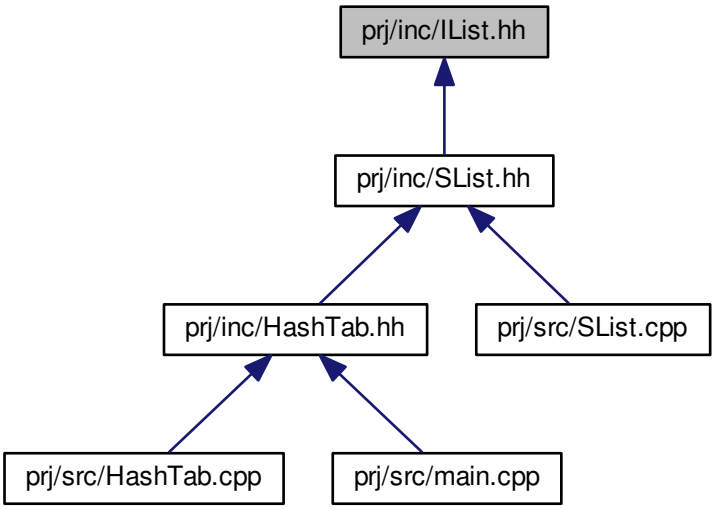
5.4 Dokumentacja pliku prj/inc/IList.hh

```
#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.

5.4.1 Opis szczegółowy

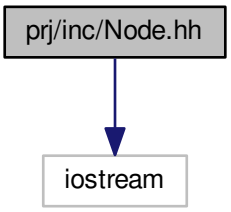
Plik zawiera interfejs listy jednokierunkową

Definicja w pliku IList.hh.

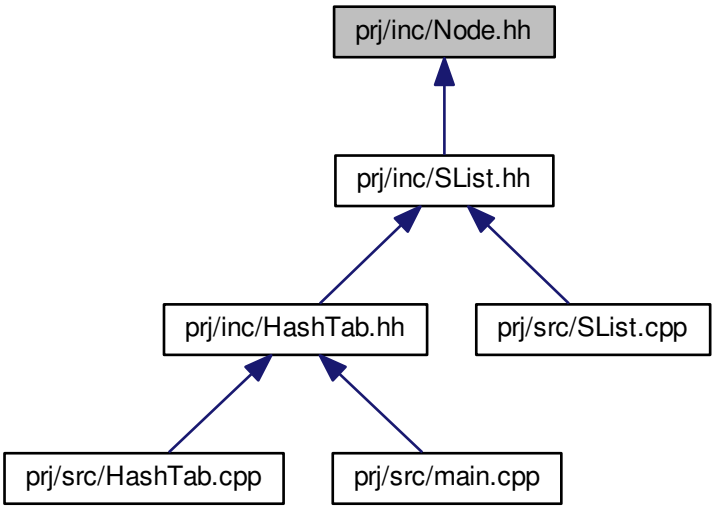
5.5 Dokumentacja pliku prj/inc/Node.hh

```
#include <iostream>
```

Wykres zależności załączania dla Node.hh:



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



Komponenty

- class Node< Object >

Klasa implementująca węzeł listy jednokierunkowej.

5.5.1 Opis szczegółowy

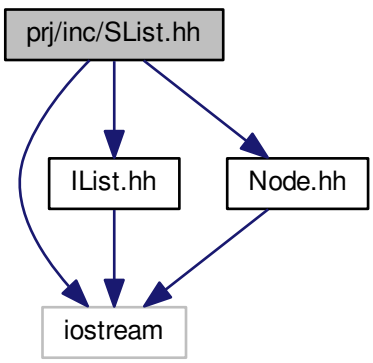
Plik zawiera definicję węzła listy jednokierunkowej.

Definicja w pliku Node.hh.

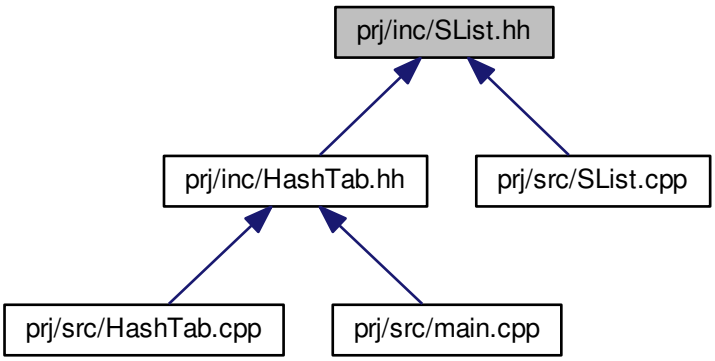
5.6 Dokumentacja pliku prj/inc/SList.hh

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

Wykres zależności załączania dla SList.hh:



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



Komponenty

- class SList< Object >

  Szablonowa klasa implementująca listę jednokierunkową

5.6.1 Opis szczegółowy

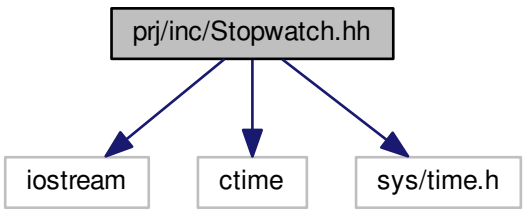
Plik zawiera definicję klasy implementującej listę jednokierunkową.

Definicja w pliku SList.hh.

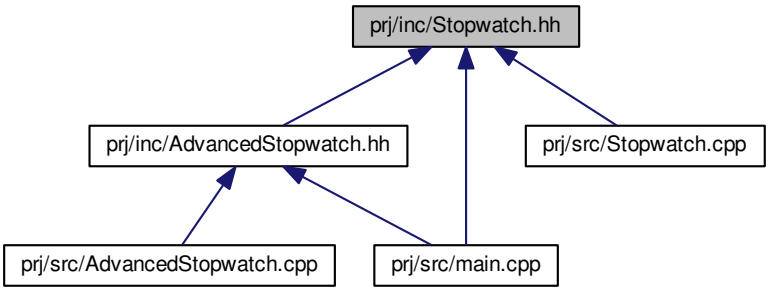
5.7 Dokumentacja pliku prj/inc/Stopwatch.hh

```
#include <iostream>
#include <ctime>
#include <sys/time.h>
```

Wykres zależności załączania dla Stopwatch.hh:



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



Komponenty

- class Stopwatch

  Klasa implementująca podstawowy stoper.

5.7.1 Opis szczegółowy

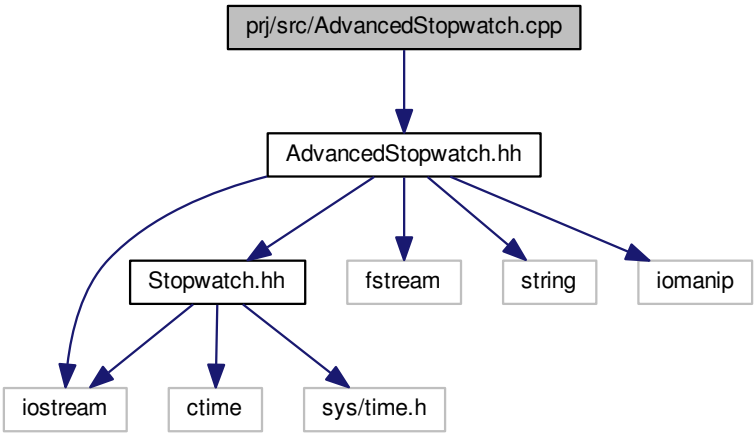
Plik zawiera implementację podstawowego stopera.

Definicja w pliku Stopwatch.hh.

5.8 Dokumentacja pliku prj/src/AdvancedStopwatch.cpp

```
#include "AdvancedStopwatch.hh"
```

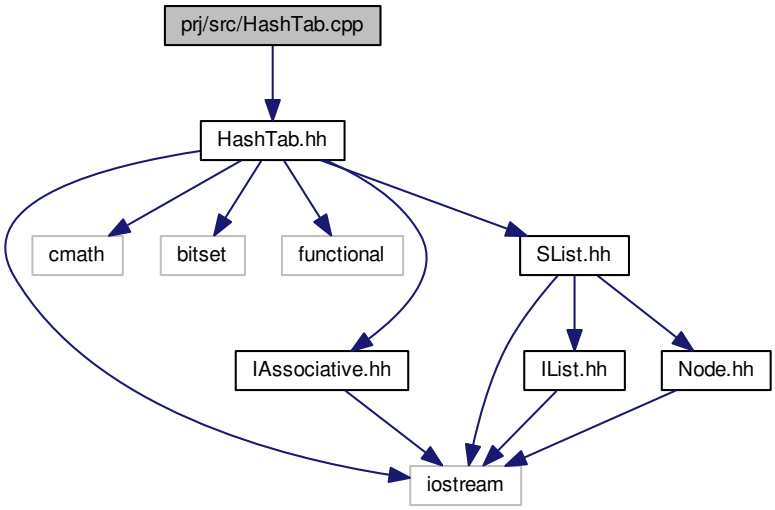
Wykres zależności załączania dla AdvancedStopwatch.cpp:



5.9 Dokumentacja pliku prj/src/HashTab.cpp

```
#include "HashTab.hh"
```

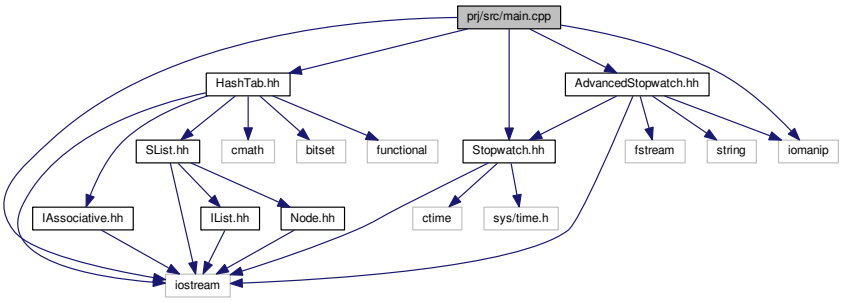
Wykres zależności załączania dla HashTab.cpp:



5.10 Dokumentacja pliku prj/src/main.cpp

```
#include <iostream>
#include <iomanip>
#include "HashTab.hh"
#include "Stopwatch.hh"
#include "AdvancedStopwatch.hh"
```

Wykres zależności załączania dla main.cpp:



Funkcje

- int main (int argc, char ∗argv[])

5.10.1 Opis szczegółowy

Zawiera kod źródłowy głównej funkcji programu.

Definicja w pliku main.cpp.

5.10.2 Dokumentacja funkcji

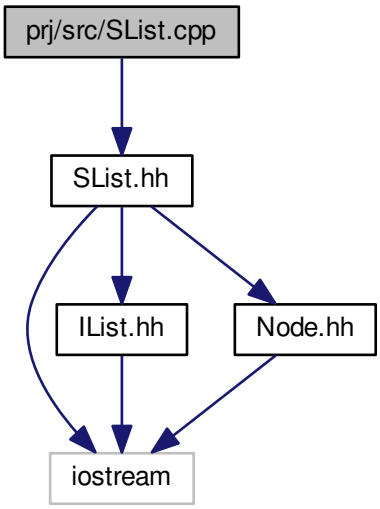
5.10.2.1 int main (int *argc,* char * *argv[]*)

Definicja w linii 15 pliku main.cpp.

5.11 Dokumentacja pliku prj/src/SList.cpp

```
#include "SList.hh"
```

Wykres zależności załączania dla SList.cpp:



5.12 Dokumentacja pliku prj/src/Stopwatch.cpp

```
#include "Stopwatch.hh"
```

Wykres zależności załączania dla Stopwatch.cpp: